VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

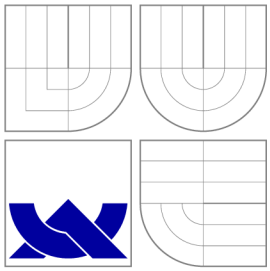
SOCIÁLNÍ SÍŤ PRO KOLEKTIVNÍ SPORTY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

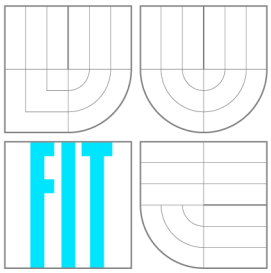
AUTOR PRÁCE
AUTHOR

Bc. IVO ADAM

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SOCIÁLNÍ SÍŤ PRO KOLEKTIVNÍ SPORTY

SOCIAL NETWORK FOR TEAM SPORTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVO ADAM

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2015

Abstrakt

Tato diplomová práce se zabývá vývojem webové sociální sítě pro organizaci amatérských utkání v kolektivních sportech. Je implementována v jazyce JavaScript. Klientská část je napsána v aplikačním rámci AngularJS. Serverová část je postavena na platformě Node.js, využívá aplikační rámec Express a NoSQL databázi MongoDB. Ke zdrojům uloženým na serveru lze přistupovat prostřednictvím aplikačního rozhraní REST. Sociální síť integruje vybrané zásuvné moduly nabízené existujícími sociálními sítěmi. Například přihlašovací dialog nebo tlačítka pro sdílení. Vytvořená aplikace funguje ve webových prohlížečích Google Chrome, Mozilla Firefox a Internet Explorer verze 10 a vyšší.

Abstract

This master thesis deals with development of a social network for organizing amateurish matches in collective sports. It is implemented in JavaScript. Client side is written in AngularJS framework. Server side is built on Node.js platform, uses framework Express and NoSQL database MongoDB. Resources stored on server are accessible via REST API. The social network integrates some plugins from existing social networks. For example login dialog or share buttons. The created application supports web browsers Google Chrome, Mozilla Firefox and Internet Explorer version 10 or higher.

Klíčová slova

sociální síť, JavaScript, Node.js, AngularJS, MongoDB

Keywords

social network, JavaScript, Node.js, AngularJS, MongoDB

Citace

Ivo Adam: Sociální síť pro kolektivní sporty, diplomová práce, Brno, FIT VUT v Brně, 2015

Sociální síť pro kolektivní sporty

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ivo Adam
26. května 2015

Poděkování

Děkuji Ing. Radku Burgetovi, Ph.D za vedení mé diplomové práce.

Obsah

1 Úvod **3**

2 World Wide Web **4**
2.1 Komunikační protokol HTTP 4
2.2 Adresa URL 5
2.3 Webové stránky 5
2.4 Webové aplikace 7

3 Tvorba webových aplikací **8**
3.1 Technologie klientské části 9
3.1.1 HTML a DOM 9
3.1.2 CSS 10
3.1.3 JavaScript 11
3.2 Technologie serverové části 13
3.2.1 Aplikační rámce 13
3.2.2 Databáze 14
3.2.3 Architektura REST pro webové API 16

4 Sociální sítě **17**
4.1 Facebook 17
4.2 Twitter 20
4.3 Google+ 21

5 Návrh sociální sítě pro kolektivní sporty **22**
5.1 Specifikace požadavků a případů použití 22
5.2 Architektura 24
5.2.1 Serverová část 24
5.2.2 Klientská část 26
5.3 Uživatelé, registrace a autentizace 28
5.4 Sportovní události 31
5.5 Upozornění a soukromé zprávy 33

6 Implementace a testování sociální sítě pro kolektivní sporty **34**
6.1 Organizace archivu se zdrojovými kódy 35
6.2 Použité knihovny a externí komponenty 36
6.3 Validace formulářů 37
6.4 Filtrování dat 38
6.5 Experimentální ověření funkčnosti 38

7 Závěr **39**

A Přístupové body serverového REST API **42**

B Testovací scénář **44**

Kapitola 1

Úvod

Přestože je sport zdraví prospěšná činnost, existuje dle průzkumu Evropské unie [1] velká část populace, která jej vůbec neprovozuje. Průzkum uvádí, že nejčastějším důvodem, proč člověk nesportuje, je nedostatek času. V České Republice tento důvod uvedlo 52% a v rámci celé EU pak 42% dotázaných. Lidé, kteří by rádi sportovali, ale nejsou nikde organizováni, mohou mít problém při shánění spoluhráčů, respektive protihráčů. Ty shání mezi kamarády, ale často se stává, že nemají čas a nikoho tak nenajdou. Nejeden pokus o zorganizování takové sportovní akce pak končí neúspěchem. Přitom je velmi pravděpodobné, že lidí, kteří by si ten den šli zahrát určitý kolektivní sport je v té dané oblasti více, jenom o sobě nevědí a nemohou se domluvit. V dnešní době, kdy jsou lidé propojeni pomocí internetu, se nabízí možnost domluvy přes web, například pomocí nějaké sociální sítě[1] nebo aplikace pro plánování schůzek. Při snaze použít zmíněné nástroje bylo zjištěno, že pro organizaci takové sportovní akce nejsou vhodně navržené. Sociální sítě sice umožňují vytvářet a spravovat události, ale vyhledávat mezi existujícími událostmi je možné pouze fulltextově a nelze tedy události filtrovat na základě typu, místa, věku účastníků či dalších kritérií. Webové aplikace pro plánování schůzek zase fungují na principu rozesílání pozvánek e-mailem. Je tedy zřejmé, že není možné naplánovat schůzku s cizími lidmi, jejichž e-mailové adresy organizátor nezná. Řešením by mohlo být vytvoření nové sociální sítě navržené pro organizování sportovních akcí. Uživatel této sítě by měl možnost organizovat sportovní akce a vyhledávat mezi nimi podle vhodných kritérií. Vytvořením takové sociální sítě se zabývá tato diplomová práce.

Kapitola 2 vysvětluje princip internetové služby *World Wide Web* a definuje s ní spojené základní pojmy.

Kapitola 3 uvádí přehled technologií používaných pro tvorbu webových aplikací, s důrazem na vývoj v jazyce JavaScript.

Kapitola 4 popisuje existující, celosvětově nejpoužívanější sociální sítě. Vysvětluje jejich typické použití a možnosti integrace do vlastní aplikace.

Kapitola 5 obsahuje návrh nové sociální sítě vhodné pro organizování amatérských sportovních akcí.

Kapitola 6 popisuje implementaci vybraných problémů navržené sociální sítě a experimentální ověření její funkčnosti.

Kapitola 7 hodnotí dosažené výsledky a uvádí možná rozšíření implementované aplikace.

Diplomová práce navazuje na semestrální projekt, ze kterého využívá kapitoly 1 až 4 a část kapitoly 5.

[1] Pojem „Sociální síť“ je v této práci chápán jako webová aplikace se zaměřením na sdílení informací a vzájemnou interakci mezi uživateli.

Kapitola 2

World Wide Web

World Wide Web (WWW, W3)[3] vynalezl v roce 1989 Tim Berners-Lee. Jedná se o distribuovaný systém využívající síť internet pro sdílení dokumentů v různých formátech. Dokumenty může nabízet libovolný počítač připojený k internetu, na kterém běží potřebný software - webový server (Apache, ISS a jiné). K dokumentům uloženým na webových serverech lze přistupovat prostřednictvím klienta - webový prohlížeč (Internet Explorer, Google Chrome a další). Ten dokáže vyžádat dokumenty od serveru a následně je zobrazit. Klíčovým typem dokumentů na webu jsou hypertextové[1] dokumenty napsané v HTML[2], které typicky obsahují textové informace, obrázky (příp. další multimediální data) a hypertextové odkazy na jiné dokumenty na webu. Tyto odkazy mohou být jak lokální (v rámci stejného serveru), tak i globální (na jiný dokument na jiném serveru). Na web lze tedy nahlížet také jako na síť vzájemně propojených zdrojů informací a proto je web často označován jako hypertextový systém. Dnes se o rozvoj jeho standardů stará *World Wide Web Consortium* (W3C), kterému předsedá jeho zakladatel Tim Berners-Lee.

2.1 Komunikační protokol HTTP

Veškerá komunikace mezi webovými klienty a servery se odehrává přes internet, prostřednictvím HTTP[3] [4]. Tento protokol aplikační vrstvy je vystavěn nad protokolem TCP[4] a typicky používá ke komunikaci port číslo 80. HTTP pracuje způsobem dotaz - odpověď a neuchovává informace o předchozích dotazech, proto je označován za bezestavový protokol. Základní dotazovací metody, kterými klient komunikuje se serverem, jsou:

- GET - požadavek na zaslání dokumentu. Nejčastěji používaná metoda.
- HEAD - požadavek na zaslání hlavičky (metadat) dokumentu.
- POST - odeslání uživatelských dat na server, například formulářová data.
- PUT - nahrání dat na server.
- DELETE - smazání dat ze serveru.

[1] Hypertext - text umožňující nesekvenční přístup pomocí sítě vazeb nazývané odkazy.[2]

[2] HTML (*HyperText Markup Language*) - značkovací jazyk pro hypertextové dokumenty. Podrobněji se jím zabývá kapitola 3.1.1

[3] HTTP (*HyperText Transfer Protocol*) - komunikační protokol pro přenos dokumentů na webu.

[4] TCP (*Transmission Control Protocol*) - spojově orientovaný protokol transportní vrstvy internetu.

Odpověď serveru na dotaz klienta se pak skládá ze stavového kódu a případně požadovaného dokumentu. Stavový kód je tříciferné číslo ve tvaru yxx, kde y určuje kategorii odpovědi a xx specifikuje konkrétní variantu z dané kategorie. Možných kategorií je celkem 5:

- $1xx$ - informační charakter
- $2xx$ - úspěšné zpracování požadavku
- $3xx$ - požadovaný dokument se nachází jinde (přesměrování)
- $4xx$ - chyba klienta (neplatný požadavek)
- $5xx$ - chyba serveru (není schopen zpracovat validní požadavek)

Nejběžněji se vyskytují stavové kódy 200 - úspěch, 404 - požadovaný dokument nebyl na serveru nalezen a 503 - server je dočasně nedostupný.

2.2 Adresa URL

Každý dokument musí být možné v rámci webu jednoznačně identifikovat, aby na něj bylo možné odkazovat, respektive aby jej mohl klient od serveru vyžádat. K tomu se používá adresa URL[5] [5]. Například URL *http://en.wikipedia.org/wiki/JavaScript* identifikuje dokument *JavaScript*, který je uložen ve složce *wiki* na serveru s doménovým jménem *en.wikipedia.org*. URL adresa je používána k identifikaci objektů i v jiných internetových službách, ne pouze na webu, proto je nutné na jejím začátku specifikovat komunikační protokol (v případě webu je to *http*). Uvedená problematika adresování objektů v rámci webu není úplná, v praxi je nutné provést překlad doménového jména serveru na IP[6] adresu prostřednictvím systému DNS[7], jehož výklad přesahuje rámec této práce.

2.3 Webové stránky

Webovou stránkou se rozumí množina tematicky zaměřených dokumentů propojených odkazy, které jsou uloženy na stejném webovém serveru. Webové stránky se dají z pohledu zpracování serverem rozdělit do dvou kategorií, na statické a dynamické.

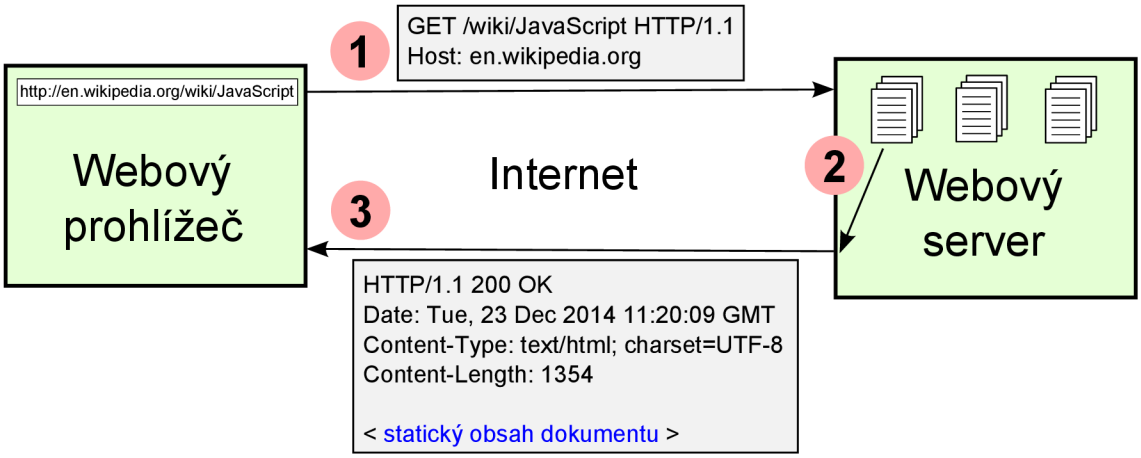
Statické webové stránky

Statická webová stránka má obsah pevně definovaný jejím tvůrcem. Když klient požádá o dokumenty, ze kterých se sestává, tak je server odešle přesně v té podobě, v jaké byly na server uloženy. To odpovídá výše popsaným principům webu, které ilustruje obrázek 2.1. Obrázek zachycuje situaci, kdy uživatel zadal do adresního řádku webového prohlížeče URL adresu stránky o JavaScriptu uložené na serveru wikipedia. Tím prohlížeč vyslal na příslušný server HTTP požadavek na daný hypertextový dokument. Ten server zpracoval a odeslal zpět HTTP odpověď s požadovaným dokumentem.

[5] URL (*Uniform Resource Locator*) - unikátní identifikátor zdroje.

[6] IP (*Internet Protocol*) - protokol síťové vrstvy internetu zajišťující směrování.

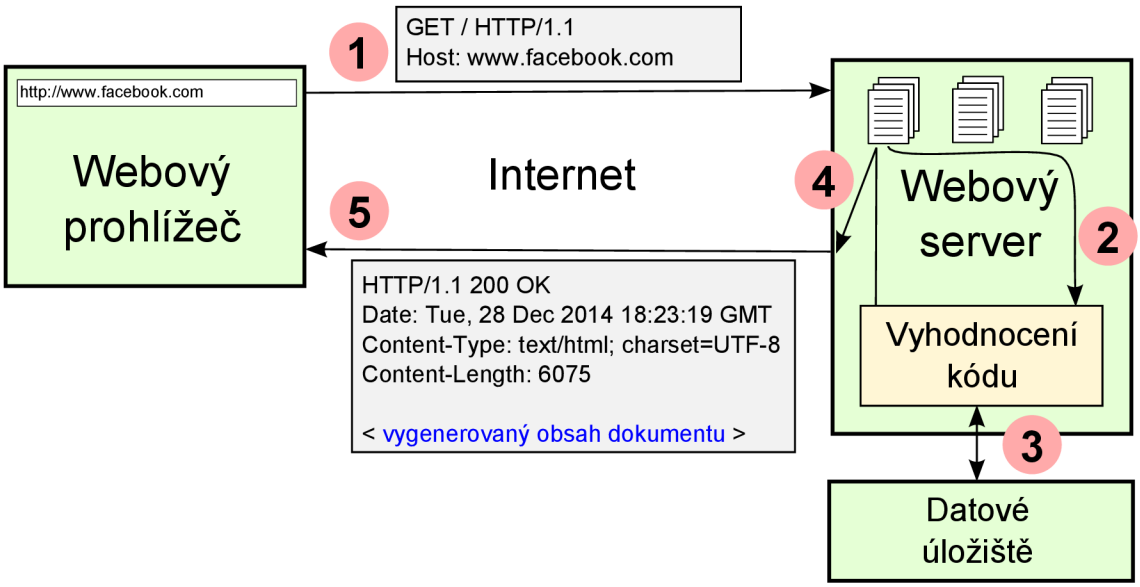
[7] DNS (*Domain Name System*) - distribuovaný systém internetových informací, umožňuje například převádět doménová jména na IP adresy a naopak.



Obrázek 2.1: Znázornění výměny statických dokumentů mezi webovým klientem a serverem pomocí aplikačního protokolu HTTP.

Dynamické webové stránky

Obsah webové stránky, respektive dokumentů ze kterých se skládá, ovšem nemusí být pouze statický a je možné jej generovat dynamicky. To je umožněno díky skriptování na straně serveru. Když klient požádá o dokument, který obsahuje programový kód určitého jazyka, tak server nejdříve kód vyhodnotí a poté klientovi odešle dokument obsahující vypočtená data. Protože server vyhodnocuje tento kód pro každý požadavek zvlášť, je možné vygenerovat rozdílnou odpověď pro stejný požadavek, například na základě času, přístupových práv nebo dotazu do datového úložiště. Princip generování obsahu webových stránek je znázorněn na obrázku 2.2.



Obrázek 2.2: Webový server při požadavku na dokument obsahující kód programovacího jazyka nejprve vyhodnotí daný kód a poté odešle klientovi dokument s vypočtenými daty.

2.4 Webové aplikace

V roce 2004 se web dostal do nové vývojové etapy, označované jako „Web 2.0“. V té době začaly vznikat funkčně propojené množiny dynamických stránek - webové aplikace. Obsah webové aplikace tvoří její uživatelé. Každý uživatel má většinou svůj vlastní účet a na základě oprávnění může přistupovat pouze k určité podmnožině dokumentů. Při tvorbě webové aplikace je tak nutné zajistit autentizaci a autorizaci uživatelů, jejich sezení a bezpečnost.

Autentizace je proces ověření identity uživatele. Nejčastějším způsobem je zadání dvojice uživatelské označení (jméno, email, apod.) a heslo, které je ověřeno vůči předem vytvořenému záznamu v databázi, jež vznikl při zaregistrování uživatele.

Autorizace je proces ověření oprávnění uživatele. Například zda-li má uživatel oprávnění pro přístup k daným dokumentům.

Sezení označuje trvalé spojení mezi klientem a serverem. Ve webové aplikaci je velmi vhodné udržovat stav komunikace s každým uživatelem (například zdali je již přihlášen či nikoliv), na základě kterého server může přizpůsobit svoji odpověď. Ovšem, jak již bylo zmíněno, HTTP je bezestavový protokol a je tedy nutné udržovat kontext komunikace na úrovni samotné aplikace. Běžně se k tomuto účelu používají HTTP cookies[8].

Bezpečnost webové aplikace je nutné zajistit na několika místech, jinak hrozí únik, zneužití či ztráta potenciálně citlivých dat uživatele. Základní předpoklad bezpečné aplikace je vyvarování se bezpečnostních chyb v naprogramovaných skriptech (především neošetřený uživatelský vstup z formulářů). Také je nutné zajistit šifrování komunikace mezi klientem a serverem. Protokol HTTP posílá veškerá data v otevřené podobě, což útočníkovi umožňuje odposlouchávat komunikaci a zjistit tím citlivé údaje (heslo atp.). Šifrování komunikace lze zajistit nahrazením klasického HTTP za HTTPS[9].

Popularita webových aplikací roste především díky všudypřítomnosti webových prohlížečů a s tím související snadné přenositelnosti mezi platformami. Také schopnost aktualizovat a spravovat webové aplikace bez nutnosti distribuovat a instalovat novou verzi software zvlášť u každého uživatele, je jejich velkou výhodou. Díky uvedeným výhodám se některé, dříve pouze desktopové aplikace (např. kancelářský balík), již běžně vytvářejí jako aplikace webové.

Způsobem a technologiemi vytváření webové aplikace se zabývá následující kapitola.

[8] HTTP cookies - technologie umožňující serveru uložit malé množství dat na straně klienta. Tato data pak klient automaticky posílá na server při každém HTTP dotazu.

[9] HTTPS (*HyperText Transfer Protocol Secure*) - nadstavba nad HTTP, která šifruje veškerá přenášená data pomocí SSL/TLS, což je kryptografický protokol fungující na principu asymetrické šifry.

Kapitola 3

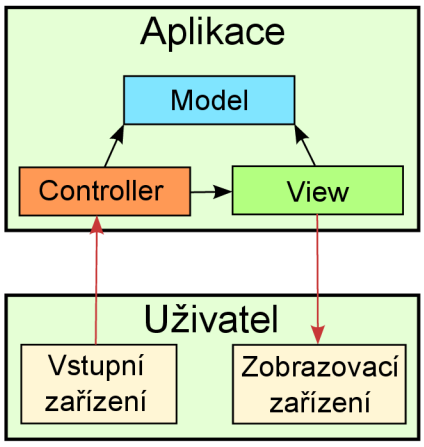
Tvorba webových aplikací

Tato kapitola shrnuje problematiku vytváření webových aplikací. Představuje základní klientské a serverové technologie i některé populární aplikační rámce[1] a knihovny[2], především pak ty, které budou použity v této práci při tvorbě sociální sítě pro kolektivní sporty. Protože téměř všechny webové aplikační rámce jsou postaveny na architektuře *Model-View-Controller*, bude zde tato architektura stručně popsána.

Architektura MVC (Model-View-Controller)

Architektura MVC[13] rozděluje aplikaci na tři logické části - Model, View a Controller. Model reprezentuje data, se kterými aplikace pracuje. View zobrazuje uživatelské rozhraní a vizualizuje data uložená v modelu. Controller má pak na starosti zpracování událostí a logiku aplikace. Výhodou rozdělení jsou snazší úpravy jednotlivých částí, které se navíc téměř nepromítají do částí ostatních. Závislosti jednotlivých komponent jsou patrné z obrázku 3.1. Je důležité si uvědomit, že model není závislý na žádné komponentě, což je důležitá vlastnost MVC architektury. Tok událostí v aplikaci probíhá následovně:

1. Uživatel vykoná nějakou akci na uživatelském rozhraní
2. Akce je zachycena Controllerem
3. Controller rozhodne, jak na akci zareagovat, a obvykle změní nějaké hodnoty v Modelu nebo přímo ovlivní View
4. View zobrazí změny uživateli



Obrázek 3.1: Blokové schéma architektury MVC se znázorněním vazby na uživatele.

[1] Aplikační rámec (angl. *Framework*) - univerzální a znovupoužitelná softwarová struktura, která usnadňuje řešení často se opakujících problémů.

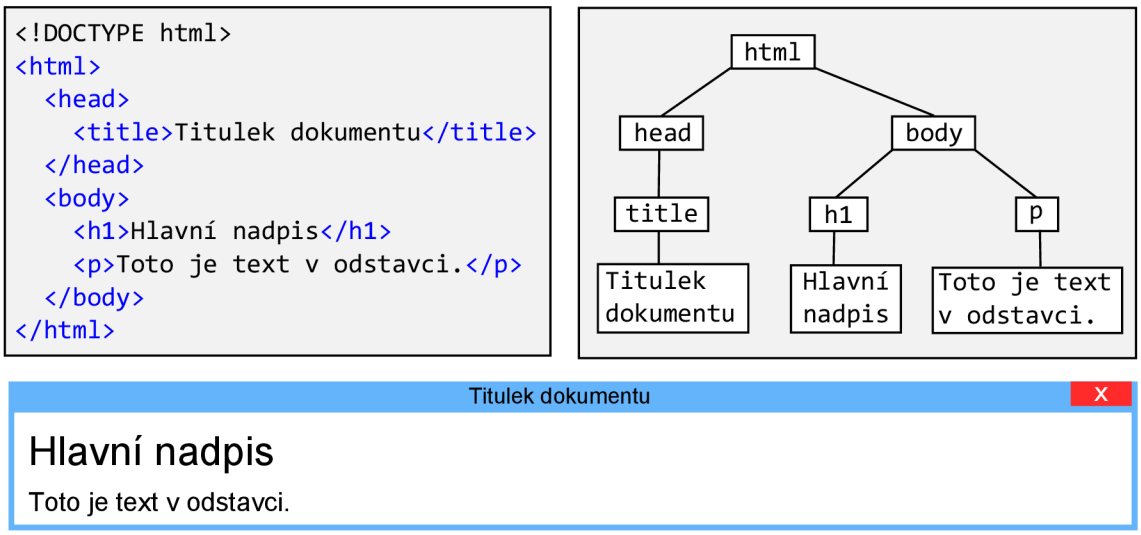
[2] Knihovna - sada funkcí (příp. tříd a objektů), která může být sdílena více programy.

3.1 Technologie klientské části

Klientská část aplikace sestává z dokumentů stažených ze serveru. Obsah dokumentů interpretuje webový prohlížeč a jsou-li v podporovaném formátu, tak jejich obsah vhodně zobrazí uživateli. Základ všech webových stránek, respektive aplikací, jsou dokumenty napsané v HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) a JS (*JavaScript*).

3.1.1 HTML a DOM

HTML[12] je značkovací jazyk, který vychází ze SGML[3]. Používá se pro definici obsahu hypertextových dokumentů. Pomocí HTML značek a jejich vlastností je možné přiřadit jednotlivým částem obsahu dokumentu význam. Například značka `<h1>` vyznačuje hlavní nadpis a značka `<p>` odstavec. Většina značek je párová, to znamená, že musí vyznačovanou část dokumentu obklopovat z obou stran, přičemž značka uzavírací (ta v textu více napravo) obsahuje navíc znak `/` za první úhlovou závorkou (viz. příklad HTML dokumentu na obrázku 3.2). Existují ale i značky nepárové, které se typicky používají pro vkládání externích zdrojů do HTML dokumentu, například nepárová značka `<img>` slouží pro vložení obrázku. Množina povolených značek závisí na konkrétní verzi HTML. Dnes je aktuální verze 5, jejíž finální specifikace byla vydána v říjnu 2014 a podporují ji všechny moderní webové prohlížeče. Struktura dokumentu v HTML je pevně daná. Nejdříve je nutné deklarovat typ dokumentu, pro HTML5 se to provede značkou `<!DOCTYPE html>`. Poté vždy následuje značka `<html>` reprezentující celý dokument, která v sobě obsahuje právě dvě značky, `<head>` pro hlavičku dokumentu a `<body>` pro tělo dokumentu. Do hlavičky se vepisují metadata o dokumentu, jako například kódování, titulek, autor, popis atp. Tělo pak zahrnuje veškerý obsah dokumentu. Když webový prohlížeč zpracovává dokument v jazyce HTML, generuje z něj DOM[4]. Rozhraní tohoto objektového modelu umožňuje programům a skriptům přistupovat k obsahu HTML dokumentu a také modifikovat jeho obsah, strukturu a styl.



Obrázek 3.2: HTML dokument, jeho reprezentace ve formě DOM a výsledek, který zobrazí webový prohlížeč.

[3]SGML (*Standard Generalized Markup Language*) - univerzální metajazyk, který dovoluje definovat značkovací jazyky.

[4]DOM (*Document Object Model*) - objektový model dokumentu se stromovou strukturou.

3.1.2 CSS

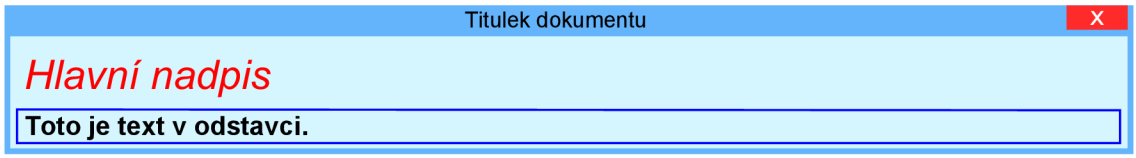
CSS[12] je jazyk pro definici vzhledu dokumentu vyznačeného pomocí jazyka HTML, nebo jemu příbuzného. Umožňuje vyznačeným částem dokumentu nastavit barvu, velikost, pozici, okraje atp. Dokument zapsaný v jazyku CSS sestává z pravidel. Každé pravidlo je tvořeno dvěma částmi, selektorem a blokem s deklaracemi vlastností. Selektor určuje, na které prvky HTML dokumentu se dané pravidlo vztahuje. Blok deklarací uvádí konkrétní vlastnosti, jež upravují vzhled daných prvků. Každá vlastnost má pevně stanovené jméno a hodnoty, kterých může nabývat. Přípustné typy selektorů a vlastností jsou definovány ve specifikaci CSS. Aby se pravidla definovaná v CSS dokumentu promítla do vzhledu HTML dokumentu, je nutné v hlavičce HTML dokumentu na daný CSS dokument odkázat nepárovou značkou `<link>`, jak je vidět na obrázku 3.3.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titulek dokumentu</title>
    <link href="style.css">
  </head>
  <body>
    <h1>Hlavní nadpis</h1>
    <p>Toto je text v odstavci.</p>
  </body>
</html>
```

```
body { background-color: #D5F6FF; }

h1 {
    color: red;
    font-style: italic;
}

p {
    border: 1px blue solid;
    font-style: bold;
}
```



Obrázek 3.3: HTML dokument a k němu připojený CSS dokument, který nastavuje vzhled značkám `<body>`, `<h1>` a `<p>`. Pod dokumenty je výsledek, který se zobrazí ve webovém prohlížeči. Zobrazený příklad předpokládá, že CSS dokument se jmenuje *style.css* a je umístěn ve stejném adresáři jako HTML dokument.

Bootstrap

Bootstrap[5] je populární aplikační rámec, který sestává ze dvou částí. První je napsána v jazyku CSS a druhá v JavaScriptu. Tento odstavec popisuje pouze CSS část, informace o JavaScriptové části jsou uvedeny v kapitole 3.1.3. CSS část Bootstrapu výrazně usnadňuje stylování dokumentů pomocí sady připravených CSS pravidel. Umožňuje například velmi snadno vytvořit mřížkové rozložení stránky (angl. *grid layout*), které se navíc může dynamicky přizpůsobovat velikosti zobrazovacího zařízení, což je vhodné při tvorbě responzivního designu. Dále pomáhá například s typografií nebo formátováním tabulek, formulářů, navigací či tlačítek.

[5]Bootstrap - `http://getbootstrap.com/`

3.1.3 JavaScript

JavaScript[16] je prototypově založený programovací jazyk. Kód v jazyce JavaScript lze vkládat přímo do HTML dokumentů, protože webové prohlížeče obsahují interpret tohoto jazyka. Pro vyznačení JavaScriptového kódu se používá HTML značka `<script>`. Jakmile na tuto značku webový prohlížeč narazí, začne její obsah interpretovat jako kód zapsaný v JavaScriptu. Použití JavaScriptu může být různorodé, ale často se používá na obsluhu událostí a uživatelských akcí (stisknutí tlačítka atp.), ovládání prohlížeče, dynamickou změnu obsahu dokumentu pomocí manipulace s DOM anebo na asynchronní komunikaci se serverem pomocí technologie AJAX[6]. Nepříjemnou věcí na prohlížečových interpretech JavaScriptu je fakt, že se ne vždy drží specifikace jazyka a stává se, že různé interprety vykonají stejný kód s odlišným výsledkem. Programátor je pak nucen na toto myslet a psát různý kód pro různé interprety, respektive prohlížeče.

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
    alert("data.txt přijata!");
}

xmlhttp.open("GET", "data.txt", true);
xmlhttp.send();
```

Obrázek 3.4: Ukázka asynchronní komunikace se serverem pomocí technologie AJAX v jazyce JavaScript. Klient odešle asynchronně požadavek na soubor *data.txt*, jakmile soubor od serveru dorazí, tak klient otevře dialogové okno a vypíše zprávu „*data.txt přijata!*“.

Bootstrap

JavaScriptová část aplikačního rámce Bootstrap usnadňuje vytváření interaktivních prvků uživatelského rozhraní. Obsahuje komponenty, které například výrazně zjednodušují tvorbu rozbalovací navigace, tlačítek uchovávajících stav, záložek, vysouvacích prvků nebo modálních dialogových oken.

jQuery

Knihovna jQuery[7] je pravděpodobně nejznámější JavaScriptová knihovna na světě. Dalo by se říci, že vytváří abstraktní vrstvu nad klientským JavaScriptem. Nabízí funkce, které výrazně usnadňují věci jako je průchod a manipulace s HTML dokumentem, zpracování událostí, animace nebo AJAX. Nespornou výhodou jQuery je také to, že odstiňuje různorodost implementace interpretů ve webových prohlížečích. Kód tak stačí napsat pouze jednou a jQuery zajistí, aby správně fungoval ve všech majoritních webových prohlížečích.

[6] AJAX (*Asynchronous JavaScript And XML*) - technologie pro asynchronní komunikaci se serverem. Umožňuje například změnit obsah v prohlížeči zobrazeného hypertextového dokumentu aniž by musel být načten celý znovu.

[7] jQuery - `http://jquery.com/`

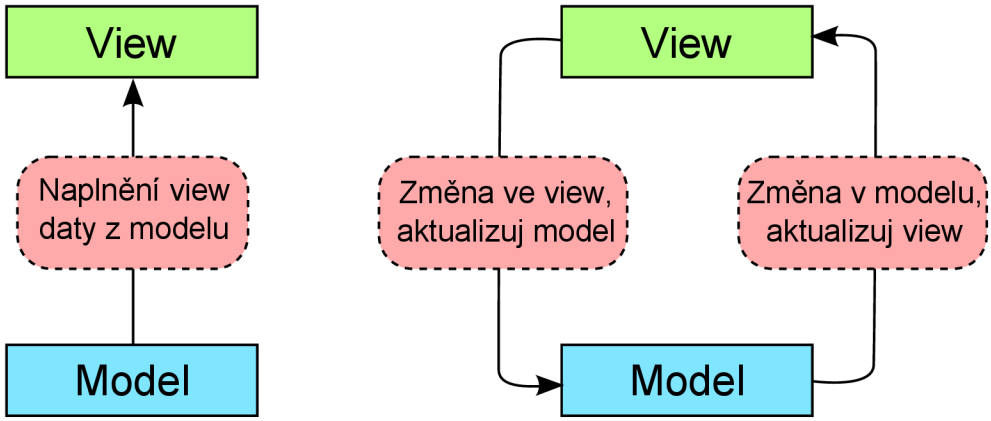
AngularJS

Klientský JavaScriptový aplikační rámec AngularJS[8] je postaven na architektuře MVC. Vyvíjí jej společnost Google a obsahuje řadu podpůrných technologií pro vytváření webových aplikací. Za nejužitečnější části Angularu se považuje *obousměrný Data-Binding*, *Dependency Injection* a *direktivy*.

Obousměrný Data-Binding automaticky zajišťuje synchronizaci mezi modelem a view, obě části se tedy neustále nacházejí ve stejném, konzistentním, stavu.

Dependency Injection (DI) je návrhový vzor usnadňující správu závislostí mezi jednotlivými softwarovými komponentami. Angular obsahuje systém implementující DI, jež automaticky řeší závislosti v celé vyvíjené aplikaci. Programátor pouze definuje, na kterých službách je daná funkce závislá a Angular je funkci předá jako parametry.

Direktivy rozšiřují množinu povolených HTML značek a atributů. Například když je direktivou `ng-model=''x''` označeno textové pole formuláře, tak se jeho obsah začne automaticky synchronizovat s proměnnou `x`. Angular obsahuje celou řadu předpřipravených direktiv, ale je možné definovat i vlastní.



Obrázek 3.5: Rozdíl mezi jednosměrným (vlevo) a obousměrným (vpravo) Data-Bindingem.

Angular umožňuje tvorbu tzv. *Single Page Applicaition* (SPA). Zatímco klasická webová aplikace, tj. *Multi Page Application* (MPA), sestává z mnoha dokumentů propojených odkazy, SPA obsahují pouze jeden dokument, jehož obsah je dynamicky měněn pomocí technologie AJAX. Jakmile se ze serveru stáhne onen jeden dokument, v tu chvíli je u klienta celá aplikace. Od serveru pak klient nežádá další stránky, ale pouze data v nějakém vhodném formátu, například v JSON[9]. Komunikace se serverem probíhá přes jeho API[10] a celková zátěž serveru je minimalizována. Mezi další výhody SPA patří například rychlejší odezva na uživatelskou interakci. Nevýhodou je mírné zesložitění aplikace a absolutní závislost na jazyku JavaScript, který nemusí být v prohlížeči vždy dostupný (např. jej uživatel zakázal).

[8] AngularJS - `https://angularjs.org/`
[9] JSON (*JavaScript Object Notation*) - textový formát pro serializaci dat.
[10] API (*Application Programming Interface*) - aplikační rozhraní.

3.2 Technologie serverové části

Serverová část aplikace slouží jako inteligentní sdílené úložiště dokumentů a dat. Přijímá od klientů požadavky na dokumenty a odesílá jim je v případě, že existují a klient má právo k nim přistupovat. Dokumenty jsou před odesláním obvykle přizpůsobeny danému požadavku pomocí předem připraveného programového kódu. Zatímco na straně klienta lze programovat pouze v jednom[11] jazyce - JavaScriptu, na serveru je možnost volby programovacího jazyka podstatně větší. Programovat lze v jazyce PHP[12], Ruby, Python, Java, C#, JavaScript a mnoha dalších. Důležité je, že programový kód se zpracuje vždy na straně serveru a klientovi se odešle pouze výstup zpracování ve formátu, který je schopen interpretovat, tedy HTML, CSS, JavaScript, případně nějaký textový serializační formát (např. JSON). Určit který z programovacích jazyků je pro vývoj serverové části aplikace nejvhodnější objektivně nelze, vždy záleží na preferencích programátora. V počtu nasazení však výrazně dominuje PHP, který je použit v 82%[6] webových aplikací.

3.2.1 Aplikační rámce

Vývoj serverové části webové aplikace většinou vychází z nějakého z existujících aplikačních rámců. Aplikace se píše čistě jen v samotném programovacím jazyce pouze v případě, že jsou na ni kladeny nějaké výjimečné požadavky, například na bezpečnost nebo efektivitu. Aplikačních rámců existuje pro každý z podporovaných jazyků mnoho, mezi populární patří například Ruby on Rails[13] (Ruby), Spring[14] (Java), ASP.NET[15] (C#) nebo Express[16] (JavaScript). Všechny aplikační rámce mají stejný cíl - usnadnit a urychlit vývoj aplikace. K dosažení tohoto cíle nabízejí vývojáři typicky následující:

- MVC architektura - oddělení uživatelského rozhraní, logiky aplikace a dat.
- Ladící nástroje - integrují je, nebo jejich použití usnadňují.
- Zabezpečení - automatická eliminace bezpečnostních děr jež vzniky nepozorností/neznalostí programátora.
- Databázová vrstva - abstraktní vrstva nad rozhraním databáze, která usnadňuje její použití (např. ORM[17]).
- Princip DRY (*Don't Repeat Yourself*) - psát stejný kód vícekrát je špatně. Dobrý kód je znovupoužitelný.

Některé aplikační rámce mohou být zaměřeny na konkrétní části z výše uvedeného, případně mohou obsahovat další podporu, nebo jim naopak může některá z uvedených částí chybět. Vybrat vhodný aplikační rámec není snadné, vždy záleží na typu řešeného projektu a preferenci programátora. Sociální síť, jež je vyvíjena v rámci této práce, bude naprogramována s využitím aplikačního rámce Express.

[11] Nebereme-li v potaz různé zásuvné moduly, které přidávají podporu dalších jazyků, jako je např. Flash, Silverlight či Java.

[12] PHP (rekurzivní zkratka *PHP: Hypertext Preprocessor*)

[13] Ruby on Rails - `http://rubyonrails.org/`

[14] Spring - `http://spring.io/`

[15] ASP.NET - `http://www.asp.net/`

[16] Express - `http://expressjs.com/`

[17] ORM (*Object-Relational Mapping*) - Softwarová vrstva, která zařídí převod objektů do relační databáze a zpět.

Express a Node.js

Express je minimalistický webový aplikační rámec napsaný v jazyce JavaScript. Přináší podporu pro MVC architekturu, routování (obousměrný překlad mezi URL adresou a akcí controlleru), testování a kešování (zrychlení komunikace mezi klientem a serverem pomocí vyrovnávací paměti). Ve srovnání s ostatními aplikačními rámci, uvedenými v této kapitole, je Express velmi odlehčený. Jeho výhoda spočívá především v rychlosti a možnostech přizpůsobení pomocí řady modulů. Aby bylo možné na serveru spustit Express, musí na něm být nainstalovaný software Node.js.

Node.js[14] je JavaScriptová platforma postavená na V8[18], která slouží pro vytváření rychlých a škálovatelných síťových aplikací. Obsahuje, mimo jiné, modul HTTP, který umožňuje vytvořit webový server. Díky webovému serveru v Node.js lze vyvíjet serverovou část aplikace v JavaScriptu, což přináší celé aplikaci jednotnost. Klient i server používají stejný programovací jazyk, mohou sdílet funkce, knihovny apod. Webový server v Node.js pracuje pouze v jednom vlákně, což je výrazný rozdíl oproti častějšímu přístupu, kdy webový server vytváří nové vlákno pro zpracování každého požadavku. Aby nedocházelo k „zamrznutí“ serveru při čekání na vstupně/výstupní operaci tak se zpracovává každá asynchronně. Součástí nástrojů spojených s Node.js je balíčkovací systém npm[19], který obsahuje již více než 115 tisíc různých balíčků. Jeden balíček odpovídá jednomu znovupoužitelnému softwarovému projektu či rozšíření souvisejícím s vývojem webových aplikací. Například zmíněný aplikační rámec Express je obsažen v balíčku `express` a lze jej přes systém npm nainstalovat příkazem `npm install express`.

```
var express = require('express');
var app = express();

app.get('/hello', function (req, res) {
  res.send('Hello World!')
});

app.listen(3000);
```

Obrázek 3.6: Vytvoření jednoduchého webového serveru v Express na platformě Node.js. Server naslouchá na portu 3000 a odpoví zprávou „*Hello World!*“ při obdržení požadavku GET na dokument *hello*. Při požadavku na jiný dokument odpoví chybovým kódem 404 - dokument nenalezen.

3.2.2 Databáze

Při vytváření webové aplikace je nezbytné ukládat na straně serveru data vytvořená používáním aplikace. K tomuto účelu se používají databáze, které jsou tvořeny dvěma částmi. Jedna část je prostor pro ukládání dat alokovaný na paměťovém médiu a druhá část je SŘBD[20]. Databáze lze z hlediska způsobu ukládání dat rozdělit do dvou skupin, na relační a NoSQL databáze.

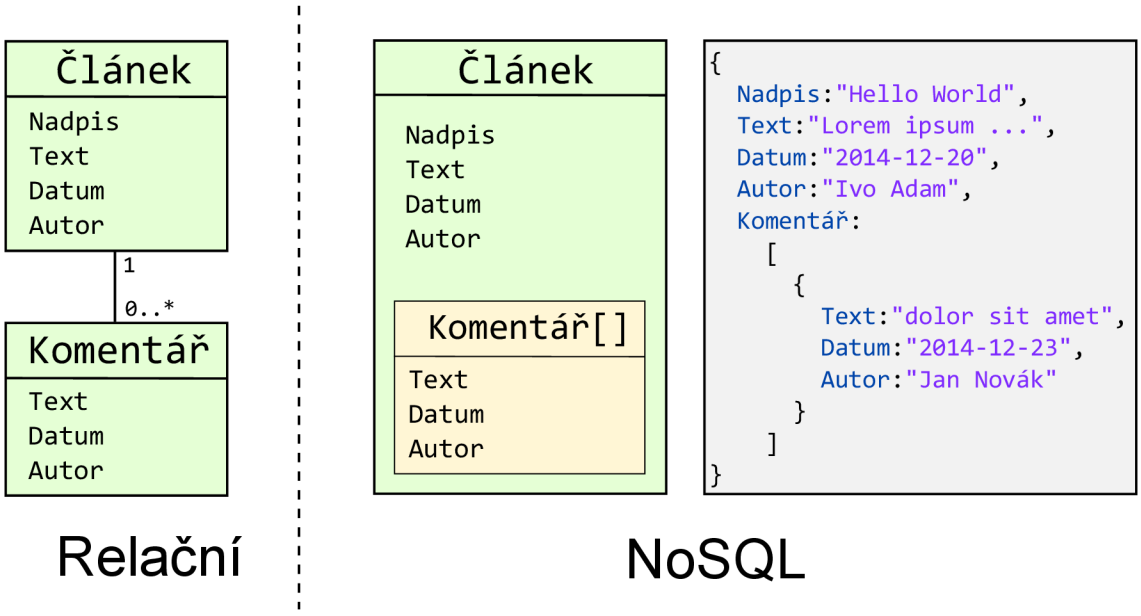
[18] V8 (*V8: JavaScript Engine*) - interpret jazyka JavaScript vyvinutý firmou Google pro jejich webový prohlížeč Chrome.

[19] npm (*Node Package Manager*) - umožňuje vývojářům sdílet a spravovat jejich znovupoužitelný kód.

[20] SŘBD (*Systém řízení báze dat*) - softwarové vrstva zajišťující přístup k uloženým datům a manipulaci s nimi.

Relační databáze jsou postaveny na pevném matematickém základu a to na podmnožině kartézského součinu množin - relaci. Základní stavební prvek relační databáze je tabulka. Její sloupce se nazývají atributy. Každý atribut má předem určený datový typ a doménu, což je množina hodnot, jakých může nabývat. Řádek tabulky je nazýván záznam a slouží pro uložení dat. Důležitým pojmem relačních databází je primární a cizí klíč. Primární klíč souží pro jednoznačné určení záznamu. Označuje se tak sloupec (příp. kombinace sloupců), ve kterém je zaručeno, že hodnoty v něm obsažené budou různé pro každý řádek. Cizí klíč je označení pro sloupec, který má stejný datový typ jako primární klíč z jiné tabulky. Tyto sloupce - primární a cizí klíč, pak mohou vytvořit vazbu mezi danými tabulkami. Pro manipulaci s daty se v relačních databázích používá jazyk SQL[21]. Relační databáze podporují transakce a zaručují ACID[22].

NoSQL databáze vznikly v reakci na stále se zvyšující požadavky na databáze. Jako je například decentralizace uložených dat, úmyslná redundance dat pro zvýšení odolnosti, časté změny schématu, vysoká míra škálovatelnosti nebo problematické datové typy - objekty, grafy, atp. NoSQL databáze podporují nerelační datový model a distribuovanou architekturu. Typů NoSQL databází existuje více, např. dokumentová, grafová, typu klíč-hodnota a další. Typ databáze určuje, jakým způsobem se budou ukládat data. Například dokumentová databáze ukládá všechna data do dokumentů ve formátu JSON (příp. jiném serializačním formátu). Jeden dokument odpovídá jednomu záznamu. Protože dodržování modelu ACID není jednoduché - omezuje změny dat a rychlost databáze, není jeho podpora do NoSQL databází zahrnuta. Místo toho je typicky dodržován model BASE[23]. Pro manipulaci s daty se většinou nepoužívá jazyk SQL, ale objektově orientované API.



Obrázek 3.7: Rozdíl mezi návrhem relační a NoSQL dokumentové databáze. Dokument ve formátu JSON zobrazuje, jak by mohl vypadat jeden záznam v dokumentové databázi.

[21] SQL (*Structured Query Language*) - dotazovací jazyk pro definici a manipulaci dat.

[22] ACID (*Atomicity, Consistency, Isolation, Durability* - zajišťuje atomičnost transakcí, bezpečné souběžné provádění transakcí, trvalost provedených transakcí a konzistentní stav dat uložených v databázi.)

[23] BASE (*Basicaly Available, Soft state, Eventualy consistency*) - aplikace je v podstatě po celou dobu dostupná, vždy v nějakém známém stavu, byť nemusí být vždy ve stavu konzistentním.

MongoDB

NoSQL databáze MongoDB[24] je napsána v programovacím jazyce C++. Jedná se o dokumentovou databázi, jejíž jeden záznam sestává z dvojic klíč-hodnota zapsaných ve formátu JSON, který je interně uložen ve formátu BSON[25]. Hodnotou může být nejen základní datový typ, ale i pole, jiný dokument, případně pole dokumentů. Tím lze jednotlivé dokumenty do sebe vnořovat, jak je vidět na obrázku 3.7. Tři klíčové vlastnosti, na kterých si MongoDB zakládá, jsou vysoký výkon, vysoká dostupnost a automatická horizontální škálovatelnost databáze.

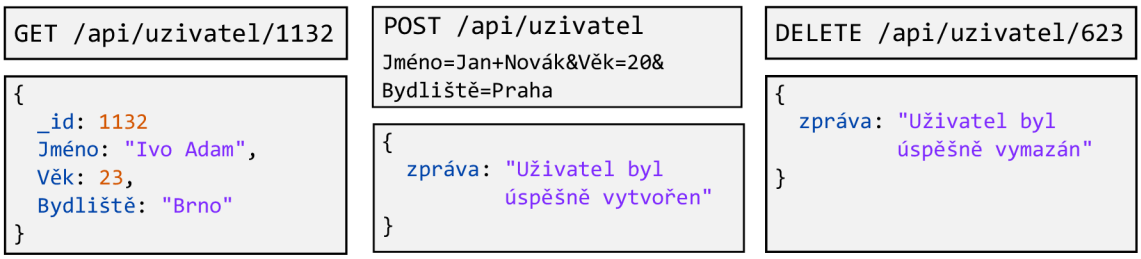
Na platformě Node.js existuje pro MongoDB softwarová mezivrstva Mongoose[26]. Ta umožňuje napevno definovat schéma databáze, provádět automatické konverze mezi datovými typy, automaticky validovat data a celkově usnadňuje práci s databází. Mongoose je dostupná prostřednictvím systému npm jako balíček `mongoose`.

3.2.3 Architektura REST pro webové API

Representational State Transfer (REST) je architektura, která specifikuje jak pomocí standardních HTTP metod vytvořit, číst, editovat nebo odstranit data ze serveru. Základní požadavek je, aby každý zdroj dat na serveru byl identifikovatelný pomocí adresy URL. Pokud server toto dodrží a implementuje rozhraní jako REST, může s ním pak po síti snadno komunikovat libovolné zařízení (počítač, telefon, kamera, televizor atp.). Manipulace s daty probíhá následujícími metodami:

- GET - získání (čtení) dat.
- POST - vytvoření dat.
- PUT - editace dat.
- DELETE - smazání dat.

Data, která server zasílá jako odpověď na uvedené metody (HTTP požadavky), jsou obvykle serializována ve formátu JSON nebo XML[27]. Příklad komunikace prostřednictvím REST API je na obrázku 3.8.



Obrázek 3.8: Získání dat o uživateli s identifikačním číslem 1132 (vlevo), vytvoření nového uživatele (uprostřed) a odstranění uživatele s číslem 623 (vpravo) pomocí REST API. Horní z obrázků vždy ukazuje HTTP požadavek a spodní pak odpověď serveru ve formátu JSON.

[24] MongoDB - `http://www.mongodb.org/`
[25] BSON - binární JSON. Navržen pro efektivní využití paměťového prostoru.
[26] Mongoose - `http://mongoosejs.com/`
[27] XML (*eXtensible Markup Language*) - značkovací jazyk, často používaný pro serializaci dat.

Kapitola 4

Sociální sítě

Jako sociální síť je v této práci chápána webová aplikace se zaměřením na sdílení informací a vzájemnou interakci mezi jejími uživateli. Obvykle umožňuje registrovaným aktérům (jednotlivci, organizaci) navázat spojení s ostatními aktéry, komunikovat s nimi a sdílet informace. V dnešní době mají nejpoužívanější sociální sítě často stovky miliónů aktivních uživatelů a jsou využívány k širokému spektru činností. Dále jsou v této kapitole popsány celosvětově nejpopulárnější sociální sítě.

4.1 Facebook

Facebook[1] umožňuje navázat kontakt s ostatními lidmi a sdílet s nimi příspěvky, což mohou být textové zprávy, fotografie, videa atp. Původní záměr Facebooku byl ale jiný, jednalo se o webový systém pro studenty Harvardovy univerzity, který založil v roce 2004 její student Mark Zuckerberg. Protože popularita Facebooku na univerzitě rostla, byl postupně rozšiřován a zpřístupňován pro studenty dalších univerzit, včetně těch zahraničních. V srpnu roku 2006 byl Facebook otevřen pro veřejnost, jediná podmínka byla věk alespoň 13 let. Od té doby se počet uživatelů Facebooku stále zvyšuje, koncem roku 2014 se na něj přihlásilo alespoň jednou měsíčně 1,35 miliardy uživatelů. Facebook se tak stal celosvětově nejpoužívanější sociální sítí.

Obrázek 4.1: Logo Facebooku.

Po přihlášení do sociální sítě Facebook může uživatel dělat celou řadu činností, mezi ty nejdůležitější patří následující:

- **Správa veřejného profilu** - Každý účet je na Facebooku reprezentován veřejným profilem, do kterého může uživatel doplnit osobní informace, záliby, fotografie, videa a další. Možnosti vyplnění profilu jsou velké a je pouze na uživateli, co vše zveřejní.
- **Navázání spojení s jiným uživatelem** - Součástí uživatelského profilu je síť kontaktů nazývaná *přátelé*. O přidání do přátel je možno požádat jiného libovolného uživatele, jestliže ten žádost příjme, stávají se z těchto uživatelů přátelé.
- **Komunikace s ostatními uživateli** - Komunikovat lze prostřednictvím textových zpráv anebo video hovoru. Komunikace se mohou zúčastnit dva a více uživatelů,

[1] Facebook - `https://www.facebook.com/`

neplatí zde tedy omezení pouze na komunikaci v rámci dvojice, jako například u SMS zpráv.

- **Vložení příspěvku** - Pro vkládání příspěvků slouží prostor v uživatelském profilu nazývaný *zeď*. Při vkládání příspěvku lze nastavit, kdo k němu bude mít přístup. Základní možnosti nastavení přístupu jsou přátelé anebo veřejnost.

- **Komentování příspěvku** - Každý příspěvek lze okomentovat anebo označit jako *To se mi líbí*. Komentář může být text, obrázek, video případně kombinace uvedeného. Označovat pomocí *To se mi líbí* lze nejen příspěvky, ale i jejich komentáře. Čím více označení příspěvek získá, tím je považován za důležitější.

- **Organizace událostí** - Tato část je z hlediska této práce důležitá, proto bude popsána podrobněji. Na Facebooku může uživatel vytvořit událost anebo avizovat, že se nějaké události vytvořené jiným uživatelem zúčastní. Při vytváření události je nutné vyplnit její název, popis, čas a místo konání a vybrat zdali je událost veřejně přístupná, nebo soukromá. Kdyby chtěl uživatel tyto události použít pro organizování sportovní akce s cizími lidmi, tak se mu to nepovede, protože pokud nastaví událost jako soukromou, tak musí všem účastníkům odeslat virtuální pozvánku, aby se o události dozvěděli. Na druhou stranu pokud organizátor nastaví událost jako veřejnou, tak se může přihlásit libovolný počet uživatelů, nezávisle na věku, pohlaví a sportovních dovednostech. Další problém je vyhledávání mezi existujícími událostmi, protože to lze jedině fulltextově. Není tedy možné vyhledat událost, která se koná v určitý den, v konkrétním městě a její název obsahuje například slovo „Tenis“.

Výše uvedený popis možností je z pohledu uživatele jakožto konkrétního jedince. Na Facebooku může mít ovšem svůj profil například také společnost, značka, produkt, umělec nebo komunita. Na takovém profilu je pak důležitá především *zeď*, na kterou její správce přidává příspěvky. Uživatelé se pak mohou přihlásit k odběru těchto příspěvků tím, že stránku označí jako *To se mi líbí*, čímž se stanou tzv. *fanoušky* dané stránky (obdoba přátelství mezi uživateli). Profil na Facebooku používají společnosti a veřejně známe osoby ke komunikaci s fanoušky a k marketingovým účelům. Mezi profily s nejvíce fanoušky patří například zpěvačka Shakira, která jich má již více než 100 miliónů.

Integrace Facebooku do vlastní aplikace

Vývojáři mohou integrovat do svých aplikací zásuvné moduly nabízené Facebookem. Tato možnost je dostupná pro webové aplikace a pro aplikace běžící na operačním systému Android nebo iOS. Jeden z velmi využívaných zásuvných modulů je tzv. *přihlašovací dialog*, který umožňuje přihlásit se do aplikace prostřednictvím Facebookového účtu. Během přihlašování si aplikace od Facebooku může vyžádat různé informace o uživateli, včetně těch osobních. Z toho důvodu je při prvním přihlášení vždy zobrazeno dialogové okno, které uživatele informuje jaké informace si aplikace od Facebooku žádá. Například při přihlašování do sociální sítě Pinterest přes Facebookový účet, musí uživatel odsouhlasit dialogové okno zachycené na obrázku 4.2. Výsledkem úspěšného přihlášení přes účet Facebooku jsou požadované informace o uživateli a jeho unikátní identifikátor, tzv. *access token*. Ten může aplikace použít pro následné posílání požadavků na Facebook API jménem přihlášeného uživatele. Dalším často využívaným modulem jsou tlačítka umožňující informaci sdílet na Facebooku anebo ji označit jako *To se mi líbí*. Uživatel těmito tlačítky snadno přenese

nějaký obsah aplikace na svoji zeď a informuje tak o něm své přátele. Další nabízené moduly dokáží například zprostředkovat platební bránu pro mezinárodní platby anebo přidat komentářové okno do aplikace, což umožní vkládat komentáře prostřednictvím Facebookových účtů. Kompletní přehled nabízených modulů je dostupný na [7]. Pro vložení zásuvného modulu do vlastní aplikace se vývojář musí přihlásit na *Facebook Developers* a vytvořit si tam účet pro svoji aplikaci. Tím získá unikátní identifikátor a heslo aplikace. Tyto údaje pak používá při volání API Facebooku, aby mohlo být rozpoznáno, ze které aplikace byl požadavek odeslán.

Obrázek 4.2: Dialogové okno, které se zobrazí při prvním pokusu o přihlášení do sociální sítě Pinterest prostřednictvím Facebookového účtu. Informuje uživatele o informacích, které budou z Facebooku předány do sítě Pinterest.

Autentizace přes Facebook na platformě Node.js

Přihlašování účtem třetí strany na platformě Node.js usnadňuje softwarová vrstva *Passport*[2]. Ta poskytuje uniformní přístup k autentizačním API více jak sta webových aplikací. Jedna z webových aplikací, jejíž API pro autentizaci je v Passport zahrnuto, je sociální síť Facebook. Přihlášení účtem Facebooku přes Passport probíhá ve stručnosti následovně. Nejprve je nutné zavolat funkci `passport.use()` a předat jí jako parametr objekt `FacebookStrategy` s identifikátorem aplikace, heslem aplikace a obslužnou funkcí (tzv. *callback*). Poté stačí zavolat funkci `passport.authenticate('facebook')`, která po úspěšné autentizaci automaticky zavolá funkci zaregistrovanou jako *callback* a předá jí informace o uživateli a jeho *access token*.

[2]Passport - `http://passportjs.org/`

4.2 Twitter

Twitter[3] vytvořil v roce 2006 Jack Dorsey, Evan Williams, Biz Stone a Noah Glass. Twitter se specializuje na sdílení krátkých textových zpráv. Součástí zpráv jsou velmi často tzv. *hashtagy*, což jsou klíčová slova (příp. fráze) uvozená znakem #, jež označují téma, kterého se daná zpráva týká. Na základě četnosti výskytu hashtagů pak Twitter sleduje popularitu jednotlivých témat - tzv. *trendy*. Přestože se Twitter zatím v České Republice příliš neprosadil, tak celosvětově je jednou z nejvíce používaných sociální sítí. Koncem roku 2014 ho alespoň jednou měsíčně používalo 284 miliónů uživatelů.

Obrázek 4.3: Logo Twitteru.

Po přihlášení do sítě Twitter může uživatel dělat následující činnosti:

- **Přidat příspěvek** - Příspěvek se nazývá *tweet* a délka jeho textové části může být maximálně 140 znaků. Součástí příspěvku mohou být rovněž fotografie.
- **Sledovat uživatele** - Má-li uživatel zájem o příspěvky nějakého jiného uživatele, například známe osobnosti, může ho tzv. *sledovat*. Sledování znamená, že se budou uživateli automaticky zobrazovat příspěvky, které sledovaný uživatel přidá.
- **Komunikace se sledujícími uživateli** - Uživatelé mohou neveřejně komunikovat prostřednictvím soukromých textových zpráv dlouhých maximálně 140 znaků. Komunikace musí být ovšem zahájena sledovaným uživatelem. Tedy sledovaný uživatel může kontaktovat své odběratele, ale nikoliv naopak.

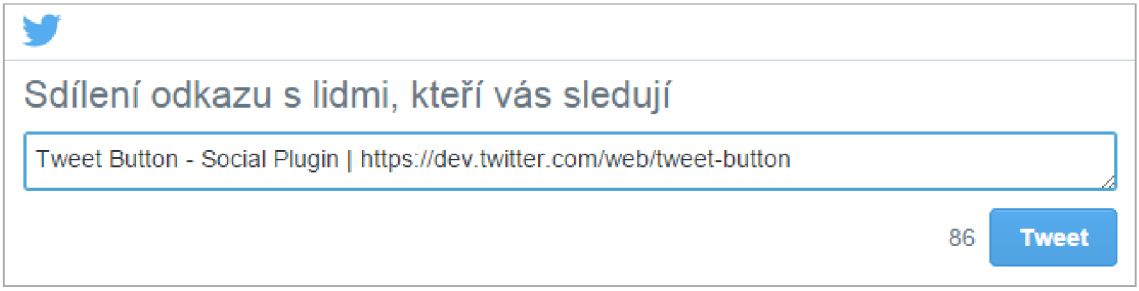
Integrace Twitteru do vlastní aplikace

Twitter nabízí vývojářům webových aplikací několik zásuvných modulů pro provázání aplikace a Twitteru. Základní moduly jsou přihlašovací dialog a tlačítka pro sdílení informace na Twitteru. Proces přihlašování probíhá stejně jako u Facebooku, tedy uživatel musí odsouhlasit předání informací z Twitteru do aplikace a výsledkem přihlášení jsou pak požadované údaje a *access token* uživatele. Tlačítko pro sdílení na Twitteru vyvolá dialogové okno, které je na obrázku 4.4. Okno obsahuje předpřipravený text, který může uživatel modifikovat a následně publikovat na svém Twitterovém účtu. Další zásuvný modul je například tzv. *vestavěný tweet*, který umožňuje vložit do aplikace okno s libovolným tweetem, včetně jeho fotografií. Přehled zásuvných modulů pro webové vývojáře je v [9]. Pro přístup k zásuvným modulům Twitteru si vývojář musí vytvořit účet na *Twitter Developers*, čímž získá identifikátor a heslo aplikace, které slouží pro stejné účely jako v případě Facebooku.

Autentizace přes Twitter na platformě Node.js

Na platformě Node.js lze pro přihlášení přes Twitter použít Passport. Princip použití je stejný jak v případě Facebooku, rozdíl je pouze v parametrech použitých funkcí. Funkci `passport.use()` je nutné předat místo objektu `FacebookStrategy` objekt `TwitterStrategy` a funkci `passport.authenticate` volat s parametrem `'twitter'`.

[3]Twitter - `https://twitter.com/`



Obrázek 4.4: Okno, které se zobrazí při požadavku na sdílení informace na Twitteru.

4.3 Google+

Mezi sociálními sítěmi se dlouhou dobu snaží prorazit i internetový gigant Google. Ten nejprve v roce 2004 spustil sociální síť *Orkut*, která se ovšem prosadila pouze v Brazílii a kvůli nedostatečnému zájmu byla v roce 2014 zrušena. Mezitím, v roce 2010, se Google zkoušel prosadit se sociální sítí *Google Buzz*, která ovšem vydržela pouze necelé dva roky a v prosinci 2011 byla rovněž zrušena. Důvodem zrušení bylo údajně soustředění veškeré pozornosti na novou sociální síť představenou v červnu 2011, nazvanou *Google+*[4]. Pro rozšíření této sociální sítě mezi uživatele zvolila společnost Google vcelku agresivní politiku, kdy účty Google+ propojila s ostatními službami jež nabízí, například s e-mailovou službou *Gmail*, nebo s *YouTube* - největším webovým serverem pro sdílení video souborů. Díky tomu je v sociální síti Google+ registrováno přes 1 miliardu uživatelů, ale z těch je alespoň jednou měsíčně aktivních „pouze“ 343 miliónů.

Obrázek 4.5: Logo Google+.

Z hlediska uživatelských možností by se dalo říci, že Google+ je velmi podobný Facebooku. Obsahuje prakticky stejnou funkcionalitu, lišící se pouze v detailech a jiných názvech. Například místo označení *To se mi líbí* používá označení *+1*, ale význam zůstává stejný. Místo *přátel* pak používá *kruhy*, princip zůstává, ale zatímco na Facebooku jsou přátelé všichni pohromadě, Google+ umožňuje vytvořit více kruhů a kontakty do nich roztřídit. Z dalších možností jako je například sdílení příspěvků, správu profilu, organizování událostí atd. nabízí Google+ přesně to stejné, co Facebook.

Integrace Google+ do vlastní aplikace

Možnosti integrace jsou rovněž velmi podobné tomu, co nabízí Facebook, respektive Twitter. Google+ umožňuje přihlašování účtem Google, tlačítka pro označování *+1* a sdílení na Google+ nebo např. *vestavěný příspěvek*, což je obdoba vestavěného tweetu. Přehled všech zásuvných modulů pro web nabízených Google+ je dostupný v [8].

Autentizace přes Google+ na platformě Node.js

Pro přihlášení přes účet Google+ je možné, stejně jako v případě Facebooku a Twitteru, použít softwarovou vrstvu Passport. Princip je opět stejný, v případě Google+ je nutné funkci `passport.use()` předat objekt `GoogleStrategy` a funkci `passport.authenticate` volat s parametrem `'google'`.

Kapitola 5

Návrh sociální sítě pro kolektivní sporty

Návrh sociální sítě vychází z principů a technologií, jež byly popsány v kapitolách 2, 3 a 4. V první části této kapitoly jsou specifikovány požadavky na navrhovanou sociální síť a její případy použití. Ve druhé části kapitoly je architektura aplikace. Třetí část se zabývá uživateli, registrací a procesem přihlašování do aplikace. Ve čtvrté části je uveden návrh sportovních událostí a v páté části návrh upozornění a soukromých zpráv.

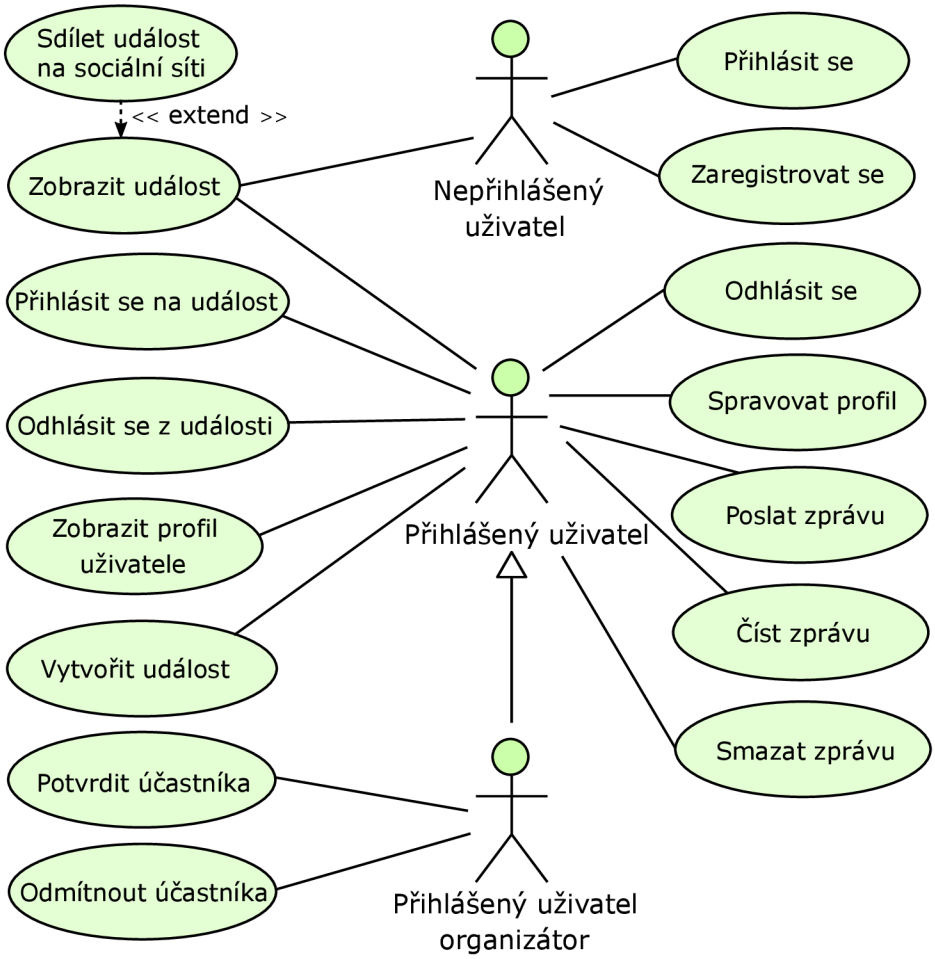
5.1 Specifikace požadavků a případů použití

Sociální síť pro organizování sportovních utkání musí splňovat tyto základní požadavky:

- **Vytvoření účtu (registrace)** - Uživatelský účet je možné vytvořit dvěma způsoby. První z nich je registrace přímo v aplikaci, kdy je požadováno zadat celé jméno, heslo, datum narození, e-mail a pohlaví. Druhou možností je přihlášení účtem Facebooku nebo Google+, od kterých budou požadovány stejné informace jako by uživatel zadal při registraci.
- **Správa účtu** - Uživatel může spravovat svůj účet. Lze do něj doplnit dovednostní úroveň v jednotlivých sportech a profilovou fotografii. Dovednostní úroveň bude možné zvolit na stupnici 1 až 10, kde 1 znamená úplný začátečník a 10 profesionální sportovec.
- **Zobrazení profilu uživatele** - Každý uživatelský účet je reprezentován veřejným profilem, který může zobrazit libovolný uživatel sítě. Na profilu bude zobrazeno jméno, věk, pohlaví, dovednostní úroveň v jednotlivých sportech a profilová fotografie.
- **Soukromé zprávy** - Libovolní dva uživatelé si mohou zasílat soukromé textové zprávy.
- **Vytvoření sportovní události** - Libovolný uživatel může vytvořit sportovní událost. Při vytváření události je povinné zadat její název, vybrat konkrétní sport, zvolit maximální počet účastníku, určit doporučenou dovednostní úroveň a nastavit datum, čas a místo konání. Uživatel, který událost vytvoří, se stává jejím *organizátorem* a jako jediný může na událost přijímat, nebo odmítat ostatní uživatele.

- **Vyhledání sportovní události** - Mezi vytvořenými událostmi musí být možné vyhledávat podle vhodných kritérií, jako je volba konkrétního sportu, místo konání, datum a čas konání a doporučená dovednostní úroveň.
- **Přidání uživatele na sportovní událost** - Uživatel se může přihlásit na libovolnou existující sportovní událost. Ovšem aby se stal platným účastníkem, musí jeho přihlášení odsouhlasit *organizátor* akce. Ten obvykle zkontroluje jeho dovednostní úroveň a věk a rozhodne se, zda-li je vhodným účastníkem či nikoliv.
- **Komentáře u události** - Pro snadnou domluvu všech účastníků události musí být možné vkládat k události komentáře. Přístupné by měli být pouze pro účastníky události.

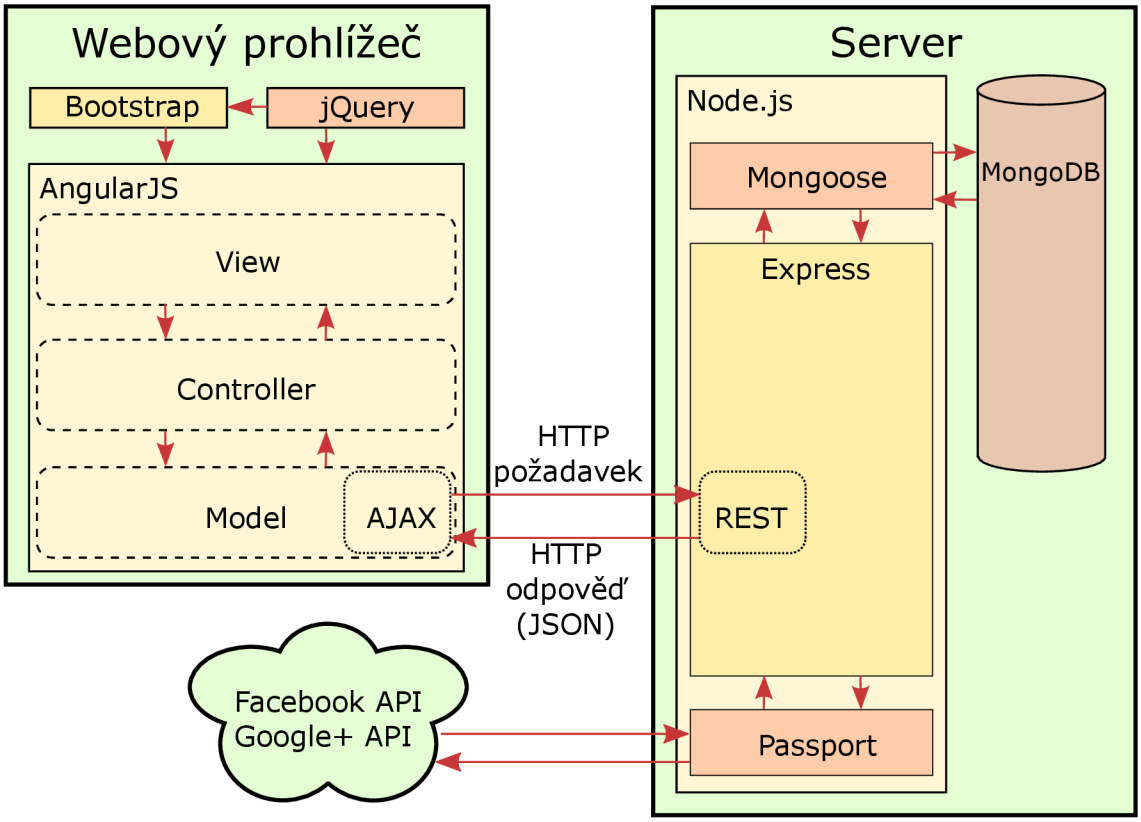
Jak bývá zvykem, je téměř všechna funkčnost aplikace dostupná teprve po přihlášení. Bez autentizace je možné zobrazovat pouze omezené informace o existujících událostech, což ovšem stačí ke sdílení událostí na ostatních sociálních sítích. Možnosti jednotlivých aktérů ilustruje diagram případů použití na obrázku 5.1.



Obrázek 5.1: Diagram případů použití.

5.2 Architektura

Sociální síť je postavena na JavaScriptových technologiích souhrnně označovaných jako MEAN (**M**ongoDB, **E**xpress, **A**ngularJS, **N**ode.js), které byly popsány v kapitole 3. Je navržena jako *Single Page Applicaition*, kde klientská část implementuje architektonický vzor MVC a se serverovou částí komunikuje prostřednictvím REST API využitím technologie AJAX. Blokové schéma architektury, s naznačenou komunikací mezi jednotlivými logickými celky, je na obrázku 5.2.



Obrázek 5.2: Blokové schéma architektury sociální sítě včetně použitých technologií v jednotlivých částech aplikace.

5.2.1 Serverová část

Serverová část aplikace je postavena na aplikačním rámci Express 4, který sám o sobě poskytuje pouze základní obsluhu HTTP požadavků [10]. Proto je potřeba jej pro účely této aplikace doplnit těmito moduly:

cookie-parser - Umožňuje vytvářet a spravovat soubory cookies.

express-session - Přidává podporu pro sezení. Využívá k tomu cookies, do kterých ukládá identifikátor sezení (tzv. *session ID*). Veškerá data sezení zůstávají na serveru, klientovi se posílá pouze onen identifikátor. Pro cookie uchovávající identifikátor sezení je

možné nastavit parametry *HttpOnly* a *Secure* pro minimalizování rizika zneužití. Výchozí úložiště pro data sezení je operační paměť serveru a není doporučené ji používat v produkčním nasazení, protože hrozí únik paměti (tzv. *memory leak*).

connect-mongo - Rozšiřuje možnosti express-session. Přidává možnost ukládání dat sezení do databáze MongoDB, čímž eliminuje problém s únikem paměti popsaný výše.

body-parser - Používá se pro parsování dat z HTTP požadavků, předaných například metodou POST. Dokáže parsovat data například ve formátu JSON (*application/json*) nebo předaná ve formě řetězce (*application/x-www-form-urlencoded*). Neumí ale zpracovat tzv. *multipart* data (*multipart/form-data*), která jsou na server posílána při nahrávání souborů, například fotografií.

multer - Používá se pro parsování *multipart* dat (*multipart/form-data*), tedy souborů předaných protokolem HTTP. Dokáže je přijmout, uložit na server a poskytnout o nich základní informace jako je jejich jméno, přípona nebo velikost.

method-override - Umožňuje používat HTTP metody PUT nebo DELETE i když je klient nepodporuje. Přidává do hlavičky HTTP komunikace vhodně nastavené pole *X-HTTP-Method-Override*, což umožní, aby se například POST požadavek interpretoval na serveru jako PUT.

passport - Slouží k autentizaci uživatelů, respektive HTTP požadavků. Autentizační mechanizmy, tzv. *strategie*, jsou dostupné zvlášť, každá ve speciálním modulu. Podpora pro přihlášení lokálním účtem je obsažena v modulu **passport-local**. Pro přihlášení účtem Facebooku přes protokol OAuth[1] je potřeba modul **passport-facebook** a pro přihlášení přes Google+ **passport-google-oauth**.

mongoose - Vytváří vrstvu nad databází MongoDB. Umožňuje napevno definovat schéma databáze, provádět automatické konverze mezi datovými typy, automaticky validovat data a celkově usnadňuje práci s databází.

serve-favicon - Přidává podporu pro zobrazení tzv. *favicon* - ikony stránky, která se zobrazuje například v záložce webového prohlížeče.

Serverové API

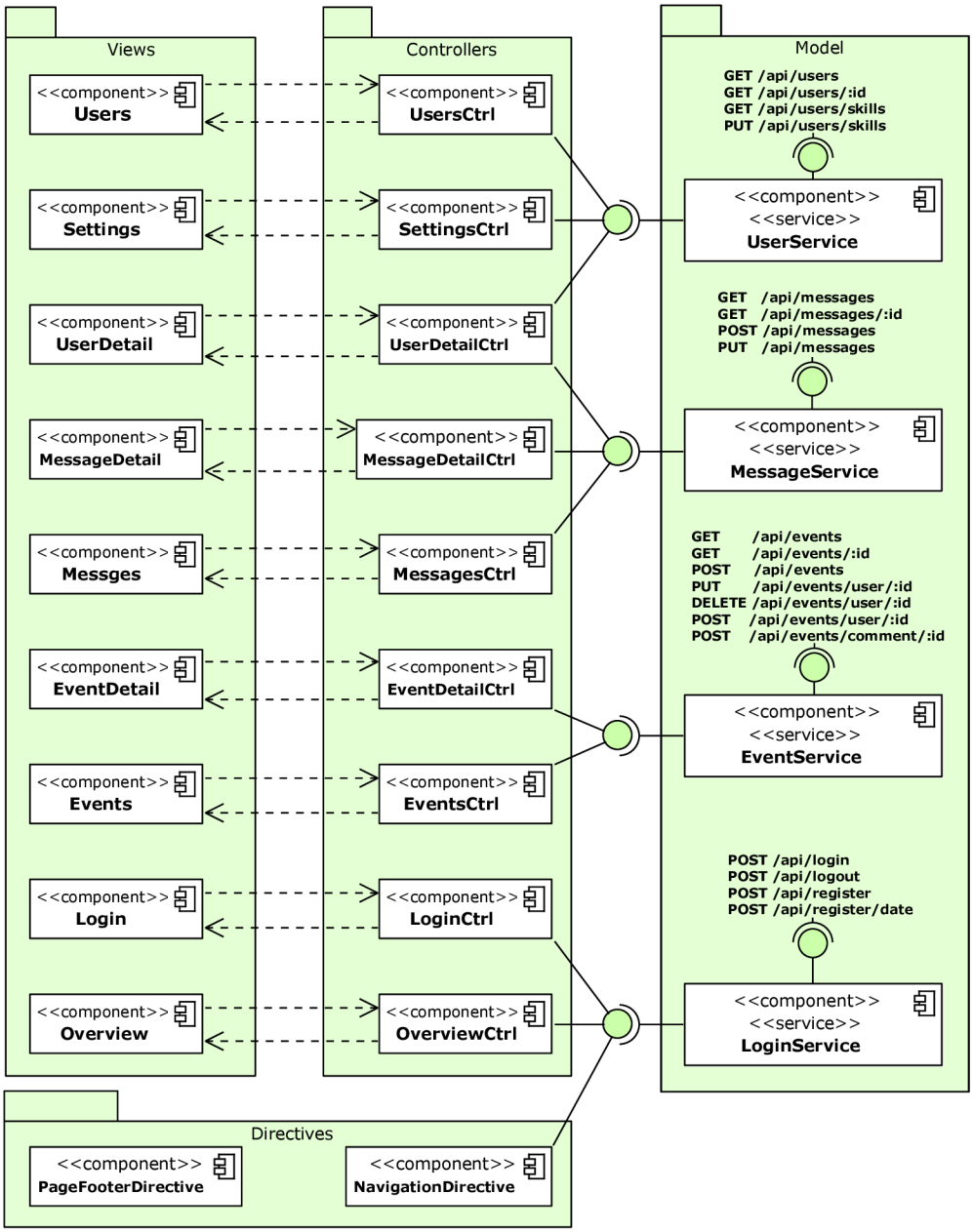
Pro komunikaci se serverem se používá rozhraní REST, každý zdroj je tedy identifikovatelný pomocí adresy URL. Většina zdrojů je zabezpečená, lze k nim přistupovat pouze po přihlášení do systému. Pokusí-li se klient přistoupit k zabezpečenému zdroji nepřihlášený, odpoví server stavovým kódem 401 (*Unauthorized*) a neposkytne žádná data. Pro některé zdroje dostupné přes metody POST a PUT jsou očekávány určité parametry v těle HTTP požadavku. Nejsou-li předány, odpoví server stavovým kódem 400 (*Bad Request*) a rovněž neposkytne data.

Jednotlivé koncové body rozhraní a stručný popis jejich obslužné funkce je uveden v dodatku A.

[1] OAuth - komunikační protokol pro bezpečnou autentizaci a autorizaci oproti aplikačnímu rozhraní různých služeb.

5.2.2 Klientská část

Klientská část sociální sítě je postavena na aplikačním rámci AngularJS [11] a sestává především z řady *controllerů* a k nim příslušných *view*. *Controllery* komunikují s API serveru přes tzv. *services*, které zprostředkovávají AJAX volání a je možné je sdílet napříč *controllery* pomocí *Dependency Injection*. Kromě uvedeného jsou v aplikaci použity i dvě *direktivy* pro sdílení stejného obsahu všech stránek sítě. Jedna zapouzdřuje hlavní navigaci a vkládá se do každého *view* pomocí nově vytvořené HTML značky <navigation></navigation>. Druhá zapouzdřuje patičku stránky a vkládá se značkou <pageFooter></pageFooter>. Celkovou architekturu klientské části sociální sítě zobrazuje diagram komponent na obrázku 5.3.



Obrázek 5.3: Diagram komponent klientské části sociální sítě, včetně naznačené komunikace se serverovým API.

Komponenty Users a UsersCtrl

Komponenty *Users* a *UsersCtrl* jsou zodpovědné za stránku zobrazující všechny uživatele v systému. Při vytvoření instance *UsersCtrl* si tento stáhne ze serveru přes *UserService* informace o všech uživatelích, které následně abecedně seřadí podle jejich jména a předá je do *view Users* k vykreslení. Uživatele lze následně filtrovat podle jména, věku nebo pohlaví.

Komponenty UserDetail a UserDetailCtrl

Komponenty *UserDetail* a *UserDetailCtrl* tvoří stránku s veřejným profilem uživatele. Informace o uživateli jsou získány ze serveru pomocí *UserService* při vytvoření instance *UserDetailCtrl*, která je následně předá do *view UserDetail* k vykreslení. Uživateli lze z jeho profilu odeslat zprávu, k tomu slouží tlačítko „Poslat zprávu“, po jehož stisknutí se zobrazí dialogové okno s formulářem pro odeslání zprávy. Událost `click` odesílacího tlačítka tohoto formuláře je napojená na funkci *controlleru* `sendMsg()` jenž přes *MessageService* zprávu odešle.

Komponenty Settings a SettingsCtrl

Komponenty *Settings* a *SettingsCtrl* umožňují zobrazit a následně upravit dovednostní úrovně uživatele v jednotlivých sportech. Instance *SettingsCtrl* si přes *UsersService* stáhne aktuální dovednostní úrovně uživatele a předá je do *view Settings* k vykreslení. *View* obsahuje posuvníky, kterými je možné dovednostní úrovně upravit a poté je tlačítkem napojeným na funkci *controlleru* `updateSkills()` odeslat přes *UserService* na server.

Komponenty Messages a MessagesCtrl

Komponenty *Messages* a *MessagesCtrl* odpovídají za stránku zobrazující všechny zprávy uživatele. Ty načte ze serveru *MessagesCtrl* přes *MessageService* při jeho instanciaci a poté je předá do *view* k vykreslení. Ve *view* jsou k dispozici 2 tlačítka pro manipulaci se zprávami, jedno k přesunutí vybraných zpráv do koše a druhé k trvalému odstranění zpráv. Událost `click` těchto tlačítek obsluhují funkce *controlleru* `moveToTrash()` a `deleteForever()`, které přes *MessageService* provedou požadovanou operaci.

Komponenty MessageDetail a MessageDetailCtrl

Komponenty *MessageDetail* a *MessageDetailCtrl* umožňují otevřít konkrétní zprávu. Data zprávy jsou načtena ze serveru přes *MessageService* do *controlleru* a předána k vykreslení do *view*. V případě, že otevřená zpráva je doručená, je možné stisknout tlačítko pro zaslání odpovědi. Událost `click` tohoto tlačítka zpracuje v *controlleru* funkce `sendMsg()`, která zprávu s odpovědí odešle přes *MessageService*.

Komponenty Events a EventsCtrl

Komponenty *Events* a *EventsCtrl* zajišťují stránku zobrazující všechny události a jejich filtrování. Data o událostech jsou stažena do *controlleru* voláním jedné z funkcí *EventService*. V *controlleru* jsou seřazena dle data a času a poté předána k vykreslení do *view*. Na stránce se všemi událostmi lze rovněž vytvořit novou událost a to pomocí tlačítka k tomu určeného. Po jeho stisknutí je zobrazen formulář pro vytvoření nové události, jehož odesílací tlačítko je napojené na funkci `create()`, jež zadaná data odešle přes *EventService* na server.

Komponenty EventDetail a EventDetailCtrl

Komponenty *EventDetail* a *EventDetailCtrl* obsluhují stránku s detailem události. Informace o dané události jsou načteny ze serveru přes *EventService* do *controlleru* a předány k vykreslení do *view*. Uživatelské rozhraní detailu události se přizpůsobí stavu uživatele. Například není-li uživatel účastníkem události, zobrazí se mu tlačítko „Chci se zúčastnit“. Jestliže je účastníkem, má naopak možnost svou účast zrušit. Celkem *controller* obsahuje čtyři funkce pro změnu vztahu uživatele a události - `signIn()` (chci se zúčastnit), `signOut()` (nechci se zúčastnit), `acceptUser()` (potvrdit účast uživatele) a `rejectUser()` (odmítnout uživatele). Celý proces přihlašování se na událost je popsán podrobněji v kapitole 5.4.

Komponenty Login a LoginCtrl

Komponenty *Login* a *LoginCtrl* jsou odpovědné za stránku umožňující přihlášení do aplikace. *View* obsahuje formulář pro přihlášení lokálním účtem, formulář pro vytvoření nového lokálního účtu a tlačítka pro přihlášení přes Facebook a Google+. Formulář pro přihlášení lokálním účtem je odesílán přímo na server metodou POST. Tlačítka pro přihlášení přes sociální sítě přímo odkazují na příslušná volání API serveru. *Controller* zpracovává pouze formulář pro vytvoření nového lokálního účtu, jehož odesílací tlačítko je navázáno na funkci `register()`. Ta vezme zadaná data z formuláře a odešle je pomocí *LoginService* na server.

Komponenty Overview a OverviewCtrl

Komponenty *Overview* a *OverviewCtrl* tvoří stránku s přehledem sportovních událostí, s nimiž má přihlášený uživatel nějaký vztah. Jedná se o úvodní stránku, na kterou je uživatel přesměrován po úspěšném přihlášení do systému. Kromě získání kolekce událostí vztahujících se k uživateli tato stránka rovněž zajišťuje dialogové okno pro zadání data narození uživatele, pokud se přihlásil přes účet sociální sítě a datum narození z ní nebylo předáno.

5.3 Uživatelé, registrace a autentizace

Po lokální registraci, nebo při prvním přihlášení přes třetí stranu, je v databázi vytvořen dokument s daty uživatele. Schéma dokumentu je na obrázku 5.4. Každý dokument v databázi má vždy unikátní identifikátor, tzv. *ObjectID*, které je vygenerováno databází při vytvoření dokumentu. Do schémat dokumentů na obrázcích níže není *ObjectID* pro jednoduchost zahrnuto.

Popis jednotlivých položek dokumentu je následující. E-mail se používá k identifikaci uživatele, proto musí být unikátní a je povinné jej zadat již při vytvoření záznamu. Heslo se ukládá pouze v případě vytvoření lokálního účtu, při autentizaci třetí stranou není potřeba. Datum narození je pro aplikaci klíčové, protože jej používá na mnoha místech k výpočtu věku uživatele. Nabízí se označit jej jako povinné, ale není to možné z důvodu, že Facebook, respektive Google+ nerad poskytuje datum narození uživatele, je nutné si ho explicitně vyžádat a i přes to není jisté, že ho do aplikace předá (například si to uživatel nepřeje). Z toho důvodu se při přihlášení přes třetí stranu datum narození neukládá, ale je po uživateli vyžadováno jej zadat ihned po úspěšném přihlášení. Při lokální registraci je povinné zadat datum narození do registračního formuláře a uloží se tak již při vytváření záznamu v databázi. Pro reprezentaci pohlaví stačí dvoustavové úložiště, proto je datový

```
{
  email: String,  (unikátní, povinný)
  heslo: String,
  jméno: String,  (povinný)
  narozen: Date,
  pohlaví: Boolean,  (povinný)
  foto: String,
  upozornění: [ Upozornění ],
  zprávy: [ Zpráva ],
  dovednosti:
    {
      fotbal: Number,
      hokej: Number,
      tenis: Number,
      basketbal: Number,
      volejbal: Number
    }
}
```

Obrázek 5.4: Reprezentace uživatele v databázi.

typ zvolen jako *Boolean*, kde `true` reprezentuje muže a `false` ženu. Další položkou dokumentu o uživateli je profilová fotografie, ta se neukládá přímo do databáze, do té se pouze uloží její URL adresa. Zprávy a upozornění uživatele jsou ukládány jako kolekce vložených dokumentů, podrobněji se jimi zabývá kapitola 5.5. Poslední prvek záznamu je objekt uchovávající dovednostní úrovně, ty jsou reprezentovány číslem a mohou nabývat hodnot od 0 do 10. Hodnota 0 znamená, že dovednostní úroveň není pro daný sport nastavena. Hodnota 1 až 10 pak značí samotný stupeň úrovně.

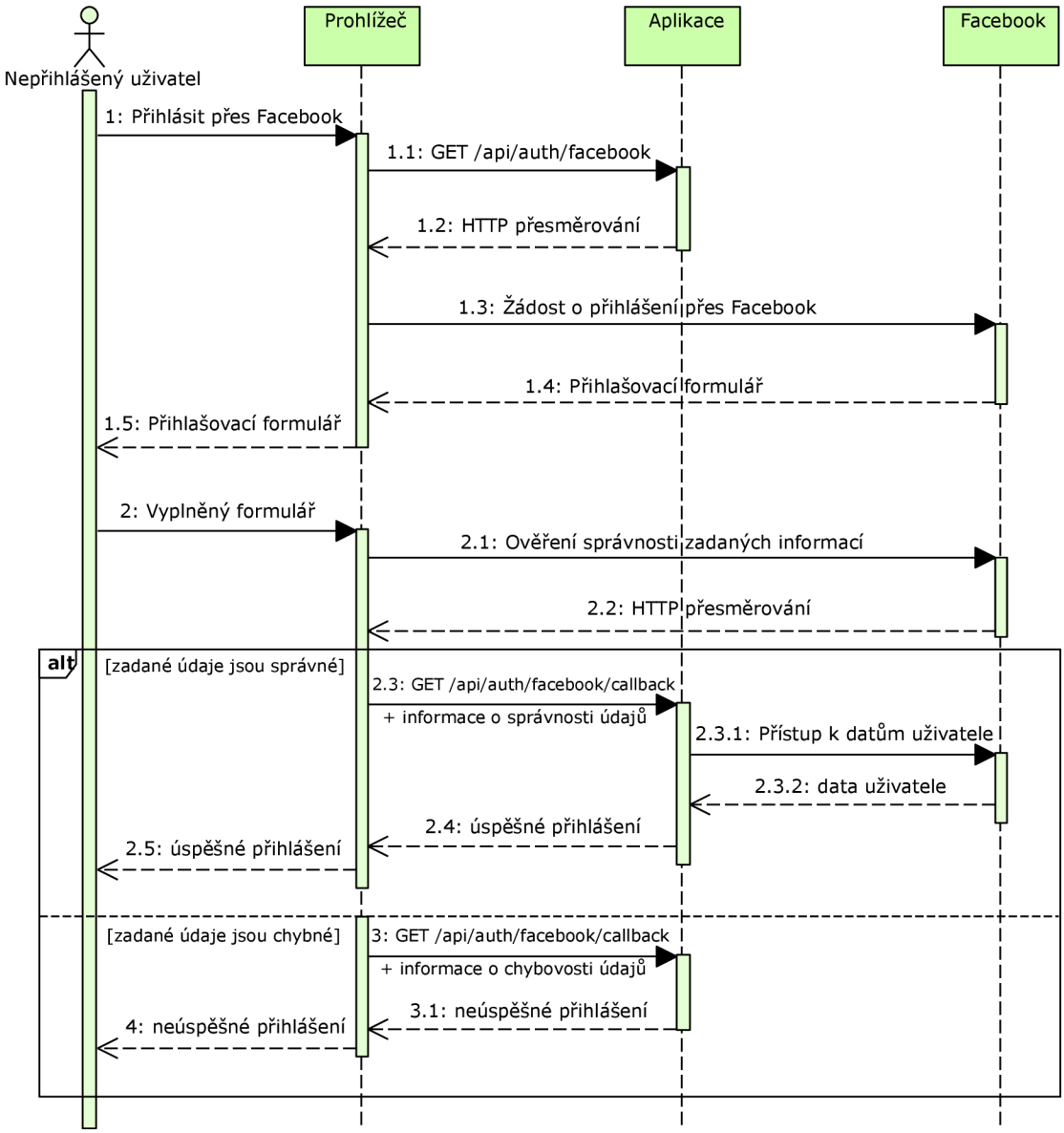
Registrace a přihlášení lokálním účtem

Při registraci lokálního účtu je potřeba řešit bezpečné uložení uživatelova hesla. Před uložením hesla do databáze je nutné jej zahashovat, aby například nebylo vyzrazeno v případě odcizení databáze. Dříve hojně používané hashovací funkce *MD5* nebo *SHA1* nelze dnes považovat za bezpečné, a to ze dvou důvodů. Prvním je existence tzv. *duhových tabulek* s předpočítanými hodnotami hashovací funkce a druhým je jejich vysoká rychlost výpočtu, která umožňuje tzv. *útok hrubou silou*. V aplikaci je jako hashovací funkce použit *bcrypt*, který vychází ze šifry *Blowfish*. Výhodou *bcryptu* je možnost zvolit počet iterací a tím určit výpočetní náročnost funkce. Při zvolení vhodné náročnosti je zcela eliminována možnost útoku hrubou silou, protože by doba jeho trvání mohla být například i v řádu tisíců let.

Přihlášení účtem třetí strany

Do aplikace je možné se přihlásit přes účet Facebooku nebo Google+. Původní záměr byl umožnit přihlášení i přes účet Twitteru, ale není možné od něj žádným způsobem získat e-mail uživatele. Z toho důvodu nebylo nakonec přihlášení přes Twitter do aplikace zahrnuto. Přihlásí-li se uživatel do aplikace různými způsoby, tak se jeho účty synchronizují, používá-li v těchto službách stejnou e-mailovou adresu. Například vytvoří-li uživatel lokální účet s e-mailovou adresou *jan.novak@email.cz* a poté se pokusí přihlásit přes účet Facebooku, který

je zaregistrován na stejný e-mail (tj. *jan.novak@email.cz*), tak se dostane do aplikace na ten stejný účet. Opačně to ovšem neplatí, pokud se nejprve přihlásí například účtem Google+ a poté se pokusí vytvořit lokální účet se stejným e-mailem, který aplikaci předal Google, tak se mu to nepodaří. Je to z důvodu bezpečnosti, aby někdo nemohl získat přístup k cizímu účtu tím, že ho „přeregistruje“ lokálním účtem. Samotný proces přihlašování třetí stranou probíhá přes protokol OAuth. Komunikace jednotlivých aktérů při přihlašování přes účet Facebooku je zobrazena v diagramu sekvence na obrázku 5.5.



Obrázek 5.5: Diagram sekvence znázorňující komunikaci při přihlašování přes sociální síť Facebook.

Detekce odpovědi se stavovým kódem 401 (*Unauthorized*)

Na straně klienta je potřeba detekovat a příslušně zareagovat na situaci, když volání serverového API pomocí technologie AJAX skončí odpovědí se stavovým kódem 401 (*Unauthorized*). V aplikaci je to řešeno pomocí tzv. *HTTP interceptoru* [15], přes který prochází každá odpověď serveru. *Interceptor* vždy zkontroluje stavový kód odpovědi a je-li roven hodnotě 401, tak provede přesměrování na úvodní stránku s přihlašovacím formulářem.

5.4 Sportovní události

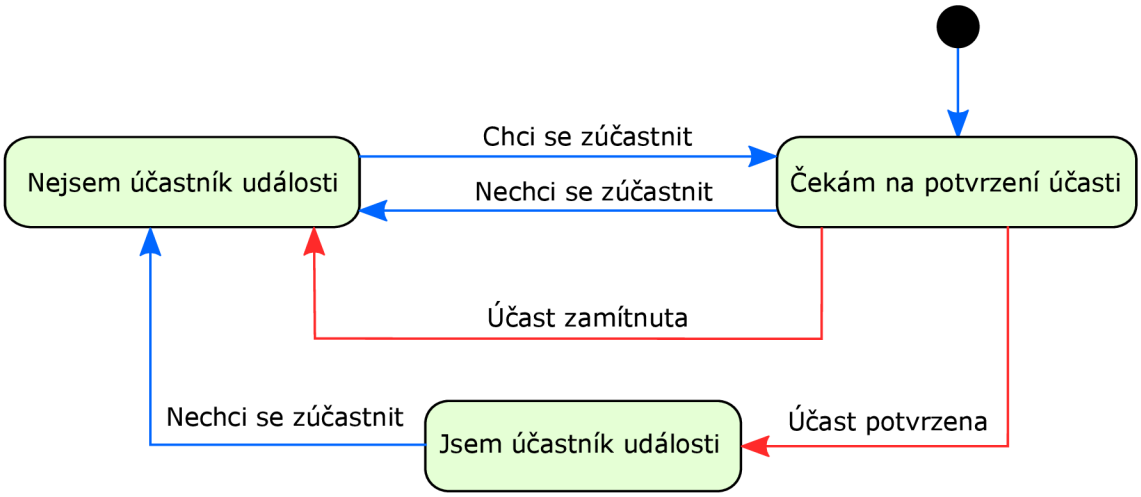
Sportovní událost je v databázi reprezentována dokumentem, jehož schéma je na na obrázku 5.6. Název události by měl vystihovat její charakter a může jim být libovolný neprázdný

```
{
  název: String,  (povinný)
  popis: String,
  sport: Number,  (povinný)
  místo: String,  (povinný)
  gps: String,
  maxÚčastníků: Number,  (povinný)
  doporučenáDovednost: Number,  (povinný)
  datum: Date,  (povinný)
  čas: String,  (povinný)
  cena: Number,  (povinný)
  uživatelé: [
    {
      id: ObjectID,
      stav: Number
    }
  ],
  komentáře: [
    {
      jméno: String,
      zpráva: String,
      datum: Date
    }
  ]
}
```

Obrázek 5.6: Reprezentace události v databázi.

řetězec. Popis události se zadává volitelně a slouží k doplnění libovolných textových informací, které by chtěl organizátor o události sdělit. Sport, který se na události bude hrát, je reprezentován číslem 0 až 4, kde 0 značí fotbal, 1 hokej, 2 tenis, 3 basketbal a 4 volejbal. GPS souřadnice místa konání slouží pro vložení interaktivní mapy a jejich zadání je nepovinné. Nejsou-li zadány, nebude se mapa u události zobrazovat. GPS souřadnice se ukládají ve formě řetězce jako dvě desetinná čísla oddělené čárkou a mezerou, s volitelným N a E označujícím *Northing* a *Easting*. Možný formát je tedy například „49.2265717N, 16.5966672E“ nebo „49.227, 16.597“. Maximální počet účastníků může nabýt hodnoty 2 až 30 a doporučená dovednostní úroveň 1 až 10. Datum, čas a místo konání události je samo

popisné. Cenou se rozumí poplatek za osobu za pronájem sportoviště, případně na pokrytí dalších nákladů s událostí souvisejících. Cena může být nastavena na libovolné přirozené číslo. Záznam o události v databázi dále obsahuje kolekci uživatelů a komentářů. Komentář sestává z objektu se jménem autora, zprávou (textem komentáře) a datem vytvoření. Kolekce uživatelů obsahuje unikátní identifikátor každého uživatele, který má s událostí něco společného a stav, který určuje jeho aktuální vztah k události. Do vztahu s událostí se dostane tím, že stiskne tlačítko „Chci se zúčastnit“ a od té chvíle může přecházet mezi 3 různými stavy způsobem, který znázorňuje stavový diagram na obrázku 5.7.



Obrázek 5.7: Stavový diagram znázorňující proces přihlašování na sportovní událost. Modré přechody provádí uživatel, který se chce zúčastnit a červené provádí organizátor události.

Sdílení na sociálních sítích

Stránku s detailem události je možné pomocí připravených tlačítek sdílet na Facebooku, Twitteru nebo Google+. Díky tomu je možné upozornit na událost i uživatele, kteří aplikaci neznají. Aby mělo sdílení praktický smysl, musí být stránka s informacemi o události dostupná i bez přihlášení. Z toho důvodu je volání `GET /api/events/:id` nezabezpečené. Ovšem nepřihlášený uživatel se nedostane ke všem informacím o události, ale pouze k těm nezbytně nutným. Neuvidí tak například seznam aktuálních účastníků či kdo je organizátorem události.

Komentáře

Komentáře u události slouží pro hromadnou domluvu zúčastněných. Z toho důvodu nejsou veřejně dostupné, ale přístup k nim mají pouze potvrzení účastníci. Serverové volání `GET /api/events/:id` tedy vrací rozdílné informace nejen pro nepřihlášeného/přihlášeného uživatele, ale rozlišuje i uživatele do aplikace přihlášené. Organizátor události získá od serveru jiná data než účastník události a ten získá jiná data než uživatel jež není účastníkem události. Na základě získaných dat se následně přizpůsobí uživatelské rozhraní.

5.5 Upozornění a soukromé zprávy

Upozornění se zobrazují ve veřejném profilu uživatele a slouží k informování o jeho nedávné aktivitě. Každé upozornění souvisí s nějakou sportovní událostí, například „Uživatel *X* organizuje událost *Y*.“, proto je součástí záznamu v databázi identifikátor události, jak je vidět na obrázku 5.8. Typ upozornění pak určuje, o kterou aktivitu se jedná, například jestli uživatel událost organizuje, nebo se jí účastní apod. Upozornění vytváří aplikace automaticky, na základě uživatelovy činnosti.

```
{
  typ: Number
  událost: ObjectId
}
```

Obrázek 5.8: Reprezentace upozornění v databázi.

Soukromé zprávy, které si mohou vyměňovat libovolní dva uživatelé, jsou v databázi reprezentovány dokumentem zachyceným na obrázku 5.9. Každý záznam obsahuje text samotné zprávy, její předmět a datum odeslání/přijetí. Kromě uvedeného se dále ukládá identifikátor uživatele, kterému je zpráva odeslána, respektive od kterého zpráva byla přijata a informace, zda-li ji již uživatel viděl či nikoli. Poslední část objektu zprávy je typ, který může nabývat hodnoty 0 až 3. Hodnota 0 znamená smazaná, 1 doručená, 2 odeslaná a 3 v koši.

```
{
  zpráva: String,
  uživatel: ObjectId,
  předmět: String,
  datum: Date,
  typ: Number
  viděna: Boolean
}
```

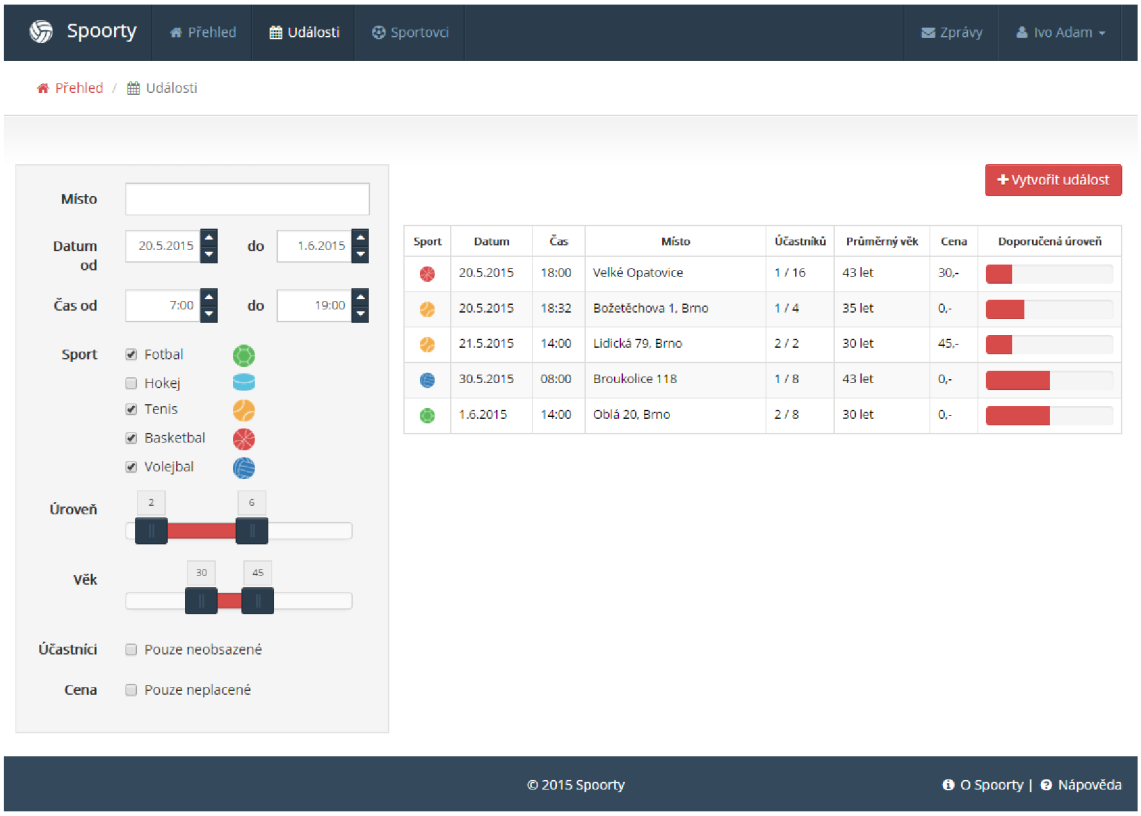
Obrázek 5.9: Reprezentace zprávy v databázi.

Odeslat zprávu je možné ze dvou míst. Jedním z nich je profil uživatele, kde je tlačítko „Poslat zprávu“, jež vyvolá dialogové okno s příslušným formulářem. Druhou možností je stisknout tlačítko „Odpovědět“ na stránce s detailem přijaté zprávy. Z uvedeného vyplývá, že chce-li uživatel kontaktovat jiného a doposud si nepsali, musí nejprve vyhledat jeho profil. Uvedený způsob byl zvolen z důvodu, že neexistuje žádný snadno zapamatovatelný a veřejně přístupný unikátní identifikátor uživatele, který by bylo možné vepsat do obecného formuláře pro odeslání zprávy pro identifikaci adresáta. Uživatel je interně identifikován 24 místným klíčem, který generuje databáze a není vhodný pro ruční zadávání.

Kapitola 6

Implementace a testování sociální sítě pro kolektivní sporty

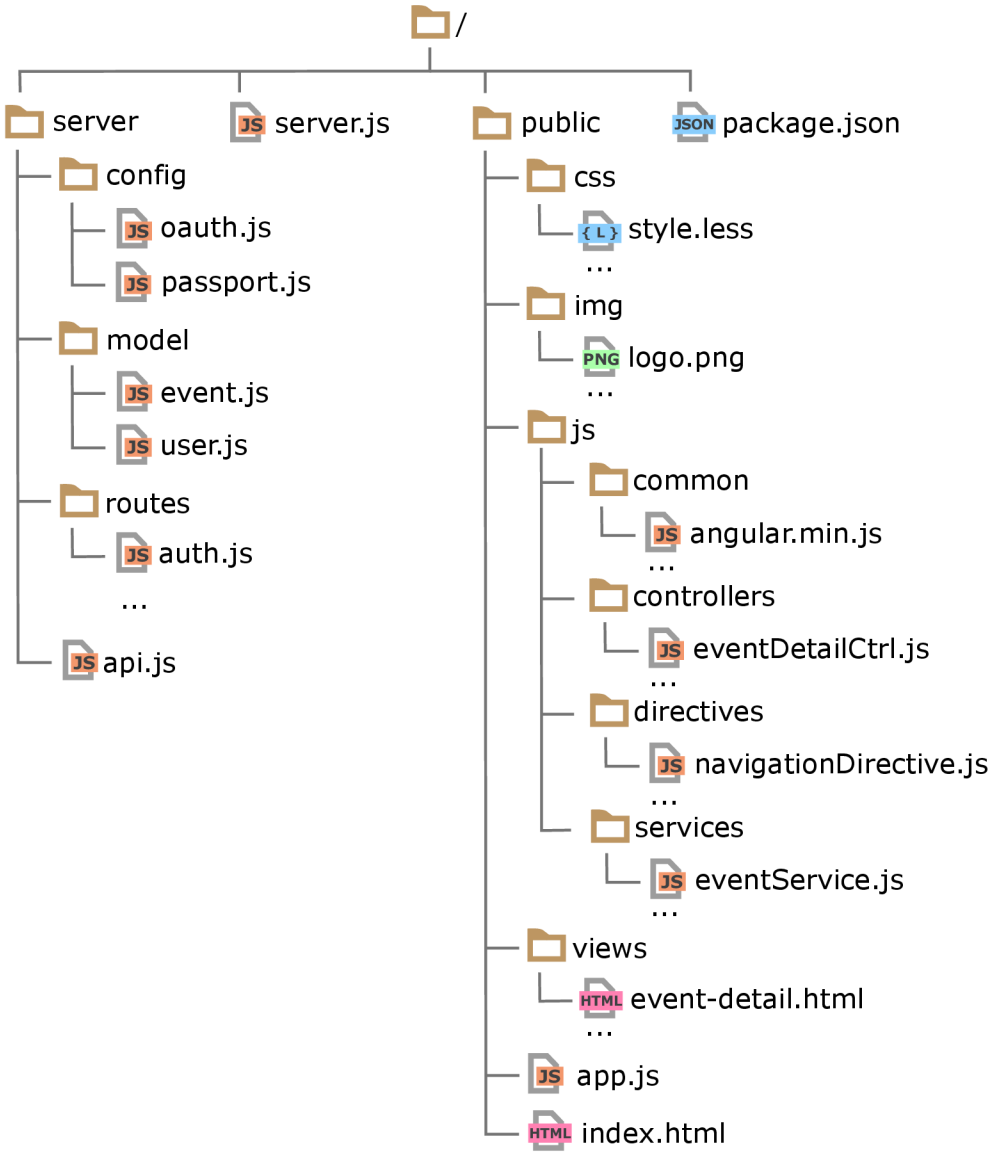
Podle návrhu uvedeném v kapitole 5 byla vytvořena sociální síť Spoorty. Je napsaná ve značkovacím jazyce HTML5, nastylovaná pomocí CSS3 a její logika je implementována v programovacím jazyce JavaScript s využitím aplikačních rámců AngularJS ve verzi 1.3.15, Express verze 4.12 a Node.js verze 0.10. Sociální síť by měla fungovat ve všech moderních webových prohlížečích. Byla experimentálně otestována v aktuálních verzích prohlížečů Google Chrome, Mozilla Firefox a Internet Explorer, ve kterých je její správná funkčnost zaručena. Snímek z aplikace je na obrázku 6.1.



Obrázek 6.1: Snímek stránky se sportovními událostmi ze sociální sítě Spoorty.

6.1 Organizace archivu se zdrojovými kódy

Zdrojové kódy jsou na nejvyšší úrovni rozděleny do dvou složek - *server* a *public*. Složka *server* obsahuje veškerý kód související s webovým serverem a její dokumenty nejsou dostupné přes web. V této složce je složka *config*, která obsahuje soubory související s autentizací uživatelů, složka *model*, která obsahuje dokumenty s databázovými schématy a složka *routes*, ve které je kód obsluhující jednotlivá volání API serveru. Ve složce *public* je celá klientská aplikace a její obsah je přes web přístupný. Vstupním bodem aplikace je soubor *index.html*, je to v podstatě jediný HTML soubor, který server vrací klientovi, neboť se jedná o *single page aplikaci*. Ostatní soubory jsou odkázány z *index.html* pomocí značek `script` a `link`, nebo staženy přes AJAX při běhu aplikace a dynamicky vloženy do stránky. Celá struktura archivu se zdrojovými kódy je vizualizována na obrázku 6.2.



Obrázek 6.2: Adresářová struktura archivu se zdrojovými kódy

6.2 Použité knihovny a externí komponenty

Kromě aplikačních rámců zmíněných na začátku této kapitoly jsou v aplikaci použity i tyto knihovny a externí komponenty:

Bootstrap - Možnosti využití Bootstrapu byly popsány v kapitole 3.1. V aplikaci je použit jako základ grafického rozhraní aplikace. Je pomocí něj vytvořeno responzivní rozložení stránky, hlavní navigační panel, tlačítka a formulářové prvky.

Font Awesome - Písmo Font Awesome umožňuje do webové stránky jednoduše vkládat předpřipravené ikony. Jedná se o font, který má ikony místo běžných znaků a díky vektorové reprezentaci je možné ikony libovolně zvětšovat bez ztráty kvality. V aplikaci se ikony používají například v navigačním panelu nebo v popiscích tlačítek.

noUiSlider - Komponenta sloužící pro vkládání posuvníku, který není běžným formulářovým prvkem jazyka HTML. V aplikaci se posuvníky používají například při nastavení dovednostní úrovně, nebo na omezení intervalu hodnot při filtrování dat.

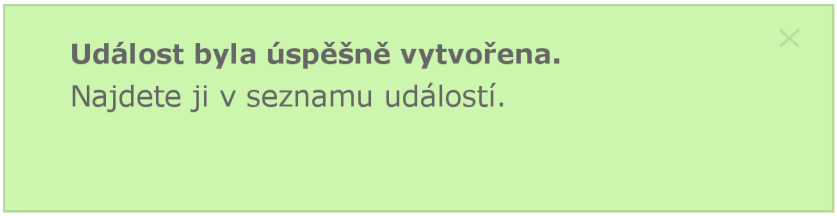
Obrázek 6.3: Posuvník vložený přes noUiSlider s nastavenou hodnotou 7 z 10.

jQuery - Schopnosti jQuery byly rovněž popsány v kapitole 3.1. V aplikaci je jQuery použito především proto, že jej vyžadují ostatní externí knihovny a komponenty, například Bootstrap či noUiSlider.

moment.js - Knihovna moment.js usnadňuje práci s datem a časem. Je schopna například rozpoznat neplatné datum, přičítat k datu dny nebo jej libovolně formátovat.

Angulike - Direktivy pro AngularJS, které zapouzdřují tlačítka pro sdílení obsahu na sociálních sítích Facebook, Twitter a Google+.

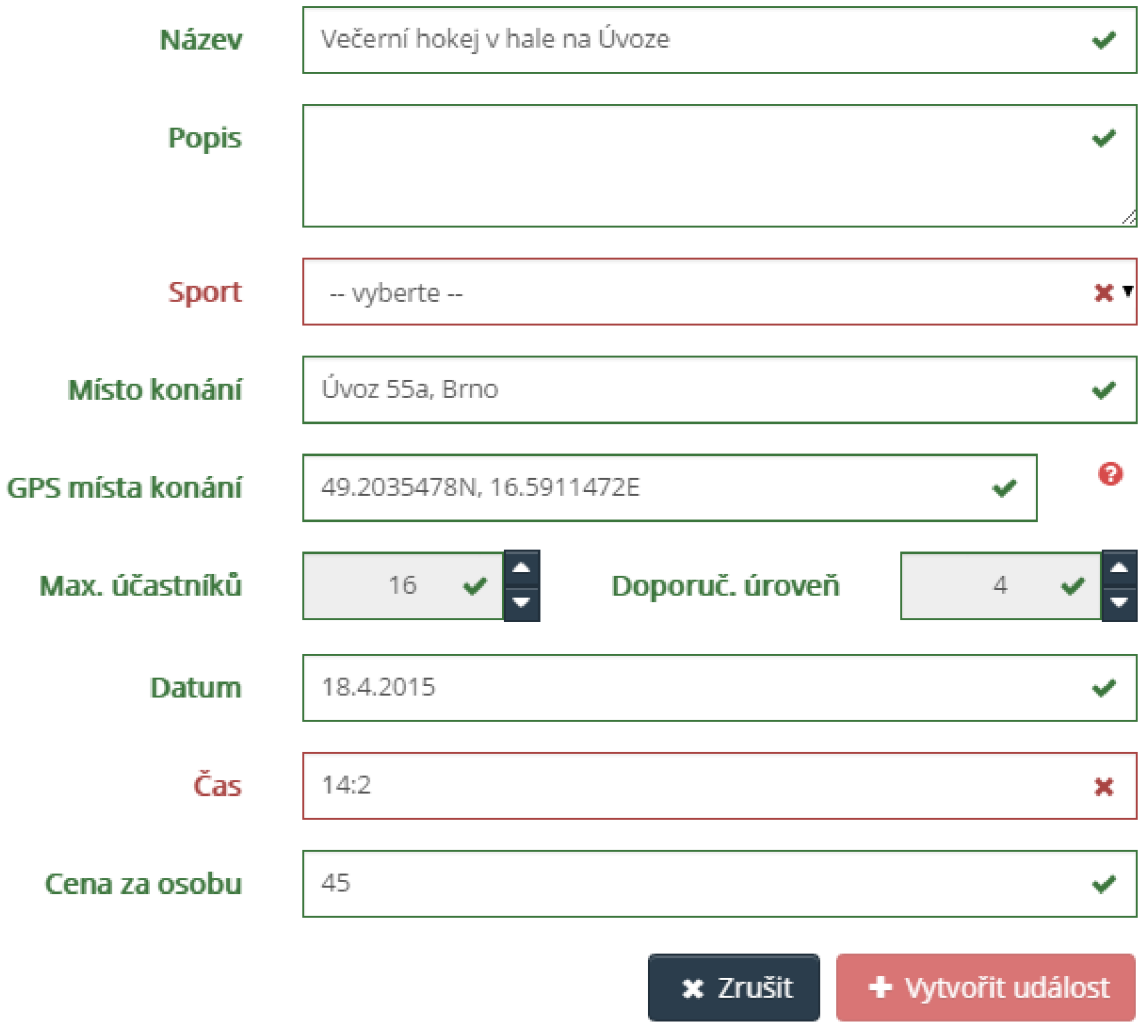
Angular-flash - Komponenta umožňující zobrazit tzv. *flash zprávy*. Jedná se o krátkou zprávu, která se zobrazí nad ostatními prvky webové stránky a po uplynutí určité doby zmizí. Používá se pro informování uživatele o výsledku volání aplikačního rozhraní serveru. Snímek flash zprávy je na obrázku 6.4.



Obrázek 6.4: Flash zpráva informující uživatele o výsledku odeslání vyplněného formuláře pro vytvoření sportovní události.

6.3 Validace formulářů

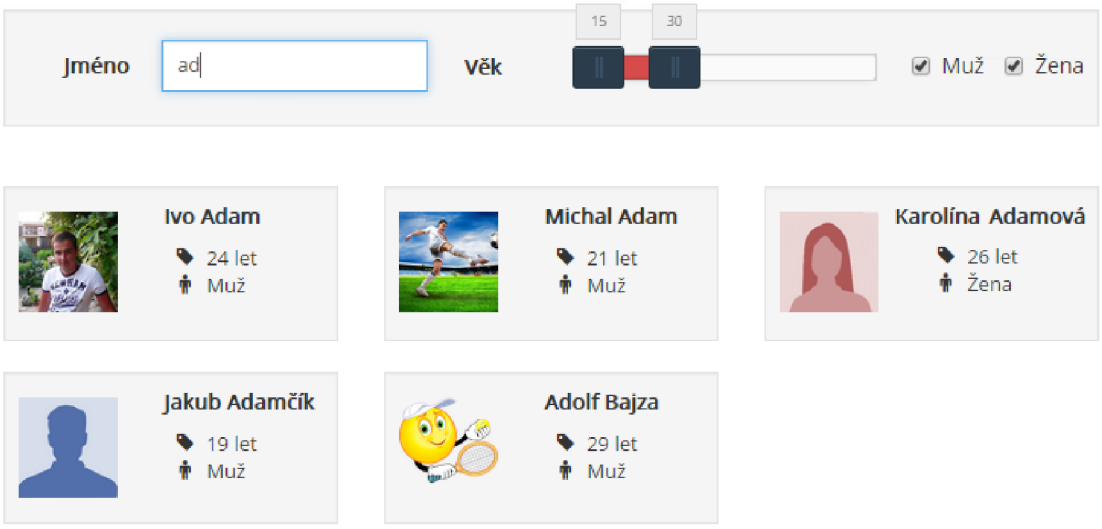
Důležitá část procesu získání informací od uživatele je ověření jejich správnosti. V aplikaci se uživatelský vstup validuje jak na serveru, tak na klientovi. Veškeré formuláře v aplikaci, které předávají svůj obsah na server, jsou zabezpečeny kódem v jazyce JavaScript. Ten porovná každý prvek formuláře s k němu určeným regulárním výrazem. Porovnání probíhá při události `oninput` vyvolané na formuláři, což v praxi znamená při každé změně libovolného prvku formuláře. Je-li formulářový prvek vyplněn správně, je v něm zobrazena ikona zatržítka a okolo něj zelený rámeček. V opačném případě je zobrazen křížek a rámeček je červený. Tlačítko pro odeslání formuláře na server je ve stavu „zakázáno“ (angl. *disabled*), dokud není celý formulář správně vyplněn. Nemělo by být tedy možné při používání aplikace odeslat na server neplatná data. Ovšem se serverem je možné komunikovat i mimo aplikaci, proto musí být data validována i na něm. Vyskytne-li se v nich neplatný údaj, server neprovede požadovanou operaci a odpoví stavovým kódem 400 (*Bad Request*).



Obrázek 6.5: Formulář pro vytvoření sportovní události, který obsahuje dvě neplatné hodnoty. Položka „Sport“ nebyla doposud vybrána z nabídky a do položky „Čas“ byl vepsán neplatný údaj.

6.4 Filtrování dat

V případě, že v aplikaci bude velké množství uložených dat (událostí, uživatelů), je pro snadné vyhledání požadovaných dat nutné umožnit je filtrovat dle vhodných kritérií. Aplikace provádí veškeré filtrování dat v klientské části pomocí aplikačního rámce Angular, který je na podobné operace připravený. Filtrování probíhá okamžitě, není nutné nastavené parametry odesílat. Ihned při změně formuláře tedy dojde k promítnutí nové hodnoty do výsledku filtrování. Uživatele aplikace lze filtrovat pomocí jména, věku a pohlaví, jak je vidět na obrázku 6.6. Ve jméně uživatele se vyhledává zadaný podřetězec, přičemž na velikosti písmen nezáleží. Věk lze omezit pomocí posuvníku na určitý interval o minimální velikosti 5 let. Jednotlivá kritéria lze samozřejmě libovolně kombinovat. Možnosti nastavení filtru u sportovních událostí bylo možné vidět v levé části obrázku 6.1.



Obrázek 6.6: Omezení zobrazených uživatelů pouze na ty co mají ve jméně podřetězec „ad“ a jsou ve věku od 15 do 30 let.

6.5 Experimentální ověření funkčnosti

Aplikační rozhraní serveru bylo testováno průběžně během celého vývoje za pomoci nástroje *Postman - REST Client*, který umožňuje odeslat HTTP dotaz na zadanou URL adresu a následně zobrazit HTTP odpověď. Klientská část aplikace byla testována nástroji *Chrome Developer Tools*, jež jsou dostupné ve webovém prohlížeči Google Chrome. Tyto nástroje byly použity pro inspekci vykresleného dokumentu, ověření výsledků AJAX volání, ladění JavaScriptového kódu a ověření správné aplikace kaskádových stylů.

Výsledná aplikace jako celek byla manuálně testována podle předem připraveného scénáře. Testování proběhlo ve webových prohlížečích Google Chrome 43, Mozilla Firefox 38 a Internet Explorer 11. Testovací scénář včetně výsledků je obsažen v dodatku B.

Kapitola 7

Závěr

Cílem této diplomové práce bylo vytvořit sociální síť pro organizování amatérských sportovních utkání v kolektivních sportech. Bylo potřeba zvolit vhodnou platformu pro vývoj, prostudovat aplikační rozhraní existujících významných sociálních sítí s ohledem na integraci informací do vlastní aplikací, navrhnout architekturu aplikace a poté ji implementovat.

Tento text postupně popsal celý vývojový proces sociální sítě. Začátek práce shrnul informace o *World Wide Webu* a definoval základní pojmy s ním spojené. V další části textu byl uveden přehled technologií pro tvorbu webových aplikací, se zaměřením na vývoj v jazyce JavaScript, ve kterém je celá aplikace implementována. Teoretickou část textu zakončila kapitola o existujících sociálních sítích a jejich typickém použití. Následující části textu se věnovaly nejprve návrhu sociální sítě pro kolektivní sporty a poté její implementaci a testování.

Výsledkem práce je funkční sociální síť, navržená jako *single page aplikace*. Klientská část je implementována v JavaScriptovém aplikačním rámci *AngularJS* a serverová v *Expressjs*, který jako běhové prostředí používá platformu *Node.js*. Business logika je implementována na straně klienta, server se používá především pro perzistentní uložení dat.

Vytvořená aplikace umožňuje uživateli organizovat sportovní události a účastnit se jich. Mezi vytvořenými událostmi lze vyhledávat na základě data, času a místa konání, doporučené dovednostní úrovně, průměrného věku účastníků a druhu sportu. Do implementované sítě je možné se přihlásit účtem sociální sítě Facebook nebo Google+. Jednotlivé sportovní události pak mohou být pomocí připravených tlačítek sdíleny na Facebooku, Google+ nebo Twitteru. Pro usnadnění domluvy uživatelů byly do aplikace implementovány soukromé zprávy, které si mohou vyměňovat libovolní dva uživatelé. Současná domluva více uživatelů přihlášených na stejnou sportovní událost je rovněž možná, a to pomocí neveřejných komentářů u události. K uživatelskému profilu, ve kterém je povinně zobrazeno jméno a věk uživatele, lze přidat profilová fotografie a dovednostní úroveň v jednotlivých sportech.

Během manuálního testování vytvořené sociální sítě nebyly objeveny žádné chyby. V případě budoucího vývoje by bylo vhodné doplnit do aplikace například možnost upravit vytvořenou sportovní událost a o provedených změnách automaticky informovat její účastníky.

Literatura

[1] Special Eurobarometer 412 „Sport and physical activity“. http://ec.europa.eu/public_opinion/archives/ebs/ebs_412_en.pdf, [cit. 2014-12-09].

[2] Hypertext. http://en.wikipedia.org/wiki/Hypertext, [cit. 2014-12-12].

[3] World Wide Web. http://en.wikipedia.org/wiki/World_Wide_Web, [cit. 2014-12-12].

[4] RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. https://tools.ietf.org/html/rfc2616, [cit. 2014-12-13].

[5] RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. https://tools.ietf.org/html/rfc3986, [cit. 2014-12-13].

[6] Usage of server-side programming languages for websites. http://w3techs.com/technologies/overview/programming_language/all, [cit. 2015-01-05].

[7] Facebook Developers. https://developers.facebook.com/, [cit. 2015-01-07].

[8] Google+ Platform - Google Developers. https://developers.google.com/+/web/, [cit. 2015-01-07].

[9] Twitter for Websites — Twitter Developers. https://dev.twitter.com/web/overview, [cit. 2015-01-07].

[10] Express - Node.js web application framework. http://expressjs.com/, [cit. 2015-03-28].

[11] AngularJS - Superheroic JavaScript MVW Framework. https://angularjs.org/, [cit. 2015-04-02].

[12] Castro, E.: *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek*. Brno : Computer Press, 2012, ISBN 978-80-251-3733-8.

[13] Greer, D.: Interactive Application Architecture Patterns. http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/, [cit. 2014-12-13].

[14] Rauch, G.: *Smashing Node.js: JavaScript Everywhere*. Wiley, 2012, ISBN 978-1-119-96259-5.

[15] Vicrey, K.: Authentication in Single Page Application. `https://vickev.com/#!/authentication-in-spa-node-passportjs-angularjs`, [cit. 2015-04-07].

[16] Zakas, N. C.: *JavaScript pro webové vývojáře : programujeme profesionálně*. Brno : Computer Press, 2009, ISBN 978-80-251-2509-0.

Dodatek A

Přístupové body serverového REST API

`GET /api/loggedin` - Test zda-li je uživatel přihlášen. Pokud ano, je vrácen objekt s informacemi o přihlášeném uživateli. Pokud ne, je vrácena hodnota 0. (*nezabezpečený zdroj*)

`POST /api/login` - Přihlášení lokálním účtem. V těle požadavku je očekáván e-mail a heslo. (*nezabezpečený zdroj*)

`POST /api/logout` - Odhlášení z aplikace. (*zabezpečený zdroj*)

`POST /api/register` - Vytvoření nového lokálního účtu. V těle požadavku je očekáván objekt s informacemi o novém uživateli. (*nezabezpečený zdroj*)

`POST /api/register/date` - Přidání data narození k účtu uživatele, který tento požadavek odeslal. Používá se po přihlášení přes účet Facebooku nebo Google+ a to v případě, že datum narození tito neposkytli. V těle požadavku je očekáváno datum narození. (*zabezpečený zdroj*)

`GET /api/users` - Vrátí kolekci objektů se základními informacemi o všech uživatelích v aplikaci. (*zabezpečený zdroj*)

`GET /api/users/:id` - Vrátí objekt s detailními informacemi o uživateli, jehož identifikátor byl předán jako parametr URL adresy. (*zabezpečený zdroj*)

`GET /api/users/skills` - Vrátí objekt s dovednostními úrovněmi uživatele, který tento požadavek odeslal. (*zabezpečený zdroj*)

`PUT /api/users/skills` - Aktualizuje dovednostní úrovně uživatele, který tento požadavek odeslal. V těle požadavku je očekáván objekt s novými hodnotami dovednostních úrovní. (*zabezpečený zdroj*)

`POST /api/users/photo` - Uloží profilovou fotografii k účtu uživatele, který tento požadavek odeslal. V těle požadavku je očekávána fotografie ve formátu JPG nebo PNG o maximální velikosti 50kB. (*zabezpečený zdroj*)

`GET /api/messages` - Vrátí kolekcí objektů s informacemi o všech soukromých zprávách uživatele, který tento požadavek odeslal. (*zabezpečený zdroj*)

`PUT /api/messages` - Přesune zprávy do koše nebo je smaže. Změní stav (doručená, odeslaná, v koši, smazaná) u 1 až *n* zprávy uživatele, který tento požadavek odeslal. V těle požadavku jsou očekávány identifikátory zpráv, kterých se změna týká a hodnota jejich nového stavu. (*zabezpečený zdroj*)

`POST /api/messages` - Posílání soukromých zpráv. Vytvoří novou odeslanou zprávu pro uživatele, který požadavek odeslal a novou doručenou zprávu pro uživatele jehož identifikátor byl předán v těle požadavku. Kromě identifikátoru se dále očekává v těle požadavku předmět a text zprávy. (*zabezpečený zdroj*)

`GET /api/messages/:id` - Vrátí objekt s detailními informacemi o zprávě, jejíž identifikátor byl předán jako parametr URL adresy. (*zabezpečený zdroj*)

`GET /api/events` - Vrátí kolekci objektů se základními informacemi o všech událostech v aplikaci. (*zabezpečený zdroj*)

`POST /api/events` - Vytvoří novou událost. V těle požadavku je očekáván objekt s informacemi o nové události. (*zabezpečený zdroj*)

`GET /api/events/:id` - Vrátí objekt s detailními informacemi o události, jejíž identifikátor byl předán jako parametr URL adresy. (*nezabezpečený zdroj*)

`POST /api/events/comment/:id` - Přidá komentář k události, jejíž identifikátor byl předán jako parametr URL adresy. V těle požadavku je očekáván text komentáře. (*zabezpečený zdroj*)

`PUT /api/events/user/:id` - Přihlášení uživatele, který požadavek odeslal, na událost, jejíž identifikátor byl předán jako parametr URL adresy. (*zabezpečený zdroj*)

`POST /api/events/user/:id` - Potvrzení/odmítnutí účasti uživatele na události, jejíž identifikátor byl předán jako parametr URL adresy. V těle zprávy je očekáván objekt s identifikátorem uživatele, kterého se změna týká a informace zda-li byl na událost přijat či nikoli. (*zabezpečený zdroj*)

`DELETE /api/events/user/:id` - Odhlášení uživatele, který požadavek odeslal, z události, jejíž identifikátor byl předán jako parametr URL adresy. (*zabezpečený zdroj*)

Následující 4 URL adresy se používají při přihlášení přes Facebook, nebo Google+. Princip jejich volání a činnosti je podrobně popsána v kapitole 5.3.

`GET /api/auth/facebook` - Požadavek na přihlášení přes Facebook. Následuje přesměrování na přihlašovací formulář Facebooku. (*nezabezpečený zdroj*)

`GET /api/auth/facebook/callback` - Na tuto URL adresu přesměrovává Facebook po úspěšné/neúspěšné autentizaci. (*nezabezpečený zdroj*)

`GET /api/auth/google` - Požadavek na přihlášení přes Google+. Následuje přesměrování na přihlašovací formulář Google+. (*nezabezpečený zdroj*)

`GET /api/auth/google/callback` - Na tuto URL adresu přesměrovává Google+ po úspěšné/neúspěšné autentizaci. (*nezabezpečený zdroj*)

Dodatek B

Testovací scénář

Test	Očekávaný výsledek	Provedení testu
Přihlášení		
lokální účet - chybné údaje	okno s informací o chybě	OK
lokální účet - správné údaje	uživatel je přihlášen do aplikace	OK
Facebook - chybné údaje	uživatel není přihlášen do aplikace	OK
Facebook - správné údaje	uživatel je přihlášen do aplikace	OK
Google+ - chybné údaje	uživatel není přihlášen do aplikace	OK
Google+ - správné údaje	uživatel je přihlášen do aplikace	OK
Odhlásit se	uživatel je odhlášen z aplikace	OK
Registrace		
formulář je chybně vyplněn	formulář nelze odeslat	OK
formulář je správně vyplněn	vytvořen nový lokální účet	OK
Sportovní události		
zobrazení přehledu událostí	zobrazen přehled událostí	OK
změna filtru událostí	zobrazené události odpovídají filtru	OK
vytvořit událost - chybné údaje	formulář nelze odeslat	OK
vytvořit událost - správné údaje	vytvořena nová sportovní událost	OK
Detail sportovní události		
sdílet na Facebooku	okno pro sdílení na Facebooku	OK
sdílet na Twitteru	okno pro sdílení na Twitteru	OK
sdílet na Google+	okno pro sdílení na Google+	OK
zobrazení - nepřihlášen do aplikace	zobrazeno datum, čas, místo, mapa, cena a doporučená úroveň	OK
zobrazení - přihlášen - neúčastník	zobrazen navíc průměrný věk, organizátor, účastníci, volná místa	OK
zobrazení - přihlášen - účastník	zobrazeny navíc komentáře	OK
zobrazení - přihlášen - organizátor	zobrazeni navíc žadatelé	OK
Sportovci		
zobrazení přehledu sportovců	zobrazen přehled sportovců	OK
změna filtru sportovců	zobrazení sportovci odpovídají filtru	OK
Detail sportovce		
sportovec není přihlášený uživatel	profil s možností poslat zprávu	OK
sportovec je přihlášený uživatel	profil s možností nastavit úrovně a nahrát profilovou fotografii	OK

Test	Očekávaný výsledek	Provedení testu
Zprávy		
poslat zprávu	zpráva uložena v odeslaných odesilatele a přijatých příjemce	OK
zobrazení doručených zpráv	zobrazeny doručené zprávy	OK
zobrazení detailu doruč. zprávy	zobrazena zpráva a tlač. odpovědět	OK
zobrazení odeslaných zpráv	zobrazeny odeslané zprávy	OK
zobrazení detailu odesl. zprávy	zobrazena zpráva	OK
vybrat vše	vybrány všechny zprávy ve složce	OK
přesunout vybrané do koše	vybrané zprávy přesunuty do koše	OK
smazat vybrané natrvalo	vybrané zprávy odstraněny	OK
Nastavení dovednostních úrovní		
zobrazit dovednostní úrovně	dovednostní úrovně zobrazeny	OK
zatrhnout sport který chci nastavit	zatržené mají aktivní posuvník	OK
nastavit dovednostní úroveň	lze nastavit posuvníkem 1 až 10	OK
uložit	uloží aktuálně nastavený stav	OK