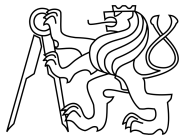
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza současných řešení Backend-as-a-Service pro vývoj mobilních a webových aplikací
Student:	Bc. Michal Májský
Vedoucí:	Ing. Vladislav Skoumal
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

1. Vymezte termín (mobile) Backend-as-a-Service (dále BaaS).
2. Analyzujte současný trh BaaS. Popište, jaká řešení jsou na trhu, jaké mají společné charakteristiky a čím se odlišují.
3. Po dohodě s vedoucím práce vyberte 3 - 5 konkrétních BaaS řešení. Na konkrétní aplikaci (či aplikacích), kterou implementujete s použitím každého z vybraných BaaS řešení, analyzujte silné a slabé stránky těchto řešení a popište, pro jaké typy projektů jsou vhodné.
4. Na vhodných příkladech demonstrujte a srovnejte ekonomické náklady využití vybraných BaaS řešení.
5. Na základě výstupů z předchozího kroku:
   a) se pokuste o zobecnění a zhodnocení, kdy je vhodné na projektu vývoje mobilní nebo webové aplikace použít BaaS řešení,
   b) vyslovte názor, jakým směrem se bude trh s BaaS ubírat.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 21. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Analýza současných řešení Backend-as-a-Service pro vývoj mobilních a webových aplikací

Bc. Michal Májský

Vedoucí práce: Ing. Vladislav Skoumal

5. května 2016

Poděkování

Děkuji Ing. Vladislavu Skoumalovi za metodické vedení této práce a flexibilní spolupráci. Děkuji Ing. Josefu Gattermayerovi za vymyšlení rámcového tématu této práce. Děkuji Ing. Pavlu Náplavovi za pomoc s výběrem tématu mé práce a metodické rady. Děkuji také rodině a blízkým za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. května 2016 .....

Odkaz na tuto práci

Májský, Michal. *Analýza současných řešení Backend-as-a-Service pro vývoj mobilních a webových aplikací.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá vysvětlením pojmu Backend as a Service (BaaS), kategorizací a popisem řešení tohoto typu dostupných na trhu, detailně srovnává BaaS řešení Firebase, Parse a Kinvey. Přečtení této práce pomůže kvalifikovaně rozhodnout o vhodnosti použití BaaS řešení pro projekt mobilní nebo webové aplikace.

Klíčová slova Backend as a Service, BaaS, mBaaS, Firebase, Kinvey, Parse

Abstract

This thesis deals with explanation of term Backend as a Service (BaaS) and categorization and description of BaaS solutions available on the market. It compares in detail BaaS solutions Firebase, Parse and Kinvey. Reading of this thesis could help to make informed decision about suitability of BaaS solution usage on mobile or web application project.

Keywords Backend as a Service, BaaS, mBaaS, Firebase, Kinvey, Parse

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tvorba prvních dynamických webových aplikací je spjata s rokem 1993, kdy byl poprvé představen protokol Common Gateway Interface [1]. Koncept webových aplikací, ve kterém (ve zjednodušené formě) na počátku stojí HTTP požadavek, který je zpracován na serveru nějakým skriptem, jehož výstupem je odpověď v podobě HTML stránky, má tak za sebou více než 20 let vývoje. Aktuálním trendem v oblasti webových aplikací a standardem v oblasti mobilních aplikací je částečné přesunutí aplikační logiky ze serveru na klienta (webový prohlížeč nebo mobilní aplikaci) za účelem zvýšení komfortu uživatelského rozhraní a lepšího tzv. „user experience“. Na straně serveru tak obvykle zůstává datová a byznys vrstva aplikace a rozhraní, které umožňuje k těmto vrstvám přistupovat (v kontextu webových aplikací nejčastěji přes RESTful API). Tento fakt, spolu s již dostatečnou vyspělostí backendových technologií, vede k možnosti komodizovat serverovou část řešení a nabídnout ji jako službu. Tyto služby jsou pak označovány jako Backend as a Service a první vznik těchto služeb se datuje do roku 2011 [2].

V současné době neexistuje mnoho případových studií popisujících využití těchto služeb. Pro manažery i vývojáře může být obtížné odhadnout, zda nějaké BaaS řešení může efektivně řešit jejich problém. Chybně zvolené řešení přitom může vést k neúspěchu celého projektu nebo k výraznému prodražení. Tato situace se stala motivací pro moji práci, ve které jsem se rozhodl prozkoumat aktuální situaci v oblasti Backend as a Service poskytovatelů.

Specifikace cíle

Hlavním cílem mé práce je poskytnout vývojářům a manažerům informace, které jim pomohou kvalifikovaně rozhodnout o využití či nevyužití BaaS na projektu webové nebo mobilní aplikace. Pro naplnění tohoto hlavního cíle jsem stanovil několik dílčích cílů.

Prvním cílem je vymezit pojem (m)BaaS, tedy poskytnout čtenáři základní

informace o tom, co se pod tímto pojmem skrývá, jaké problémy BaaS může pomáhat řešit a jaká rizika jeho využití přináší.

Druhým cílem je popsat současný trh s BaaS, tedy identifikovat jaké služby jsou aktuálně na trhu, popsat co mají společné, čím se liší a pokusit se je určitým způsobem kategorizovat.

Třetím cílem je srovnat několik vybraných konkrétních BaaS řešení z technického hlediska za účelem odhalení silných a slabých stránek těchto řešení, prozkoumat jejich použitelnosti pro specifické typy aplikací a získat praktické zkušenosti s použitím BaaS řešení.

Čtvrtým cílem je srovnat ekonomické náklady na užívání vybraných BaaS řešení - analyzovat, jak rozdílné účtovací přístupy jednotlivých BaaS poskytovatelů ovlivňují výslednou cenu, a poskytnout čtenáři návod jak odhadnout náklady na užívání BaaS.

Pátým cílem je na základě zkušeností a výstupů ze všech předchozích kroků zhodnotit skutečnou použitelnost BaaS řešení pro vývoj webové nebo mobilní aplikace.

Šestým cílem je na základě předchozích zkušeností odhadnout, jakým směrem se bude trh s BaaS řešeními dále ubírat.

Použité termíny

Některé termíny uváděné v této práci zůstávají nepřeložené v původním anglickém znění. Domnívám se, že v současné době neexistují zavedené české překlady a že zavedení vlastních překladů v této práci by mohlo vést k dezinformaci čtenáře a komplikovalo jeho další zkoumání problematiky. Některé termíny jsou přeložené, ale při jejich prvním použití je uveden také původní anglický výraz.

Rešerše

V oblasti odborných publikací lze dohledat jen několik zdrojů věnujících se problematice BaaS z jakéhokoliv pohledu. Mezi ně patří diplomové práce Tomáše Hermana [3] a Tomáše Linharta [4]. Obě práce se soustředí především na tvorbu vlastní služby Backend as a Service. Součástí jejich práce je ale také definice pojmu BaaS a popis trhu v době psaní jejich prací. Diplomová práce Johana Laanstra z TU Delft [5] se zaměřuje na problematiku implementace funkcionality Offline & sync (viz 3.4.3) pro BaaS řešení a v rámci tohoto tématu se věnuje také srovnání této funkcionality u významných hráčů na trhu v té době.

Na internetu lze dohledat poměrně hodně článků, které se v nějaké formě věnují vymezení BaaS, popisují jaké benefity by mělo BaaS uživateli přinést a srovnávají vybraná BaaS řešení. Problémem většiny z těchto článků ovšem je, že se nezaobírají problematikou v dostatečně širokém kontextu a v případě

srovnávání služeb se často zaměřují jen na určitý aspekt těchto služeb (např. řeší nabízenou funkcionalitu, ale již neřeší možnosti nasazení) nebo jen na několik poskytovatelů, aniž by bylo dostatečně vysvětleno, proč jsou porovnáváni právě tito poskytovatelé a ne jiní.

Dalším problémem mnoha internetových článků je, že jejich autory jsou osoby, které se nějakou formou podílí na vývoji konkrétní BaaS služby. Tento fakt je třeba zohlednit při přebírání informací z těchto článků.

Mezi kvalitnější internetové články, věnující se problematice, bych zařadil sumarizační článek o BaaS [2], jehož autorem je Kin Layne. Tato práce se v nějaké podobě zaobírá všemi tématy, kterými se zaobírá má práce, avšak v mnohem menším rozsahu. Druhým článkem, který svojí kvalitou, dle mého názoru, převyšuje jiné internetové články je článek Davida Tuckera [6], který se také zaobírá jak definicí BaaS, tak stručnou analýzou trhu. Dalším zajímavým článkem je krátká analýza společnosti Gartner [7], která se na problematiku BaaS zaměřuje více z pohledu velkých firem. Posledním článkem, který bych zmínil, je poměrně obsáhlé srovnání pěti řešení od Martina Hellera (problém tohoto článku vidím ve faktu, že některé ze srovnávaných služeb jsou dle mého názoru MEAP - viz kapitola 2.2) [8].

Detailním technickým srovnáním vybraných BaaS služeb, konkrétně služeb SynergyKit a Firebase, se zabývají Jakub Nidžeradže a Tomáš Mistrik, vývojáři SynergyKit, v článku Vývoj mobilních aplikací na platformách Backend as a Service [9].

Mimo webové stránky provozovatelů samotných BaaS jsem nedohledal případové studie popisující úspěšné či neúspěšné adaptace BaaS řešení na projektech.

Kapitola 1

Vymezení pojmu Backend as a Service

Cílem této kapitoly je poskytnout čtenáři základní vysvětlení pojmu BaaS. Po přečtení této kapitoly by měl být čtenář schopen posoudit, zda je pro něj vhodné se tématem BaaS dále zabývat.

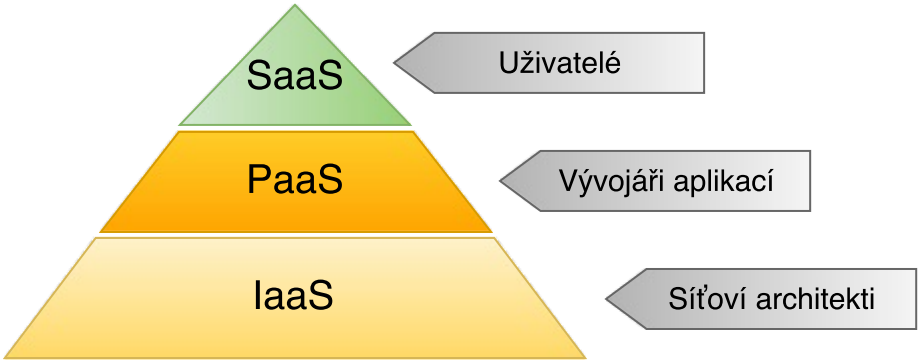
1.1 Rozbor pojmu a zařazení v cloud computingu

Termín „Backend as a Service“ lze rozebrat na dvě částí - „Backend“ a „as a Service“. Termín backend odkazuje na tradiční rozdělení aplikací na frontend a backend, přičemž jako frontend je chápána ta část aplikace, která je viditelná uživateli a se kterou uživatel komunikuje. Backend je uživateli skrytý a zpravidla sestává z API, databáze a určité aplikační logiky.

Spojení „as a Service“ je spojeno s cloud computingem a jeho distribučními modely, kde jsou již zažité a běžně používané termíny Infrastructure as a Service (IaaS), Platform as a Service (PaaS) a Software as a Service (SaaS) [10]. Pokud se na tyto distribuční modely budeme koukat tradičně jako na na sebe navazující vrstvy (viz obrázek 1.1), pak o BaaS lze uvažovat jako o mezivrstvě mezi PaaS a SaaS [4]. Existují ale i takové názory, že „není třeba rozlišovat mezi PaaS a BaaS, protože cílí na stejný trh a postupem času se spojí do jedné služby“ [11].

Tomáš Herma ve své práci k dané problematice píše: „BaaS lze považovat za určitou evoluci, na jejímž počátku bylo IaaS a PaaS. Z důvodů, že nasazení aplikace na platformu, jako je např. Amazon Web Services, bylo složité, bylo třeba vymyslet lepší způsob, tedy jakousi nadstavbu nad PaaS, kerou představuje právě BaaS“ [3]. Z tohoto úhlu pohledu lze tedy o BaaS přemýšlet také jako o SDK nebo API, které vývojářům jednoduše zpřístupňují funkcionalitu určité platformy.

V kontextu problematiky Backend as a Service se používají dva termíny -



Obrázek 1.1: Tradiční distribuční modely cloud computingu

„Backend as a Service“ (BaaS[1]) a „mobile Backend as a Service“ (mBaaS). Pro účely této práce můžeme tyto pojmy považovat za ekvivalentní. Někteří poskytovatelé hovoří o svém řešení jako o BaaS, jiní jako o mBaaS. V druhém případě je kladen důraz na to, že řešení je vhodné především pro mobilní aplikace. Zpravidla se ovšem jedná spíše o marketingovou strategii a řešení lze využít i pro vývoj webových aplikací.

1.2 Definice

Nejpoužívanější definice pro BaaS, které lze na internetu dohledat, se zpravidla opírají především o funkcionalitu a komponenty, ze kterých BaaS sestává, a příliš se neliší např. od této [12]:

„Backend as a Service provides web and mobile app developers with a way to link their applications to backend cloud storage while also providing features such as user management, push notifications, and integration with social networking services. These services are provided through customized Software Development Kits (SDKs) and Application Programming Interfaces (APIs).“

Pro názornost citujme ještě jednu definici uváděnou společností Gartner [7], která se zaměřuje více na podnikové BaaS (viz. kapitola 2.3.4):

„Mobile back end as a service (mBaaS) vendors deliver capabilities to mobile apps via published APIs that can be incorporated into mobile apps. mBaaS services are typically hosted services delivered as middleware between the client-based mobile apps and the back-end databases and applications that enterprises use to run the business. Typical services offered via these REST APIs include data hosting; identity and access management (IAM);

[1] V jiných zdrojích lze narazit na používání termínu BaaS jako zkratky pro Backup as a Service, což je zcela jiný segment služeb. V případě této práce se bude vždy jednat o zkratku pro Backend as a Service.

push notifications; business system integrations, such as CRM or ERP; and location. They are typically multitenant services that are hosted in the cloud, and many can be delivered as on-premises or virtual, private, cloud-hosted services. Many services offered in the mBaaS category can be used by other app projects, including Web apps, and investment in mobile support can and will benefit other areas.“

Jak výše uvedené definice říkají, BaaS tedy zpravidla sestává z těchto „komponent“:

- **Serverová část**, která poskytuje určitou funkcionalitu a je spravována uživatelem prostřednictvím webového rozhraní (to je označováno jako (webová) konzole nebo dashboard), případně prostřednictvím CLI.
- **REST API**, které vystavuje funkcionalitu serverové části a na které se buď přímo nebo nepřímo (pomocí SDK) napojuje uživatel se svojí aplikací.
- **SDK**, tedy knihovna, která je již specifická pro konkrétní programovací jazyk. Zpravidla „pod pokličkou“ pracuje s REST API, ale uživateli nabízí komfortnější a efektivnější rozhraní pro práci, než kdyby komunikoval s REST API napřímo.

Na obrázku 1.2 je přehledně znázorněno, jaké komponenty BaaS řešení obsahuje a jakým způsobem s nimi vývojáři pracují.

1.3 Funkcionalita poskytovaná BaaS

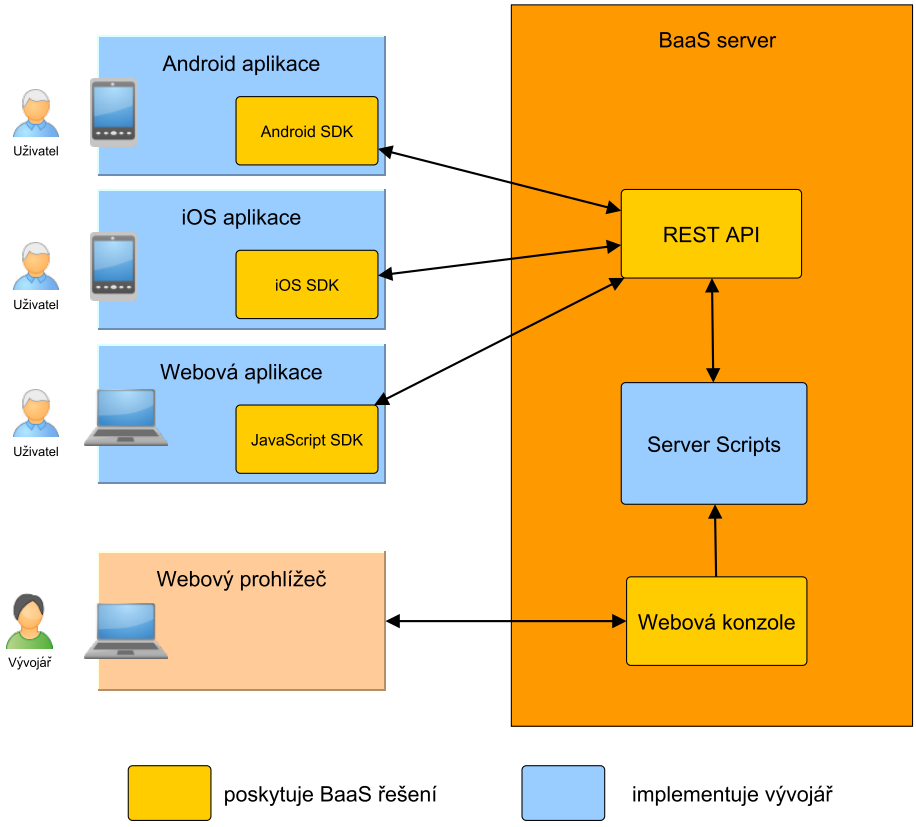
Ačkoliv přesná funkcionalita poskytovaná jednotlivými BaaS řešeními se liší, lze identifikovat funkcionalitu, která je pro BaaS řešení „typická“. Takovou funkcionalitu v nějaké formě poskytuje naprostá většina BaaS řešení.

Následující přehled a popis funkcionality je přesto třeba brát jako orientační. Jeho účelem je poskytnout čtenáři základní představu o tom, jaké možnosti BaaS nabízí. Přehled vychází z mé analýzy současných poskytovatelů a z práce dalších autorů [3][4][6].

Funkcionalita, která je již specifická pro každého poskytovatele, je popsána v kapitole 2.4.

1.3.1 Datové úložiště

Součástí BaaS je vždy úložiště pro (semi) strukturovaná data. Ve většině případů se jedná o nějaký typ NO-SQL databáze založené na ukládání dat ve formátu JSON (nejčastěji je v této kategorii používána MongoDB). Méně obvyklá jsou BaaS řešení, která používají tradiční relační databáze. Poskytovatelé většinou oficiálně neuvádějí, jaký konkrétní databázový stroj používají.



Obrázek 1.2: Architektura a komponenty BaaS řešení

Součástí SDK od BaaS poskytovatele je zpravidla jazyk pro provádění CRUD operací nad datovým úložištěm. Síla a možnosti dotazování se u jednotlivých poskytovatelů liší a závisí především na typu databáze. Zpravidla bývá také možnost komunikovat s datovým úložištěm použitím REST API.

1.3.2 Správa uživatelů

Jiné používané názvy: Authentication, Users, Identity and Access Management

Pod správou uživatelů se rozumí úložiště autentizačních a autorizačních informací a podpora na ně navázaných procesů, tedy:

- Registrace
- Přihlášení
- Změna hesla

- Obnova zapomenutého hesla

Dále je často poskytnuta podpora pro jednoduchou integraci autentizace přes sociální sítě či v případě enterprise BaaS podpora pro integraci LDAP, SSO apod.

1.3.3 Messaging

Jiné používané názvy: Publish-Subscribe messaging, Communication

BaaS řešení zpravidla podporuje jeden nebo více typů zasílání zpráv uživatelům aplikace. Nejčastěji je poskytováno jednoduché API pro zasílání tzv. push notifikací na mobilní zařízení, obvyklá je také možnost zasílání e-mailových zpráv. Méně častá je podpora SMS.

1.3.4 Server code

Jiné používané názvy: Custom Business logic, Custom server-side-code, KScripts, Cloud logic, Cloud code, Server Extension

Tato funkcionalita BaaS umožňuje implementovat vlastní aplikační logiku na straně serveru ve formě skriptů psaných nejčastěji prostřednictvím webového editoru.

Tyto skripty zpravidla mohou být spouštěny jedním z následujících způsobů:

- „Manuálně“ zavoláním prostřednictvím SDK či API.
- Navázáním na určitou událost (typicky databázovou operaci).
- Periodicky prostřednictvím plánovače.

Nejčastěji dostupným jazykem pro psaní serverového kódu je JavaScript.

1.4 Benefity využití BaaS

Obecným benefitem, který by mělo využití BaaS přinést uživateli[2] oproti vývoji backendu „na zelené louce“, je úspora času a peněz. Podrobněji lze identifikovat následující benefity:

- Schopnost vyvinout funkční aplikaci bez potřeby backend vývojářů.
- Rychlé získání určité funkcionality. Může se jednat o zcela běžnou funkcionalitu (správa uživatelů) nebo naopak specializovanou / nestandardní, jejíž vývoj „na zelené louce“ může být velmi náročný (např. real-time komunikace).

[2]Uživatelem v tomto kontextu míním osobu, která přímo pracuje s BaaS. Ve většině případech se tedy jedná o vývojáře (a vývojářské firmy)

- Škálovatelnost platformy / infrastruktury. BaaS řešení, stejně jako jiné formy cloud computingu, může účinně „za pochodu“ přizpůsobovat alokované zdroje aktuální poptávce a tím minimalizovat riziko nedostupnosti aplikace nebo zbytečných výdajů za naddimenzované řešení.
- Komfortní knihovna pro přenos dat mezi klientem a serverem. Úzkým hrdlem SPA nebo mobilní aplikace je často komunikace (synchronizace dat) mezi klientem a serverem (jako jeden z největších problému při vývoji webových aplikací to uvádí např. [13]). Ta je zpravidla postavena na RESTful API popř. SOAP. Implementace takového API a jeho konzumace na straně frontendu je však časově náročná. BaaS zpravidla poskytuje komfortnější rozhraní pro komunikaci specifické pro konkrétní jazyk (SDK) a uživatele abstrahuje od nutnosti pracovat přímo s rozhraním založeném na HTTP.

1.5 Rizika užívání BaaS

Využití BaaS na projektu sebou přináší (mimo potenciálních příležitostí popsaných v předchozích kapitolách) také rizika, která by měla být brána v potaz při rozhodování o tom, zda BaaS využít na daném projektu:

- **Čas a náklady na integraci BaaS řešení přerostou náklady na vývoj vlastního backendu.** Každé BaaS řešení, stejně jako jakákoliv technologie, vyžaduje od vývojáře osvojit si určitý způsob uvažování a naučit se technologii používat. Záleží na mnoha faktorech (např. využití technologie v budoucnu, současné znalosti vývojářů), zda se vyplatí do nové technologie investovat.
- **Kritické blokátory identifikované až v průběhu implementace.** I při detailní analýze BaaS řešení je obtížné domyslet všechny důsledky použití vybraného řešení. Dokumentace z principu popisuje co daná technologie umí a již ne to, co neumí. Vzorové příklady uváděné v dokumentaci zpravidla demonstrují řešení jednoho konkrétního izolovaného problému. Proto je zde riziko, že si vývojáři až v průběhu implementace uvědomí, že vybrané BaaS řešení „trpí“ závažným nedostatkem pro jejich projekt.
- **Neodhadnutí přímých nákladů na využívání BaaS.** Podobně jako u jiných modelů cloud computingu, s ohledem na způsob účtování BaaS, může být poměrně obtížné odhadnout, jaké budou výsledné náklady. Ty je možné zpravidla spočítat, pokud dokážeme odhadnout, kolik uživatelů bude naše aplikace mít a jak ji tito uživatelé budou používat.
- **Závislost na poskytovateli BaaS.** U projektů, kde předpokládáme dlouhou životnost, je třeba počítat s rizikem, že poskytovatel BaaS může tuto službu ukončit.

- **Náročná údržba a další rozvoj aplikace.** Důvodů pro problematickou údržbu a rozvoj projektu může být více. Např. špatná podpora migrace dat nebo změny struktury databáze či chybějící podpora pro více instancí aplikace (testovací, produkční, ...). Také je třeba zvážit, zda nehrozí, že s postupným rozvojem projektu funkcionalita poskytovaná BaaS přestane být pro projekt dostačující.

Kapitola 2

Analýza trhu s BaaS

Tato kapitola navazuje na kapitolu 1 a věnuje se hlouběji popisu BaaS řešení dostupných na trhu. Na základě přečtení této kapitoly by čtenář měl získat ucelenou představu o různých typech BaaS řešení, na jejímž základě je schopen zvolit kandidáty na vhodné BaaS řešení pro svůj projekt.

Obecně lze konstatovat, že trh s BaaS řešeními je v době psaní této práce velmi dynamický. Dynamický jednak ve smyslu, že stále vznikají nová řešení a zanikají stará, ale také ve smyslu, že řada existujících BaaS řešení prochází intenzivním rozvojem.

V době psaní této práce neexistuje komplexní katalog, který by shromažďoval informace o existujících poskytovatelích BaaS. Při jejich hledání je třeba spoléhat především na internetové vyhledávače. Na internetu lze sice nalézt řadu článků, které nějakým způsobem popisují existující BaaS řešení, ale zpravidla jen úzký výběr řešení, přičemž kritéria, jakým způsobem byla tato řešení vybrána, nejsou uvedena. Lze odhadnout, že na trhu se nachází několik desítek BaaS řešení.

Během práce na této kapitole jsem dohledal okolo 20 BaaS řešení[3]. Část z dohledaných BaaS řešení jsem rozdělil dle kategorií definovaných v kapitole 2.3. Vytvořený přehled je ve zjednodušené formě dostupný v příloze D a v plné verzi na CD ve formátu PDF.

2.1 Historie trhu

Vznik prvních BaaS řešení je zpravidla datován do roku 2011 [14][11], popřípadě do roku 2010 [7]. První BaaS řešení vznikla jako startup a jednalo se o consumer-oriented BaaS [7] (viz kapitola 2.3.4).

Později začali svá BaaS řešení vyvíjet také „tradiční velcí hráči“ (IBM, Microsoft, Oracle) či velcí cloudoví poskytovatelé jako Amazon.

[3]Je však třeba zmínit, že mým cílem nebylo poskytnout vyčerpávající seznam, ale získat přehled o tom, jaké typy BaaS řešení existují.

Za další důležité události lze označit zrušení BaaS StackMob v roce 2014 [15] a oznámení o ukončení hostované verze Parse v roce 2016 (této události se více věnuji v kapitole 3.2.2). V obou případech se jednalo o dominantní hráče na trhu ve své době.

2.2 Vztah BaaS a MEAP

V kapitole 1.1 jsem již popsal vztah PaaS, BaaS a SaaS. V této kapitole bych chtěl vymezit své pojetí pojmu BaaS vůči tzv. Mobile Enterprise Application Platforms, protože mezi těmito dvěma pojmy často není rozlišováno, ale podle mého názoru je jejich odlišení důležité.

MEAP je komplexní platforma pro vývoj podnikových mobilních aplikací s cílem překlenout problémy, které vznikají s množstvím existujících mobilních platforem. Tento segment jako takový vznikl dříve, než BaaS [16].

Mezi BaaS a MEAP lze identifikovat několik důležitých rozdílů:

1. MEAP je určený pouze pro vývoj podnikových mobilních aplikací. BaaS se nezaměřuje pouze na segment velkých podniků a je obvykle možné ho použít také pro vývoj webových, a v některých případech i desktopových, aplikací.

2. BaaS by měl umožnit vývojáři vyvíjet aplikace pomocí libovolného SDK či frameworku, který si zvolí. MEAP zpravidla přichází s vlastním frameworkem či tzv. development stackem, který je třeba pro vývoj aplikace použít [16]. Tedy, zatímco BaaS je opravdu backend, ke kterému je poskytnuto SDK ke zjednodušení komunikace s tímto backendem, MEAP vyvojářům určuje i technologie pro vývoj frontendu aplikace.

3. Použití BaaS je zpravidla poměrně jednoduché. Vývojář je schopný od „prvního dne“ bez hlubšího zaškolení či samostudia začít službu konzumovat. Použití MEAP zpravidla vyžaduje naučení se novému přístupu a vyžaduje řádově více času k osvojení [16].

Lze pozorovat, že součástí MEAP je v některých případech také BaaS. Tento BaaS je však integrovaný do MEAP řešení a není primárně určen pro použití s libovolným SDK (popř. není vůbec možné ho použít s jinými technologiemi, než těmi, které jsou součástí MEAP řešení).

Řada internetových článků porovnává řešení, která naplňují znaky MEAP (např. appcelerator[4] nebo appery.io[5]) vedle BaaS řešení. Netvrdím, že BaaS a MEAP nelze porovnávat, je však třeba si uvědomit, že debata o (ne)použití MEAP by se měla vést na jiné úrovni, než debata o (ne)použití BaaS, protože použití MEAP má na firmu či projekt větší dopady z těchto důvodů:

[4] `http://www.appcelerator.com`
[5] `https://appery.io/`

1. Použitím MEAP se zpravidla firma stane technologický závislá na dodavateli MEAP.

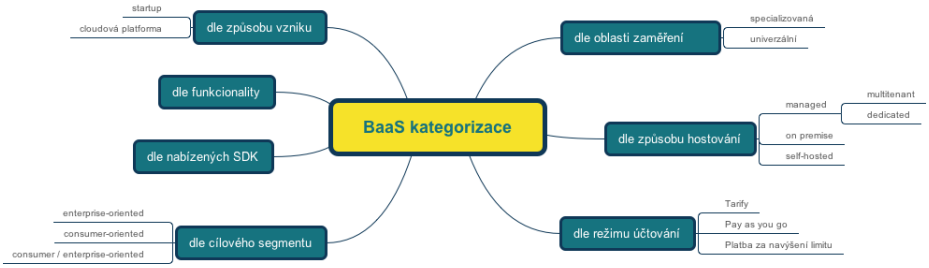
2. Náklady spojené s využíváním MEAP jsou zpravidla řádově vyšší ve srovnání s využíváním BaaS, protože je třeba vynaložit mnoho času na osvojení si technologie a také platit licence za používání MEAP.

2.3 Kategorizace BaaS řešení

Tomáš Linhart [4] ve své práci uvádí, že: „Provést vyhodnocení nebo srovnání jednotlivých mobilních služeb bohužel není dost dobře možné. Každý se orientuje na jiný druh zákazníků. Někdo na menší týmy a někdo na velké korporace. Plus i v případě, že se orientují na stejný segment zákazníků, tak se snaží odlišit, aby byly konkurence schopné. Původně jsem tedy zamýšlel sestrojit tabulku jednotlivých služeb, kde by bylo porovnání. Jediný možný případ by byl udělat nějaké základní kategorie, ale zde je problém, že prostě z definice BaaS je jasné, které funkce by služby měly mít a tím pádem tyto základní kategorie by splňovala téměř naprostá většina služeb.“.

Souhlasím, že provést vyhodnocení ve smyslu nalézt nejlepšího poskytovatele BaaS řešení, není možné, protože pro každý projekt bude nejvhodnější BaaS řešení jiné. Myslím si ovšem, že lze identifikovat určitá kritéria, která jednotlivé BaaS řešení odlišují a tímto způsobem pomoci čtenáři zorientovat se rychleji na trhu a zefektivnit jeho rozhodovací proces při výběru BaaS řešení.

Na základě analýzy webových prezentací a dokumentace pro vývojáře řady BaaS řešení a studia dalších pramenů zabývajících se BaaS jsem navrhl několik kritérii, dle kterých můžeme provést základní členění BaaS řešení. Tato kritéria jsou shrnuta na obrázku 2.1 a rozebrána v podkapitolách, které následují.



Obrázek 2.1: Kategorie BaaS

2.3.1 Dle oblasti zaměření

Dle tohoto kritéria lze identifikovat dvě kategorie BaaS řešení. První kategorii lze nazvat kategorií **univerzální**. Do této kategorie patří většina řešení. Řešení v této kategorii se nesoustředí na podporu určitého typu aplikace nebo technologie a lze je tedy použít pro vývoj různých aplikací. Druhou kategorií jsou řešení **specializovaná**, která se soustředí na podporu specifického typu aplikace nebo specifické technologie. Vhodným příkladem je např. Quickblox[6], který se soustředí na podporu aplikací pro Instant messaging nebo Backand[7], který se prezentuje jako BaaS řešení pro AngularJS[8].

2.3.2 Dle způsobu hostování

Toto kritérium dělí BaaS řešení podle toho, jakým způsobem je backend poskytovaný BaaS poskytovatelem hostovaný (toto dělení vychází z [6]). Pro toto kritérium lze identifikovat tři hlavní kategorie:

- **Managed** - O hostování BaaS řešení se stará poskytovatel. Je obvyklé, že BaaS poskytovatel pro hostování svého řešení využívá jiné cloudové PaaS popř. IaaS řešení. Tuto kategorii lze ještě rozdělit na tzv. **multi-tenant** či **dedicated** řešení, dle toho zda poskytovatel garantuje, že servery jsou vyhrazeny pouze pro zákazníka (dedicated) či nikoliv a tedy mohou být sdíleny s ostatními uživateli cloudového řešení (multi-tenant).

- **On-premise** - Poskytoval BaaS řešení nasadí řešení na servery zákazníka a spravuje je. To zpravidla obnáší, aby zákazník dodal infrastrukturu splňující určité požadavky poskytovatele. Toto řešení může být vhodné pro firmy pracující s citlivými údaji.

- **Self-hosted** - Odpovědnost a řešení nasazení a údržby je na straně zákazníka (uživatele BaaS). Tento způsob je typický pro open-source BaaS řešení. Při této variantě se vytrácí jedna z hlavních výhod využívání BaaS a lze diskutovat o tom, zda se stále jedná o BaaS.

Je poměrně běžné, že poskytovatel BaaS nabízí několik z výše uvedených možností hostování řešení. Consumer-oriented poskytovatelé si mohou dovolit nabízet pouze managed - multitenant variantu, zatímco pro enteprise-oriented poskytovatele je v podstatě nutností nabízet své řešení ve variantě managed-dedicated a on-premise. To vychází z faktu, že velké podniky zpravidla potřebují řešit bezpečnost a ochranu soukromí dat a s tím je spjatá potřeba mít pod kontrolou infrastrukturu, na které jsou data uložena.

[6] `http://quickblox.com/`
[7] `http//backand.com`
[8] MVC framework pro vývoj webových aplikací, `https://angularjs.org/`

2.3.3 Dle způsobu účtování

Pro všechna BaaS řešení, která byla zkoumána, platí, že nabízejí nějakou variantu zdarma, která je vhodná na vyzkoušení služby. Tato neplacená varianta je omezena buď nabízenými prostředky (a ve své podstatě se tedy jedná o tzv. freemium byznys model) nebo časem, po který může být služba využívána zdarma (pak lze hovořit o tzv. free trial byznys modelu). Případně je neplacená varianta omezena obojím - nabízenými prostředky i časem.

Lze vypozorovat, že BaaS poskytovatelé, kteří jsou primárně consumer-oriented (viz 2.3.4), dávají obvykle přednost freemium modelu, zatímco enterprise-oriented poskytovatelé zpravidla nabízí free trial.

Obecně lze konstatovat, že zpoplatněny jsou v důsledku různé „aspekty" BaaS řešení, pro které jsou nastaveny určité limity. Typickými příklady těchto zpoplatněných aspektů jsou:

- Velikost úložiště
- Počet uživatelů či zařízení
- Objem toku dat

Na základě toho, jak poskytovatel s těmito aspekty pracuje, lze poskytovatele rozdělit do tří kategorií:

- **Platba za tarif** - Poskytovatel nabízí různé tarify, které jsou charakterizovány maximálními limity pro řadu aspektů. Překročením limitu jednoho aspektu je tedy zákazník nucen přejít na vyšší tarif, popř. platit za překročené limity dle zvláštní sazby.
- **Platba za navýšení limitu** - V tomto případě poskytovatel účtuje překročení určitého limitu pro definované aspekty za každý zvlášť. Základní verze do určitých limitů je buď zdarma nebo za předem definovaný měsíční poplatek.
- **„Pay as you go"** - Tento model je podobný modelu obvyklého u PaaS / IaaS poskytovatelů. Zákazník platí za jednotku každého aspektu (např. za každý GB úložiště). Není tedy překvapením, že tento platební model nalezneme především u BaaS řešení od firem vlastnících cloudové platformy (viz kapitola 2.3.7).

2.3.4 Dle cílového segmentu trhu

Na základě nabízené funkcionality, marketingové strategie, způsobu účtování a dalších aspektů lze BaaS řešení rozdělit do dvou skupin.

První jsou tzv. **consumer-oriented**[9] BaaS, které cílí primárně na freelancery a malé, popř. střední firmy specializované na mobilní / webový vývoj.

[9] V jiných zdrojích se používá pro tuto skupinu také zkratka SMB - Small and Medium Business

Druhou skupinou jsou tzv. **enterprise-oriented** BaaS, které cílí zejména na velké firmy.

Lze vysledovat, že někteří poskytovatelé BaaS, kteří začali jako consumer-oriented, se postupně snaží doplnit funkcionalitu a možnosti nasazení tak, aby se stali zajímavými také pro velké podniky.

Obdobně platí, že i enterprise-oriented BaaS poskytovatelé se snaží zaujmout consumer trh - na svých webových prezentacích zveřejňují zvláštní sekce, kde popisují, jak může tato skupina zákazníků jejich řešení využívat.

2.3.5 Dle funkcionality

V kapitole 1.3 byla popsána funkcionalita, kterou v nějaké formě disponuje naprostá většina BaaS řešení. Jednotliví BaaS poskytovatelé ale zpravidla disponují také další funkcionalitou. Tu lze nějakým způsobem kategorizovat (tímto tématem se více zabývá kapitola 2.4), ovšem výsledná množina kategorií funkcionality poskytovaná každým BaaS poskytovatelem je víceméně unikátní, a proto nelze vymezit konkrétní skupiny BaaS poskytovatelů dle tohoto kritéria. V každém případě je ale nabízená funkcionalita dalším z klíčovým kritérií v procesu výběru BaaS poskytovatele.

2.3.6 Dle nabízených SDK

SDK, která daný poskytovatel BaaS nabízí, do značné míru určují, na kterých platformách[10] lze dané řešení pro vývoj použít. Nejčastěji jsou zastoupeny SDK pro JavaScript (někdy též nazývaná SDK pro HTML5 či web), Android a iOS. Méně často jsou dostupná SDK pro Windows Phone / .NET či Blackberry.

Je poměrně obvyklé, že BaaS poskytovatel nabízí také SDK pro určitý framework (např. AngularJS) nebo vývojářskou platformu (např. Apache Cordova či Xamarin pro vývoj hybridních aplikací). V této oblasti je však nabídka každého poskytovatele již velmi specifická.

BaaS lze zpravidla využít i bez SDK komunikováním prostřednictvím REST API, uživatel tak ale může být ochuzen o část funkcionality, která je na SDK závislá (např. podpora cachování pro mobilní vývoj) a zároveň přijde o jeden z významných benefitů využívání BaaS (viz kapitola 1.4).

2.3.7 Dle způsobu vzniku

Na trhu lze rozlišit dvě hlavní kategorie BaaS poskytovatelů dle způsobu vzniku. První kategorií jsou BaaS řešení, která začala jako startup (popř. je za startup lze stále označit). Druhou kategorií jsou BaaS řešení, která vznikla u velkých IT firem, které disponují cloudovou platformou[11].

[10] V tomto kontextu platformou míním např. Android, iOS nebo web

[11] Cloudovými platformami míním Microsoft Azure, IBM Bluemix, Oracle Cloud, Amazon Web Services a další.

BaaS řešení ze „startup kategorie“ zpravidla také využívají některou z těchto cloudových platforem pro hostování svého řešení, ale uživatel je od tohoto faktu abstrahován a BaaS řešení se mu jeví jako jednotný celek. Oproti tomu proces vzniku BaaS řešení velkých IT firem, která disponují cloudovými platformi, je obrácený. Tyto firmy využily postavení na trhu - tedy situace, že disponují cloudovou platformou a hotovými PaaS / SaaS produkty a pokusily se eliminovat problém, který se stal původně motivací pro vznik BaaS - tedy že využití těchto PaaS / SaaS řešení je pro vývojáře mobilních či webových aplikací příliš obtížné [14]. Tento problém řeší zpravidla vytvořením zvláštní webové konzole pro BaaS[12], kde je snaha uživatele abstrahovat od standardního „product-oriented“ přístupu, který je pro tyto cloudové platformy typický a místo toho uživateli prezentovat a nabízet funkcionalitu[13].

Z mého pozorování plyne, že i přes výše popsanou snahu, je použití BaaS velkých firem pro uživatele náročnější, protože nejsou zcela abstrahováni od jednotlivých produktů a je třeba vynaložit více úsilí na orientaci v problematice a integraci jednotlivých produktů. Na druhou stranu má uživatel detailní informace o produktech, které jsou na pozadí používány a „zázemí“ stabilní firmy na trhu.

2.4 Další funkcionalita nabízená BaaS

Tato kapitola navazuje na kapitolu 1.3 a popisuje jakou další funkcionalitu BaaS poskytovatelé nabízejí. Na rozdíl od funkcionality popsané v kapitole 1.3, funkcionalita popsaná v této kapitole je již více specifická a jedná se o jakousi nadstavbu, kterou nabízí jen někteří BaaS poskytovatelé.

2.4.1 Realtime synchronizace dat

Jiné používané názvy: Live queries, Realtime

Pod pojmem Realtime synchronizace dat je míněna synchronizace dat mezi klienty[14] a databází na serveru a to takovým způsobem, že každý klient obdrží okamžitě informace o změnách provedených v databázi. Vývojář je tak ušetřen nutnosti manuálně řešit synchronizaci dat mezi klientem a serverem a zároveň je tato automatizovaná synchronizace zpravidla rychlá, neboť komunikace probíhá prostřednictvím Websockets.

RT synchronizaci dat lze dobře vysvětlit na jednoduchém příkladu. Na ukázce zdrojového kódu 2.1 je vyobrazena naivní implementace okamžitého načítání zpráv z databáze bez podpory RT synchronizace dat ze strany BaaS

[12] např. v případe Amazon tzv. Amazon Mobile Hub

[13] Demonstrujme tuto myšlenku na konkrétním příkladu. Zatímco ve standardní webové konzoli AWS si uživatel objednává Amazon S3. V AWS Mobile Hub si uživatel aktivuje File management.

[14] Klientem je v tomto kontextu míněna instance mobilní aplikace nebo webového prohlížeče

řešení. „Okamžitosti“ je zde docíleno pravidelným dotazováním na data na serveru v intervalu 500 ms, tedy pomocí tzv. pull strategie, přičemž pokaždé jsou vždy načtena a zobrazena všechna data z databáze (problém by šel optimalizovat dotazováním pouze na data novější, než byl čas posledního dotaz). Oproti tomu na ukázce zdrojového kódu 2.2 je řešen stejný problém za podpory RT synchronizace dat ze strany BaaS řešení. „Okamžitosti“ je zde docíleno definicí posluchače události.

```
setInterval(function() {
  var messages = Backendless.Persistence.of(Message).find().data;
  clearMessages();
  for (var i = 0, i < messages.length; i++) {
      var message = messages[i];
      displayChatMessage(message.name, message.text);
  }
}, 500);
```

Výpis zdrojového kódu 2.1: Výpis zpráv bez použití RT synchronizace dat (Backendless)

```
myFirebaseRef.on('child_added', function(snapshot) {
  var message = snapshot.val();
  displayChatMessage(message['name'], message['text']);
});
```

Výpis zdrojového kódu 2.2: Výpis zpráv s použitím RT synchronizace dat (Firebase)

2.4.2 Geolokace

Jiné používané názvy: Geolocation, Location, Geofire

Funkcionalita geolokace umožňuje spravovat (tedy provádět CRUD operace) nad GPS souřadnicemi a následně provádět vyhledávací dotazy nad těmito daty. Obvyklé podporované vyhledávací dotazy jsou

- **Vyhledávání na základě poloměru**, kde vstupem je GPS souřadnice a poloměr vzdálenosti, výstupem jsou všechny záznamy nacházející se v definované kružnicové oblasti
- **Vyhledávání v obdélníkové oblasti**, kde vstupem je dvojice souřadnic definující levý horní a pravý dolní roh oblasi a výstupem jsou všechny záznamy v definované obdélníkové oblasti.

Typickým scénářem využití funkcionality geolokace může být např. nalezení restaurací v okolí uživatele aplikace.

2.4.3 Integrace externích datových zdrojů

Externím datovým zdrojem míním jiný datový zdroj, než je standardní datové úložiště poskytované BaaS službou (viz kapitola 1.3.1). Takovým externím datovým zdrojem může být např. databáze, API nebo aplikace.

Cílem této funkcionality je poskytnout jednotný dotazovací jazyk (tedy stejný jako pro standardní datové úložiště BaaS) pro tyto externí zdroje. To je realizováno nastavením přístupu k vybranému externímu zdroji ve webové konzoli poskytovatele BaaS.

Z popisu plyne, že poskytovatel BaaS musí připravit tuto integraci pro každý typ externího zdroje. Možnosti, jaké externí zdroje lze integrovat, se tedy u každého BaaS poskytovatele liší.

Někteří BaaS poskytovatelé umožňují integrovat pouze stejný typ databáze, jako používají pro své standardní datové úložiště. Poskytovatelé zaměřující se na enterprise trh umožňují často integrovat některé z typických enterprise produktů jako je např. Microsoft Dynamics nebo SAP.

Někdy bývá také dostupná možnost integrovat libovolný datový zdroj implementací rozhraní pro tento zdroj. Podoba toho rozhraní je definována BaaS poskytovatelem.

2.4.4 File & Media management

File management znamená poskytnutí úložiště pro soubory a API pro práci se soubory (zejména upload a download).

Některé BaaS podporují také streamování videa či audia a broadcasting. Tato funkcionalita bývá v dokumentacích některých BaaS uváděna zvlášť jako media management.

2.4.5 Caching

Možnosti cachování se u BaaS poskytovatelů objevují ve dvou podobách.

První možností je poskytnutí API k rychlé key-value in-memory databázi na straně serveru. Tuto možnost zpravidla nabízejí BaaS poskytovatelé, jejichž standardní datové úložiště je postaveno na relační databázi.

Druhou možností je cachování na straně klienta (používané zejména pro výsledky databázových dotazů), které je v různé míře zautomatizováno SDK od BaaS (uživatel BaaS je od problematiky částečně abstrahován).

2.4.6 Offline & Client-Server Synchronization

Tato funkcionalita umožňuje ukládat dočasně data na klientském zařízení (tzv. offline saving) po dobu, kdy zařízení není připojeno k serveru a následně data synchronizovat se serverem, když dojde k obnově připojení. Vývojář je od této problematiky z velké části abstrahován (pokud chce být) a SDK BaaS

poskytovatele řeší rozhodnutí kam data uložit (zda na klienta nebo na server) na pozadí.

Některá BaaS řešení v rámci této funkcionality poskytují vývojářům další funkce:

- Možnost implementovat vlastní pravidla pro řešení konfliktních situací.
- Možnost spustit databázovou operaci po odpojení klienta ze serveru či připojení klienta k serveru. Takovou funkcionalitu lze např. použít pro indikaci offline / online stavu uživatele.

2.4.6.1 Web hosting

Web hostingem je míněna možnost hostovat webovou aplikaci v rámci služeb BaaS poskytovatele. V případě vývoje webové aplikace uživateli BaaS tak odpadá starost se zajišťováním dalšího dodavatele pro tuto službu.

Web hosting je zpravidla omezen pouze na tzv. statické soubory.

2.4.7 Analytika

Analytika umožňuje:

1. Definovat metriky, které budou sledovány. Ty se zpravidla definují ve webové konzoli.
2. Prostřednictvím daného API sbírat data k definovaným metrikám.
3. Zpracovat nashromážděná data. Zpracování dat řeší BaaS zpravidla bez povědomí uživatele BaaS periodicky na pozadí.
4. Zobrazit zpracovaná data. Uživateli je prostřednictvím webové konzole umožněno prohlížet zpracovaná data v podobě tabulek, grafů a dalších vizualizačních nástrojů. Bývá dostupná také funkcionalita pro export získaných dat.

2.4.8 Další méně obvyklá funkcionalita

V této sekci popisuji funkcionalitu, kterou z analyzovaných BaaS poskytovatelů v době psaní této práce nabízel pouze jeden poskytovatel. Jedná se tedy o funkcionalitu, která určitým způsobem odlišuje daného BaaS poskytovatele od konkurence.

2.4.8.1 A/B testování

A/B testování je ve své podstatě podobné Analytice (viz kapitola 2.4.7). Umožňuje:

1. Definovat experiment s dvěma variantami. Pro každou variantu je pak třeba definovat událost, kdy je varianta zobrazena a kdy dojde ke konverzi. Tento krok se provádí ve webové konzoli.

2. Integrovat experiment do aplikace. Integrace se provádí prostřednictvím SDK BaaS dodavatele přímo v kódu aplikace.

3. Spuštění A/B testu a sledování výsledků. Prostřednictvím webové konzole lze test spustit, ukončit a vyhodnocovat výsledky. O rovnoměrnou distribuci A / B variant mezi uživatele se automaticky stará logika BaaS na pozadí.

   Tuto funkcionalitu v době psané této práce nabízel BaaS Kii.

2.4.8.2 Šifrování

Umožňuje šifrovat data, která se nachází v klientském zařízení. Typicky se tedy jedná o soubory, interní datové úložiště a přihlašovací údaje

Šifrování umožňoval v době psaní této práce BaaS Kinvey.

2.5 Alternativy a doplňující služby k BaaS

MEAP jako produkt BaaS podobný, který svojí funkcionalitou zpravidla přesahuje možnosti BaaS, byl popsán v kapitole 2.2. Účelem této kapitoly je poskytnout základní přehled o dalších typech cloudových služeb, které může být vhodné použít např. v následujících situacích:

1. Pro projekt bylo vybráno vhodné BaaS řešení, kterému ovšem chybí určitá funkcionalita.

2. Zvolené BaaS řešení je vhodné, ale určitou funkcionalitu nepodporuje v dostatečném rozsahu.

3. Dospělo se ke zjištění, že z rozsahu nabízené funkcionality BaaS řešení by byla pro projekt použita jen úzká podmnožina, je proto vhodné zvolit jiné úzce specializované řešení.

2.5.1 Database as a Service

Database as a Service je dnes již velmi standardizovaná služba. Tradiční velcí cloudoví poskytovatelé (IBM, Microsoft, ...) nabízejí své řešení zpravidla jak relační tak NO-SQL databáze. Existují ovšem i méně známá řešení. Např. Flybase[15] je služba velmi podobná Firebase, která se ovšem soustředí pouze na poskytování Real-time databáze bez další podpůrné funkcionality.

[15] `http://flybase.io`

2.5.2 Push notification services

Na trhu existuje celá řada cloudových služeb specializujících se na problematiku zasílání tzv. push notifikací[16] a s tím souvisejících procesů (např. vyhodnocování dopadů).

2.5.3 Velké cloudové platformy

V kapitole 2.3.7 byl popsán přístup velkých cloudových platforem k poskytování BaaS. V textu je vysvětleno, že BaaS v těchto případech je spíše „balíček“ několika produktů. Nabízí se tedy inspirovat v těchto nabídkách a využít nakonec jen některý z těchto produktů.

2.5.4 Authentication as a Service

Správa uživatelů poskytovaná BaaS může být nahrazena některým z poskytovatelů tzv. Auth as a Service [17].

[16] např. `http://carnival.io/` nebo `http://www.accengage.com/`
[17] např. `https://authrocket.com`

Kapitola 3

Technická analýza vybraných řešení

Cílem této kapitoly je provést detailní technické srovnání vybraných služeb v definované podmnožině BaaS řešení dostupných na trhu, identifikovat jejich silné a slabé stránky, popsat, jak vybrané BaaS řešení ovlivňuje další aspekty aplikace (např. architekturu, uživatelské rozhraní, ...) a popsat, pro jaké typy projektu jsou daná řešení vhodná.

3.1 Metodika výběru řešení

Ve spolupráci s vedoucím práce byl stanoven cíl vybrat BaaS řešení, která mají největší šanci na úspěšné použití na projektech, které lze popsat podle následujících charakteristik:

1. Nativní aplikace pro iOS a Android nebo webová tzv. single page aplikace.
2. Aplikace jsou malého či středního rozsahu.[18]
3. Cílovou skupinou jsou zpravidla běžní uživatelé.[19]
4. Charakter a funkční požadavky na aplikaci nelze obecně specifikovat.
5. Zvláštní nároky na bezpečnost nebo vlastnictví dat nejsou kladeny.

Výběr řešení k detailní technické analýze proběhl následujícím způsobem. Na základě kategorizace BaaS řešení, která byla vydefinována v kapitole 2.3, proběhlo posouzení každého kritéria s ohledem na stanovený cíl, což je zaznamenáno v tabulce 3.1.

[18] Vývoj těchto aplikací probíha zpravidla nejdéle v rozsahu měsíců.

[19] Běžným uživatelem je míněna potenciálně jakákoliv osobu užívající chytrý telefon nebo zařízení s přístupem k internetu bez zvláštního zaškolení.

Tabulka 3.1: Posouzení kritérií při volbě BaaS řešení

Kritérium	**Zvolená kategorie**	**Zdůvodnění**
Dle oblasti zaměření	Univerzální	S ohledem na charakteristiku č. 4 bylo rozhodnuto zaměřit se na univerzální řešení.
Dle způsobu hostování	Bez preference	S ohledem na charakteristiku č. 5 není toto kritérium rozhodující.
Dle režimu účtování	Bez preference	Ekonomické analýze řešení se věnuje zvláštní kapitola.
Dle cílového segmentu	Consumer-oriented, Consumer / Enterprise oriented	S ohledem na charakteristiku č. 3 se nebudeme zaměřovat na BaaS řešení, která cílí na enterprise aplikace.
Dle nabízených SDK	Android, iOS, JavaScript	S ohledem na charakteristiku č. 1 se zaměříme na BaaS poskytující SDK pro všechny tři vydefinované platformy.
Dle funkcionality	Bez preference	S ohledem na charakteristiku č. 4 a množinu funkcionality je toto kritérium bez přímé preference. Jako relevantní funkcionalita byly identifikovány: Datové úložiště, Správa uživatelů, Messaging, Server code, RT synchronizace dat, Geolokace, File management, Caching, Offline & Sync. Detailní analýza použitelnosti této funkcionality je předmětem této kapitoly.
Dle způsobu vzniku	startup	S ohledem na charateristiku č. 2 preferujeme BaaS řešení jednoduché na použití (viz kapitola 2.3.7) se strmou křivkou učení

Tabulka 3.2: Analýza popularity BaaS řešení ke dni 18. 3. 2016

	Stackoverflow		Quora		Vlastní platforma	
BaaS	Dotazů celkem	Dotazů poslední týden	Dotazů	Počet sledujících	Dotazů celkem	Skóre
Parse	18133	98	750	3900	0	22881
Firebase	5827	117	205	1600	0	7749
Kinvey	118	4	13	827	1420	2382
Backendless	62	7	5	11	861	946
BaaSBox	30	0	0	0	470	500
Apache Usergrid	85	0	3	13	0	101
SynergyKit	0	0	0	0	0	0

Takto vydefinovaným kritériím z analyzovaných BaaS řešení vyhovělo sedm kandidátů.

Provést detailní technickou analýzu všech sedmi řešení by bylo nad možnosti rozsahu této práce. Z tohoto důvodu bylo rozhodnuto o vybrání tří „nejpopulárnějších“ BaaS řešení z těchto kandidátů.

Přidání kritéria popularity je postaveno na předpokladu, že BaaS řešení, která mají nejvíce uživatelů, mají vyšší pravděpodobnost, že budou po technické stránce vyspělá a nedojde tedy ke srovnání vyspělých řešení s nevyspělými.

Zhodnocení popularity proběhlo na základě analýzy počtů dotazů týkajících se daných BaaS na široce používaných Q & A webových portálech Stackoverflow[20] a Quora[21,22] a počtu dotazů na vlastní support platformě, pokud takovou platformu daný BaaS poskytovatel používá.

Na základě těchto výstupů bylo rozhodnuto o detailní technické analýze těchto řešení:

- Firebase[23]
- Parse

[20] `http://stackoverflow.com`

[21] `http://quora.com`

[22] Záměrem bylo použít pro analýzu popularity také službu Google Trends, ovšem pro většinu BaaS poskytovatelů tato služba neposkytovala žádná data z důvodu nedostatečného objemu vyhledávání relevantních dotazů

[23] V době psaní této kapitoly byl Firebase kategorizován v oblasti zaměření jako univerzální. Změna kategorie na specializovaný proběhla na základě výstupů z této části práce, viz kapitola 3.6.5.1 .

- Kinvey

3.2 Stručná charakteristika vybraných řešení

3.2.1 Firebase

Firebase je BaaS, které vznikalo původně pod názvem Plankton a počátek jeho vzniku se datuje do roku 2011. V únoru roku 2013 přešel Firebase z beta verze do ostrého provozu. V roce 2014 byl Firebase koupen společností Google [17].

3.2.2 Parse

Parse je BaaS řešení, které začalo vznikat okolo roku 2011. V roce 2013 došlo k akvizici společností Facebook. V únoru roku 2016 Parse oznámil ukončení své služby k datu 28.1. 2017. O pár týdnů později bylo oznámeno uvolnění Parse serveru jako open-source (SDK byla vydána jako open-source již dříve).

Parse server nedisponuje veškerou funkcionalitou, kterou disponovalo původní hostované řešení. Ovšem komunita na GitHub, kde jsou zdrojové kódy zveřejněny, je velmi aktivní [18] a lze říci, že Parse server prochází bouřlivým vývojem (např. již byla implementována RT synchronizace dat, kterou původní hostovaný Parse nedisponoval).

Hodnocení Parse v rámci této práce je obtížné, protože v době psaní je situace okolo Parse serveru proměnlivá a nepřehledná. Na druhou stranu se jedná o pravděpodobně nejpoužívanější BaaS řešení na trhu a i s ohledem na další cíle práce je zajímavé sledovat jeho další vývoj. Z tohoto důvodu bylo rozhodnuto i přes tuto situaci Parse server do srovnání zařadit. Veškeré informace ohledně funkcionality Parse v této práci se tedy týkají open-source varianty v době psaní této práce, pokud není uvedeno explicitně jinak.

3.2.3 Kinvey

Kinvey BaaS bylo založeno v roce 2010 [19]. Z Vybraných BaaS řešení nabízí nejvíce funkcionality a jako jediné z vybraných řešení nabízí funkcionalitu tvořenou zejména pro podniky.

3.3 Metodika srovnání řešení

Srovnání řešení probíhalo na třech úrovních.

Zaprvé byla detailně zkoumána funkcionalita BaaS, která byla identifikována jako relevantní (viz tabulka 3.1). Toto srovnání je provedeno v kapitole 3.4.

Zadruhé, na základě zkušeností s vývojem mobilních a webových SPA aplikací na jedné straně a dosavadního zkoumání vybraných BaaS řešení na straně

druhé, byly vytipovány funkční scénáře nebo klíčové oblasti, jejichž řešení v aplikacích vydefinovaných v kapitole 3.1 může být potřebné a které naplňují jednu z následujících charakteristik:

- Jejich řešení / implementace ve vybraných BaaS řešeních by mohlo být problematické.
- Jejich řešení ze strany BaaS řešení má potenciál ulehčit práci vývojářům aplikace.

Tyto scénáře nebo klíčové oblasti byly zkoumány buď přímo v kapitole 3.4 nebo v kapitole 3.5 v případě, že svým rozsahem nebo povahou překračují hranice jednotlivých funkčních oblastí.

Funkcionalita, scénáře a klíčové oblasti byly srovnávány na základě dokumentace a implementace jednoduchých testovacích programů v případě, kdy nebylo možno učinit závěry jen na základě dokumentace.

Zatřetí byla implementována vzorová SPA webová aplikace Cost Tracker pro všechna zvolená BaaS řešení (viz kapitola 3.6). Cílem této implementace bylo otestovat funkcionalitu BaaS řešení na komplexnějším příkladu.

3.4 Srovnání nabízené funkcionality

3.4.1 Datové úložiště, dotazovací jazyk a RT synchronizace

Každý BaaS je specifický vlastnostmi databáze, kterou nabízí. Lze konstatovat, že určitý datový scénář se bude implementovat v každém BaaS jiným způsobem a v některých případech bude možných (a zároveň správných) přístupů více. V přímé souvislosti s databází je dotazovací jazyk poskytovaný SDK, jehož možnosti a „síla“ jsou do velké míry určeny vlastnostmi databáze.

3.4.1.1 Firebase

Data ve Firebase jsou ukládána ve stromové struktuře jako JSON objekty (JSON pole nejsou povolena[24]). Celou databázi lze tedy reprezentovat jako jeden JSON dokument.

Komunikace s databází je založena na RT synchronizaci dat (viz 2.4.1). Ta je funkčně postavena na dvou principech.

1. **Databázová reference**, která se skládá vždy z odkazu na určitý podstrom databáze. Tento podstrom je jednoznačně reprezentovaný URL.
2. **Databázový posluchač**, který:
    - Poslouchá na určité databázové referenci.

[24]V důsledku jsou pole implementována jako JSON objekt s číselnými klíči 0, 1, ...

- Poslouchá určitou událost. Událostí je několik typů. Pro demonstraci uveďme např. událost „child_added“, která je vyvolána vždy, když je přidán další JSON objekt do podstromu.

Databázové referenci lze dále volitelně předat informaci o tom, v jakém pořadí budou dostávány výsledky z databáze (lze řadit dle klíče, dle hodnoty, či dle priority). Pokud určíme pořadí, lze ještě filtrovat výsledky pomocí několika metod. Metody limitToLast a limitToFirst umožňují omezit počet výsledků na pevně stanovený počet. Metody startAt, endAt a equalTo umožňují dále vyfiltrovat výsledky. Filtrovací metoda filtruje dle aspektu, podle kterého výsledky řadíme.

Dotazovací možnosti tedy nejsou u Firebase příliš široké, lze se domnívat, že se jedná o určitý „tradeoff“ za real-time funkcionalitu.

Firebase nechává uživatelům velkou volnost v oblasti jak navrhnout strukturu databáze. Správnou strukturu je třeba odvozovat od způsobu, na jakém bude na data nahlíženo[25].

3.4.1.2 Parse

Ukládání dat v Parse je postaveno na objektech a principiálně je podobné ORM přístupu. Každý objekt se skládá z dvojic klíč-hodnota JSON kompatibilních dat. Objekty jsou vždy instancí určité třídy (např. osoba), což znamená že mají stejnou strukturu. Třída se definuje buď explicitně ve webové konzoli nebo implicitně při vytvoření první instance třídy z klienta.

Parse umožňuje mezi třídami definovat vazby všech typů - tedy one-to-one, one-to-many a many-to-many.

Dotazovací jazyk Parse je podobný dotazovacím jazykům používaných u ORM a je poměrně silný. Obsahuje mnoho operátorů a umožňuje také dotazování skrz vazby (ovšem pouze v tom směru, ve kterém byly definovány - neumožňuje tedy tzv. reverse relational lookup). Z agregačních dotazů je podporována pouze operace count.

V březnu 2016 Parse uvedl také funkcionalitu RT synchronizace dat pod názvem Live Queries, která je v době psaní práce podporována v JavaScript SDK a ve zvláštním SDK pro iOS. RT synchronizace v Parse funguje následovně:

- Na straně serveru je třeba explicitně uvést pro které databázové kolekce je RT synchronizace povolena.

- Objekt Query je rozšířen o metodu subscribe, která vrací tzv.subscription. Subscription umožňuje poslouchat události create, enter, update, leave a delete. Filtrace, které lze na Query provést jsou pravděpodobně určitým

[25] „Best practices“ přímo od vývojářů jsou dostupné na `https://www.firebase.com/docs/web/guide/structuring-data.html`

způsobem omezené, v tuto chvíli tato problematika není zcela zdokumentovaná (bylo vyzkoušeno, že např. nelze query filtrovat na základě vazeb na jiné kolekce).

3.4.1.3 Kinvey

Základní datovou jednotkou ukládanou do databáze je entita, což je množina klíč-hodnota JSON kompatibilních dvojic. Entity jsou organizovány do kolekcí. Názvosloví se liší, ale princip je velmi podobný Parse.

Mezi enitami kolekcí lze definovat vazbu typu one-to-many.

Dotazovací jazyk je syntakticky i principiálně podobný dotazovacímu jazyku Parse. Ve srovnání s Parse je silnější v agregačních dotazech, ale slabší v dotazech skrz vazby - u dotazů skrz vazby umožňuje dotaz pouze na základě ID vázaného objektu.

3.4.1.4 Shrnutí

Porovnávat v této oblasti Firebase s Kinvey a Parse je bezpředmětné, protože se jedná o dva odlišné přístupy - RT synchronizaci a dotazování. Možnosti Kinvey a Parse jsou velmi podobné. Parse lépe pracuje se vztahy, Kinvey disponuje silnějšími operátory pro agregační dotazy.

3.4.2 Správa uživatelů & Social Auth

Na základě popisu této problematiky v kapitole 1.3.2 je u jednotlivých služeb zkoumána podpora zmíněných procesů a dále podpora přihlašování přes sociální sítě.

Testování této funkcionality probíhalo na webové aplikaci.

3.4.2.1 Možnosti Firebase

Firebase ukládá autentizační informace o uživatelích mimo databázové úložiště, se kterým pracuje uživatel BaaS. Z tohoto důvodu je zpravidla třeba na registrační událost navázat vytvoření databázového záznamu, ke kterému jsou poté ukládána všechna data pro daného uživatele.

- Registrace - Firebase nativně umožňuje registraci uživatele prostřednictvím e-mailu (lze použít e-mail) a hesla. Proces ověření pravosti e-mailu není podporován. Unikátní možnost je tzv. anonymní autentizace, kdy je uživatel vytvořen dočasný účet bez nutnosti zadat z jeho strany jakékoliv údaje.
- Přihlášení - Podporováno.
- Změna hesla - Funkce pro změnu hesla je v SDK implementována.

- Obnova zapomenutého hesla - SDK implementuje metodu resetPassword, která uživateli na e-mail zašle dočasné heslo, kterým se může přihlásit. Dále je předpokládáno použití formuláře pro změnu hesla, který implementuje vývojář sám.

Firebase podporuje autentizaci prostřednictvím těchto třetích providerů: Facebook, Twitter, Github a Google. Z pohledu uživatele BaaS je tato funkcionalita řešena velmi dobře a konzistentně - implementace podpory pro jednotlivé providery se neliší. Lze říci, že základní zprovoznění některé z těchto autentizací je otázka několika minut.

3.4.2.2 Možnosti Parse

Parse ukládá uživatele jako standardní databázovou kolekci. Rozšíření o případná další data o uživatelích je tedy jednoduché.

- Registrace - Parse umožňuje registraci uživatele prostřednictvím uživatelského jména (lze použít e-mail) a hesla. Proces ověření pravosti e-mailu není podporován.
- Přihlášení - Podporováno.
- Změna hesla - Pro změnu hesla není v SDK speciální funkce, ale lze ji provést standardní úpravou entity Usera.
- Obnova zapomenutého hesla - Není nativně podporována. Odpovídající funkce se v SDK nalézá, ale v open-source variantě v době psaní této práce nefunguje (mimojiné z důvodu, že Parse Server neumí nativně posílat e-maily).

V oblasti Social Auth Parse podporuje autentizaci pouze přes Facebook, ta je z pohledu uživatele BaaS řešena efektivně a její zprovoznění je otázka několika minut.

3.4.2.3 Možnosti Kinvey

Kinvey obdobně jako Parse ukládá uživatele jako standardní databázovou kolekci.

- Registrace - Kinvey standardně nabízí dvě varianty registrace prostřednictvím uživatelského jména (může být použit e-mail) a hesla. K dispozici jsou dvě varianty - registrace bez nutnosti ověření e-mailu a registrace s nutností ověřit e-mail předtím, než bude uživateli dovoleno se přihlásit.[26]. Ve webové konzoli lze nastavit textaci e-mailů spojených se

[26]Dodejme, že tato varianta se v době testování nechovala zcela korektně, neboť se autentizační mechanismus dostal do určitého nekonzistentního stavu, ve kterém Kinvey evidentně interně evidovalo uživatele jako přihlášeného, ale zároveň metoda getActiveUser padala na chybu 401 - unauthorized.

správou uživatelů i podobu webové stránky, která se zobrazí po ověření platného e-mailu.

- Přihlášení - Podporováno.
- Změna hesla - Pro změnu hesla není v SDK speciální funkce, ale lze ji provést standardní úpravou entity Usera.
- Zapomenuté heslo - Kinvey implementuje funkci v SDK, která uživateli zašle e-mail s odkazem na zvláštní stránku, kde může nastavit nové heslo.
- Zapomenuté uživatelské jméno - V případě, že se uživatele přihlašují uživatelským jménem, ale zároveň je k dispozici jejich e-mail, umožňuje Kinvey zaslání uživatelského jména na e-mail.

Kinvey podporuje autentizaci prostřednictvím těchto třetích providerů: Facebook, Google, LinkedIn a Twitter.

Z pohledu uživatele BaaS je tato funkcionalita řešena poměrně komplikovaně. Implementace jednotlivých providerů se liší, samotná konfigurace této funkcionality vyžaduje kroky (např. založení databázové kolekce), které by mohly být ze strany Kinvey zautomatizovány.

3.4.2.4 Shrnutí

V oblasti standardní registrace (uživ. jméno / e-mail a heslo) vychází v době psaní této práce nejhůře Parse, který trpí nedostatky v důsledku přechodu na open-source. Nejvíce možností nabízí Kinvey, což je vykoupeno ne zcela jasnou dokumentací pro tuto problematiku.

V oblasti Social auth vychází nejlépe Firebase, který podporuje nejvíce providerů a zároveň komfortním způsobem. Kinvey trpí složitou konfigurací propojení s jednotlivými providery. Parse podporuje pouze Facebook.

3.4.3 Offline & Client-Server sync

Zejména u mobilních aplikací je třeba řešit chování aplikace v případě, kdy zrovna zařízení není připojeno k internetu. Pro názornost si představme například aplikaci úkolníčku, kdy se uživatel nachází v situaci vytvoření nového úkolu nebo upravuje již existující úkol ve chvíli, kdy není připojen k internetu.

V případě vlastního backendu by se tato situace pravděpodobně musela řešit implementací vlastní logiky na klientovi, která by detekovala stav připojení k internetu a na tomto základě odesílala data buď okamžitě na server nebo implementovala určitou frontu požadavků, které budou vykonány ve chvíli, kdy dojde k obnově připojení.

Implementace této vlastní logiky není triviální, přitom lze tento problém částečně zobecnit a BaaS poskytovatel prostřednictvím SDK může vývojáře tak od této problematiky určitým způsobem abstrahovat.

3.4.3.1 Možnosti ve Firebase

Obecně lze konstatovat, že Firebase poskytuje vysokoúrovňovou abstrakci v této problematice. Dle dokumentace platí, že každý klient si udržuje interní verzi serverové databáze.

V případě mobilních klientů (iOS, Android) Firebase SDK v offline stavu řeší jak dotazování na databázi (data jsou načtena z lokální verze databáze), tak zápis či úpravu dat (změny jsou uloženy do lokální databáze a později propagovány na server). V případě webového klienta Firebase SDK řeší jen zápis a úpravu dat.

Na mobilních zařízeních lze zapnout „Disk persistence“, což znamená, že interní verze databáze je zapisována na disk, tudíž jsou lokální data zachována i po restartu aplikace.

Dále Firebase SDK disponuje funkcionalitou pro zjištění aktuálního stavu (zda je zařízení offline či online) a umožňuje provést určitou databázovou operaci na serveru při změně tohoto stavu (tato funkcionalita může sloužit např. pro informování ostatních uživatelů o tom, zda je daný uživatel offline či online).

Firebase SDK lze manuálně přepínat do režimu offline / online.

3.4.3.2 Možnosti v Kinvey

V případě SDK pro web Kinvey umožňuje více úrovní podpory práce offline. První je definována jako „Automated control“, kde lze chování popsat následovně:

- Při každém dotazu na data na serveru jsou tyto data také uložena do cache. Dotazování na cache je poté aplikováno v případě, kdy je aplikace offline.
- Při ukládání či úpravě dat jsou data automaticky uložena na server v případě, že je aplikace online. V opačném případě jsou uložena lokálně a synchronizována automaticky ve chvíli, kdy aplikace přejde do stavu online.

Druhou možností je pracovat na nižší úrovni a pro každý dotaz na databázi definovat, zda se má primárně dotazovat na server či na lokální úložiště. U operace ukládání pak lze obdobně vynutit ukládání do lokálního úložiště. Synchronizační proces pak musí být spuštěn manuálně voláním příslušné funkce.

V případě synchronizačních konfliktů lze nastavit jednu z předpřipravených „politik“ řešení - client always wins nebo server always wins, popřípadě je možné definovat politiku vlastní.

V případě SDK pro Android má uživatel možnost nastavit chování pro dvě související problematiky:

- Caching policy - Ta určuje, jakým způsobem se mají cachovat výsledky dotazů do serverové databáze a nastavuje se na úrovni jednotlivých dotazů.

- Offline policy - Ta určuje, v jakém pořadí se mají provádět dotazy do lokální databáze nebo serverové databáze. Offline policy předpokládá zapnutí služby, která se stará o synchronizaci serverové a lokální databáze (pro lokální databázi je použita Sqlite).

V případě SDK pro iOS jsou možnosti principiálně podobné jako pro Android, ale API se liší.

3.4.3.3 Možnosti v Parse

SDK pro web neimplementuje podporu pro tuto problematiku.

Možnosti SDK pro iOS a Android jsou v této problematice totožné.

Parse SDK implementuje lokální databázi, která disponuje stejným dotazovacím API jako databáze serverová.

Každá entita disponuje metodou save, pin, delete a unpin, přičemž save a delete slouží pro uložení resp. smazání entity na serveru a pin / unpin pro ty samé operace v lokální databázi. Obdobné operace lze aplikovat také na celou množinu entit. Operaci pin lze přiřadit zvláštní klíč, pod kterým budou ukládané entity dostupné, což lze použít jako cache.

Uživatel SDK musí tedy sám řešit problematiku kdy pracovat s lokální databází a kdy se serverovou. Jedinou výjimkou jsou metody entity saveEventually a deleteEventually. Chování saveEventually lze popsat následovně (chování deleteEventually lze odvodit.)

- Entita je nejprve uložena do lokání databáze (operace pin).

- Ve chvíli, kdy je obnoveno připojení k internetu, je entita uložena do serverové databáze (operace save).

- Entita je odstraněna z lokální databáze (operace unpin).

3.4.3.4 Shrnutí

Každý z BaaS poskytovatelů volí v této problematice odlišný přístup. Firebase je snadno pochopitelný, poměrně konzistentní napříč platformami, ale jeho chování nelze customizovat. Parse nenabízí tuto funkcionalitu pro web, jinak je dobře pochopitelný a nabízí dobrý poměr mezi abstrakcí uživatele BaaS a možností manuálně s touto funkcionalitou pracovat. Kinvey nabízí nejvíce možností jak s touto problematikou pracovat, ovšem je nekonzistentní napříč platformami a obtížně pochopitelný.

3.4.4 File management

Na projektu mobilní nebo webové aplikace je třeba často pracovat se soubory. Může se jednat např. o profilové fotky uživatelů, fotografie z mobilních zařízení či soubory PDF. Od backendu, který nám umožní komfortně tuto problematiku řešit, bychom očekávali podporu následujících scénářů:

1. Základní možnost nahrát, skladovat a opět stahovat soubory ze serveru. Možnost tyto soubory sdílet mezi uživateli.

2. Přerušovaný upload souborů (v případě nestabilního internetového připojení).

3. Vytváření náhledů z fotografií.

4. Transformace videí do různých formátů.

3.4.4.1 Implementace ve Firebase

Firebase explicitně neřeší File management. Soubory je možné ukládat do databáze po zakódování pomocí Base64[27] a to do velikosti 10MB, což je maximální velikost hodnoty ve Firebase databázi.

Firebase tedy pokrývá pouze první scénář a to do velikosti souborů 10 MB. Ostatní scénáře nejsou podporovány.

3.4.4.2 Implementace v Kinvey

Dle dokumentace Kinvey používá jako úložiště souborů Google Cloud Storage[28] a poskytuje v rámci svých SDK API pro download a upload souborů, které mimojiné řeší práva na soubory. Práva na soubor v rámci Kinvey může nabývat dvou typů:

1. Private - soubor může být stáhnut pouze uživatelem, který jej nahrál.

2. Publicly - readable - Soubor může být stažen kýmkoliv.

Ukládaný soubor může nabývat velikosti až 5 TB.

Ohledně podpory definovaných scénářů lze konstatovat následující závěry. První scénář je podporován zcela. Systém práv souborů je sice omezený a neumožňuje přímo sdílení vybraným uživatelům, ale toto sdílení lze vyřešit nastavením práva souboru na publicly-readable a zároveň volbou nepredikovatelného názvu souboru, který by pak byl předán jen vybraným uživatelům.

Druhý scénář není podporován. Google Cloud Storage sice podporuje přerušovaný upload, ovšem tato možnost není zabudována v Kinvey SDK. Lze

[27] Base64 převádí binární data do ASCII a zpět.

[28] `https://cloud.google.com/storage/`

zvažovat možnost vlastní implementace, kdy na straně klienta soubor rozdělíme na více částí, které budeme posílat vlastnímu skriptu Business Logic typu Endpoint, který dočasně jednotlivé části uloží do databáze (nebo do úložiště souborů) a na závěr je spojí v jeden celek, který nahraje do úložiště souborů. Problém tohoto řešení je náročnost na vlastní implementaci a časová omezenost běhu Business logic (5 sekund pro základní tarif, možnost navýšit až na 60 sekund).

Scénáře tři a čtyři nejsou podporovány.

Mimo rámec námi definovaných scénářů Kinvey nabízí možnost streamování videa, nastavení expirace souborů stažených do klienta, možnost navázat odkaz na soubor na databázovou entitu a na mobilních platformách také informování o pokroku nahrávání.

3.4.4.3 Implementace v Parse

File Management od Parse nabízí více „adaptérů“ pro různé úložiště souborů. Výchozí je adaptér pro MongoDB GridFS[29], dále jsou v době psaní této práce k dispozici adaptéry pro Amazon S3 a Google Cloud Storage (to předpokládá mít v u těchto cloudových služeb vytvořený účet). V rámci této práce byl testován první zmíněný adaptér pro MongoDB. Platí však, že uživatel by právě díky adaptéru měl být abstrahován od konkrétního uložiště. Maximální velikost souboru není explicitně stanovena (lze ji uměle nastavit v konfiguraci Parse Server, výchozí je 20 MB) a bude záviset pravděpodobně na konfiguraci infrastruktury popř. přímo MongoDB.

Podpora námi definovaných scénářů je podobná jako u Kinvey (první je podporován, ostatní nikoliv). Třetí a čtvrtý scénář by šlo řešit použitím dalších knihoven v rámci Cloud skriptů v případě, že máme plný přístup k serveru[30]. Mimo rámec definovaných scénářů Parse nabízí možnost navázat odkaz na soubor na databázovou entitu a na mobilních platformách také informování o pokroku nahrávání.

3.4.4.4 Shrnutí

Firebase nepodporuje práci se soubory. Možnosti Kinvey a Parse jsou podobné, avšak nedostatečné pro „běžné použití“ - chybí např. podpora práce s obrázky.

3.4.5 Messaging

Často je žádoucí uživatele aplikace informovat o nějaké události prostřednictvím e-mailu a v případě mobilních aplikací také prostřednictvím push notifikací.

[29] `https://docs.mongodb.org/manual/core/gridfs/`

[30] Pro úplnost doplňme, že hostovaný Parse nabízel v rámci Cloud Code modul pro práci s obrázky, Parse Server již nyní tuto možnost nenabízí a je doporučeno použít jiné knihovny.

3.4.5.1 Firebase

Firebase neposkytuje rozhraní pro zasílání push notifikací ani pro zasílání e-mailů (vyjma e-mailů, které jsou spojené s procesy okolo správy uživatele - tyto e-maily jsou však zasílány automaticky ze strany Firebase a uživatel BaaS může ovlivnit pouze jejich obsah ve webové konzoli).

3.4.5.2 Kinvey

Kinvey umožňuje zasílání e-mailů i push notifikací pro Android a iOS prostřednictvím modulů dostupných pro Server Scripts (moduly Push a Email).

Nezbytné přístupové údaje ke konfiguraci rozesílání push notifikací se nastavují ve webové konzoli.

Registraci zařízení k Push notifikacím lze provést buď prostřednictvím SDK pro Android a iOS nebo prostřednictvím REST API. Push notifikace lze zaslat buď vybraným uživatelům nebo všem uživatelům aplikace.

3.4.5.3 Parse

Parse Server nativně nepodporuje rozesílání e-mailů. Tuto funkcionalitu je třeba implementovat vlastním způsobem např. napojením na další službu (např. SendGrid[31]).

Push notifikace pro Android a iOS jsou podporovány. Konfigurační přístupové údaje k rozesílání push notifikací je třeba vložit přímo do spouštěcího skriptu Parse serveru.

Registrace zařízení k push notifikacím je poměrně komplexní a je popsána v dokumentaci[32].

Samotné push notifikace je možné rozesílat programově prostřednictvím modulu Server Scriptu, prostřednictvím REST API nebo ve webové konzoli. Příjemci jsou definovány buď odebíráním určitého kanálu, na který se notifikace rozesílají, nebo lze použít tzv. pokročilé cílení, kde lze pracovat i s filtrováním uživatelů dle určitých kritérií.

3.4.5.4 Shrnutí

Firebase messaging nepodporuje. Parse v době psaní této práce trpí nedostatky z důvodu přechodu na open-source (složitá konfigurace push, chybějící e-mailing). Kinvey nabízí solidní podporu této funkcionality.

[31] `http://sendgrid.com`

[32] `https://github.com/ParsePlatform/Parse-Server/wiki/Push-Configuring-Clients`

3.4.6 Geolokace

3.4.6.1 Firebase

Firebase umožňuje práci s lokalitami prostřednictvím rozšiřující knihovny GeoFire, která je dostupná pro JavaScript, Java a Objective C. GeoFire umožňuje:

- Uložit souřadnice k libovolné databázové referenci.
- Definovat dotaz na vrácení objektů, které se nachází v definovaném okruhu (ten je definován středem a poloměrem v kilometrech).
- Vytvořit posluchače na události přidání / odebrání či změny souřadnic.

3.4.6.2 Kinvey

Pro práci s geo lokacemi je třeba k databázovým entitám ukládat souřadnice do atributu _geoloc. Kinvey poté umožňuje provádět dotazy pomocí následujících operátorů:

- near - Vrací entity, které se nachází v definovaném okruhu (určený středem a poloměrem v mílích).
- withinBox - Vrací entity, které se nachází v zadaném obdélníku (definovaným dvěma body)
- withinPolygon - Vrací entity, které se nachází v mnohoúhelníku definovaném $n > 2$ body.

3.4.6.3 Parse

V případě Parse se podobně jako u Kinvey ukládají souřadnice k databázovým entitám. K dispozici jsou následující dotazovací operátory:

- near - Vrací entity nacházející se „poblíž“ zadaného bodu. Konkrétněji není funkce zdokumentována.
- withinGeoBox - Vrací entity nacházející se v obdélníku definovaným dvěma body.
- withinKilometers / withinMiles / withinRadians - vrací Entity nacházející se v definovaném okruhu (určený středem a vzdáleností v jednotkách odpovídajících názvu funkce).

3.4.6.4 Shrnutí

Možnosti všech tří zkoumaných BaaS jsou srovnatelné a pro běžné použití pravděpodobně dostatečné.

3.4.7 Server code

Firebase Server code nepodporuje. Možnosti Parse a Kinvey v této oblasti jsou podobné, ale terminologie se mírně liší. V případě Parse hovoříme o Cloud Code, v případě Kinvey o Business Logic.

Server code se píší u obou služeb v JavaScriptu. V případě Kinvey se kód píše ve webovém rozhraní. V případě Parse do souboru umístěného na serveru.

Obě služby nabízí dva typy Server code:

- Collection Hooks / Triggers - umožňují spustit skript v návaznosti na operaci uložení či smazání databázové entity. Kinvey navíc nabízí možnost spustit skript při operaci čtení.
- Endpoints / Cloud functions - umožňují vytvořit skript, který lze volat z klienta.

Kinvey umožňuje nastavit pravidelné automatizované spouštění Endpointů (tedy bez potřeby aktivovat jej voláním z klienta). U Parse je třeba problematiku automatizovaného spouštění řešit vlastní implementací.

Obě služby v rámci Server code disponují tzv. moduly, které zpřístupňují určitou funkcionalitu (např. rozesílání e-mailů, možnost provádět HTTP dotazy).

V případě Kinvey je síla Server code limitována právě moduly, které nabízí. Uživatel BaaS nemůže použít libovolnou knihovnu dostupnou pro daný jazyk pro získání určité funkcionality. V Případě Parse jsou moduly spíše pozůstatkem z doby, kdy nebyl open-source, neboť nyní lze použít libovolnou knihovnu pro JavaScript (za předpokladu, že disponujeme plným přístupem k serveru).

3.4.7.1 Shrnutí

Firebase Server code nepodporuje. Kinvey nabízí oproti Parse jednodušší konfiguraci skriptů. Výhodou Parse je možnost použít libovolné knihovny třetích, což je dáno tím, že Parse je třeba hostovat na vlastním serveru.

3.5 Řešení funkčních scénářů a klíčových oblastí

3.5.1 Implementace nestandardního registračního procesu

Předpokládejme, že v našem projektu je potřeba implementovat registrační proces, který není nativně BaaS podporován. Demonstrujme možnosti jednotlivých BaaS v této problematice na pokusu implementovat registrační proces, který lze popsat následujícími po sobě jdoucími kroky:

1. Uživatel se registruje do aplikace zadáním svého e-mailu a hesla.
2. Uživateli je odeslán e-mail pro ověření pravosti e-mailové adresy.

3. Vstup do aplikace je uživateli umožněn okamžitě bez čekání na ověření e-mailové adresy.

4. Uživatelský účet je však zablokován, pokud e-mailová adresa není ověřena do x dní po provedení registrace.

3.5.1.1 Implementace v Kinvey

Implementaci procesu tak, jak je navržen, by mohla být provedena následujícím způsobem:

1. Vytvořením skriptu byznys logiky typu endpoint pro odeslání verifikačního e-mailu, hrubá kostra tohoto skriptu se nachází na obrázku 3.1, tento skript bude na klientovi[33] volán po registraci uživatele.

2. Vytvořením malé webové aplikace, která se bude nacházet na URL, která je obsažena ve verifikačním e-mailu. Tato webová aplikace nastaví příznak uživatele verified. Tuto aplikaci by bylo třeba hostovat mimo BaaS řešení, neboť Kinvey hosting webových aplikací nenabízí. Uživatel by se musel v této webové aplikaci přihlásit, aby bylo možné informaci o ověření e-mailu uložit nebo bychom mohli pro zaznamenání informace do databáze použit tzv. master key, pak by tato aplikace ale musela mít také serverovou část, neboť master key nelze z důvodu bezpečnosti vystavit na klientovi. Pokud bychom přistoupili na variantu, že lze tuto informaci zaznamenat bez autentizace uživatele, pak vytváříme bezpečnostní díru v aplikaci. Pro úplnost dodejme, že alternativním řešením, které se nabízí, je vytvoření dalšího endpointu. Toto alternativní řešení však není možné, neboť veškeré endpointy implementované v Kinvey jsou dostupné jen pod HTTP metodou POST.

3. Na klientovi je třeba po autentizaci uživatele zkontrolovat, zda je e-mail ověřený a pokud ne (a zároveň uběhla doba po kterou je možné uživatelský účet použít bez ověření), přístup do aplikace zablokovat.

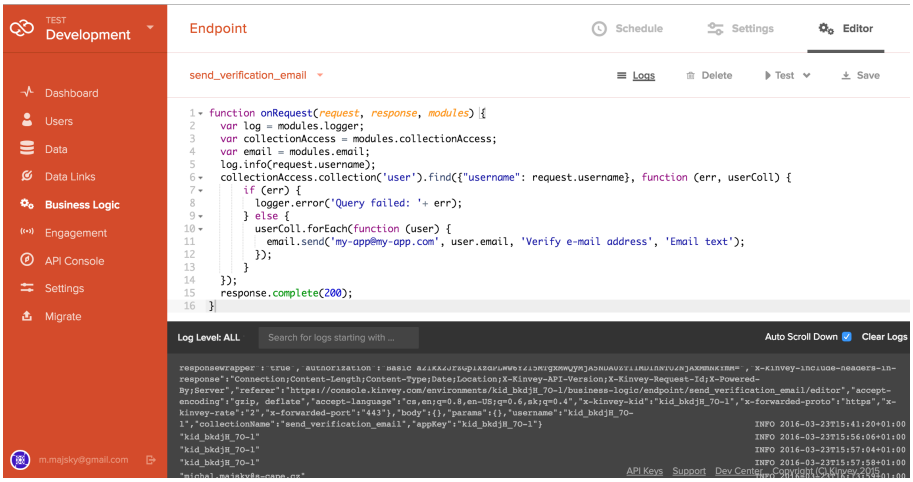
Z důvodu popsaných v druhém bodu lze konstatovat, že implementace toho scénáře v Kinvey je proveditelná, ale velice problematická.

3.5.1.2 Implementace ve Firebase

Vzhledem k tomu, že Firebase nenabízí funkcionalitu Server Code (viz kapitola 1.3.4), proces ověřování e-mailu včetně námi definovaného by bylo třeba implementovat prostřednictvím vlastního řešení - serveru, který:

- Obslouží zaslání verifikačního e-mailu.

[33] tedy webové nebo mobilní aplikaci, kterou implementuje uživatel BaaS



Obrázek 3.1: Kinvey: skript pro odeslání e-mailu

- Vystaví URL dostupnou z prohlížeče - verifikační odkaz .
- Komunikuje s Firebase prostřednictvím REST API a zapisuje údaje o ověření uživatele.

Lze tedy konstatovat, že Firebase tento scénář zcela nepodporuje.

3.5.1.3 Implementace v Parse

Vzhledem ke skutečnostem, že předpokládáme hostování Parse na vlastním serveru, kde lze plně přistoupit ke zdrojovým kódům, a vzhledem k faktu, že Parse je postaven nad frameworkem Express[34], jeví se jako pravděpodobně nejjednodušší řešení doimplementovat potřebnou funkcionalitu (tak, jak je popsána v předchozí kapitole 3.5.1.2) nad tímto frameworkem.

3.5.1.4 Shrnutí

Byť možnosti implementace u jednotlivých BaaS se mírně liší, lze konstatovat, že vyřešení tohoto scénáře je ve všech případech srovnatelně obtížné a problematické.

3.5.2 Přesun logiky aplikace z klienta na server

V případě vývoje aplikace, která má více klientů (např. pro iOS, Android a webový klient), je zpravidla žádoucí maximum z logiky aplikace přesunout na server z následujících důvodů:

[34] `http://expressjs.com/`

- V případě, že danou logiku implementujeme na klientovi, musíme ji implementovat v každém klientovi - vzniká tak duplikace kódu, hrozba nekonzistence a odlišného chování napříč klienty.
- V případě chyby v této logice je jednodušší tuto chybu opravit na serveru, než na klientech (klientů je více, v případě mobilních aplikací je třeba zveřejnit novou verzi aplikace na distribučních serverech (např. Google play).

Vhodným příkladem takovéto logiky může být libovolný složitější výpočet.

3.5.2.1 Implementace v Kinvey

V Kinvey je řešení této úlohy obecně realizovatelné prostřednictvím funkcionality Custom Endpoint v Business logic (což je jiný název pro Server Code).

3.5.2.2 Implementace ve Firebase

Firebase nenabízí funkcionalitu Server Code, tudíž jakákoliv logika musí být implementována na klientovi nebo musí být vytvořen vlastní server.

3.5.2.3 Implementace v Parse

V případě Parse je řešení této úlohy podobné jako u Kinvey - lze použít Cloud code (což je jiný název pro Server code) typu function.

3.5.2.4 Shrnutí

Firebase neumožňuje přesunout logiku aplikace na server. V případě Kinvey a Parse lze pro řešení této problematiky použít Server code.

3.5.3 Přístupová práva k datům

Nejprve popišme kontext problematiky. Přístupové údaje, kterými se v aplikaci inicializuje SDK poskytované BaaS, jsou přístupné a tedy potenciálně zneužitelné (v případě webové SPA aplikace jsou jednoduše čitelná ze zdrojových kódu, u mobilní aplikace by bylo třeba aplikaci dekompilovat). Z tohoto důvodu je třeba řešit explicitně zabezpečení dat na straně serveru a nespoléhat na „filtrování dat“ implementované logikou aplikace.

U BaaS se obáváme, že řešení přístupových práv nebude dostatečně jemnozrnné pro komplexnější scénáře. Otestujeme možnosti BaaS na následujícím příkladu: V databázi evidujeme uživatele a každý uživatel má přiřazené svá auta. Všichni uživatelé mohou vidět kolik mají ostatní uživatelé aut, ale konkrétní výpis aut mohou vidět jen přátelé uživatele.

Vzorový příklad lze rozložit na dva podproblémy:

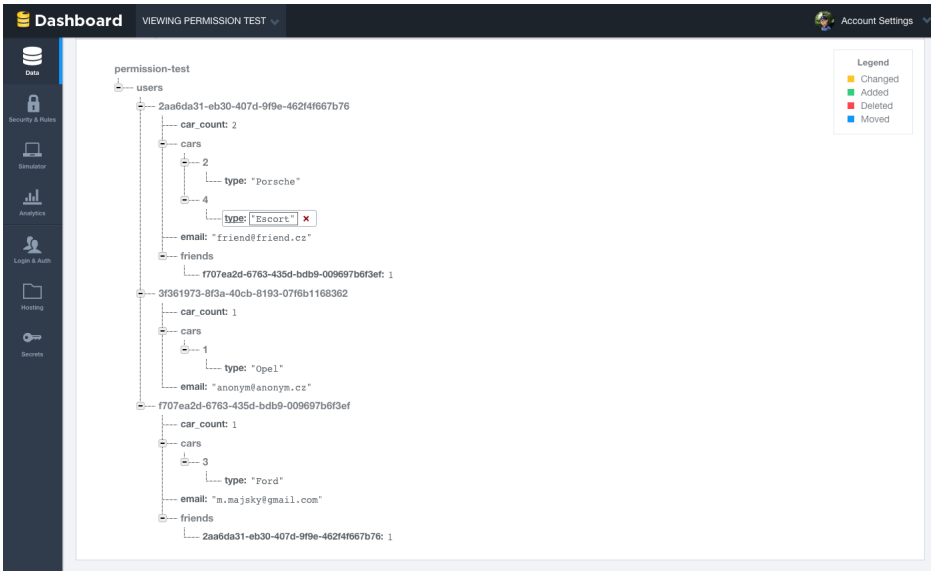
1. Povolit zobrazení seznamu aut jen pro přátele uživatele.

2. Povolit zobrazení počtu aut pro všechny uživatele.

3.5.3.1 Implementace ve Firebase

Konfigurace zabezpečení dat se realizuje ve Firebase „centrálně“ prostřednictvím webové konzole v jazyce JSON. Ve výchozím nastavením může „kdokoliv číst a zapisovat cokoliv“. Nastavení přístupových práv je tedy třeba se před spuštěním aplikace věnovat explicitně. S ohledem na přístup k datům je třeba také navrhovat strukturu databáze, v opačném případě se může stát, že z bezpečnostních důvodů bude třeba strukturu databáze předělat (viz příklad, který následuje).

Implementace vzorového problému z datového pohledu je vyobrazena na obrázku 3.2.



Obrázek 3.2: Firebase: Struktura databáze pro vzorový příklad

První podproblém je ve Firebase řešitelný. Druhý podproblém nelze řešit přímo již z toho důvodu, že dotazovací jazyk Firebase neumožňuje dotázat se pouze na počet určitých položek. Alternativním řešením je tedy počet aut ukládat do databáze explicitně (viz klíč cars_count na obrázku 3.2). Vzhledem k tomu, že Firebase neumožňuje žádnou formu triggeru, je třeba se o aktuálnost hodnoty v cars_count starat na klientovi.

Implementace výsledných oprávnění je na ukázce zdrojového kódu 3.1.

3.5.3.2 Implementace v Kinvey a Parse

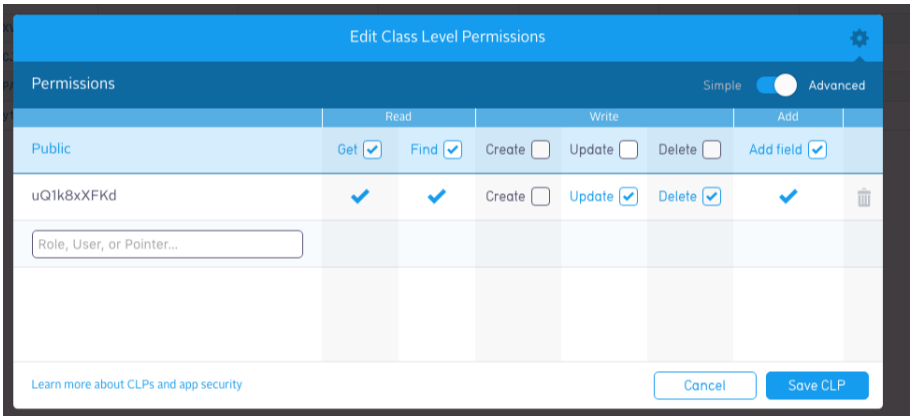
Kinvey i Parse disponují dvěma úrovněmi nastavení přístupu k datům.

```
{
    "rules": {
        "users": {
          "$user_id": {
            ".write": "auth.uid === $user_id",
            "cars": {
              ".read": "root.child('users/'+$user_id+'/friends/'+
    auth.uid).exists()"
            },
            "car_count": {
              ".read": "auth !== null"
            }

          }
        }
    }
}
```

Výpis zdrojového kódu 3.1: Implementace zabezpečení dat (Firebase)

První úroveň nastavení je na úrovni kolekce (Kinvey) / třídy (Parse), které se nastavuje z webové konzole. Na této úrovni Parse disponuje silnějším nástrojem (viz také obrázek 3.3), který umožňuje jednak rozlišit mezi právem na operaci get (uživatel musí znát ID entity, aby ji mohl přečíst) a operaci find (uživatel může vyhledat a přečíst ID entity na základě různých kritérií), jednak umožňuje nastavit přístup pro určité uživatelské role nebo konkrétní uživatele. Kinvey na této úrovni nabízí pouze nastavení jednoho ze 4 módů - Shared (read-all, write-yours), Private (read-yours, write-yours), Read Only (readl-all, write-none), Full (read-all, write-all).



Obrázek 3.3: Parse: Zabezpečení dat na úrovni tříd

Druhá úroveň nastavení je na úrovni instancí jednotlivých entit (tzv. Ac-

cess Control List). Je důležité zmínit, že v Případě Parse je toto nastavení doplňkem k první úrovni nastavení, v případě Kinvey má toto nastavení vyšší prioritu (nastavení na první úrovni je tedy přepsáno).

ACL je implementováno jako JSON objekt ukládaný u každé entity, který není nutné upravovat přímo, ale prostřednictvím předpřipravených metod na úrovni SDK.

Síla ACL pro obě služby je podobná a umožňuje nastavit práva na čtení nebo zápis u konkrétních entit konkrétním uživatelům nebo uživatelským skupinám.

S ohledem na srovnatelné možnosti obou služeb v této problematice bude dále popsáno řešení vzorového příkladu pouze prostřednictvím Kinvey (pro Parse je řešení principiálně stejné).

Auta uchováváme jako zvláštní kolekci „cars“, kde vlastnost entity owner je referencí na uživatele. Nejvhodnější implementace seznamu přátel uživatele by musela být vyhodnocena v širším kontextu, pro tento příklad jej implementujeme jako vlastnost entity User, která je seznamem ID uživatelů, kteří jsou přátelé. Dále pro řešení prvního podproblému postupujeme podle následujících kroků:

- Kolekci cars nastavíme na první úrovni zabezpečení mód Private.
- Implementujeme kód, který nejprve vymaže seznam všech uživatelů s právem číst a poté všem přátelům uživatele přiřadí právo na čtení všech aut uživatele.
- Vzniklý kód budeme spouštět vždy při vytvoření nového auta nebo při přidání / odebrání nového přítele. Spouštění lze řešit dvěma způsoby:
  - Voláním na vhodných místech na straně klienta.
  - Implementací tzv. postSave webhooks, tedy Server kódů, které jsou volány automaticky po uložení entity z vybrané kolekce.

Pro řešení druhého podproblému můžeme zvolit jedno z následujících dvou řešení:

- Data o počtu aut jednotlivých uživatelů uchováváme ve zvláštní kolekci, která má na první úrovni zabezpečení nastavený mód Shared. Aktuálnost těchto dat řešíme buď na klientovi nebo pomocí implementace postSave a postDelete webhooks.
- Implementujeme Server code typu endpoint (tedy dostupný voláním z SDK), který bude vracet počet aut pro konkrétního uživatele.

3.5.3.3 Shrnutí

Firebase řeší práva k datům centrálně konfiguračním souborem ve webové konzoli. Kinvey a Parse řeší problematiku podobně: práva lze definovat na úrovni kolekce prostřednictvím webové konzole nebo na úrovni entity. Práva na úrovni entity je třeba zpravidla řešit na klientovi.

V obou případech se jedná o neobvyklé řešení pro vývojáře, kteří nemají zkušenost s BaaS. Obě řešení jsou dle mého názoru náchylná k chybám - zejména k opomenutí vyřešení určitých práv.

3.5.4 Validace a integrita dat

Pro zamezení jak úmyslného (určitý typ útoku) tak neúmyslného (chybná implementace na straně klienta) ukládání dat v jiném, než očekávaném formátu, je vhodné mít na straně BaaS řešení nástroje k ošetření takovýchto situací. Typickými situacemi, které je třeba ošetřit může být:

1. Kontrola, že daná hodnota je v ukládaném objektu obsažena.
2. Kontrola, že hodnota je určitého typu.
3. Kontrola, že řetězcová hodnota splňuje určitý formát.
4. V případě, že hodnota je cizí klíč (tedy pravděpodobně ID nějakého objektu v jiné kolekci), zkontrolovat, že tento objekt existuje.
5. Komplexní kontrola hodnoty - např. kontrola platnosti rodného čísla.

3.5.4.1 Implementace ve Firebase

Validace dat ve Firebase je řešena podobně jako přístupová práva (viz kapitola 3.5.3.1). Vedle pravidel .write a .read lze specifikovat další pravidlo .validate, které je kontrolováno před uložením objektu do databáze. Pomocí těchto pravidel lze ošetřit situace 1 - 4, komplexní kontrolu hodnoty by bylo třeba implementovat na straně klienta (což nezabrání úmyslnému podvržení hodnoty). Demonstrace použití validačních pravidel je na ukázce zdrojového kódu 3.2.

3.5.4.2 Implementace v Parse

Atributům datových objektů v Parse je vždy přiřazen datový typ. Datové typy tedy řeší situaci č. 2.

Ostatní situace lze řešit prostřednictvím Server code typu BeforeSave Trigger (tedy skript, který je vždy spuštěn před uložením objektu určitého typu), které se programují v JavaScript.

Na ukázce zdrojového kódu 3.3 je demonstrace jednoduché validace v Parse s použitím server skriptu.

```
{
  "rules": {
    //zapis je globalne povolen
    ".write": true,
    "widget": {
      //platny widget musi mit atributy "color" a "size"
      ".validate": "newData.hasChildren(['color', 'size'])",
      "size": {
        //hodnota "size" musi byt cislo mezi 0 a 99
        ".validate": "newData.isNumber() &&
                      newData.val() >= 0 &&
                      newData.val() <= 99"
      },
      "color": {
        //hodnota "color" musi existovat jako klic v kolekci valid_colors
        ".validate": "root.child('valid_colors/' + newData.val()).exists()"
      }
    }
  }
}
```

Výpis zdrojového kódu 3.2: Ukázka validate dat (Firebase)

```
Parse.Cloud.beforeSave(Parse.User, function(request, response) {
  if (!request.object.get("email")) {
    response.error("email is required for signup");
  } else {
    response.success();
  }
});
```

Výpis zdrojového kódu 3.3: Ukázka validate dat (Parse)

3.5.4.3 Implementace v Kinvey

Kinvey datové objekty nedisponují datovými typy, tudíž veškeré situace 1 - 5 je třeba řešit prostřednictvím Server code. Použití Server kódu je principiálně stejné jako v případě Parse. Server skripty Kinvey disponují validačním modulem, který ulehčuje implementaci některých validačních případů.

3.5.4.4 Shrnutí

Firebase řeší tuto problematiku centrálním konfiguračním souborem, přičemž toto řešení nepokrývá validaci komplexní hodnoty. V Parse a Kinvey je tato problematika řešena Server kódem, který je navazán na operaci uložení objektu.

3.5.5 Plánované spouštění úloh

Žádoucí funkcionalitu backendu v případě některých aplikací je možnost spouštět pravidelně určitou úlohu např. pro dopočítání údajů z velkého množství dat nebo z důvodu importu dat z externího API.

3.5.5.1 Implementace ve Firebase

Jedinou možností, jak implementovat plánované úlohy nad daty nacházející se ve Firebase databázi, je vytvořit vlastní server, který se postará o pravidelné spouštění námi vytvořeného skriptu (v libovolném jazyce). Tento skript bude komunikovat s Firebase databází prostřednictvím REST API.

3.5.5.2 Implementace v Kinvey

Kinvey umožňuje nastavit plánované spouštění pro Business logic typu Endpoint, což zcela řeší tuto problematiku.

Je třeba zmínit, že maximální běh úlohy je omezen dle placeného tarifu (v nejlevnější variantě je běh omezen na 2 sekundy).

3.5.5.3 Implementace v Parse

V době psaní této práce Parse server nepodporuje plánované úlohy[35]. Řešení je tedy podobné jako v případě Firebase s tím rozdílem, že můžeme problematiku řešit na stejném serveru jako hostujeme Parse, pokud k němu máme plný přístup.

3.5.5.4 Shrnutí

Plánované spouštění úloh není podporováno u Firebase ani Parse (v případě Parse se pravděpodobně jedná o dočasný stav). Kinvey tuto funkcionalitu podporuje, ale je třeba dát si pozor na omezení maximálního dobu běhu skriptu.

3.5.6 Propagace změn dat do zařízení - synchronizace Client to Client

Touto problematikou jsou míněny zejména situace, kdy uživatel aplikace pracuje v zařízení A nad nějakými daty, která jsou ukládána na server a tato data jsou upravena ze zařízení B téhož uživatele nebo ze zařízení C jiným uživatelem s tím, že cílem je zajistit, aby uživatel v každém zařízení viděl pokud možno (tedy pokud je zařízení zrovna připojeno k internetu) aktuální podobu dat.

[35] Hostovaná verze Parse umožňovala nastavit pravidelné spouštění pro Cloud code typu Function

Tuto problematiku lze řešit několika způsoby, přičemž je třeba vždy zohlednit kontext. Zejména u mobilních aplikací je třeba problém optimalizovat z hlediska objemu komunikace a náročnosti na spotřebu baterie zařízení.

Níže jsou popsány obecně možnosti řešení této problematiky.

3.5.6.1 RT synchronizace dat

Princip tohoto řešení je popsán detailně v kapitole 2.4.1. V případě webových aplikací může být toto řešení pro většinu případů plně vyhovující.

V případě mobilních aplikací vývojář pravděpodobně narazí na následující problém: když je aplikace přepnuta do režimu na pozadí, operační systém odpojí komunikaci probíhající prostřednictvím Websockets. Ta je obnovena ve chvíli, kdy aplikace přejde do popředí, data se pak pravděpodobně velmi rychle aktualizují v případě, že zrovna existuje připojení k internetu. Určitým workaroundem může být na chvíli „probudit“ aplikaci push notifikací, což způsobí obnovení komunikace prostřednictvím Websockets a tedy synchronizaci dat.

3.5.6.2 Polling

Princip pollingu, tedy pravidelného dotazování na server, je demonstrován na ukázce zdrojového kódu 2.1. Polling může být vhodným řešením pro webové aplikace, u mobilních aplikací nastává problém se zbytečným tokem dat a plýtváním energie baterie v důsledku nepotřebných dotazů na server.

3.5.6.3 Polling aktivovaný push notifikací

Toto řešení se týká výhradně mobilních aplikací.

Princip tohoto řešení spočívá v mechanismu, kdy server (popř. jiný klient) zašle push notifikaci, která obsahuje informaci o tom, že došlo ke změně určitých dat. Aplikace se následně dotáže serveru na potřebná data.

Výhodou tohoto řešení je šetrnost pro baterii i možnost optimalizovat množství přenášených dat mezi klientem a serverem. Nevýhodou je náročnost implementace takového řešení.

3.5.6.4 Implementace ve Firebase

Firebase implementuje RT synchronizaci dat nativně. Toto řešení je tedy optimální pro použití s Firebase. Implementovat polling u Firebase je v zásadě nesmysl, protože při práci s Firebase se na data nedotazujeme, nýbrž dochází k jejich synchronizaci (to opět plyne z nativně podporované RT synchronizace).

Použití zmiňovaného workaroundu pro aktivaci RT synchronizace v době, kdy je aplikace v pozadí, se nabízí. Zasílání push notifikací by však muselo být řešeno mimo Firebase, neboť funkcionalitou push notifikací Firebase nedisponuje.

3.5.6.5 Implementace v Kinvey

Kinvey nepodporuje RT synchronizaci dat. Polling lze v Kinvey bezproblémově realizovat. Polling aktivovaný Push notifikací lze taktéž realizovat. Zasílání Push notifikací lze např. navázat na událost vytvoření nebo smazání entity prostřednictvím tzv. Business logic collection hooks. Také je možné realizovat Business logic typu endpoint, který např. může vracet jen datové záznamy, které se změnily od určité doby, a optimalizovat tak množství přenášených dat a zjednodušit tak logiku implementovanou na straně klienta.

3.5.6.6 Implementace v Parse

Parse podporuje tzv. „Live queries“, což je v této práci definovaná RT synchronizace dat (v době psaní této práce byly Live Queries podporovány pouze pro JavaScript a iOS SDK).

Možnosti pollingu a pollingu aktivovaného push notifikací jsou stejné jako v případě Kinvey.

3.5.6.7 Shrnutí

Pokud se omezíme na mobilní aplikace, u Firebase je tato problematika dobře vyřešena pro aplikace, které je třeba synchronizovat pouze v době, kdy běží na popředí. V případě Parse a Kinvey se tato problematika bude řešit pravděpodobně použitím push notifikací - v takovém případě vývojářům BaaS ulehčí práci řešením push notifikací, samotnou synchronizaci je třeba implementovat vlastními silami.

3.5.7 Podpora odesílání a přijímání dat pouze na Wi-Fi

Typickým příkladem této situace může být nahrávání nebo stahování fotografií do zařízení, které chceme realizovat pouze v případě, kdy je uživatel připojen na internet bez omezení FUP.

3.5.7.1 Implementace ve vybraných BaaS

Tato funkcionalita není v žádném z vybraných BaaS přímo implementována. Logika detekující typ připojení a na to navázání určitých operací musí být tedy implementována vývojářem.

3.5.8 Debugging

Při vývoji jakékoliv aplikace je třeba disponovat nástroji pro debugging. Proto je vhodné zaměřit se také na to, jaké ladící nástroje a možnosti nabízí vybraná BaaS řešení. Definujme nejprve oblasti, v jakých bychom mohli potřebovat určitou podporu debugování:

1. Prohlížení serverové databáze.

2. Prohlížení souborů uložených na serverů.

3. Debugování funkcí volaných prostřednictvím SDK.

4. Debugování vytvářených Server scripts.

5. Prohlížení lokální databáze.

6. Možnost logovat vlastní informace z klientů a shromažďovat je na serveru.

Čísla položky seznamu v implementačních podkapitolách rozebírá možnosti debuggingu v oblasti odpovídající číslu ze seznamu výše.

3.5.8.1 Implementace ve Firebase

1. Prohlížení (a ostatní operace) databáze je možné realizovat ve webové konzoli a prostřednictvím rozšíření do prohlížeče Chrome Vulcan[36]. Databáze je zobrazována formou stromové struktury, kterou lze postupně rozbalovat. Případně jde zobrazit jen určitý podstrom zadáním konkrétního URL do prohlížeče (a tak např. zobrazit položku s konkrétním ID). Toto uživatelské rozhraní je k náhlednutí na obrázku 3.2. Jinými filtrovacími a vyhledavacími nástroji toto rozhraní nedisponuje, což může být při velkém počtu položek v databázi problematické. Data lze v tomto rozhraní také vytvářet, modifikovat a mazat.

2. Není relevantní (Firebase nenabízí File management)

3. API metodám se dává jako parametr callback, který dostává jako parametr objekt typu error s informacemi o typu chyby.

4. Není relevantní (Firebase nenabízí Server code).

5. Neexistuje zdokumentovaná cesta, jak prohlížet lokální verzi databáze.

6. Přímé logování vlastních informací není podporováno. Určitým částečným workaroundem může být tyto informace zapisovat do databáze.

[36] `https://chrome.google.com/webstore/detail/vulcan-by-firebase/oippbnlmebalopjbkemajgfbglcjhnbl?hl=en`

3.5.8.2 Implementace v Kinvey

1. Prohlížení (a ostatní operace) databáze je možné realizovat ve webové konzoli. Na data lze nahlížet vždy z pohledu kolekce (tedy entit určitého typu). Rozhraní disponuje rychlým vyhledáváním, které prohledává všechny atributy, a nebo pokročilým vyhledáváním, jehož síla odpovídá síle dotazovacího jazyku. Prostřednictvím webové rozhraní lze data také vkládat nebo modifikovat.

2. Soubory jsou ve webové konzoli dostupné jako jedna z databázových kolekcí.

3. API metodám se dává jako parametr callback, který dostává jako parametr objekt typu error s informacemi o typu chyby.

4. Server scripts disponují logovacím modulem. Výstupy z logovacího modulu se vypisují vždy u každého server skriptu ve webové konzoli. Server script lze ve webové konzoli přímo spustit a tak otestovat jeho funkčnost.

5. Neexistuje zdokumentovaná cesta, jak prohlížet lokální databázi.

6. Přímé logování vlastních informací není podporováno. Určitým částečným workaroundem může být tyto informace zapisovat do databáze.

U jednotlivých SDK lze zapnout debugovací mód, což znamená, že SDK bude do konzole vypisovat informace o tom, jaké operace jsou prováděny.

3.5.8.3 Implementace v Parse

1. Prohlížení (a ostatní operace) databáze je možné realizovat ve webové konzoli[37]. Na data lze nahlížet vždy z pohledu třídy (class) - tedy entit určitého typu. Entity lze v rozhraní také úpravovat, přidávat a mazat. Rozhraní disponuje možností filtrování podle konkrétních atributů a několika operátorů. Pokud disponujeme plným přístupem k serveru, lze také použit pro prohlížení dat libovolný nástroj pro MongoDB.

2. Možnost prohlížení souborů závisí na adaptéru, který je použit pro ukládání souborů (viz kapitola 3.4.4.3). Přímo ve webové konzoli lze soubory prohlížet jen pokud jsou navázány na nějakou databázovou entitu.

3. API metodám se dává jako parametr callback, který dostává jako parametr objekt typu error s informacemi o typu chyby.

4. Jakákoliv práce se Server scripts není v době psaní práce v dashboardu podporována. Je možné použít libovolné debugovací knihovny při psaní server scripts, pokud máme plný přístup k serveru.

[37] Je důležité zmínit, že Parse Dashboard je samostatnou aplikací, kterou je třeba instalovat a není součástí Parse server. Viz `https://github.com/ParsePlatform/parse-dashboard`

5. Neexistuje zdokumentovaná cesta, jak prohlížet lokální databázi. Libovolný dotaz prostřednictvím SDK lze však na lokální databázi aplikovat.

6. Dokumentace navrhuje, že Parse Analytics může být použito jako jednoduchý error tracker. V době psaní této práce však webová konzole nedisponuje nástrojem pro zobrazování dat z Analytics (ačkoliv mechanismus ukládání dat do analytiky funguje).

3.5.8.4 Shrnutí

Možnosti debugování u všech tří BaaS řešení existují a jsou na podobné úrovni. Společným nedostatkem všech řešení je nemožnost prohlížet lokální databázi na klientovi.

3.5.9 Škálovatelnost a geolokační distribuovatelnost

V případě, že je u aplikace očekáván velký počet uživatelů nebo je obtížně odhadnutelný nárůst používanosti, je třeba zajímat se o to jakým způsobem je BaaS poskytovatel schopen škálovat své řešení a zda existují nějaké limity.

Pokud bude vyvíjená aplikace užívaná po celém světe napříč kontinenty, je dobré vědět, zda BaaS poskytovatel zajistí, že aplikace bude reagovat srovnatelně odkudkoliv.

3.5.9.1 Firebase

V oblasti škálovatelnosti, prvním aspektem, který by v případě Firebase měl být brán v potaz, jsou oficiálně uvedené limity databáze[38].

Pokud budeme vycházet z informací uvedených u cenových tarifů Firebase, pak dalším potenciálně úzkým hrdlem je maximální počet připojení k databázi v jedné chvíli, které je standardně omezeno na 10 000. Dle technické podpory však toto omezení není limitní a po domluvě může být limit navýšen či zcela zrušen (žádná jasně stanovená horní hranice tedy dle technické podpory neexistuje).

V případě, že jsou překročeny limity zakoupeného tarifu, jsou účtovány poplatky za určitou jednotku každého aspektu dle zvláštní sazby (např. 5 dolarů za GB databáze za měsíc). Toto neplatí pro maximální počet připojení k databázi, kde definované omezení platí.

V oblasti geolokační distribuovatelnosti, dle technické podpory, nejsou významné rozdíly mezi dobou odezvy řešení napříč různými kontinenty.

[38] Viz `https://www.firebase.com/docs/web/guide/understanding-data.html#section-limitations`

3.5.9.2 Kinvey

Jediný explicitně stanovený limit, který by měl být dle oficiálních zdrojů brán v potaz, je maximálně velikost databáze omezená na 30 GB.

Více detailů o škálovatelnosti nebo geolokační distribuovatelnosti Kinvey řešení se nepodařilo zjistit. Kinvey doporučuje v případě pochybností obrátit se na technickou podporu s dotazem ohledně konkrétního projektu.

3.5.9.3 Parse

S ohledem na to, že Parse nenabízí hostování svého řešení, je problematika škálovatelnosti přenesena na uživatele BaaS řešení. Parse jako celé řešení se skládá z několika komponent (MongoDB, Parse Server, Webový server (např. Nginx), Parse Dashboard), tyto komponenty jsou nezávislé a každá z nich může být škálována zvlášť.

Jednotlivé komponenty mohou být hostovány u různých poskytovatelů cloudových řešení typu PaaS (např. Heroku) nebo IaaS (např. DigitalOcean).

3.5.9.4 Shrnutí

U Firebase nebyly zjištěny žádné explicitní limity. V případě Kinvey může být limitující maximální velikost databáze. Problematika škálování Parse je přenesena na uživatele řešení.

3.6 Aplikace Cost Tracker

3.6.1 Cíl aplikace

Cílem aplikace je otestovat vybraná BaaS řešení na komplexním příkladu.

3.6.2 Funkcionalita aplikace

Aplikace umožní uživateli stanovit si maximální výši svých výdajů v různých kategoriích pro jím definované období. Dále umožní zanášet informace o svých výdajích a během období sledovat výši svých nákladů vzhledem k definovaným maximálním výším.

3.6.3 Návrh

S ohledem na cíl aplikace a znalosti autora bylo rozhodnuto, že aplikace bude vytvářena jako webová SPA.

Je vhodné zmínit, že samotný koncept a návrh aplikace vznikl ještě před detailní analýzou vybraných BaaS řešení.

3.6.3.1 Vybrané technologie

Použité technologie pro vývoj aplikace v kontextu cíle aplikace nejsou příliš důležité, proto následuje jen stručný přehled a popis technologií použitých na tomto projektu:

- Node.js a NPM - instalace knihoven a správa závislostí
- Bower - instalace knihoven a správa závislostí pro frontend.
- Yeoman - Vytvoření základní kostry projektu.
- AngularJS - MVC framework pro JavaScript.
- Foundation - Tvorba uživatelského rozhraní.
- SASS - Tvorba CSS.
- Babel - kompilace ECMAScript2015 do ECMAScript5
- Gulp - automatizace podpůrných procesů (kompilace, spouštění serveru).

3.6.3.2 Uživatelské rozhraní

Uživatelské rozhraní bylo navrhováno přímo pomocí frameworku Foundation a je složeno z následujících obrazovek:

- Přihlášení a registrace
- Přehled a správa období
- Přehled a správa kategorií
- Přehled a správa plateb
- „Dashboard“ - vložení platby a přehled sledovaného období

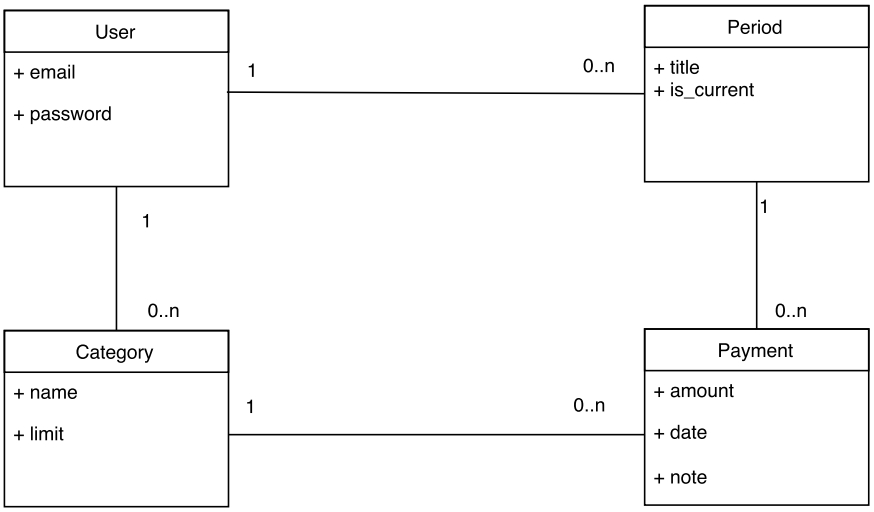
Náhledy uživatelského rozhraní se nachází v příloze C.

3.6.3.3 Struktura databáze

Implementačné nezávislý návrh databáze pro aplikaci je zobrazen na obrázku 3.4.

Následuje stručný popis entit:

- User - Reprezentuje uživatele aplikace.
- Period - Období, za které jsou sledovány výdaje. Příznak is_current značí, ve kterém období uživatel právě pracuje.



Obrázek 3.4: Návrh databáze aplikace Cost tracker

- Category - Kategorie, ve kterých uživatel sleduje své výdaje. Atribut limit je maximální výše výdajů, kterou uživatel nechce za jedno období přesáhnout.

- Payment - Reprezentuje platbu - tedy výdaj uživatele ve výši amount vázající se k určité kategorií a určitému období.

3.6.4 Postup implementace

Postup implementace byl následující:

1. Bylo vytvořeno uživatelské rozhraní.

2. Uživatelské rozhraní bylo nasazeno na MVC framework AngularJS.

3. Byla vytvořena maketa modelové vrstvy poskytující vzorová data a byla vytvořena logická vrstva aplikace komunikující s touto maketou.

4. Byla implementována integrace s jednotlivými BaaS řešeními se záměrem nahradit modelovou vrstvu a udělat jen minimální nezbytné zásahy do ostatních vrstev aplikace.

Následující tři kapitoly shrnují zkušenosti s integrací jednotlivých BaaS řešení.

3.6.5 Implementace s Firebase

Pro integraci Firebase s AngularJS byla použita knihovna AngularFire[39], která rozšiřuje koncept AngularJS nazývaný two-way data binding. Two-way data binding je koncept, kdy dochází z pohledu vývojáře k automatické synchronizaci dat mezi modelem a view. Tedy jakákoliv změna provedená v modelu se ihned projeví v UI aplikace a obráceně. AngularFire tento koncept rozšiřuje o automatickou okamžitou synchronizaci s Firebase databází - tzv. three-way data binding. Pokud tedy uživatel pracuje s nějakými daty v UI, změna těchto dat je okamžitě propagována nejen do modelové vrstvy, ale také do serverové databáze a obráceně.

3.6.5.1 Zkušenosti z implementace

Netriviálním problémem je správný návrh struktury Firebase databáze. Řešení není jednoznačné a správný návrh závisí na způsobu použití (čtení) dat[40]. Výsledné převedení navrženého datového modelu na strukturu Firebase databáze je vyobrazeno na ukázce zdrojového kódu 3.4.

Lze konstatovat, že používání Firebase může obnášet pro vývojáře určitou změnu paradigmatu způsobenou zejména neobvyklou prací s databází.

Obtížné je řešení zejména vztahů mezi daty při zachování RT synchronizace. Pokud je např. odebírán seznam plateb aktuálního období, je vhodné implementovat řešení tak, aby pružně reflektovalo změnu uživatele i aktuálního období. Tento problém je demonstrován na ukázce zdrojového kódu 3.5, která pracuje s databází z předešlého zdrojového kódu 3.4.

Jelikož možná manipulace s daty na straně databáze je značně omezená, je třeba řadu věcí řešit přímo algoritmy na úrovni controlleru. Původní předpoklad nahradit jen modelovou vrstvu databáze nebyl u Firebase zcela možný.

Lze říci, že ačkoliv navrženou aplikaci se podařilo ve Firebase realizovat, aplikace jako taková není vhodným typem aplikace pro realizaci na Firebase z důvodu, že nevyužívá hlavní přednosti Firebase - RT synchronizaci dat (nevyužívá ve smyslu, že tuto funkcionalitu ze své podstaty nepotřebuje) a její implementace je tedy zbytečně náročná[41].

3.6.6 Implementace s Parse

Pro integraci na straně klienta byla použita standardní knihovna pro JavaScript. V rámci implementace proběhlo také nasazení Parse Server a Parse

[39] https://www.firebase.com/docs/web/libraries/angular/

[40] Více informací ke správnému návrhu jsou dostupné na `https://www.firebase.com/docs/web/guide/structuring-data.html`

[41] Na základě této zkušenosti byl v přehledu BaaS v příloze D změněna oblast zaměření z univerzální na specializovaný.

Dashboard k IaaS poskytovateli DigitalOcean. Řešení bylo nasazeno na VPS s OS Ubuntu 14.04.4 x64[42].

3.6.6.1 Zkušenosti z implementace

Navržený databázový model byl převeden do Parse databáze poměrně přímočaře - každá entita odpovídá jedné Parse třídě.

Nahrazení maketové modelové vrstvy modelovou vrstvou komunikující přímo s Parse se obešlo bez velkých zásahů do původního rozhraní této vrstvy. Určité změny na úrovni controllerů musely být provedeny, což bylo způsobeno zejména tím, že instance Parse modelových tříd kvůli svým getterům a setterům nejsou implicitně příliš dobře kompatibilní s AngularJS.

Obecně lze konstatovat, že implementace v Parse byla bezproblémová a Parse jako takový lze doporučit jako vhodný BaaS pro takovýto typ aplikace.

3.6.7 Implementace s Kinvey

Pro integraci byla použito oficiální SDK pro AngularJS, které je lehkou nadstavbou nad oficiálním SDK pro JavaScript obalující volání Kinvey SDK do tzv. services angularJS.

3.6.7.1 Zkušenosti z implementace

Lze konstatovat, že implementace Kinvey byla velmi podobná implementaci Parse.

Navržený databázový model byl převeden takovým způsobem, že každá entita z návrhu odpovídá jedné databázové kolekci v Kinvey.

Kinvey lze doporučit jako vhodný BaaS pro takovýto typ aplikace.

3.7 Silné a slabé stránky vybraných řešení

Na základě zkušeností získaných při detailním srovnávání funkcionality a při tvorbě aplikace Cost Tracker se pokusím v následujících podkapitolách shrnout silné a slabé stránky zkoumaných řešení.

3.7.0.2 Firebase

Firebase lze doporučit jako BaaS vhodné pro aplikace, které dokáží využít RT synchronizaci dat. O Firebase není vhodné uvažovat jako o BaaS vhodném pro libovolný typ aplikace. Limity a omezení, kterými funkcionalita Firebase disponuje v důsledku toho, že zachovává RT synchronizaci dat, jsou zbytečnou komplikací pro aplikace, které RT synchronizaci dat nevyužijí. Funkcionalita, kterou Firebase disponuje, je dobře zpracována a zdokumentovaná,

[42]Nasazení proběhlo na základě návodu `https://www.digitalocean.com/community/tutorials/how-to-run-parse-server-on-ubuntu-14-04`

přesto může vývojářům zpočátku trvat déle, než se s Firebase naučí pracovat z důvodu, že práce s Firebase vyžaduje osvojení si určitého paradigmatu.

Vzhledem k vlastnostem Firebase databáze je dobré již při návrhu uživatelského rozhraní aplikace uvažovat v kontextu možností Firebase. Lze tedy říci, že rozhodnutí o (ne)využití Firebase zasahuje zásadním způsobem do návrhu a architektury aplikace.

3.7.0.3 Parse

Parse lze označit za vyspělé řešení, kterému se podařilo najít velmi dobrý kompromis mezi použitelností a množstvím funkcionality, kterou disponuje. Jako takové může být obecně vhodným řešením pro většinu consumer-oriented aplikací. Dokumentace je velmi dobrá a křivka učení strmá.

Největším nedostatkem Parse je zrušení jeho hostované verze a s tím spjatý přechod na open-source (viz kapitola 3.2.2). Touto situací odpadá jeden z hlavních benefitů BaaS - totiž nutnost nestarat se o provoz backendu jako takového a zároveň přináší výzvu v oblasti řešení škálovatelnosti pro aplikaci s velkým počten uživatelů. Tuto situaci lze z druhé strany vidět také jako příležitost. Přechodem na open-source má Parse potenciál překlenout problém uzavřeného řešení (viz kapitola 5.1) a transformovat se spíše do podoby platformy či frameworku pro vývoj aplikací jako je např. Meteor[43].

V době psaní této práce bych však rozhodně nedoporučil Parse pro použití na jakémkoliv projektu do té doby, než se situace okolo tohoto BaaS více stabilizuje.

3.7.0.4 Kinvey

Kinvey je řešení, které se svým přístupem velmi podobá Parse před zrušením hostované verze. Oproti Parse nabízí více funkcionality (ne všechna funkcionalita Kinvey byla v rámci této analýzy zkoumána). Pokud srovnáme stejný typ funkcionality u Parse a Kinvey, Kinvey často nabízí více prostoru pro customizaci ze strany uživatele BaaS (viz kapitoly 3.4.2 a 3.4.3). Během testování Kinvey jsem však nenabyl přesvědčení, že tyto možnosti customizace v konečném důsledku přináší benefit pro uživatele BaaS. Působí to na mě dojmem, že autoři nedokázali najít ideální řešení, tak jich raději nabídli více.

Celkově vzato lze Kinvey považovat za vyspělé univerzální BaaS vhodné pro většinu consumer-oriented aplikací a lze ho doporučit pro použití na projektech.

[43] https://www.meteor.com/

```
{
  "users" : {
    "6e2e7e70-0c13-4697-9daa-b378a320f38a" : { //generovane id uzivatele
      "categories" : { //kategorie uzivatele
        "-K4OwPpp8XujwsGfenDO" : {
          "limit" : 200,
          "title" : "Jidlo"
        },
        "-K4OwWylXCGPq7maLmJ_" : {
          "limit" : 200,
          "title" : "Zabava"
        },
      },
      "current_period" : "-K4OwA1nnZ4WOswJtddg", //id aktivniho obdobi
      "email" : "m.majsky@gmail.com",
      "payments" : { // seznam plateb
        "-K4OwA1nnZ4WOswJtddg" : { //platby pro obdobi s danym ID
          "-K4Rgb49igDRnqkYCTXW" : {
            "amount" : 15,
            "category" : "-K4OwdWDfbUWuFon0V-r",
            "date" : 1448959158314,
            "note" : "Mobil"
          },
          "-K4RglR7ZErRW1OblYKw" : {
            "amount" : 30,
            "category" : "-K4OwgnyT0W3BeG94Tga",
            "date" : 1448959180176,
            "note" : "cepice"
          },

        }
      },
      "periods" : { //obdobi
        "-K4OwA1nnZ4WOswJtddg" : {
          "title" : "December 2015"
        },
        "-K8TVJkUfBgRlUM52DuC" : {
          "title" : "Test"
        }
      }
    },
    "c1f3db12-3b93-4022-b31c-027079c87208" : { //dalsi uzivatel se stejnou strukturuu dat
      "email" : "m.majsky+test@gmail.com"
      //...
      }
    }
  }
}
```

Výpis zdrojového kódu 3.4: Firebase databáze aplikace Cost Tracker

```
var subscribePayments = function(callback) {
  $firebaseAuth(firebaseRef).$onAuth(function (authData) {
    if (authData) {
      var userDataRef = firebaseRef.child('users').child(authData.
    uid);
      var curPeriodRef = userDataRef.child('current_period');
      curPeriodRef.on('value', function (data) {
        if (data.val() != null) {
          var currentPeriod = $firebaseObject(ref.child('periods')
    .child(data.val()));
          callback(costFirebaseArray(firebaseRef.child('users').
    child(authData.uid).child('payments').child(currentPeriod.$id)
    ));
        }
      });
    }
  });
};

subscribePayments(function(data) {
  $scope.payments = data;
});
```

Výpis zdrojového kódu 3.5: Firebase: Demonstrace problematiky vztahů mezi daty

Kapitola 4

Ekonomická analýza vybraných řešení

Cílem této kapitoly je poskytnout informace ohledně tématu jakým způsobem odhadnout náklady na používání BaaS řešení a jakým způsobem srovnat z ekonomického hlediska jednotlivá BaaS řešení mezi sebou.

4.1 Postup

Pro ekonomickou analýzu byl vymyšlen následující postup, který lze obecně použít pro srovnání ekonomických nákladů jednotlivých BaaS řešení ze všech kategorií definovaných v kapitole 2.3.3 pro konkrétní projekt. Lze jej popsat následujícími kroky:

1. **Identifikace relevantních aspektů**. U každého srovnávaného BaaS řešení je třeba identifikovat aspekty, které mají vliv na projekt, který bude využívat BaaS.

2. **Nalezení vztahů mezi aspekty**. Jelikož každý BaaS poskytovatel používá jiné aspekty, je vhodné provést úvahu, zda některé aspekty, byť mají různý název, neznamenají v důsledku totéž nebo zda jeden na druhý nelze snadno převést.

3. **Analýza vlivu aspektů na cenu**. V tomto kroku se zjišťuje, jaký vliv má daný aspekt u BaaS poskytovatele na výslednou cenu.

4. **Výpočet ceny pro jednotlivá BaaS**. V tomto kroku se provede odhad výše relevantních aspektů pro nasazovaný projekt a na základě tohoto odhadu je spočítána cena pro každé BaaS řešení.

Tento postup je v následujících kapitolách prakticky předveden na srovnání třech BaaS řešení, který byla vybrána v kapitole 3.2.

4.2 Identifikace relevantních aspektů

V kapitole 2.3.3 byla provedena kategorizace BaaS řešení dle způsobu účtování. Z definovaných kategorií BaaS Firebase a Kinvey patří do kategorie platba za tarif, přičemž obě řešení nabízejí základní tarif zdarma. Parse řešení je open-source a problematika nasazení je přenesena na uživatele. Náklady na provoz Parse budou tedy záviset na zvoleném způsobu hostování.

Firebase nabízí 6 tarifů. Každý tarif je určen měsíčními limity pro 5 aspektů (Database Connections, Database Storage, Database Transfer, Hosting Storage, Hosting Transfer). Pro zjednodušení budou ve výpočetním modelu uvažovány pouze následující aspekty týkající se databáze[44].

Vysvětlení aspektů je následující:

- **Database Connections (dále DC_{fire})** je počet zařízení, která paralelně v jednu chvíli přistupují k databázi.
- **Database Storage (dále DS_{fire})** je celkový objem dat uložených v databázi v GB.
- **Database Transfer** je celkový objem dat přenášený z databáze a do databáze v GB.

Kinvey nabízí čtyři tarify pro managed hostovanou variantu řešení (vysvětlení viz kapitola 2.3.2.), přičemž první dva tarify cílí na malé firmy. Tyto tarify jsou účtovány měsíčně. Druhé dva tarify cílí na velké podniky (vysvětlení viz kapitola 2.3.4) a jsou účtovány ročně. Tarify jsou složeny z velkého množství aspektů, většina z nich však nejsou klasické „škálovatelné aspekty“ typu max. počet uživatelů, ale (ne)poskytnutí určité funkcionality / podpory. Vzhledem k tomu, že tato dodatečná funkcionalita / podpora cílí spíše na velké podniky, v rámci dalších výpočtu budou brány v potaz tyto aspekty:

- **Maximální počet aktivních uživatelů (dále U_{kinvey})** - aktivní uživatel je jakýkoliv uživatel, který v daném měsíci provedl dotaz na API.
- **Business logic timeout (dále BL_{kinvey})** je maximální doba běhu Server skriptu.
- **Database Storage (dále DS_{kinvey})** - vysvětlení viz odstavec výše.

V případě Parse předpokládáme pro ilustraci jeho nasazení na Heroku, kde pro projekt využijeme následující služby[45]:

[44] Toto zjednodušení je zdůvodněno následovně: Firebase hosting nemusí být na projektu vůbec využit. Pokud využit je, limity jsou dle názoru autora nastaveny natolik vysoko, že se s vysokou pravděpodobností nestanou důvodem pro přechod na vyšší tarif

[45] Využití těchto služeb vychází z návodu jak zmigrovat Parse na heroku `https://devcenter.heroku.com/articles/deploying-a-parse-server-to-heroku`

- mLab MongoDB[46]
- Dynos pro obsluhu požadavků na Parse Server

Náklady na MongoDB lze určit snadno - cena je stanovena tarifem, který má jediný aspekt - velikosti databáze (dále DS_{parse}). Oproti tomu určit počet Dynos (dále D_{parse}). na obsluhu Parse Server je netriviální problém. Pro výpočet byl aplikován následující vzorec:

$$D_{parse} = \left\lceil \frac{\frac{N_s.T_s}{N_d}}{T_{max}} \right\rceil = \left\lceil N_s.\frac{T_s}{N_d.T_{max}} \right\rceil$$

Způsob účtování Dynos lze tedy označit za „Pay as you go“, kde

- T_s je doba odbavení jednoho požadavku na server.
- N_s je počet souběžných požadavků přicházejících na server
- N_d je počet současných požadavků, které dokáže odbavit jedno Dyno,
- T_{max} je maximální přípustná doba pro odbavení jednoho požadavku (tedy nejvyšší možná doba, po jaké klient musí dostat odpověď ze serveru).

T_s, N_d, T_{max} jsou konstanty, které je třeba vhodně zvolit. Pro srovnání, která jsou dále prezentována, byly tyto konstanty zvoleny následovně. $N_d = 1$ z důvodu, že předpokládáme použití Dynos standard-1x (cena 25 $ měsíčně za 1 dyno) a budeme se řídit výchozím doporučením 1 worker pro 1 dyno tohoto typu [20]. Dále předpokládáme, že doba obsloužení jednoho požadavku $T_s = 10\,ms$ a klient by na obsloužení neměl čekat déle, než 0,5 sekundy, tedy $T_{max} = 500\,ms$.

4.3 Nalezení vztahů mezi aspekty

Aby bylo možné BaaS mezi sebou určitým způsobem srovnat, je třeba zavést několik dalších předpokladů:

$$DS_{fire} = DS_{kinvey} = DS_{parse}$$

Tedy, že aplikace by ve všech BaaS využívala stejný objem dat v databázi. Dále předpokládáme, že

$$DC_{fire} = k_1.U_{kinvey}; k_1 \in (0, 1\rangle$$

$$N_s = k_2.U_{kinvey}; k_2 \in (0, 1\rangle$$

[46] viz `https://elements.heroku.com/addons/mongolab`

$$k_2 < k_1$$

Tedy, že počet současných připojení na Firebase databázi je k_1-tina celkového počtu uživatelů a počet souběžných požadavků na Parse server je k_2-tina celkového počtu uživatelů. Pro dále demonstrované srovnání uvažujeme $k_1 = 0.1, k_2 = 0.01$, což přeneseně říká, že z celkového počtu aktivních uživatelů aplikace tuto aplikaci bude využívat maximálně desetina z těchto uživatelů současně, což vytvoří maximálně ještě desetkrát méně souběžných požadavků na server. Pro uznání této úvahy je třeba pochopit, že zatímco připojení DC_{fire} mezi serverem a klientem je otevřené po celou dobu používání aplikace, požadavky na server v případě Parse N_s vznikají (např. při otevření aplikace, provedení určité akce) a jsou ihned odbaveny.

4.4 Analýza vlivů aspektů na cenu

Na základě vztahu popsaných v předchozí kapitole byl implementován kalkulátor nákladů jako skript naprogramovaný nad Google Spreadsheet[47], kde jako vstup uživatel zadává[48]:

- počet aktivních uživatelů měsíčně
- velikost databázového uložiště
- objem přenášených dat z databáze a do databáze
- max. dobu běhu server skriptu

Dále může uživatel upravit výstup konfigurací výše definovaných konstant $k_1, k_2, T_s, N_d, T_{max}$ a upravovat účtované ceny nebo limity aspektů tarifů.

Tabulka 4.1 srovnává ceny BaaS jen na základě počtu aktivních uživatelů, tedy při nulové velikosti databáze. Zvolené hodnoty Max. počtu uživatelů představují limitní hodnoty pro tarify Kinvey nebo Firebase. Ačkoliv tento příklad neodpovídá reálné situaci, lze z výsledných čísel pozorovat tyto závěry:

- Cena Parse na Heroku za uživatele roste lineárně. Zde je třeba zmínit, že cena by v případě Heroku šla pravděpodobně výrazně optimalizovat dynamickým škálováním počtu Dynos na základě aktuálního vytížení - tedy např. dle aktuální doby obsloužení požadavku[49].
- Počet uživatelů sám o sobě v případě Firebase nemá rozhodující vliv na cenu.

[47] Dostupné z `https://docs.google.com/spreadsheets/d/1TSnKCEKiRESYMzxAjlcWN0f1jY91yx9JxvfLAaYqkko/edit?usp=sharing`

[48] Pokud uživatel některou z těchto vstupních proměnných nevyplní, pak s ní není kalkulováno - její hodnota je nula.

[49] Dynamické škálování pro Heroku nabízí např. služba HireFire - `https://www.hirefire.io`

Tabulka 4.1: Srovnání ceny BaaS dle počtu uživatelů

Max. počet uživatelů	1000	10 000	100 000	1 000 000
Počet současných připojení	100	1000	10 000	100 000
Počet souběžných dotazů	10	100	1 000	10 000
Cena Parse na Heroku ($)	25	50	500	5 000
Cena Kinvey ($)	0	200	200	2 000
Cena Firebase ($)	0	49	49	49

Tabulka 4.2 nabízí opačný pohled - srovnání ceny pouze na základě velikosti databáze (tedy při nulovém počtu uživatelů)[50]. Zvolené hodnoty velikosti DB zanesené v tabulce jsou limitní hodnoty pro tarify alespoň jednoho z poskytovatelů. Z tabulky lze vyvodit tyto závěry:

- Velikost databáze u Kinvey nemá vliv na cenu. Zároveň databázi u Kinvey nelze škálovat nad 30 GB.
- V případě Firebase lze říci, že velikost databáze má zásadní vliv na cenu.
- Samotné hostování databáze u Heroku je ve ve většině případů (10 z 15) dražší než tarif u Firebase zahrnující i další aspekty v ceně.

4.5 Výpočet ceny pro jednotlivá BaaS

Pro demonstraci předpokládejme, že je plánováno nasazení aplikace, jejíž náklady predikujeme na prvních 6 měsíců provozu. Počet uživatelů v jednotlivých měsících je stanoven odhadem. Dále je odhadováno, že data jednoho uživatele budou průměrně odpovídat velikosti 5 MB. V tabulce 4.3 jsou zobrazeny výsledné údaje. Ceny služeb pro jednotlivé měsíce byly jednoduše spočítány prostřednictvím kalkulátoru popsaného v předchozí kapitole.

Pro tento příklad tedy vychází jako nejlevnější řešení Firebase. Je důležité zmínit, že předpokládáme, že k překročení limitu Database Transfer nedojde dřív, než k překročení limitu DS_{fire}, což by ovlivnilo výslednou cenu Firebase.

[50]Znak „-“ znamená, že tato konfigurace není dostupná.

Tabulka 4.2: Srovnání ceny BaaS dle velikosti databáze

Max. velikost DB (GB)	0.496	1	2	4	8	10	30	40	60	100	120	240	300	480	700
Cena Parse na Heroku ($)	0	18	36	72	144	240	240	240	480	960	960	1920	3040	3040	4740
Cena Kinvey ($)	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-
Cena Firebase ($)	0	0	49	49	49	49	149	449	449	449	1449	1449	1449	-	-

Tabulka 4.3: Výpočet ceny pro jednotlivá BaaS - vzorový příklad

Měsíc	1	2	3	4	5	6	celkem
Počet uživatelů	100	500	1000	2500	5000	6000	
Velikost dabáze (GB)	0.5	2.5	5	12.5	25	30	
Cena Parse na Heroku ($)	18	72	144	240	240	240	954
Cena Kinvey ($)	0	0	0	200	200	200	600
Cena Firebase ($)	5	49	49	149	149	149	550

Kapitola 5

Využitelnost BaaS řešení a výhled do budoucna

Cílem této kapitoly je na základě zkušeností získaných při práci na předchozích kapitolách popsat, kdy je vhodné použít BaaS řešení na projektu, a pokusit se odhadnout další vývoj na tomto trhu.

5.1 Limity BaaS

Během práce s jednotlivými BaaS řešeními jsem došel k následujícím závěrům.

1. BaaS je uzavřené řešení, které poskytuje vývojářům určitou funkcionalitu výměnou za to, že se vyvíjená aplikace musí přizpůsobit tomu, jak je tato funkcionalita připravena či navržena.

   O BaaS nelze tedy hovořit jako o univerzální platformě, která poskytuje určitou funkcionalitu a zároveň zůstává otevřená do té míry, že na ni lze postavit libovolnou aplikační logiku. Toto tvrzení je postaveno zejména na zkušenostech popsaných v kapitole 3.5.1.

   To znamená, že při použití BaaS na projektu, je třeba počítat s tím, že bude třeba v některých případech funkcionalitu aplikace přizpůsobit možnostem BaaS řešení.

2. Při vývoji komplexnější aplikace se projekt při použití BaaS pravděpodobně neobejde bez dalšího podpůrného vlastního backendu pro řešení určitých funkčních scénářů, přičemž tyto dva backendy bude třeba určitým způsobem integrovat. Příkladem je např. práce se soubory, kterou dostatečně neřeší ani jeden z detailně analyzovaných BaaS, viz kapitola 3.4.4.

3. Jeden z klíčových benefitů BaaS - přístup k databázi přímo z klienta - se stává zároveň úzkým hrdlem řešení.

Úzkým hrdlem je ze dvou důvodů. Zaprvé přináší bezpečnostní rizika, která je třeba explicitně řešit technikami, které jsou pro vývojáře bez zkušeností s BaaS neobvyklé (viz kapitoly 3.5.3 a 3.5.4). Zadruhé tento přístup vede implicitně k přesunu logiky aplikace na klienta, což může být kontraproduktivní u multiplatformních aplikací, neboť vede k duplikaci aplikační logiky na straně klientů.

5.2 Kdy je na projektu vhodné použít BaaS

Následující seznam výroků byl vytvořen jako pomůcka pro rozhodnutí o použití BaaS na projektu. Výroky vychází z benefitů a rizik popsaných v kapitolách 1.4 a 1.5. Souhlas s výrokem značí podporu pro užití BaaS.

- **Nedisponuji zdroji nebo znalostmi pro vývoj vlastního backendu.** V případě, že projektový tým nemá zkušenosti s vývojem backendu a použití externího dodavatele není žádoucí / možné, pak BaaS může efektivně řešit tento problém.
- **Potřebuji pro projekt funkcionalitu, kterou je obtížné vyvinout vlastními silami a BaaS tuto funkcionalitu nabízí.** V případě, že vyvíjená aplikace disponuje specifickou potřebou a BaaS tuto potřebu řeší, může být BaaS vhodné řešení, které bude např. použito společně s vlastním backendem.
- **Vyvíjený projekt má definovanou omezenou časovou životnost.** V případě, že se jedná o krátkodobý projekt, dochází ke snížení rizika závislosti na poskytovateli BaaS (resp. tato závislost je pouze krátkodobá).
- **Vysokou prioritou je minimalizace doby dodání projektu.** Lze předpokládat, že použití BaaS zabere méně času, než vývoj vlastního backendu a jeho integrace s klienty - za předpokladu, že se podaří vybrat vhodné BaaS řešení.
- **Vysokou prioritou je minimalizace počátečních nákladů na projekt.** V případě, že je vytvářen např. proof of concept, může být použití BaaS nejlevnější a nejrychlejší způsob jak jej implementovat.
- **Projekt má jasně definovaný rozsah a funkcionalitu.** V případě, že ano, lze minimalizovat riziko, že BaaS řešení z hlediska funkcionality nepokryje požadavky projektu.

5.3 Volba vhodného BaaS řešení

Cílem této podkapitoly je poskytnout rady pro volbu konkrétního BaaS řešení v situaci, kdy již bylo rozhodnuto o tom, že BaaS jako takové může být

vhodnou volbou pro projekt. Pro volbu vhodného řešení je možné postupovat podle následujících kroků.

1. **Na základě kategorizace popsané v kapitole 2.3 vydefinovat kritéria a kategorie BaaS, které jsou pro projekt relevantní**. Praktická ukázka tohoto kroku je obsažena v tabulce 3.1.

2. **Vyhledat BaaS poskytovatele, kteří splňují definovaná kritéria.** Jako základ pro hledání může posloužit seznam poskytovatelů nacházející se v příloze D.

3. **Provést hrubou analýzu funkcionality nabízeného BaaS a srovnat ji s požavavky projektu**. Také je možné prozkoumat případové studie, které zpravidla zveřejňují BaaS poskytovatelé na svých webových stránkách. Na základě této analýzy dojde možná k dalšímu zúžení vhodných kandidátů.

4. **Analyzovat důvěryhodnost a vyspělost BaaS poskytovatelů.** Cílem tohoto kroku je minimalizovat riziko, že dojde k výběru poskytovatele, který brzo ukončí svoji činnost nebo jehož řešení nebude vyspělé (chybovost). Inspiraci pro praktický postup v tomto kroku lze najít v kapitole 3.1.

5. **Prozkoumat proveditelnost klíčových funkčních scénářů či user stories u jednotlivých BaaS**. Cílem tohoto kroku je minimalizovat riziko, že během implementační fáze projektu dojde ke zjištění, že některé klíčové aspekty projektu nejde s daným BaaS realizovat nebo je realizace značně náročná. Inspirace proto jak tuto analýzu realizovat lze najít v kapitole 3.5.

6. **Srovnat náklady na využití jednotlivých BaaS**. Této problematice se detailně věnuje kapitola 4. Dle významosti nákladů při rozhodování lze tento krok zařadit v rozhodovacím procesu dříve.

5.4 Budoucnost BaaS

Odhadnout, jakým směrem se bude trh s BaaS řešeními nadále ubírat, je velkou výzvou. Z analyzovaných zdrojů a BaaS řešení se pokusím odhadnout několik trendů, které bude možné sledovat na trhu s BaaS v následujících několika letech. Tyto trendy jsou popsané v následujících podkapitolách.

5.4.1 Vznik a další rozvoj BaaS pro Internet of Things (IoT)

IoT jako trend ovlivňuje celý IT sektor a BaaS dle mého názoru není a nebude výjimkou. Zatímco v tuto chvíli je BaaS stále chápan jako služba související především s vývojem mobilních a webových aplikací, již v době psaní této

práce se profilují BaaS řešení prezentující se jako platformy / BaaS pro Internet of things [21]. Je otázkou, zda vznikne spíš nový segment trhu nebo bude rozšířeno chápání BaaS jako takového.

5.4.2 Koncentrace univerzálních řešení na enterprise-oriented segment trhu

Univerzální BaaS, které se na trhu udrží, se budou orientovat primárně na velké podniky ze dvou důvodů. Prvním důvod je, že v tomto segmentu trhu je potenciál větší ziskovosti [22]. Druhým důvodem je můj názor, že univerzální BaaS nejsou schopny nabídnout dostatečně pružnou funkcionalitu (viz kapitola 5.1), která je zpravidla vyžadována pro aplikace orientující se na koncové zákazníky.

5.4.3 Úspěch consumer-oriented BaaS, které se zaměřují na specializované oblasti

V oblasti consumer-oriented BaaS uspějí řešení, která poskytnou funkcionalitu, která je pro freelancery a menší vývojářské firmy obtížně dostupná - tedy ji tuto subjekty nejsou schopny vyvinout vlastními silami. V době psaní této práce je to např. RT synchronizace dat, která se ovšem pomalu stává více dostupnou v různých technologiích. V budoucnu se může jednat o něco zcela jiného.

Závěr

V závěru své práce nejdříve provedu konfrontaci formálního zadání s výstupy své práce, poté se pokusím shrnout přínos této práce a nakonec nastíním možnosti dalšího výzkumu v této oblasti.

Konfrontace zadání s výstupy práce

Cílem mé diplomové práce bylo analyzovat současná řešení Backend as a Service pro vývoj mobilních a webových aplikací. Pro realizaci tohoto cíle bylo vytyčeno několik dílčích cílů. Níže popíši jakým způsobem byl naplněn každý z těchto cílů.

1. *Vymezte termín (mobile) Backend-as-a-Service*

   Vymezení tohoto pojmu a popisu významu tohoto pojmu je věnována kapitola 1 - Vymezení pojmu Backend as a Service.

2. *Analyzujte současný trh BaaS. Popište, jaká řešení jsou na trhu, jaké mají společné charakteristiky a čím se odlišují.*

   Analýze současného trhu se věnuje kapitola 2. Problematice, jaké typy řešení na trhu existují, a jejich společným charakteristikám se pak věnuje zejména podkapitola 2.3 - Kategorizace BaaS řešení. Analýza proběhla na základě zkoumání čtrnácti BaaS řešení, jejichž seznam je uveden v příloze D.

3. *Po dohodě s vedoucím práce vyberte 3 - 5 konkrétních BaaS řešení. Na konkrétní aplikaci (či aplikacích), kterou implementujete s použitím každého z vybraných BaaS řešení, analyzujte silné a slabé stránky těchto řešení a popište pro jaké typy projektů jsou vhodné.*

   Byly vybrány tři BaaS řešení k detailní analýze. Způsob jejich výběru je popsán v kapitole 3.1. S každým z těchto BaaS řešení byla implementována aplikace Cost Tracker, která je popsána v kapitole 3.6. Silné a

slabé stránky řešení byly dále analyzovány prostřednictvím detailního srovnání funkcionality těchto řešení a testováním vybraných problémů, což je popsáno v kapitolách 3.4 a 3.5. Celkové zhodnocení těchto BaaS a jejich vhodnost pro různé projekty je pak popsána v kapitole 3.7.

4. *Na vhodných příkladech demonstrujte a srovnejte ekonomické náklady využití vybraných BaaS řešení.*

   Srovnání ekonomických nákladů vybraných BaaS řešení a návodu jak obecně odhadnout cenu BaaS řešení se věnuje kapitola 4.

5. *Na základě výstupů z předchozích kroků: (1) Se pokuste o zobecnění a zhodnocení, kdy je vhodné na projektu vývoje mobilní nebo webové aplikace použít BaaS řešení. (2) Vyslovte názor jakým směrem se bude trh s BaaS ubírat.*

   O zhodnocení, kdy je vhodné použít na projektu BaaS, a odhadnutí budoucnosti BaaS se na základě všech zkušeností nabytých na této práci snažím v závěrečné kapitole 5.

Přínos práce

Věřím, že tato práce splnila svůj hlavní cíl popsaný v úvodu práce. Tedy, že na základě přečtení této práce budou vývojáři a manažeři schopni kvalifikovaně rozhodnout o (ne)využití Backend as a Service řešení na projektu webové nebo mobilní aplikace.

Další výzkum

Téma Backend as a Service jako takové je velmi široké a v době psaní této práce neexistovalo mnoho prací zabývajících se tímto tématem. Během psaní této práce jsem narazil na následující témata, která by dle mého názoru stála za hlubší zkoumání a která s tématem mé práce určitým způsobem souvisí.

- Srovnání BaaS a MEAP.
- Analýza a srovnání různých MEAP řešení.
- Analýza s srovnání BaaS řešení zaměřených na enterprise sektor.
- Srovnání řešení, které se zaměřují na real-time komunikaci (Firebase, Flybase, Quikblox, SynergyKit).

Literatura

[1] Shearer, D.: A history of the dynamic web. 2007, [cit. 2016-04-29]. Dostupné z: http://royal.pingdom.com/2007/12/07/a-history-of-the-dynamic-web/

[2] Lane, K.: Overview Of The Backend as a Service (BaaS) Space. 2013. Dostupné z: http://www.kinvey.com/baas-whitepaper

[3] Herma, T.: *Platformě nezávislé aplikační rozhraní na architektuře REST*. Diplomová práce, Vysoká škola ekonomická v Praze, 2014. Dostupné z: https://www.vse.cz/vskp/41109_platforme_nezavisle_aplikacni_rozhrani_na_architekture_rest

[4] Linhart, T.: *Backendové řešení služeb pro vývojáře mobilních aplikací*. Diplomová práce, České vysoké učení technické v Praze, 2013. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/linhato2_2013dipl.pdf

[5] Laanstra, J.: *Offline Data and Synchronization for a Mobile Backend as a Service system*. Diplomová práce, Delft University of Technology, 2014. Dostupné z: http://repository.tudelft.nl/view/ir/uuid%3A16a1d98d-2a2e-44bc-9051-35c1942f9e9b/

[6] Tucker, D.: Understanding Mobile Back End As A Service. 2014, [cit. 2016-28-01]. Dostupné z: https://www.smashingmagazine.com/2014/12/understanding-mobile-back-end-as-a-service/

[7] Marshall, R.; Baker, V. L.; Wong, J.: Market Guide for Cloud Mobile Back-End Services. 2015, [cit. 2016-03-03]. Dostupné z: https://www1.good.com/forms/gartner-cloud-mbaas.html

[8] Heller, M.: MBaaS shoot-out: 5 clouds for building mobile apps. [cit. 03-12-2016]. Dostupné z: http://www.infoworld.com/article/2842791/application-development/mbaas-shoot-out-5-cloud-platforms-for-building-mobile-apps.html

[9] Nižaradze, J.; Mistrik, T.: Vývoj mobilních aplikací na platformách Backend as a Service. 2015, [cit. 2015-09-25]. Dostupné z: https://www.zdrojak.cz/clanky/vyvoj-mobilnich-aplikaci-na-platformach-backend-service/

[10] Making sense of the cloud: SaaS, PaaS and IaaS explained. 2014, [cit. 2016-02-22]. Dostupné z: http://www.clouderpc.com/making-sense-of-the-cloud-saas-paas-and-iaas-explained/

[11] Aaron, T.: FatFractal ups the ante in backend-as-a-service market. 2012, [cit. 2016-02-22]. Dostupné z: http://www.techgoondu.com/2012/09/30/fatfractal-ups-the-ante-in-backend-as-a-service/

[12] BAASBOX: What is backend as a service? [cit. 2015-25-09]. Dostupné z: http://www.baasbox.com/what-is-backend-as-a-service/

[13] Sinha, N.; Karim, R.; Gupta, M.: Simplifying Web Programming. 2015: s. 80–89, doi:10.1145/2723742.2723750. Dostupné z: http://doi.acm.org/10.1145/2723742.2723750

[14] Sridhar, S.: Mobile Cloud Backend as a Service Ecosystem Map - All roads lead to BaaS. [cit. 03-11-2016]. Dostupné z: http://www.kinvey.com/blog/65/mobile-cloud-backend-as-a-service-ecosystem-map-8211-all-roads-lead-to-baas

[15] Novet, J.: PayPal closing down backend service StackMob mere months after buying it. 2014, [cit. 2016-04-23]. Dostupné z: http://venturebeat.com/2014/02/12/paypal-closing-down-backend-service-stackmob-months-after-buying-it/

[16] Rice, K.: 4 Major Differences between BaaS and MEAP. 2013, [cit. 2016-03-06]. Dostupné z: http://www.kinvey.com/blog/3380/4-major-differences-between-baas-and-meap

[17] About Firebase. [cit. 03-28-2016]. Dostupné z: https://www.firebase.com/about.html

[18] Yarmosh, K.: Parse Server: Missing Features, Workarounds, and What to Do About Migration. 2016, [cit. 04-14-2016]. Dostupné z: http://savvyapps.com/blog/parse-server-missing-features-solutions-migration

[19] Williams, A.: Kinvey Raises $5 Million For Mobile And Web App Backend As A Service. 2012, [cit. 2016-04-24]. Dostupné z: http://techcrunch.com/2012/07/11/kinvey-raises-5-million-for-mobile-and-web-app-back-end-as-a-service/

[20] Optimizing Node.js Application Concurrency. 2015, [cit. 04-19-2016]. Dostupné z: https://devcenter.heroku.com/articles/node-concurrency

[21] Kelly, W.: MBaaS and its role in the future of IoT infrastructure. 2015, [cit. 2016-04-23]. Dostupné z: http://www.techrepublic.com/article/mbaas-and-its-role-in-the-future-of-iot-infrastructure/

[22] Lane, K.: What Is The Future Of Backend As A Service (BaaS)? 2014, [cit. 2016-04-23]. Dostupné z: http://apievangelist.com/2014/02/14/what-is-the-future-of-backend-as-a-service-baas/

Příloha **A**

Seznam použitých zkratek

API Application Programing Interface

BaaS Backend as a Service

CLI Command Line Interface

CRUD Create, Read, Update, Delete (typické databázové operace)

DBaaS Database as a Service

IaaS Infrastructure as a Service

mBaaS Mobile Backend as a Service

MEAP Mobile Enterprise Application Platform

ORM Object Relational Mapping

PaaS Platform as a Service

Q & A Question and Answer

REST Representational State Transfer

RESTful dodržující principy REST

RT Realtime

SaaS Software as a Service

SDK Software development kit

SMB Small and Medium Business

SOAP Simple Object Access Protocol

SPA Single Page Application

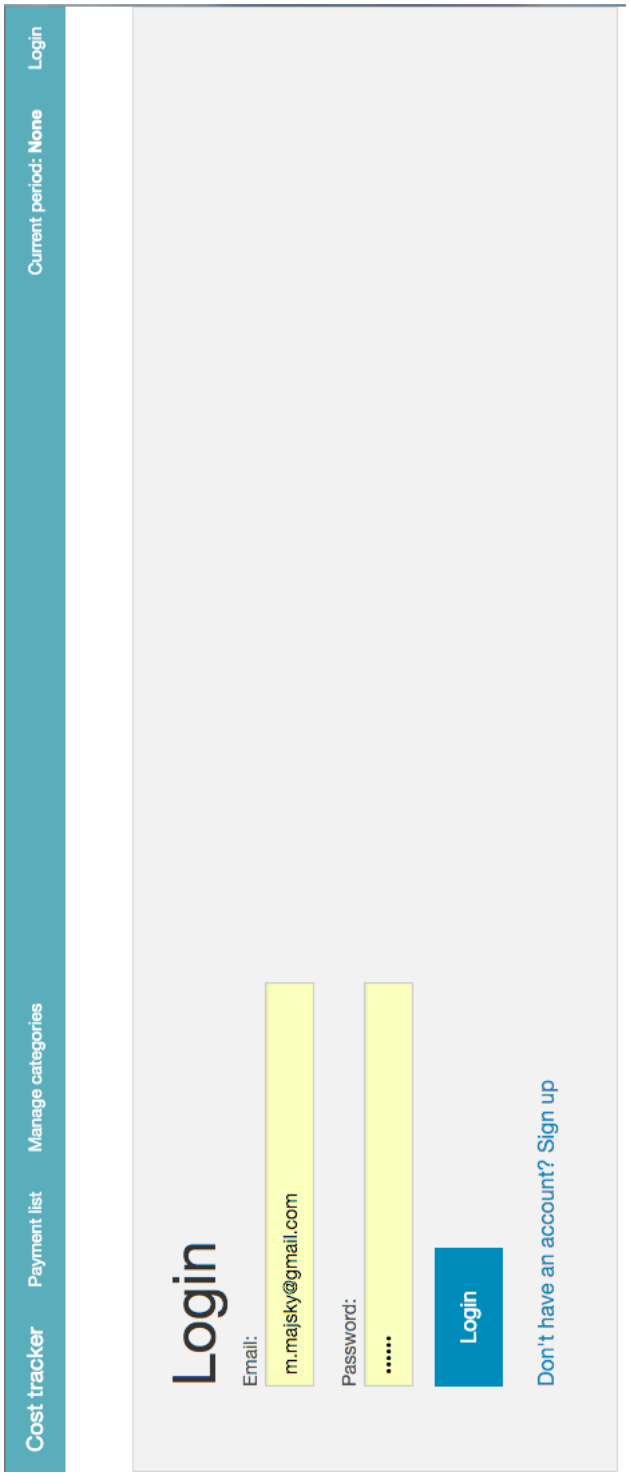
Příloha **B**

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
aplikace_cost_tracker
    deploy ..... verze aplikací Cost Tracker připravené k nasazení
    src ..... zdrojové kódy aplikace Cost Tracker
kalkulator_nakladu ..... podklady pro vytvoření kalkulátoru nákladů
prehled_baas.pdf .přehled poskytovatelů BaaS (příloha D v plné verzi)
thesis_src ..... zdrojová forma práce ve formátu LaTeX
thesis.pdf ..... text práce ve formátu PDF
```

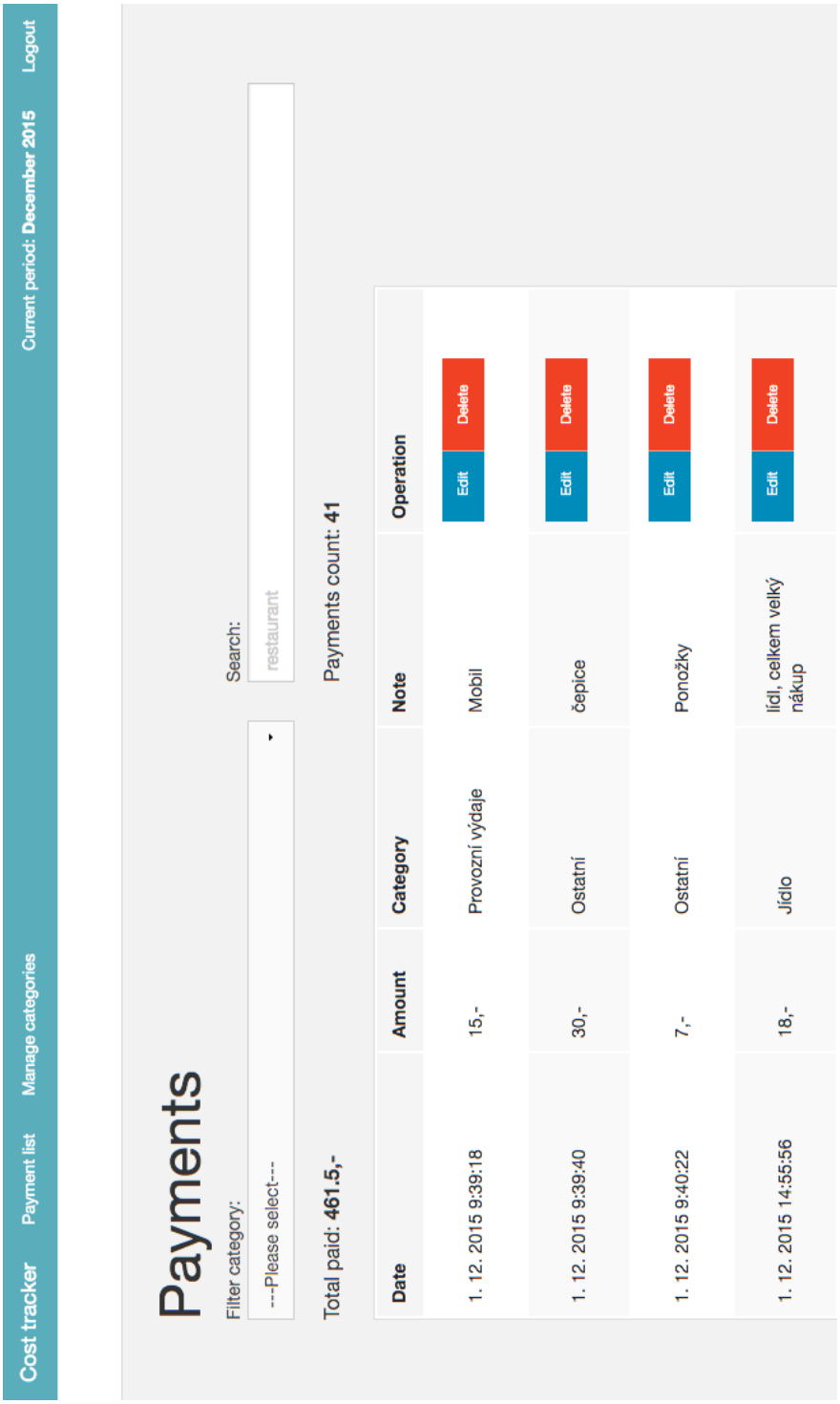
Příloha C

Uživatelské rozhraní aplikace Cost Tracker



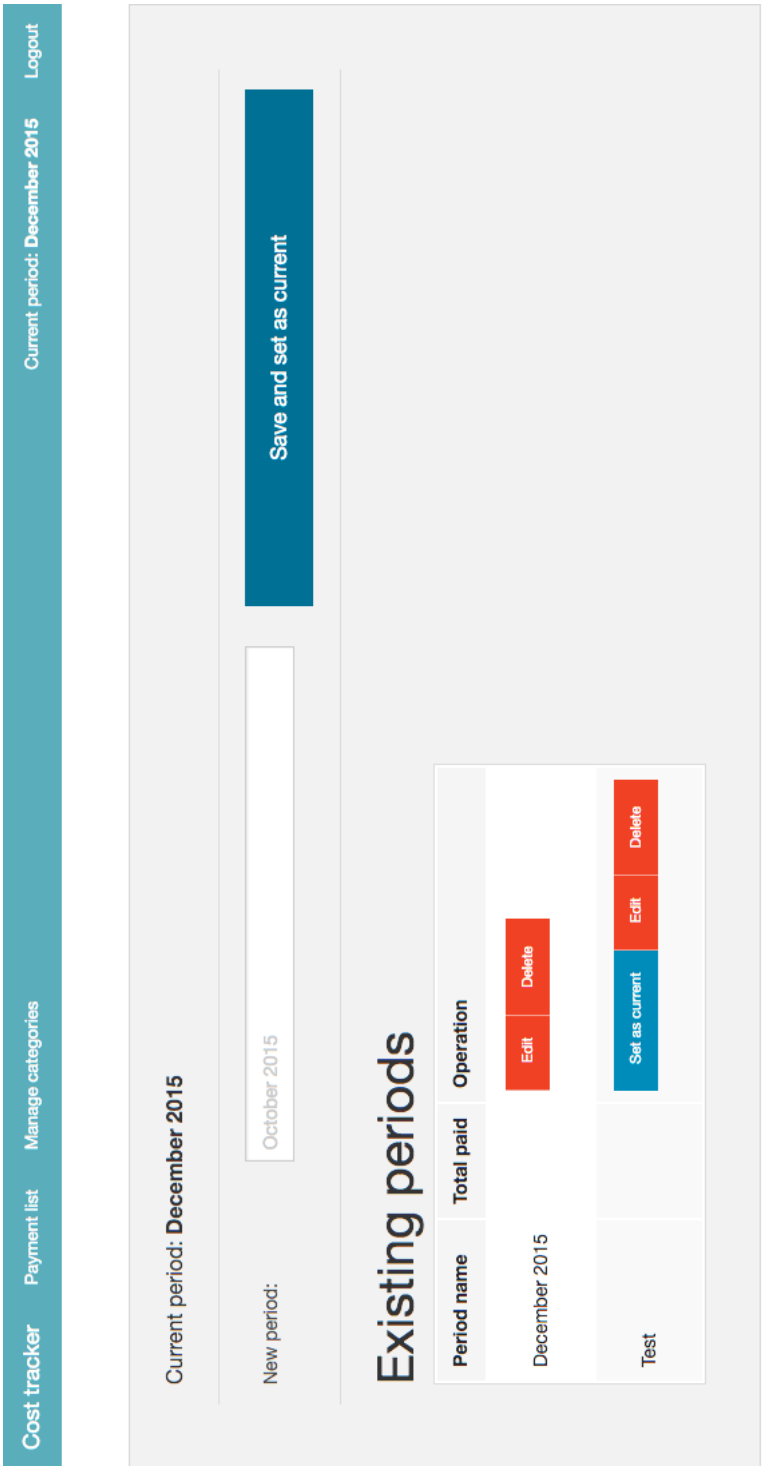
Obrázek C.1: Přihlášení

Obrázek C.2: „Dashboard“ - přidání platby a přehled výdajů za období



Obrázek C.3: Přehled a správa plateb

Obrázek C.4: Přehled a správa kategorií



Obrázek C.5: Přehled a správa období

Příloha D

Přehled poskytovatelů BaaS

				Způsob hostování						SDK			
Název	URL	Cílový segment	Oblast zaměření	Managed - multi tenant	Managed - dedicated	On premise	Self hosted	Dle režimu účtování	REST API	JavaScript	Android	iOS	Win phone /.NET
Firebase	https://www.firebase.com/	Consumer	Specializovaný	✓	☐	☐	☐	Tarify	✓	✓	✓	✓	☐
Backendless	https://backendless.com/	Consumer / Enterprise	Univerzální	✓	✓	✓	☐	Platba za navýšení limitu	✓	✓	✓	✓	✓
SynergyKit	https://synergykit.com	Consumer	Univerzální	✓	☐	☐	☐	Tarify	✓	✓	✓	✓	☐
Kinvey	http://kinvey.com	Enterprise / Consumer	Univerzální	✓	✓	☐	☐	Tarify	✓	✓	✓	✓	☐
Quickblox	http://quickblox.com/	Consumer / Enterprise	Specializovaný	✓	✓	✓	☐	Tarify	✓	✓	✓	✓	☐
Kumulos	https://www.kumulos.com/	Consumer	Univerzální	✓	☐	☐	☐	Platba za navýšení limitu	✓	☐	✓	✓	☐
BlueMix	http://www.ibm.com/cloud-computing/bluemix/mobile/	Enterprise	Univerzální	☐	☐	☐	☐	Platba za navýšení limitu	☐	☐	✓	✓	☐
AWS Mobile Hub	https://aws.amazon.com/mobile/	Consumer / Enterprise	Univerzální	✓	☐	☐	☐	Pay as you go	☐	☐	✓	✓	☐
BaaSBox	http://www.baasbox.com/	Consumer	Univerzální	✓	☐	☐	✓	Tarify	✓	✓	✓	✓	☐
Kii	https://en.kii.com/	Enterprise	Univerzální	✓	✓	✓	☐	Tarify	✓	✓	✓	✓	☐
AnyPresence	http://www.anypresence.com/	Enterprise	Univerzální	✓	✓	✓	☐	Neveřejné	✓	✓	✓	✓	☐
ApiOmat	https://apiomat.com/	Enterprise	Univerzální	☐	☐	☐	☐	Neveřejné	✓	✓	✓	✓	✓
Apache Usergrid	http://usergrid.apache.org/	Consumer	Univerzální	☐	☐	☐	✓	Zdarma	✓	✓	✓	✓	✓
Parse	https://parse.com/	Consumer	Univerzální	☐	☐	☐	✓	Zdarma	✓	✓	✓	✓	✓

	Funkcionalita															
Název	Data storage	External Data sources Integration	User management	Social integration	SSO / LDAP / Enterprise integrations	Messaging - Push notifications	Messaging - Email	Realtime Data Sync	GeoLocation	Web Hosting	Cloud code	Media management	File management	Caching	Offline & Sync	Analytics
Firebase	✓	☐	✓	✓	☐	☐	☐	✓	✓	✓	☐	☐	☐	☐	✓	☐
Backendless	✓	☐	✓	✓	☐	✓	✓	☐	✓	✓	✓	✓	✓	✓	☐	☐
SynergyKit	✓	☐	✓	☐	☐	✓	✓	✓	☐	☐	✓	☐	✓	✓	☐	☐
Kinvey	✓	✓	✓	☐	✓	✓	✓	☐	✓	☐	✓	✓	✓	✓	✓	☐
Quickblox	✓	☐	✓	✓	☐	✓	☐	✓	✓	☐	☐	✓	☐	✓	☐	☐
Kumulos	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
BlueMix	✓	☐	✓	✓	☐	✓	☐	☐	☐	☐	✓	☐	✓	☐	☐	☐
AWS Mobile Hub	✓	☐	✓	✓	☐	✓	☐	☐	☐	☐	✓	☐	✓	☐	☐	✓
BaaSBox	✓	☐	✓	✓	☐	✓	☐	☐	☐	☐	✓	☐	✓	☐	☐	☐
Kii	✓	☐	✓	✓	☐	✓	☐	☐	☐	☐	✓	☐	☐	☐	☐	✓
AnyPresence	✓	✓	✓	✓	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ApiOmat	✓	☐	✓	☐	☐	✓	☐	☐	✓	☐	✓	☐	☐	☐	✓	☐
Apache Usergrid	✓	☐	✓	✓	☐	✓	☐	☐	✓	☐	☐	☐	✓	☐	☐	☐
Parse	✓	☐	✓	✓	☐	✓	☐	✓	✓	☐	✓	☐	✓	✓	✓	✓

Obrázek D.1: Přehled poskytovatelů BaaS