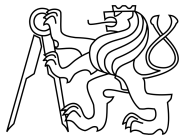
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Vývoj mobilních aplikací pomocí nenativního frameworku Meteor.js
Student:	Michal Artazov
Vedoucí:	Ing. Josef Gattermayer
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem bakalářské práce je zjistit, zda je framework Meteor.js vhodným řešením pro vývojáře mobilních aplikací jako náhrada za nativní vývoj. Úkolem je implementovat prototypovou aplikaci podobnou aplikaci Tinder s pomocí Meteor.js s důrazem na detaily uživatelského rozhraní.

1. Analyzujte rozdíly ve vývoji pomocí nativních řešení a pomocí Meteor.js.
2. Popište vývoj pomocí frameworku Meteor.js.
3. Implementujte prototyp mobilní aplikace Tinder pro Android a iOS pomocí Meteor.js, cílem není implementovat celou aplikaci, ale pouze vybrané obrazovky, ale zato jako co nejpřesnější kopii původního UX.
4. Implementujte REST API na serveru, se kterým bude aplikace komunikovat.
5. Navrhněte a implementujte webové administrační rozhraní pro aplikaci.
6. Analyzujte, na jaké limity frameworku Meteor.js jste narazili během vývoje.
7. Analyzujte rizika a ekonomické dopady použití Meteor.js při vývoji mobilních aplikací místo nativního řešení.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 23. prosince 2015

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

Bakalářská práce

Vývoj mobilních aplikací pomocí nenativního frameworku Meteor.js

Michal Artazov

Vedoucí práce: Ing. Josef Gattermayer

16. května 2016

Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Josefu Gattermayerovi, za podporu a cenné rady. Také bych rád poděkoval rodině za podporu při studiu.

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 16. května 2016 .....

Odkaz na tuto práci

Artazov, Michal. *Vývoj mobilních aplikací pomocí nenativního frameworku Meteor.js*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá analýzou hybridního frameworku pro vývoj mobilních aplikací Meteor.js. V této práci jsou analyzovány rozdíly mezi implementací mobilních aplikací s pomocí nativních řešení a s pomocí frameworku Meteor.js. Součástí práce je implementace prototypu populární mobilní aplikace Tinder pro operační systémy Android a iOS s pomocí frameworku Meteor.js. Na základě poznatků, získaných při implementaci pak jsou analyzovány přínosy, limity, rizika a ekonomické dopady použití frameworku Meteor.js pro vývoj mobilních aplikací místo nativních řešení.

Klíčová slova mobilní aplikace, Meteor.js, nativní řešení, hybridní framework, Android, iOS

Abstract

The objective of this thesis is to analyse the hybrid framework Meteor.js for development of mobile applications. The analysis focuses on the differences between using native solutions or Meteor.js for mobile application development. As part of the thesis, a working prototype of the popular mobile application Tinder is implemented for operating systems Android and iOS using Meteor.js framework. Finally, based on findings, acquired during the development stage, the benefits, risks and economical impact of using Meteor.js framework for mobile application development are analysed.

Keywords mobile application, Meteor.js, native solution, hybrid framework, Android, iOS

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Mobilní aplikace jsou od svého vzniku každým rokem čím dál tím víc populární. Jejich počet, ale i počet jejich uživatelů rapidně roste. Jak roste trh s mobilními aplikacemi, tak roste i počet vývojářů, kteří se zabývají právě vývojem mobilních aplikací.

Na poli mobilních aplikací jsou dominantní zejména dva operační systémy, pro které jsou tyto aplikace vyvíjeny. První z nich – operační systém Android od společnosti Google – sází na otevřený svět open-source a komunitu vývojářů. Druhý – iOS od společnosti Apple — je proprietární software, který sází na uzavřený systém a přísnou kontrolu kvality distribuovaných aplikací.

Jelikož si tyto operační systémy navzájem konkurují, každý z nich přichází s vlastním nativním řešením implementace aplikací a tato řešení nejsou navzájem kompatibilní. Pokud tedy vývojáři vytvářejí aplikaci pro oba systémy, jsou nuceni vyvíjet ji dvakrát – jednou pro každý systém s pomocí nativních nástrojů každého z těchto systémů.

Kromě toho začaly časem vznikat nenativní, tzv. hybridní frameworky, které nabízejí možnost z jediného zdrojového kódu vytvořit aplikaci pro oba systémy. Tento koncept představuje velký potenciál, jak ušetřit vývojářům čas i peníze na vývoj mobilních aplikací. Stojí proto za to se na tyto technologie podívat blíž.

Tato práce se zabývá srovnáním nativních řešení pro vývoj mobilních aplikací pro operační systémy Android a iOS a jednoho z hybridních frameworků – Meteor.js.

Vize této práce vznikla, když jsem pracoval v jedné nejmenované firmě. Jednalo se o mladou firmu s omezeným rozpočtem, která potřebovala implementovat mobilní aplikaci. Kvůli omezenému rozpočtu bylo rozhodnuto, že se aplikace bude vyvíjet právě s pomocí frameworku Meteor.js. Tehdy mě to zaujalo a zajímalo mě, jestli je Meteor.js skutečně vhodným řešením pro vývoj mobilních aplikací, které může ušetřit čas i peníze a přitom zachovat kvalitu výsledného produktu. Tato práce se zaměřuje právě na to.

Kapitola 1

Cíl práce

Cílem této práce je zjistit, zda je framework Meteor.js vhodným řešením pro vývojáře mobilních aplikací jako náhrada za nativní vývoj.

V práci budou nejdříve analyzovány hlavní rozdíly mezi implementací mobilních aplikací s pomocí nativních řešení a s pomocí frameworku Meteor.js. Dále bude popsán vývoj s pomocí frameworku Meteor.js. Poté bude následovat návrh implementace a samotná implementace prototypu mobilní aplikace. V implementaci bude kladen důraz zejména na zpracování uživatelského rozhraní. Nakonec budou na základě poznatků, získaných při implementaci, analyzovány limity frameworku Meteor.js a rizika a ekonomické dopady jeho použití pro vývoj mobilních aplikací místo nativních řešení.

Pro účely této práce byla vybrána populární mobilní aplikaci Tinder, protože má velice dobře zpracované uživatelské rozhraní, které je velice intuitivní, rychlé a přirozeně reaguje na dotek prstu.

Takové uživatelské rozhraní představuje výzvu pro Meteor.js. Obsahuje mnoho drobných detailů, u kterých není jisté, jestli je bude vůbec možné s pomocí Meteor.js implementovat, případně, zdali bude výsledné řešení dostatečně efektivně využívat dostupných prostředků, aby se aplikace dala používat v praxi.

Kapitola 2

Použitá terminologie

Kapitola vysvětluje termíny, které jsou používány v kontextu vyvíjené aplikace a budou se vyskytovat v textu následujících kapitol.

Like

Like je jedna z možností, které má uživatel na výběr, když v aplikaci hlasuje, zdali považuje danou osobu za atraktivní. V překladu z angličtiny znamená slovo like doslova „líbí se“, tudíž tímto typem hlasu uživatel vyjadřuje, že danou osobu považuje za atraktivní.

Nope

Nope je druhá možnost, kterou má uživatel na výběr při hlasování. V překladu znamená „ne“, tedy je to opak Like a uživatel tímto typem hlasu vyjadřuje, že danou osobu nepovažuje za atraktivní.

Super like

Super like je speciální varianta Like, kterou uživatel vyjadřuje, že danou osobu považuje za obzvlášť atraktivní.

Kapitola 3

Analýza

3.1 Rozdíly ve vývoji pomocí nativních řešení a pomocí Meteor.js

3.1.1 Vývoj nativních aplikací pro Android

Text této sekce vychází z článku na webu TutsPlus.com[1].

Nativní mobilní aplikace pro Operační systém (OS) Android jsou vyvíjeny s pomocí programovacího jazyka Java. Většina uživatelů k vývoji používá Integrated Development Environment (IDE) Android Studio, které vyvíjí a poskytuje zadarmo společnost Google.

Aktuální verze Android Studio poskytuje mnoho nástrojů, usnadňujících vývoj mobilních aplikací v mnoha ohledech, např. textový editor pro psaní zdrojového kódu, vizuální editor pro tvorbu uživatelských rozhraní, správu vektorových a obrázkových souborů a další.

Android Studio také obsahuje realistický emulátor. Vzhledem k tomu, že všechny verze OS Android jsou volně dostupné, lze si ve virtuálním prostředí spustit jakoukoliv verzi OS Android a testovat na ní vyvíjenou aplikaci, i když vývojář nemá k dispozici fyzické zařízení s požadovanou verzí OS.

Vývojář mobilních aplikací pro OS Android běžně řeší mnoho různých problémů, specifických právě pro tuto platformu. Je nutné, aby aplikace podporovala starší verze OS, protože výrobci mobilních zařízení zřídkakdy nabízejí možnost aktualizace systému na novější verzi. Aplikace musí také podporovat různé velikosti a rozlišení obrazovek, protože v dnešní době existuje mnoho různých zařízení, používajících OS Android, nejen mobilní telefony. Dnes jsou to např. i tablety nebo televize.

Jakmile je aplikace hotová, je možné jí publikovat online. Nejčastějším místem pro to je služba Google Play. Autor aplikace zkompiluje aplikaci a výsledný soubor nahraje na Google Play.

3.1.2 Vývoj nativních aplikací pro iOS

Text této sekce vychází z dokumentace iOS[2].

iOS je operační systém, který běží na dotykových zařízeních iPhone, iPad a iPod od společnosti Apple.

K vývoji nativních aplikací pro iOS s pomocí moderních nástrojů je potřeba počítač Mac s operačním systémem OS X ve verzi 10.10 nebo novější a aktuální verze Xcode.

Xcode je IDE od společnosti Apple, obsahující různé nástroje pro návrh, vývoj a ladění aplikací. Také obsahuje iOS SDK, což je sada nástrojů, kompilátorů a frameworků pro vývoj aplikací konkrétně pro iOS.

Aplikace pro iOS jsou vyvíjeny s pomocí programovacího jazyka Swift. Xcode obsahuje i emulátor, kde lze emulovat všechna aktuálně dostupná iOS zařízení a testovat na nich vyvíjenou aplikaci.

Společnost Apple provozuje službu App Store, kde je možné publikovat hotové aplikace. Autor nahraje zkompilovanou aplikaci do App Store a po ověření a schválení je aplikace dostupná ke stažení všem uživatelům podporovaných zařízení.[3]

3.1.3 Vývoj hybridních mobilních aplikací s Meteor.js

Text této sekce vychází z dokumentace Meteor.js[4].

Meteor.js je platforma pro vývoj moderních webových a mobilních aplikací s pomocí programovacího jazyka Javascript. Obsahuje sadu klíčových technologií pro vývoj aplikací, sestavovací nástroj a sadu balíčků poskytovaných Node.js a komunitou vývojářů používajících Javascript.

Meteor.js umožňuje vyvíjet vše v jednom programovacím jazyce – Javascriptu – pro všechna prostředí – aplikační server, webový prohlížeč i mobilní zařízení.

3.2 Existující mobilní aplikace, implementované s pomocí Meteor.js

Kapitola obsahuje příklady existujících mobilních aplikací, které jsou implementovány s pomocí Meteor.js.

3.2.1 Down To Chill

Mobilní aplikace Down To Chill umožňuje uživateli vidět, kteří z jeho přátel mají aktuálně chuť odpočívat. Vybraným uživatelům lze posílat soukromé zprávy, zakládat s nimi skupinové konverzace a plánovat společné aktivity. Dále uživatel vidí všechny plánované aktivity, kterých se účastní a může do nich zvát další uživatele.[5]

3.2.2 Verso Campus

Mobilní aplikace Verso Campus slouží jako doplněk ke standardní výuce ve škole. Poskytuje nástroje, pomáhající vzbudit ve studentech aktivitu a originální myšlení. Podporuje komunikaci a spolupráci mezi studenty, kritické myšlení, kreativitu, osobnost, atd. Také poskytuje platformu, kde mohou učitelé sdílet své poznatky, rady a tipy.[6]

3.3 Funkční požadavky

Kapitola obsahuje seznam vybraných funkčních požadavků, které musí výsledný prototyp mobilní aplikace a webové administrační rozhraní splňovat.

3.3.1 Mobilní aplikace

- Uživatel může aplikaci používat pouze, pokud je zaregistrován a přihlášen.
- Registrace i přihlášení uživatele budou realizovány ověřením uživatele ve webové službě Facebook.
- Uživatel bude moct dávat hlasy dalším uživatelům. Tímto hlasem vyjadřuje, zdali mu daná osoba připadá atraktivní nebo ne. Uživatel bude mít na výběr tři možnosti – Like, Nope a Super like.
- Uživateli se bude zobrazovat fronta uživatelů, kterým může dávat hlasy. Vždy může dávat hlas pouze uživateli, který se nachází na začátku fronty. Jakmile tomuto uživateli dá hlas, je z fronty odstraněn a na začátek fronty se přesune další uživatel.
- Uživatel může kdykoliv zrušit svůj poslední hlas, čímž vrátí uživatele, kterého se hlas týkal, zpět na začátek fronty a opětovně pro něj hlasuje.
- V případě, že uživatel dá jinému uživateli hlas typu Like nebo Super like a zároveň v minulosti od tohoto uživatele také dostal hlas typu Like nebo Super like, dochází k tzv. shodě a uživatel je o tom informován.
- Uživatel si může volitelně zobrazit podrobnější informace o uživateli, který se právě nachází na začátku fronty.

3.3.2 Webové administrační rozhraní

- Uživatel se bude do administračního rozhraní přihlašovat pod stejným účtem, jako se přihlašuje do mobilní aplikace. Aby měl do rozhraní přístup, musí jeho účet mít práva administrátora.

- Rozhraní bude zobrazovat seznam všech uživatelů v systému a informace o nich.
- Přihlášený uživatel bude moct kteréhokoliv uživatele smazat ze systému.
- Přihlášený uživatel bude mot kterémukoliv uživateli přidat nebo odebrat práva administrátora.

3.4 Nefunkční požadavky

3.4.1 Mobilní aplikace

- Aplikace bude dostupná jako mobilní aplikace pro operační systémy Android a iOS.
- Aplikace bude podporovat český jazyk.
- Aplikace bude podporovat responzivní design.
- Aplikace bude získávat informace o uživatelích z webové služby Facebook.
- Aplikace bude podporovat dotykové rozhraní.

3.4.2 Webové administrační rozhraní

- Rozhraní bude dostupné jako webová aplikace.
- Rozhraní bude podporovat český jazyk.
- Rozhraní bude podporovat responzivní design.

Kapitola 4

Architektura Meteor.js

Text této kapitoly vychází z dokumentace Meteor.js.[4]

4.1 Organizace zdrojového kódu

Jak už bylo zmíněno výše, aplikace, postavené na platformě Meteor.js se píší s pomocí programovacího jazyka Javascript.

Různé části zdrojového kódu mohou být spuštěny v různých prostředích. Některé části běží v klientské části, tedy v prohlížeči nebo mobilní aplikaci. Jiné části běží na serveru v Node.js kontejneru. A pak jsou tu části kódu, které běží v obou prostředích.

Od verze 1.3 Meteor.js plně podporuje EcmaScript 2015 moduly. Tento standard je náhradou za starší standardy CommonJS a Asynchronous Module Definition (AMD).

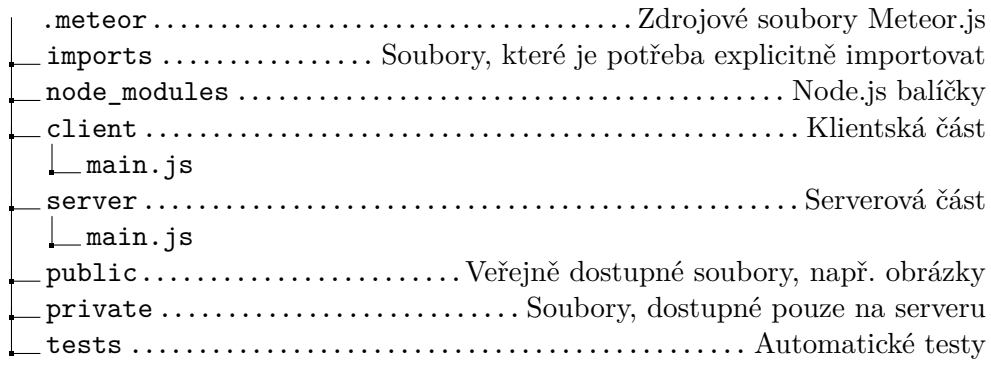
Meteor.js má určitá základní pravidla (tzv. „eager evaluation or loading“), dle kterých načítá soubory při spuštění aplikace. Soubory, které se nachází v adresáři `server/` budou k dispozici pouze na serveru. Stejně tak soubory v adresáři `client/` budou k dispozici pouze v klientské části aplikace. Existuje několik dalších speciálních adresářů, které mají speciální význam, viz. 4.1.

Soubory, umístěné mimo výše uvedené adresáře, sou načítány jak na serveru, tak v klientské části aplikace. Pro tento případ poskytuje Meteor.js proměnné `Meteor.isClient` a `Meteor.isServer`, pomocí kterých může vývojář upravit chování podle prostředí, ve kterém je daný zdrojový kód spuštěn.

4.2 Datová vrstva

4.2.1 Databáze

V Meteor.js se jako databázové úložiště typicky používá MongoDB. Souboru dat, které spolu souvisí, se v MongoDB říká kolekce. Meteor.js poskytuje třídu `Meteor.Collection`, která slouží jako abstrakce těchto kolekcí a vývojář přes



Obrázek 4.1: Základní adresářová struktura

ní přistupuje k jednotlivým kolekcím. Tato třída slouží jako primární mechanismus pro persistenci dat v aplikaci.

S kolekcemi v Meteor.js lze pracovat jak na serveru, tak v klientské části aplikace.

Když vytvoříme instanci třídy `Meteor.Collection` na serveru, vytváříme zároveň kolekci v databázi a tato instance slouží jako asynchronní rozhraní pro tuto kolekci.

Instanci třídy `Meteor.Collection` lze vytvořit i v klientské části, ale má to jiný efekt. Z klientské aplikace se nelze přímo připojit k databázi. Místo toho reprezentuje kolekce v klientské části cache databáze. Toho je dosaženo s pomocí knihovny Minimongo. Knihovna Minimongo poskytuje datovou strukturu, která je uložena v paměti a implementuje stejné rozhraní jako MongoDB.

4.2.2 Tok dat mezi serverem a klientskou aplikací

V klasických webových aplikacích spolu klientská aplikace a server komunikují dle protokolu Hypertext Transfer Protocol (HTTP), tzn. klientská aplikace posílá požadavky na server a dostává zpět odpověď v podobě HyperText Markup Language (HTML) kódu nebo dat ve formátu JavaScript Object Notation (JSON). V takových aplikacích není možnost, jak automaticky odeslat data ze serveru do klientské aplikace, když dojde k nějaké události, bez toho, aby klientská aplikace nejdříve odeslala požadavek na server.

Meteor.js je postaven na protokolu Distributed Data Protocol (DDP), což umožňuje tok dat v obou směrech. Není třeba vytvářet žádné serverové rozhraní, přes které by klientská aplikace komunikovala se serverem. Místo toho vývojář definuje tzv. publikace (publications), které automaticky odesílají data ze serveru do klientské aplikace.

Publikace je rozhraní, které odesílá určitou podmnožinu dat klientské aplikaci. Klientská aplikace spustí tzv. předplacení (subscription) určité publikace a obdrží aktuální data, která daná publikace nabízí. Tato data jsou dále automaticky aktualizována, když dojde k jejich změně na serveru.

4.2.3 Manipulace dat

Aby mohla klientská aplikace nejen číst, ale i upravovat data na serveru, poskytuje Meteor.js tzv. metody (methods).

Metody jsou systémem vzdáleného volání procedur (Remote Procedure Call) v Meteor.js a umožňují zpracovávat uživatelské akce a ukládat jejich výsledky na serveru. V praxi je metoda funkce, definovaná vývojářem. Tato funkce je pak spuštěna na serveru s parametry, odeslanými z klientské aplikace. Tato funkce provede nějaké změny v databázi a vrátí nějaký výsledek. Zároveň je ta samá funkce spuštěna i v klientské aplikaci a její výsledek uložen do paměti do té doby, než je dokončena komunikace se serverem. Této technice se říká „optimistické uživatelské rozhraní“ (optimistic UI) a používá se, aby se zdánlivě urychlil běh aplikace.

4.3 Uživatelské rozhraní

Meteor.js oficiálně podporuje tři knihovny pro implementaci uživatelského rozhraní – Blaze, React a Angular.

4.3.1 Blaze

Blaze je základní knihovnou, kterou Meteor.js používá k vykreslování uživatelského rozhraní.

Vývojář píše šablony jednotlivých komponent s pomocí speciálního jazyka Spacebars, což je variace jazyka Handlebars, upravená pro potřeby Meteor.js. Šablony jsou poté zkompilovány do kódu v jazyce Javascript a vykresleny s pomocí knihovny Blaze.

4.3.2 React

React je knihovna pro tvorbu uživatelských rozhraní v jazyce Javascript. Vytvořila jí společnost Facebook.[7]

React dělí uživatelské rozhraní na tzv. komponenty. Tyto komponenty jsou psány přímo v jazyce Javascript. K tomu React navíc přidává volitelné rozšíření JSX, umožňující v kódu používat syntaxi podobnou jazyku HTML.

Největší síla Reactu se projeví ve chvíli, kdy se ve vykreslené aplikaci mění data a je potřeba překreslit uživatelské rozhraní. Když má být komponenta poprvé vykreslena, React vygeneruje kód HTML, reprezentující aktuální stav komponenty a vykreslí ho. Jakmile se změní data, React vygeneruje nový kód a porovná ho s předchozí verzí kódu. Na základě tohoto porovnání vygeneruje minimální nutnou sadu úprav, které je třeba provést k dosažení požadovaného výsledku. Díky tomuto přístupu je překreslování komponenty velmi rychlé.[8]

4.3.3 Angular

Angular je framework pro tvorbu dynamických webových aplikací od společnosti Google. Používá HTML jako jazyk pro psaní šablon a rozšiřuje jeho syntaxi o další prvky. Poskytuje nástroje pro synchronizaci dat mezi datovým modelem a komponentami (data binding), vkládání závislostí (dependency injection), práci s formuláři a další, čímž snižuje množství kódu, který je třeba napsat.[9]

4.4 Balíčkovací systém

Meteor integruje dva balíčkovací systémy – Node Package Manager (npm) a Atmosphere.

4.4.1 npm

npm je repositář obecných balíčků pro Javascript. Původně byly tyto balíčky určeny výhradně pro serverové prostředí Node.js, ale časem vznikla různá řešení pro použití těchto balíčků i v jiných prostředích, např. ve webovém prohlížeči. Dnes je npm používáno pro všechny možné typy balíčků pro Javascript.

4.4.2 Atmosphere

Atmosphere je repositář balíčků, psaných přímo pro Meteor.js. Díky tomu mají tyto balíčky několik výhod oproti npm balíčkům:

- Mohou přímo využívat základních funkcí Meteor.js
- Mohou zahrnovat i jiné typy souborů, než Javascript, např. Cascading Style Sheets (CSS) nebo obrázky
- Mohou definovat různé chování pro server a klientskou aplikaci, čímž dochází k odlišnému chování v různých prostředích
- Mohou obsahovat rozšíření pro sestavovací program platformy Meteor.js

4.5 Sestavení hotové aplikace

Meteor.js poskytuje nástroj pro příkazovou řádku, který slouží ke kompilaci zdrojového kódu, sestavení aplikace, jejímu umístění na server a spuštění.

Tento nástroj slouží i k pohodlnému vývoji a testování aplikace. Vývojář spustí sestavovací proces a nechá ho spuštěný, zatímco píše kód. Jakmile změní část kódu, sestavovací proces tuto změnu zaregistruje a automaticky znovu zkompiluje daný soubor a restartuje klientskou aplikaci nebo server, pokud je to potřeba.

Další důležitou funkcí sestavovacího nástroje je, že při sestavování produkční verze aplikace zkombinuje veškerý kód do jednoho souboru, který následně očistí od veškerých nepotřebných znaků, čímž se výrazně zmenší jeho velikost a tudíž i zvýší rychlost běhu aplikace (proces, známý jako „minification“).

Sestavovací nástroj také poskytuje možnost kompilace zdrojového kódu, psaného v některé z verzí jazyka, která není podporována webovými prohlížeči. Mezi hlavní dva jazyky, které sestavovací nástroj umí zkompilovat do klasického kódu jazyka Javascript, patří EcmaScript 2015 a CoffeeScript.

V neposlední řádě podporuje sestavovací nástroj populární preprocesory jazyka CSS, mezi něž patří Syntactically Awesome Stylesheets (SASS), LESS a Stylus.

4.6 Mobilní aplikace

Meteor.js integruje technologii Cordova.

Cordova je známý open-source projekt od společnosti Apache, umožňující sestavování mobilních aplikací ze stejného zdrojového kódu jako klasické webové aplikace. Webová aplikace, implementovaná s pomocí jazyků HTML, CSS a Javascript, běží ve speciálním prostředí, nazývaném web view. Web view je v podstatě webový prohlížeč bez uživatelského rozhraní. Díky Cordově lze existující webovou aplikaci distribuovat jako nativní mobilní aplikaci.

Důležitou výhodou distribuce v podobě Cordova aplikace je, že jsou k aplikaci přibaleny i všechny potřebné soubory, např. obrázky. Díky tomu se aplikace načítá rychleji, než ekvivalentní webová aplikace, což je důležité v případě, že má uživatel pomalé připojení k internetu.

Další důležitou funkcí je tzv. „hot code push“, což je technologie, umožňující automatickou aktualizaci aplikace na zařízeních uživatelů bez toho, aby se musela nová verze znovu publikovat na Google Play nebo App Store.

Cordova také nabízí možnost integrace dalších rozšíření. Tato rozšíření umožňují využívání funkcí, specifických pro mobilní zařízení, které většinou nejsou dostupné u webových aplikací, např. fotoaparát zařízení, přístup do lokálního souborového systému, interakce s čtečkou čárových kódů, atp.

Meteor.js integruje proces sestavení Cordova aplikace do svého základního sestavovacího nástroje.

Kapitola 5

Návrh

5.1 Použité technologie

Kapitola popisuje hlavní technologie, které jsou použity k realizaci této práce s výjimkou technologie Meteor.js, která je podrobně popsána v kapitole 4.

5.1.1 Javascript

Javascript je nenáročný, interpretovaný, objektově orientovaný jazyk. Je znám zejména jako skriptovací jazyk pro webové stránky, nicméně je v dnešní době používán i v jiných prostředích, např. v Node.js. Standardem pro Javascript je EcmaScript.[10]

5.1.2 EcmaScript 2015

EcmaScript 2015 je šestou verzí standardu EcmaScript. Cílem EcmaScript 2015 je poskytnout lepší podporu pro velké projekty, tvorbu knihoven a podpořit použití EcmaScriptu jako výsledného produktu kompilace jiných jazyků. Mezi hlavní přínosy EcmaScript 2015 patří moduly, deklarace tříd, iterátory a generátory, sliby pro asynchronní programování a další. Mezi vestavěné třídy knihovny EcmaScriptu byly přidány nové datové struktury, např. mapy a množiny. EcmaScript 2015 navíc nově umožňuje deklarovat třídy, které dědí od vestavěných tříd.[11]

5.1.3 Webpack

Webpack je nástroj, který z použitých modulů a jejich závislostí generuje statické soubory. Cíle Webpacku jsou:

- Dělit strom závislostí jednotlivých modulů na menší celky, které jsou načítány dle potřeby

- Minimalizovat dobu prvotního načtení aplikace
- Mít možnost přeměnit jakékoliv statické soubory na moduly
- Mít možnost integrovat knihovny třetích stran jako moduly
- Optimalizovat tento proces pro velké projekty

Samotný Webpack umí zpracovávat pouze kód v jazyce Javascript, ale poskytuje podporu pro rozšíření (tzv. „loaders“), která umožňují transformovat jiné typy souborů do kódu jazyka Javascript.[12]

Vzhledem k tomu, že Meteor.js má implicitně zabudovaný sestavovací nástroj, který funguje podobně (viz. 4.5), není nutné používat Webpack. Důvodem, proč je v této práci použit je, že dle našeho pozorování Webpack rychleji generuje hotové statické soubory a tudíž umožňuje pohodlnější vývoj.

5.1.4 MongoDB

MongoDB je open-source dokumentová databáze, poskytující vysoký výkon, vysokou dostupnost a automatické škálování.

Záznam v MongoDB se nazývá dokument, což je datová struktura, skládající se z dvojic údajů klíč a hodnota. Jednotlivé hodnoty mohou být několika typů, včetně dalších dokumentů, polí nebo polí dokumentů. Dokumenty se svojí strukturou podobají JSON objektům, proto se MongoDB snadno používá v kombinaci s jazykem Javascript.[13]

5.1.5 React

React je jedna ze tří knihoven, které Meteor.js oficiálně podporuje pro implementaci uživatelského rozhraní (viz. 4.3).

5.1.6 React Router

React Router je knihovna, sloužící jako doplněk knihovny React. Slouží k vykreslování různých komponent podle aktuální adresy Uniform Resource Locator (URL) a udržuje tak synchronizaci uživatelského rozhraní a aktuální adresy URL. Poskytuje jednoduché rozhraní a užitečné funkce, např, načítání různých komponent a ž ve chvíli kdy jsou potřeba (tzv. „lazy loading“).[14]

5.1.7 Redux

Redux je předvídatelný stavový kontejner pro aplikace psané v jazyce Javascript. Pomáhá vytvářet aplikace, které se chovají předvídatelně a snadno se testují.[15]

Redux vychází z návrhového vzoru Flux[16], ale snaží se vyhnout jeho komplexitě. Také se inspiruje návrhovým vzorem Elm.[17]

5.1.8 CSS

CSS je jazyk, popisující způsob, jakým má být prezentován dokument, napsaný v jazyce HTML nebo Extensible Markup Language (XML). CSS popisuje, jak má být dokument vykreslen na obrazovce, na papíře a dalších médiích.[18]

5.1.9 Twitter Bootstrap

Twitter Bootstrap je frontendový framework, umožňující rychlejší a jednodušší vývoj webových aplikací. Obsahuje sadu nástrojů pro implementaci klasických prvků uživatelského rozhraní, např. typografie, formuláře, tlačítka, tabulky, upozornění a další. Kromě toho obsahuje další volitelné rozšíření pro Javascript.[19]

Největší výhodou Twitter Bootstrapu je, že usnadňuje tvorbu responzivního designu. Responzivní design znamená, že se uživatelské rozhraní aktivně přizpůsobuje prostředí, ve kterém je spuštěna, zejména velikosti obrazovky a platformě.[20]

5.2 Doménový model

Doménový model 5.1 je navržen tak, aby výsledná aplikace splňovala cílové požadavky a slouží k identifikaci klíčových entit a vztahů mezi nimi.

Javascript je sice objektově orientovaný jazyk, podobně jako C++ nebo Java, ale na rozdíl od těchto jazyků, kde je objektové programování založené na bázi tříd a instancí těchto tříd, je Javascript založen na bázi prototypů. [21] Z tohoto důvodu nejsou v návrhu modelu použity termíny „třída“ a „instance“. Místo toho se používá termín „objekt“.

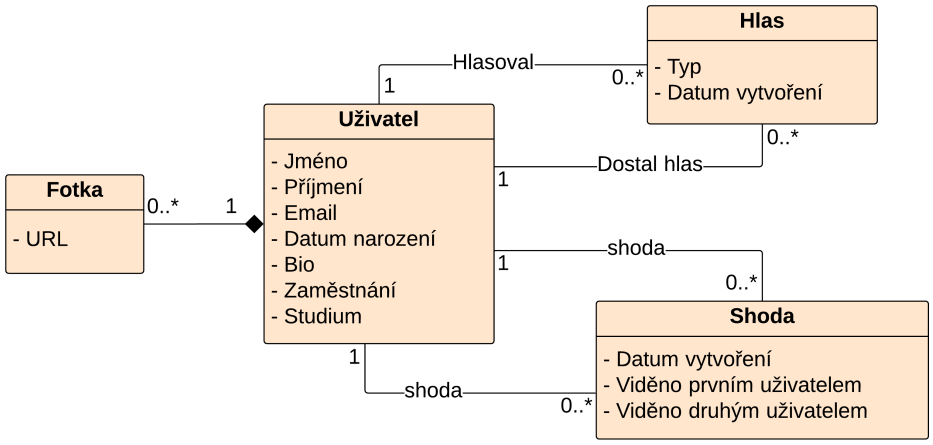
Mezi entitami jsou použity následující vztahy:

- Asociace – znázorňuje se plnou čarou a vyjadřuje, že entity mohou existovat nezávisle na sobě.
- Kompozice – znázorňuje se plnou čarou s vyplněným kosočtvercem na konci. Vyjadřuje vztah celek-část, přičemž část nemůže existovat bez celku.

V následujících podkapitolách jsou podrobněji popsány jednotlivé entity.

5.2.1 Uživatel

Entita reprezentuje uživatele v systému. Při registraci jsou data o uživateli získána z rozhraní webové služby Facebook a struktura těchto dat do značné míry diktuje strukturu výsledné entity.



Obrázek 5.1: Doménový model

5.2.2 Hlas

Entita reprezentuje hlas, který udělil uživatel jinému uživateli. Obsahuje informaci o typu hlasu (Like, Nope nebo Super like) a datum a čas vytvoření hlasu.

5.2.3 Shoda

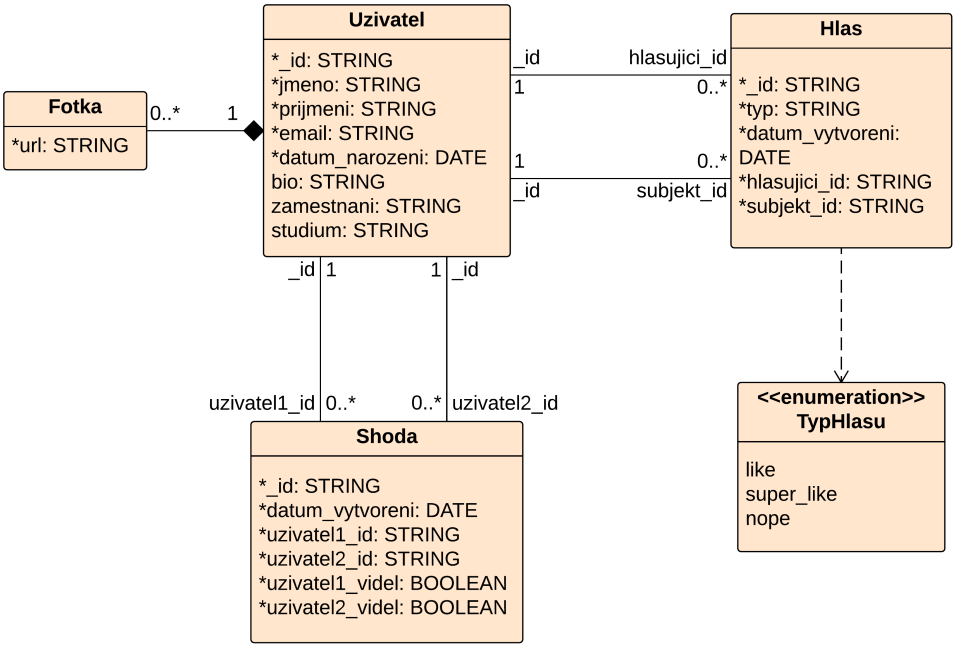
Jakmile si dva uživatelé navzájem udělí hlas typu Like nebo Super like, dochází ke shodě. Tato entita reprezentuje tuto shodu a obsahuje informaci o tom, o které dva uživatele se jedná, kdy došlo ke shodě a zdali už byli tito uživatelé o shodě informováni.

5.3 Databázový model

Databázový model 5.2 popisuje uložení dat v dokumentové databázi. Databázový model je vytvořen podle jednotlivých entit doménového modelu. Unified Modeling Language (UML) notace je upravena pro potřeby znázornění schématu dokumentové databáze.

Mezi entitami jsou použity následující vztahy:

- Asociace – znázorňuje se plnou čarou a vyjadřuje vztah mezi entitami, podobně jako v relační databázi. Místo cizího klíče jsou v asociaci uvedeny názvy hodnot, které danou asociaci tvoří.
- Kompozice (vnořený dokument) – znázorňuje se plnou čarou s vyplněným kosočtvercem na konci. Vyjadřuje vztah celek-část.



Obrázek 5.2: Databázový model

Kapitola 6

Implementace

Kapitola popisuje pracovní prostředí a vybrané části implementace. Implementace zahrnuje všechny funkční i nefunkční požadavky, uvedené v kapitole 3.

6.1 Implementační nástroje

Pro zvýšení přehlednosti a produktivity byl využit nástroj Git pro verzování kódu. Je to užitečný nástroj, umožňující inkrementální změny zdrojového kódu, revize změn před každým commitem a vracení změn.

K napsání zdrojového kódu bylo použito IDE WebStorm.

6.2 Poznámky k implementaci

6.2.1 Adresářová struktura

V rámci implementace byly stanoveny určité konvence pro organizování souborů. Tyto konvence vycházejí ze základní adresářové struktury, popsané na obrázku 4.1 a rozšiřují ji.

Je důležité zmínit zejména, jak jsou organizovány soubory klientské části aplikace. Jak je vidět na obrázku 6.1, adresář *client/modules* obsahuje adresáře, které reprezentují jednotlivé moduly, které dohromady tvoří klientskou část aplikace. Moduly mají jednotnou adresářovou strukturu, což zajišťuje přehlednost a konzistenci zdrojového kódu, viz. 6.2.

6.2.2 Redux

Text této sekce vychází z dokumentace knihovny Redux.[15]

Jak již bylo zmíněno v kapitole 5.1, v implementaci je použit návrhový vzor Redux. Konkrétně je tento návrhový vzor použit v kombinaci s knihovnou React pro implementaci uživatelského rozhraní.

- `.meteor` — Zdrojové soubory Meteor.js
- `client` — Klientská část
  - `configs` — Konfigurační soubory klientské části aplikace
  - `modules` — Jednotlivé moduly, tvořící aplikaci
    - `admin` — Modul administračního rozhraní
    - `core` — Modul základních funkcí
    - `users` — Modul správy uživatelů
    - `voting` — Modul hlasování
- `lib` — Soubory, dostupné ve všech prostředích
  - `collections.js` — Implementace objektů databázových kolekcí
  - `helpers.js` — Pomocné funkce
- `node_modules` — Node.js balíčky, stažené s pomocí npm
- `public` — Veřejně dostupné soubory, např. obrázky
  - `fonts` — Fonty
  - `images` — Obrázky
- `server` — Serverová část
  - `bootstrap` — Dílčí scripty, které se spustí při startu serveru
  - `methods` — Metody, tvořící webové rozhraní pro klientskou část
  - `publications` — Publikace dat
  - `services` — Pomocné funkce pro práci se službami třetích stran

Obrázek 6.1: Adresářová struktura aplikace

- `module` — Modul, nacházející se v adresáři `client/modules`
  - `actions` — Tzv. „akce“, využívané knihovnou Redux
  - `components` — Vykreslitelné komponenty
  - `containers` — Kontejnery, obalující komponenty
  - `reducers` — Tzv. „r“educery, využívané knihovnou Redux
  - `stylesheets` — Soubory CSS, definující grafické rozvržení komponent daného modulu
  - `index.js` — Soubor, který poskytuje určité části modulu k využití zvenčí

Obrázek 6.2: Adresářová struktura modulu

Redux je tvořen několika prvky, které je potřeba zmínit v kontextu implementace:

Skladiště (Store)

Skladiště je objekt, který obsahuje kompletní aktuální stav aplikace. Zároveň poskytuje rozhraní pro získání aktuálního stavu nebo jeho části a také poskytuje rozhraní pro jeho modifikaci.

Je důležité zmínit, že aplikace má pouze jedno skladiště, které obsahuje kompletní stav celé aplikace.

Akce

Akce je nosičem informace, kterou aplikace odesílá skladišti. Je to zároveň jediný způsob komunikace mezi aplikací a skladištěm. Akce jsou odesílány prostřednictvím rozhraní skladiště.

Aby byla zachována znovupoužitelnost různých typů akcí, které jsou v implementaci definovány, byly implementovány tzv „action creatory“. Action creator je speciální funkce, která vytváří instance určitého typu akce.

Reducer

Akce informuje skladiště o tom, že se něco stalo. Už ale nedefinuje, jak se na základě této informace má změnit stav aplikace. K tomu slouží právě reducer.

Reducer je funkce se dvěma argumenty – aktuální stav a akce – a vrací nový stav.

Stejně jako je v rámci aplikace jen jedno skladiště, je i jen jeden reducer, který toto skladiště registruje. To ale neznamená, že pro rozsáhlou aplikaci musíme implementovat rozsáhlou monolitickou funkci, která bude reducerem této aplikace. S tímto případem knihovna Redux počítá a nabízí funkci `combineReducers()`, která přijímá jako argument seznam reducerů a vrací jeden velký reducer, kde se jednotlivé reducery starají pouze o část stavu. Tato funkce umožňuje použití návrhového vzoru, který je důležitou součástí implementace této práce.

Každý z modulů, ze kterých se implementace skládá, implementuje vlastní reducer. V případě rozsáhlého modulu může definovat i více reducerů, které jsou pak zkombinovány na úrovni daného modulu. Pro získání výsledného globálního reduceru pak stačí zkombinovat reducery jednotlivých modulů.

6.2.3 React

Uživatelské rozhraní je implementováno s pomocí knihovny React.

Implementace je rozdělena do mnoha komponent, což jsou objekty, které dědí od objektu `React.Component` a implementují metody životního cyklu komponent dle specifikace knihovny React.[22]

Návrh komponent se řídí striktními pravidly, které mají zajistit jejich znovupoužitelnost. Nejdůležitějším pravidlem je, že samotná komponenta se nestará o zisk dat, která potřebuje, ale očekává, že je obdrží jako argument od nadřazené komponenty, která ji bude vykreslovat. Je to tedy určitá obdoba Dependency Injection (DI). Často ale komponenty očekávají data z Redux skladiště. K tomu slouží druhý druh komponent, kterým se říká „kontejnery“.

Kontejner obaluje komponentu a má dvě funkce. Za prvé propojuje komponentu s Redux skladištěm a dodává jí aktuální data a action creatory. Za druhé, pokud je to potřeba, před prvním vykreslením komponenty spustí asynchronní načítání dat ze serveru, které komponenta potřebuje.

6.2.4 Asynchronní načítání dat ze serveru

Jak již bylo popsáno v kapitole 4.2.2, komunikace mezi klientskou aplikací a serverem probíhá asynchronně s pomocí protokolu DDP, což je v praxi realizováno s pomocí publikací a předplacení.

Jelikož je v implementaci použit návrhový vzor Redux, je nutné data, získaná ze serveru, ukládat do skladiště, stejně jako kterákoliv jiná data. Zároveň je nutné brát v potaz to, že komunikace mezi klientskou aplikací a serverem v Meteor.js je tzv. reaktivní. To znamená, že pokud má klientská aplikace předplacenou nějakou podmnožinu dat, tak všechny změny v rámci této podmnožiny, ke kterým dojde na serveru, jsou automaticky odesílány do klientské aplikace, která je předplatila.

Aby data, získaná ze serveru, byla vždy synchronizována s daty ve skladišti, jsou v implementaci využity dva balíčky:

Redux Thunk

Redux Thunk je volitelné rozšíření Redux skladiště. Standardně lze do skladiště odesílat pouze objekty akcí. Redux Thunk přidává možnost odesílat funkce. Odesílaná funkce dostane jako argument funkci `dispatch()`, která slouží k odesílání akcí do skladiště.[23]

Tracker

Tracker je knihovna, poskytující jednoduché rozhraní pro reaktivní programování. Rozhraní Trackeru představuje metoda `Tracker.autorun()`. Tato metoda přijímá jako argument libovolnou funkci. Uvnitř této funkce lze provádět libovolné operace, ale smysl Trackeru se projeví ve chvíli, kdy je v této funkci volána nějaká funkce, která je tzv. „tracker-aware“.[24]

„Tracker-aware“ funkce je jakákoliv funkce, která informuje Tracker pokaždé, když se změní data, která poskytuje. Pokud byla tato funkce zavolána uvnitř funkce, která byla předána jako argument metodě `Tracker.autorun()`, je celá tato funkce zavolána znovu pokaždé, kdy je Tracker informován o změně dat.

Metody `Meteor.subscribe()`, `Collection.find()` a další, které slouží k předplacení dat a jejich načítání z paměti, jsou „tracker-aware“.

Synchronizace dat mezi serverem a Redux skladištěm je v implementaci dosaženo následujícím způsobem:

1. Pro každou množinu dat, načítaných ze serveru, existuje action creator a ten vrací funkci.

2. V této funkci se volá metoda `Tracker.autorun()`, které je jako argument předána další funkce.

3. V této další funkci je volána metoda `Meteor.subscribe()`, která předplatí určitá data. Tato metoda vrací objekt, na kterém lze zavolat metodu `ready()`, jejíž návratová hodnota je typu `boolean` a je `true`, pokud data již byla načtena ze serveru.

4. Pokud jsou data již načtena, získají se s pomocí jedné z metod, které implementuje objekt `Collection`.

5. Nakonec se získaná data odešlou jako akce do skladiště.

6. Reducer příjme akci se získanými daty a data uloží do globálního stavu.

7. Pokaždé, když na serveru dojde ke změně dat, je o tom Tracker informován a celý proces se opakuje, čímž jsou data ve skladišti průběžně aktualizována.

6.2.5 CSS styly

Struktura zdrojového kódu CSS stylů kopíruje strukturu komponent uživatelského rozhraní, tzn. pro každou komponentu je vytvořen samostatný soubor. V tomto souboru jsou definovány všechny styly pro danou komponentu.

Sestavovací proces Meteor.js má zabudovanou funkci, díky které automaticky objeví všechny CSS soubory v projektu a vytvoří z nich jeden výsledný soubor, který obsahuje všechny styly a tento soubor navíc optimalizuje (proces, rovněž známý jako tzv. „minifikace“).[4]

6.2.6 Uživatelské rozhraní

Aby uživatelské rozhraní vizuálně co nejvěrněji kopírovalo původní aplikaci, bylo implementováno několik animovaných vizuálních efektů:

- Fronta uživatelů, kterým může uživatel dávat hlasy, se zobrazuje formou jakéhosi balíčku karet. Tzn. je použit grafický efekt, který vytváří iluzi toho, že jednotlivé fotky leží jedna na druhé. Aktuální uživatel, pro kterého může uživatel hlasovat, se nachází na vrchu tohoto balíčku.

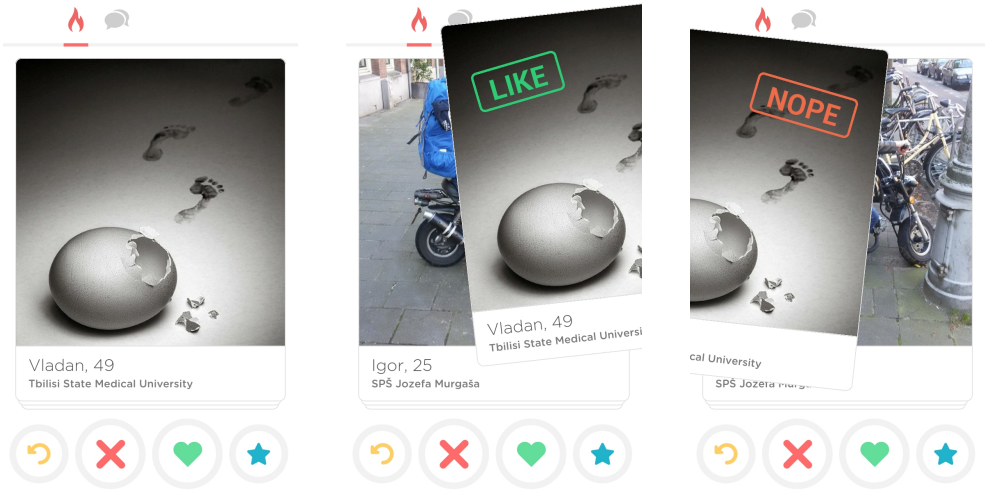
- Uživatel může hlasovat přiložením prstu na zobrazenou fotku uživatele a potažením do strany. Potažení doleva znamená hlas typu Nope a potažení doprava znamená hlas typu Like. Tento efekt má dále několik důležitých vedlejších efektů:

  – Zatímco uživatel drží prst na obrazovce, zobrazená fotka uživatele následuje pohyb prstu.

  – Při potažení fotky do strany začíná fotka rotovat ve směru pohybu a úhel rotace roste úměrně vzdálenosti, o kterou byla fotka potažena.

  - Při potažení fotky do strany se další fotky, nacházející se pod ní, posouvají do popředí. Tento posun roste úměrně vzdálenosti, o kterou byla fotka potažena.
  - Při potažení fotky do strany se na ní zobrazí grafický indikátor, napovídající typ hlasu, který bude uživateli udělen při potažení daným směrem.

- Na obrazovce detailu profilu uživatele lze potažením prstem procházet jednotlivé fotky daného uživatele. Animovaný efekt reaguje na rychlost pohybu prstu.
- Na obrazovce detailu profilu uživatele lze potažením prstem směrem dolů zvětšit právě zobrazenou fotku.



Obrázek 6.3: Ukázka vizuálního efektu na obrazovce hlasování

Kapitola 7

Vyhodnocení

Prototyp aplikace byl implementován dle zadání a implementace splňuje všechny funkční i nefunkční požadavky.

7.1 Analýza limitů Meteor.js

7.1.1 Implementace

Během implementace bylo identifikováno několik problémů, specifických pro vývoj mobilních aplikací s pomocí Meteor.js.

Podpora dotykového rozhraní

Jazyk Javascript má velice základní podporu dotykového rozhraní. Jediné možnosti, co samotný jazyk poskytuje, jsou rozpoznání, že byl prst přiložen na obrazovku, že se pohnul po obrazovce a že pustil obrazovku.

Pokud je třeba implementovat nějaké efekty, založené na konkrétních vzorcích pohybu prstu, musí si vývojář poradit s tímto základním rozhraním a vše implementovat sám, což značně zpomaluje vývoj. Tento problém by do značné míry zmírnilo použití nějaké knihovny, která by poskytovala abstrakci akcí, běžných pro dotykové rozhraní. Bohužel se nám ale nepodařilo nalézt žádnou knihovnu, která by splňovala požadavky pro implementaci této práce.

Situace se velmi liší, pokud je aplikace vyvíjena s pomocí nativního řešení. Jak nativní řešení pro OS Android, tak i pro iOS poskytují abstrakci rozpoznávání vzorců pohybu prstu po obrazovce (klik, pohyb vlevo, pohyb vpravo, rychlost pohybu).[25, 26] Na základě toho lze považovat neexistenci potřebné abstrakce jako limit frameworku Meteor.js při implementaci mobilních aplikací.

Podpora CSS3

I přesto, že se aplikace, implementovaná s pomocí Meteor.js, na mobilním zařízení tváří jako nativní aplikace, je fakticky spouštěna v prostředí webového prohlížeče. O jaký webový prohlížeč a jakou verzi se jedná, závisí na konkrétním OS a jeho verzi. Základní webové prohlížeče starších verzí OS Android a iOS nepodporují některé moderní funkce standardu CSS3. Např. funkce flexbox je plně podporována OS Android až od verze 4.4 a iOS od verze 8.4.[27]

Pro vývojáře je to limit, který může zpomalit a zneefektivnit vývoj. Pokud má mobilní aplikace podporovat starší verze OS a vývojář chce implementovat uživatelské rozhraní, na jehož implementaci by se hodily funkce CSS3, může se stát, že některá z funkcí nebude podporována ve starších verzích OS. V takovém případě musí vývojář zvolit jiné řešení, které mu pravděpodobně zabere více času a bude více náchylné na chyby.

Další problém je, že vývojář nemůže přesně vědět, jaké z jím použitých funkcí budou podporovány jakým zařízením. Nemá jinou možnost, než implementovanou aplikaci otestovat na co nejvíce zařízeních, aby si mohl být jistý, že aplikace bude fungovat správně. Tento problém je značně zmírněn při vývoji pomocí nativních řešení, kde musí vývojář specifikovat, od jaké verze OS bude aplikace podporována a na základě toho je mu poskytnuta určitá sada nástrojů a verze kompilátoru, odpovídající specifikované verzi OS. Díky tomu se nemůže stát, že by se do výsledné aplikace dostala nějaká funkce, která by nebyla podporována některou z cílových verzí OS.

Tato fakta jsou dalším limitem pro vývoj s pomocí Meteor.js a mohou zpomalit vývoj nebo způsobit nečekané problémy při běhu výsledné aplikace.

7.1.2 Výkon

Během implementace byly veškeré algoritmy navrženy pro maximální efektivitu a výkon. Použité algoritmy byly dále konzultovány v internetové komunitě vývojářů, používajících Meteor.js, a dle výsledků konzultace byly zrevidovány. I přes veškeré úsilí nebylo možné dosáhnout dostačujícího výkonu implementované aplikace.

Problematickou částí jsou grafické animované efekty, popsané v kapitole 6.2.6. Důvodem je, že tyto efekty jsou založeny na pohybu prstu po obrazovce, což nelze vyřešit dostatečně efektivně, protože je rozhraní implementováno v jazyce Javascript.

Moderní webové prohlížeče podporují a optimálně vykreslují jednoduchý efekt potažení elementu po obrazovce (efekt, rovněž známý jako „drag'n'drop“). Požadavky efektů, implementovaných v této práci ale přesahují možnosti tohoto jednoduchého řešení a je tedy nutné je řešit explicitně a na míru, s pomocí sady velmi jednoduchých a základních funkcí jazyka Javascript. Tento fakt má za následek, že výsledné řešení nemá nativní podporu webového prohlížeče a má tedy znatelně nižší výkon.

Nedostatečný výkon je nejvíce znát na efektu potažení fotky uživatele na obrazovce hlasování. Tento efekt má několik vedlejších efektů, spojených s dalšími elementy na obrazovce. Tato vyšší komplexita dále snižuje výsledný výkon a zpomaluje aplikaci.

Kromě vykreslování animovaných efektů je výkon implementovaného prototypu dostačující. Nízký výkon v mobilních aplikacích, implementovaných s pomocí Meteor.js, tedy hrozí hlavně tehdy, když jsou součástí implementace uživatelského rozhraní právě podobné animované efekty.

7.2 Analýza rizik a ekonomických dopadů použití Meteor.js

V kapitole je analyzováno použití Meteor.js v praxi z manažersko-ekonomického hlediska ve srovnání s vývojem stejné aplikace s pomocí nativních řešení. Jsou analyzována rizika a ekonomické dopady této volby.

7.2.1 Rizika

Z informací, uvedených v kapitole 7.1 vyplývá, že největší rizika při použití Meteor.js pro vývoj mobilních aplikací hrozí vývojářům při implementaci rozsáhlejších aplikací.

Zejména pokud má výsledná mobilní aplikace disponovat interaktivními animovanými efekty, je poměrně velké riziko, že nakonec nebude mít dostačující výkon. V takovém případě vývojářům nezbude nic jiného, než buď zvolit jiné řešení a implementovat celou aplikaci znovu a nebo odstranit z návrhu aplikace efekty, které nejvíc ovlivňují výkon.

Rizika spojená s nízkým výkonem aplikace lze částečně mitigovat. Vývojáři mohou ve fázi návrhu identifikovat části aplikace, které budou mít největší vliv na výkon. Mohou pak tyto části implementovat samostatně jako prototyp. Na základě pozorování tohoto prototypu pak lze stanovit, zdali je vhodné aplikaci implementovat s pomocí Meteor.js.

7.2.2 Ekonomické dopady

V případě nativních řešení je důležité, aby firma, která má vývoj na starosti, zaměstnávala experty na vývoj jak pro OS Android, tak pro iOS. Většinou to v praxi znamená, že firma musí zaměstnat více různých lidí, z nichž každý se věnuje pouze vývoji pro jeden z OS, což má několik praktických důvodů. Zaprvé, kvalitní vývoj pro každý z OS s pomocí nativních řešení vyžaduje vysokou úroveň znalostí daného řešení a OS a pro firmu je jednodušší sehnat dva lidi, z nichž každý je expertem na jeden z OS, než najít jediného člověka, který by byl expertem na oba. Druhým důvodem je čas. Více vývojářů, vyvíjejících paralelně stejnou aplikaci pro více OS logicky dokončí projekt dříve, než jediný vývojář, který by musel nejdříve vyvinout aplikaci pro jeden OS

a následně pro druhý. To ale není všechno, protože mobilní aplikace většinou vyžaduje serverové rozhraní, se kterým komunikuje a získává od něj data. Pro implementaci tohoto rozhraní je standardně potřeba vývojář webových aplikací.

V případě vývoje s pomocí Meteor.js si firma vystačí s menším počtem vývojářů. Tam kde by v případě nativních řešení byli potřeba tři vývojáři – jeden pro OS Android, jeden pro iOS a jeden pro vývoj serverového rozhraní – tam v případě Meteor.js stačí jediný vývojář se znalostí vývoje webových aplikací a platformy Meteor.js.

Z toho plynou určité ekonomické dopady pro firmu, která tyto vývojáře zaměstnává. Zaprvé se v obou případech liší počet vývojářů, potřebných pro vývoj. Zadruhé se velmi liší znalosti, které musí vývojáři v obou případech ovládat.

V průzkumu na webu Stack Overflow z roku 2015[28] jsou uvedeny průměrné roční platy mobilních a webových vývojářů. Pokud budeme předpokládat, že člověk pracuje průměrně 40 hodin týdně, pak z uvedených čísel vyplývají hodinové sazby, uvedené v tabulce 7.1.

Pozice	Hodinová sazba (USD)
Android vývojář	13
iOS vývojář	13
Webový vývojář	12

Tabulka 7.1: Průměrná hodinová sazba

Podle webu The Nine Hertz[29] trvá vývoj serverového rozhraní (backend) standardní mobilní aplikace 10 týdnů a vývoj samotné aplikace (frontend) trvá 8 týdnů. Pokud firma vyvíjí mobilní aplikaci pro OS Android i iOS, musí vyvinout dvě aplikace, tedy vývoj trvá 8 týdnů pro každý z OS. Naopak, pokud firma vyvíjí aplikaci s pomocí Meteor.js, trvá vývoj aplikace pouze 8 týdnů.

Z těchto informací lze vypočítat odhadovanou cenu vývoje standardní mobilní aplikace. Výsledky jsou uvedeny v tabulce 7.2.

Řešení	Cena (USD)
Nativní	13 120
Meteor.js	8 640

Tabulka 7.2: Odhadovaná cena vývoje standardní mobilní aplikace

Je tedy vidět, že vývoj s pomocí Meteor.js může výrazně snížit náklady na vývojáře.

7.3 Meteor.js v praxi

Z výsledků analýzy v této kapitole plyne, že Meteor.js je vhodné použít pro implementaci mobilních aplikací v několika případech:

- Jedná se o jednoduchou aplikaci
- Vývojář implementoval webovou aplikaci s pomocí Meteor.js a chce jí vydat jako mobilní aplikaci
- Jedná se o složitější aplikaci, kde bylo ověřeno, že nedojde k problémům s výkonem
- Jedná se o prototyp aplikace, který je nutné vyvinout co nejrychleji

V takových případech může Meteor.js vývojářům ušetřit čas i peníze. V opačném případě může volba použít Meteor.js způsobit více problémů než užitku a spíše se doporučuje zvolit nativní řešení nebo případně jiné vhodné nenativní řešení, s kterým by v daném případě bylo dosaženo lepších výsledků.

Závěr

Cílem práce bylo zjistit, zdali je framework Meteor.js vhodným řešením pro vývojáře mobilních aplikací jako náhrada za nativní vývoj.

Nejdříve byly v práci analyzovány rozdíly mezi vývojem s pomocí nativních řešení a Meteor.js. Také byl podrobně popsán vývoj s pomocí Meteor.js. S pomocí Meteor.js byl implementován prototyp aplikace Tinder a také webové administrační rozhraní pro tuto aplikaci. Na základě poznatků, získaných během implementace byly nakonec analyzovány limity Meteor.js v rámci vývoje mobilních aplikací a dále byla analyzována rizika a ekonomické dopady použití Meteor.js při vývoji mobilních aplikací místo nativních řešení.

Bylo vyhodnoceno, že framework Meteor.js může být vhodným řešením pro implementaci menších mobilních aplikací a mobilních aplikací bez komplexních interaktivních vizuálních efektů uživatelského rozhraní. V takovém případě může použití Meteor.js ušetřit čas i peníze. V případě rozsáhlejších aplikací může dojít k problémům s výkonem, přesně tak, jak se stalo v případě aplikace Tinder, implementované v této práci.

Kvůli tomu, že uživatelské rozhraní aplikace obsahovalo animované prvky a komplexní interaktivní vizuální efekty došlo k tomu, že aplikace měla i přes všechny optimalizace problémy s výkonem. Z tohoto důvodu bylo v této práci vyhodnoceno, že pro takový typ aplikací je vhodnější zvolit jiné řešení, které nabídne vyšší výkon a takové řešení je vhodnější i za cenu toho, že vývoj může trvat delší dobu a stát víc peněz.

Budoucí práce

Jako další krok by mohlo být zajímavé zkusit implementovat stejný prototyp mobilní aplikace v dalších hybridních frameworcích, např. React Native nebo Xamarin. I když všechny tyto frameworky, včetně Meteor.js nabízejí podobné výhody, každý z nich funguje úplně jinak a tak pravděpodobně poskytne jiný výsledek.

Literatura

[1] HATHIBELAGAL, A.: *Android From Scratch: An Overview of Android Application Development* [online]. 2016, [cit. 2016-04-28]. Dostupné z: http://code.tutsplus.com/articles/android-from-scratch-an-overview-of-android-application-development--cms-25972

[2] Apple, inc.: *Start Developing iOS Apps (Swift)* [online]. 2015, [cit. 2016-04-28]. Dostupné z: https://developer.apple.com/library/ios/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html

[3] JACOBS, B.: *How to Submit an iOS App to the App Store* [online]. 2013, [cit. 2016-04-28]. Dostupné z: http://code.tutsplus.com/tutorials/how-to-submit-an-ios-app-to-the-app-store--mobile-16812

[4] Meteor Develop Group: *Meteor Guide* [online]. [cit. 2016-04-27]. Dostupné z: http://guide.meteor.com/

[5] Down to Chill: *Down To Chill* [online]. ©2015, [cit. 2016-05-05]. Dostupné z: http://downtochill.com/

[6] Verso Learning: *Verso Campus* [online]. ©2016, [cit. 2016-05-05]. Dostupné z: http://versoapp.com/

[7] Facebook, inc.: *Why React?* [online]. [cit. 2016-04-28]. Dostupné z: \unhbox\voidb@x\hbox{https://}facebook.github.io/react/docs/why-react.html

[8] HUNT, P.: *Why did we build React?* [online]. 2013, [cit. 2016-04-28]. Dostupné z: https://facebook.github.io/react/blog/2013/06/05/why-react.html

[9] Google: *What Is Angular?* [online]. [cit. 2016-04-28]. Dostupné z: https://docs.angularjs.org/guide/introduction

[10] KACEROVSKÝ, M.; SCHOLZ, F.; GRŇO, M.: *JavaScript* [online]. Mozilla Developer Network and individual contributors, 2016, [cit. 2016-04-28]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript

[11] Ecma International: *ECMAScript® 2015 Language Specification* [online]. ©2015, [cit. 2016-04-28]. Dostupné z: http://www.ecma-international.org/ecma-262/6.0

[12] KOPPERS, T.: *What is Webpack* [online]. ©2012-2016, [cit. 2016-04-28]. Dostupné z: http://webpack.github.io/docs/what-is-webpack.html

[13] MongoDB, inc.: *Introduction to MongoDB* [online]. ©2008-2016, [cit. 2016-04-28]. Dostupné z: https://docs.mongodb.org/manual/introduction/

[14] React Community: *React Router* [online]. [cit. 2016-05-05]. Dostupné z: https://github.com/reactjs/react-router

[15] ABRAMOV, D.: *Redux* [online]. React Community, ©2015, [cit. 2016-05-05]. Dostupné z: http://redux.js.org/

[16] Facebook Inc.: *Flux: Application architecture for building user interfaces* [online]. ©2004-2015, [cit. 2016-05-05]. Dostupné z: http://facebook.github.io/flux/

[17] CZAPLICKI, E.: *The Elm Architecture* [online]. ©2014, [cit. 2016-05-05]. Dostupné z: https://github.com/evancz/elm-architecture-tutorial/

[18] SCHOLZ, F.; DONALDSON, B.; SWISHER, J.; aj.: *CSS* [online]. Mozilla Developer Network and individual contributors, 2016, [cit. 2016-04-29]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[19] Tutorial Republic: *What is Bootstrap* [online]. ©2016, [cit. 2016-04-29]. Dostupné z: http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php

[20] KNIGHT, K.: *Responsive Web Design: What It Is and How To Use It* [online]. 2011, [cit. 2016-04-29]. Dostupné z: https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/

[21] SCHOLZ, F.; MCNALLY, M.; PORTIOLI, G.; aj.: *Details of the object model* [online]. Mozilla Developer Network and individual contributors, 2016, [cit. 2016-04-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

[22] Facebook Inc.: *Component Specs and Lifecycle* [online]. ©2013-2016, [cit. 2016-05-07]. Dostupné z: https://facebook.github.io/react/docs/component-specs.html

[23] ABRAMOV, D.: *Redux Thunk* [online]. ©2015, [cit. 2016-05-06]. Dostupné z: https://github.com/gaearon/redux-thunk

[24] Meteor Development Group Inc.: *Meteor Tracker* [online]. ©2016, [cit. 2016-05-06]. Dostupné z: https://atmospherejs.com/meteor/tracker

[25] Google: *GestureDetector* [online]. [cit. 2016-05-08]. Dostupné z: http://developer.android.com/reference/android/view/GestureDetector.html

[26] Apple Inc.: *Gesture Recognizers* [online]. ©2016, [cit. 2016-05-08]. Dostupné z: https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer_basics/GestureRecognizer_basics.html

[27] DEVERIA, A.: *Can I use flexbox?* [online]. [cit. 2016-05-08]. Dostupné z: http://caniuse.com/#feat=flexbox

[28] Stack Exchange Inc: *2015 Developer Survey* [online]. ©2016, [cit. 2016-05-08]. Dostupné z: http://stackoverflow.com/research/developer-survey-2015

[29] SHARMA, G.: *Insight of Mobile App Development Process [Infographic]* [online]. Nine Hertz, ©2008-2016, [cit. 2016-05-08]. Dostupné z: http://theninehertz.com/insight-of-mobile-app-development-process

[30] Refsnes Data: *CSS3 Introduction* [online]. ©1999-2016, [cit. 2016-05-08]. Dostupné z: http://www.w3schools.com/css/css3_intro.asp

[31] GLOVER, R.: *What is ES2015?* [online]. 2015, [cit. 2016-04-28]. Dostupné z: https://themeteorchef.com/blog/what-is-es2015/

[32] NEŠETŘIL, J.: *JavaScript na serveru: Začínáme s Node.js* [online]. 2010, [cit. 2016-04-27]. Dostupné z: https://www.zdrojak.cz/clanky/javascript-na-serveru-zaciname-s-node-js/

Příloha A

Slovník

CSS3 CSS3 je nejnovějším standardem CSS.[30].

EcmaScript 2015 EcmaScript 2015 je oficiální název poslední verze programovacího jazyka Javascript, která byla vydána v roce 2015.[31].

MongoDB MongoDB je open-source dokumentová databáze, poskytující vysoký výkon, vysokou dostupnost a automatické škálování.[13].

Node.js Node.js je vysoce výkonné, událostmi řízené prostředí pro programovací jazyk Javascript, umožňující kompilaci zdrojového kódu pro běh na straně serveru.[32].

Příloha B

Seznam použitých zkratek

AMD Asynchronous Module Definition.

CSS Cascading Style Sheets.

DDP Distributed Data Protocol.

DI Dependency Injection.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

IDE Integrated Development Environment.

JSON JavaScript Object Notation.

npm Node Package Manager.

OS Operační systém.

SASS Syntactically Awesome Stylesheets.

UML Unified Modeling Language.

URL Uniform Resource Locator.

XML Extensible Markup Language.

Příloha C

Obsah přiloženého CD

- `README.txt` ..... stručný popis obsahu CD
- `bin` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF