VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ SLUŽBA PRO TVORBU INFORMAČNÍCH STRÁNEK K UDÁLOSTEM

WEB SERVICE FOR CREATION OF INFORMATIVE WEB PAGES ABOUT EVENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ADAM LINKA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Linka Adam**
Program: Informační technologie
Název: **Webová služba pro tvorbu informačních stránek k událostem**
Web Service for Creation of Informative Web Pages about Events
Kategorie: Web

Zadání:

1. Seznamte se s principy tvorby webových aplikaci, dostupnými vývojovými prostředími a frameworky.
2. Analyzujte požadavky na webovou službu umožňující vytvářet libovolné stránky informující o soukromých událostech včetně odeslání emailu zúčastněným, možnost se událost přihlásit a být informován o změnách týkajících se události.
3. Navrhněte webovou službu dle požadavků, využijte k tomu vhodné modelovací techniky.
4. Navrženou webovou službu implementujte a otestujte její funkčnost na vhodném datovém vzorku.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování a rozšíření tohoto projektu.

Literatura:

- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 24. října 2019

Abstrakt

Tato bakalářská práce se zabývá analýzou technologií webových aplikací, dostupných vývojových prostředí, aplikačních rámců a v neposlední řadě také návrhem a vývojem webové služby. Tato služba svým uživatelům poskytuje možnost vytvořit informační webové stránky k pořádaným událostem soukromého charakteru. Služba je implementována v jazyce JavaScript. Na serverové straně je použit aplikační rámec Express, běhové prostředí Node.js a databáze MongoDB. Klientská část služby je implementována za použití aplikačního rámce React.

Abstract

This bachelor's thesis deals with analyzing technologies of web applications, available development environments, frameworks and last but not least also with designing and developing a web service. Such service provides creation of informational web pages about upcoming private events. The service is implemented in JavaScript. Express framework running in Node.js environment is used on the server-side. MongoDB is used as a database. Client-side is implemented with use of React framework.

Klíčová slova

událost, web, webová aplikace, aplikační rámec, server, klient, MongoDB, Node.js, Express, React, JavaScript, MERN

Keywords

event, web, web application, framework, server, client, MongoDB, Node.js, Express, React, JavaScript, MERN

Citace

LINKA, Adam. *Webová služba pro tvorbu informačních stránek k událostem.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Webová služba pro tvorbu informačních stránek k událostem

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Adam Linka
28. května 2020

Poděkování

Mé poděkování patří panu Ing. Vladímíru Bartíkovi, Ph.D. za vedení této práce, kontrolu uvedených informací a za jeho odborné rady. Dále bych chtěl poděkovat Lence Linkové za jazykovou korekturu a Pavlíně Dratvové za otestování výsledné webové služby.

Obsah

1 Úvod **2**

2 Web a jeho technologie **3**
2.1 Webová stránka . . . 3
2.2 HyperText Transfer Protocol . . . 5
2.3 Uniform Resource Locator . . . 6

3 Technologie a tvorba webových aplikací **7**
3.1 Model View Controller . . . 7
3.2 Technologie klientské části . . . 8
3.3 Aplikační rámce klientské části . . . 10
3.4 Technologie serverové části . . . 11
3.5 Aplikační rámce serverové části . . . 13
3.6 Databáze . . . 14
3.7 Zabezpečení . . . 16

4 Požadavky na službu a její návrh **20**
4.1 Existující řešení . . . 20
4.2 Požadavky . . . 23
4.3 Návrh architektury . . . 26
4.4 Návrh struktury dat . . . 27
4.5 Uživatel . . . 27
4.6 Událost . . . 29

5 Implementace **34**
5.1 Klientská část . . . 35
5.2 Serverová část . . . 38
5.3 Validace dat . . . 40

6 Testování **41**

7 Závěr **45**

Literatura **46**

A Seznam koncových bodů API webové služby **48**

B Struktura adresáře webové služby **50**

Kapitola 1

Úvod

Setkávání je nedílnou součástí každého života. Již od narození se setkáváme s ostatními lidmi, zvířaty, věcmi, živou i neživou přírodou, setkat se zkrátka můžeme s kýmkoliv a čímkoliv. Jsou to ale právě lidská setkání, při kterých vznikají velké věci a vzpomínky na celý život. Dva lidé nebo menší skupina se na setkání snadno domluví přes telefon, chat či email. Co ale větší soukromé akce – svatby, oslavy, pohřby, třídní srazy, přátelská sportovní klání a podobné události? Jak účastníky hromadně a přehledně informovat o všech detailech dané události a případně o změnách programu? Jak a kde sdílet poděkování, výsledky či odkaz na fotografie z proběhlé akce?

Řešení nabízejí sociální sítě, například Facebook. Ne však všichni lidé sociální sítě používají – buď je ke svému životu nepotřebují, nebo vyloženě nechtějí poskytovat své osobní informace třetím stranám a potenciálním obchodníkům s daty. Cílem této práce je proto vytvořit jednoduchou webovou službu, která bude sloužit právě ke sdílení informací o soukromých událostech. Vytváření ani prohlížení událostí by nemělo vyžadovat registraci. Založení uživatelského účtu, které přinese jednodušší správu událostí, by ale mělo být umožněno. Služba by dále měla nabízet odběr aktualizací skrz zasílání notifikací na zadanou emailovou adresu.

K implementaci webové aplikace/služby je zapotřebí získat znalosti o fungování webu, jeho standardech a úskalích. Je také vhodné zanalyzovat dostupné aplikační rámce a knihovny, a vybrat z nich ty, které se pro daný projekt nejvíce hodí, a především které nejvíce vyhovují samotnému vývojáři, případně vývojářům. Tato bakalářská práce tedy vysvětluje všechny tyto pojmy, standardy a technologie, jejichž znalosti jsou následně uplatňovány při návrhu a samotné implementaci aplikace.

Ve druhé kapitole je vysvětlen princip fungování webu včetně popisu a objasnění technologií a standardů, které s ním souvisí. Třetí kapitola obsahuje vysvětlení technologií a principů, které se uplatňují při vývoji webových aplikací. Dále je zde výčet a srovnání nejpoužívanějších aplikačních rámců a knihoven, a to jak z klientské, tak i ze serverové části webové aplikace. V neposlední řadě se zde také nachází srovnání relačních a tzv. NoSQL databází. Čtvrtá kapitola se zabývá analýzou existujících řešení pro sdílení informací o událostech, dále pak detailním popisem případů užití nově vznikající aplikace a také jejím návrhem. V páté kapitole je přiblížena samotná implementace webové aplikace a způsoby využití zvolených technologií, aplikačních rámců a knihoven. Šestá kapitola je věnována testování implementované aplikace, tedy jakým způsobem byla aplikace testována a jaké byly výsledky provedených testů. V závěru je pak zhodnoceno naplnění cílů bakalářské práce včetně zhodnocení výsledné webové aplikace. Rovněž jsou zmíněny způsoby, jakými by se dala výsledná služba v budoucnu vylepšit nebo rozšířit.

Kapitola 2

Web a jeho technologie

World Wide Web (také zkráceně web) je internetový systém sloužící k distribuci hypertextových dokumentů. Vynalezl ho v roce 1989 Tim Berners-Lee, který o rok později naprogramoval také první webový prohlížeč.

Dokumenty na webu mohou být v různých formátech a jsou identifikovány pomocí adresy URL. Dokumenty jsou většinou psány ve značkovacím jazyce HTML, obsahují text, multimediální data, a mohou na sebe vzájemně odkazovat pomocí hypertextových odkazů. Tyto odkazy mohou směřovat k dokumentům uloženým na stejném i vzdáleném serveru. Díky tomu také vzniklo označení web – síť.

Jednotlivé dokumenty jsou tedy uložené na serverech, a klient k nim přistupuje většinou skrz webový prohlížeč zadáním URL adresy nebo kliknutím na hypertextový odkaz. Webový prohlížeč slouží k intuitivnímu vyžádání daného dokumentu od příslušného serveru a k jeho následnému zobrazení. Pro vyžádání a přenos dokumentů je využíván protokol HTTP, případně jeho zabezpečená verze HTTPS. Od roku 2004 se začal používat pojem Web 2.0, který kromě posunu v používaných technologiích a standardech přinesl především novou koncepci tvorby webového obsahu. Rozmohly se stránky/služby, jejichž obsah tvoří sami uživatelé. Jsou to například sociální sítě, blogy, služby pro sdílení fotografií a videí, wiki a v neposlední řadě webové aplikace. [3][4]

2.1 Webová stránka

Pojem webová stránka (anglicky web page) je možno chápat jako jeden HTML dokument, který lze zobrazit ve webovém prohlížeči. Všechny webové stránky na webu jsou dostupné skrz unikátní adresu – URL.

Pojmem webová stránka (anglicky web site) se ale také rozumí kolekce vzájemně propojených webových stránek. Webové stránky v této kolekci mají stejné doménové jméno a většinou na sebe tematicky navazují.

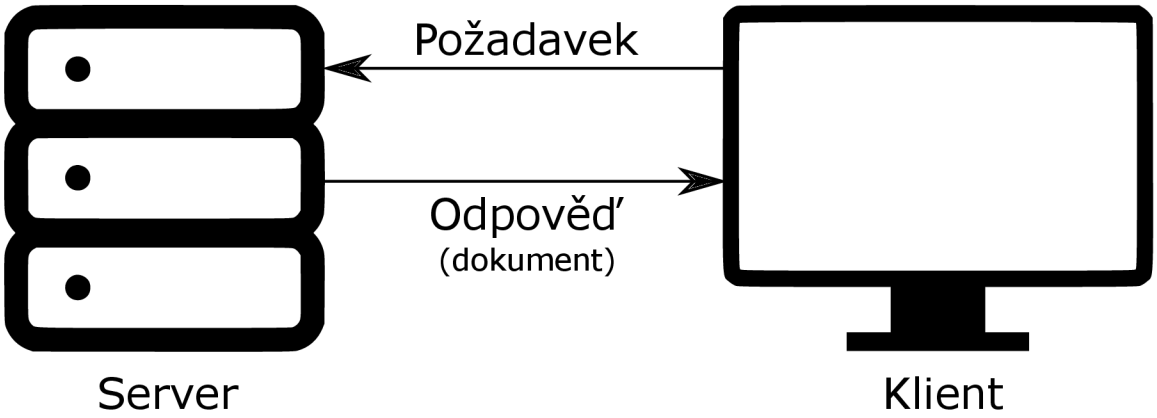
Webové stránky se dělí na statické a dynamické.

Statická stránka

Statická webová stránka se vyznačuje tím, že informace zobrazované uživateli jsou umístěny přímo v HTML dokumentu. Webový prohlížeč si tedy po zadání adresy od serveru vyžádá dokument, který následně zobrazí přesně tak, jak je uložený na serveru (viz. obr. 2.1).

Výhodou statických webových stránek je možnost jednoduše individuálně přizpůsobit každou stránku. Počáteční cena je většinou nižší než u dynamických stránek. Při úpravě nebo přidávání obsahu však celková cena rychle roste.

Statické webové stránky je tedy vhodné použít v projektech, kde ke změně nebo přidávání obsahu dochází zřídkakdy, např. informační webové stránky společnosti. [18]



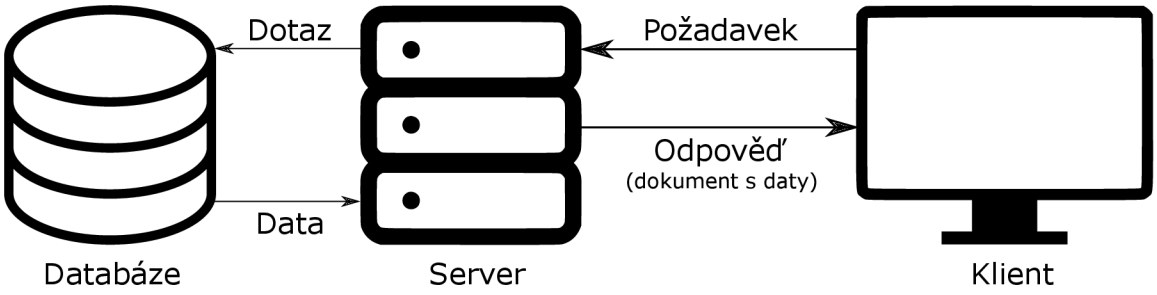
Obrázek 2.1: Princip získání statické stránky ze serveru

Dynamická stránka

Dynamická webová stránka na druhou stranu vyžaduje logiku minimálně na straně serveru. Základní struktura stránky je stále založena na HTML dokumentu. Místo informací je ale v dokumentu uložen kód, který je před odesláním dokumentu klientovi vyhodnocen na serveru (viz. obr. 2.2). Díky tomu může serverová logika do výsledného dokumentu před odesláním vložit aktuální informace z databáze nebo jiného datového úložiště. Zároveň může uživatel ovlivnit, jaká data a jakým způsobem je chce zobrazit.

I přes náročnější vývoj a vyšší počáteční cenu tak mají dynamické webové stránky obrovskou výhodu. Následná změna nebo přidání obsahu je jen otázkou aktualizace, resp. přidání obsahu v databázi, což může být uživatelsky přívětivě provedeno například přes Systém pro správu obsahu (také známé pod anglickou zkratkou CMS).

Na dynamických stránkách je založen výše zmíněný Web 2.0, kdy jeho obsah tvoří sami uživatelé. Už z tohoto principu by bylo využití statických stránek nemyslitelné. [18]



Obrázek 2.2: Princip získání dynamické stránky ze serveru

2.2 HyperText Transfer Protocol

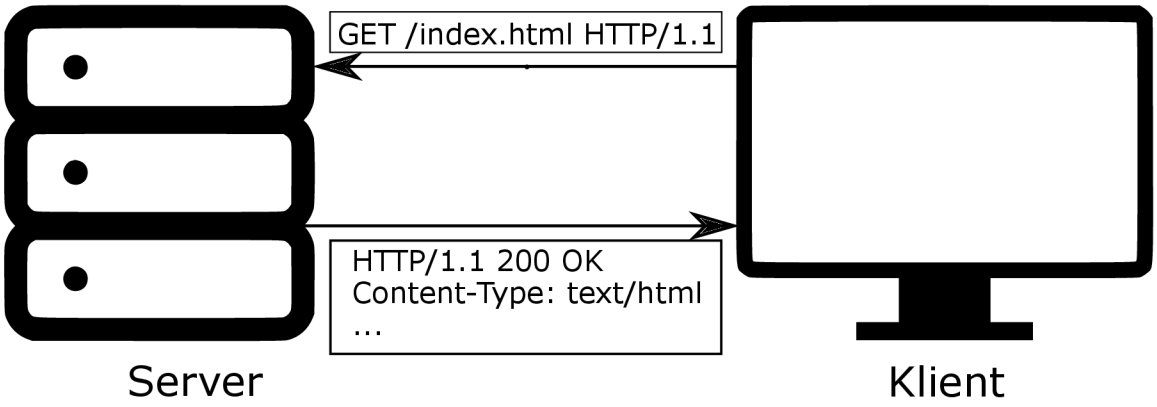
Hypertext Transfer Protocol je internetový protokol pro komunikaci se servery a přenos informací. Řadí se mezi protokoly aplikační vrstvy, pro přenos dat využívá protokol TCP a standardně je dostupný na portu číslo 80. Jeho šifrovaná verze HTTPS využívá port 403. HTTP je bezstavový protokol, což znamená, že sám o sobě neuchovává informace o uskutečněných požadavcích a odpovědích. Pracuje na principu požadavek – odpověď. V kontextu webu tedy klient (webový prohlížeč) posílá požadavek, který server zpracuje a zpátky zašle odpověď (viz. obr. 2.3). HTTP definuje metody, kterými je možno zaslat požadavek, a to následující:

- GET – požadavek na zaslání dokumentu, výchozí metoda ve webových prohlížečích,
- HEAD – požadavek na zaslání metadat dokumentu (velikost, typ, datum změny…),
- POST – odeslání dat na server, často používaná metoda pro odeslání dat z formuláře,
- PUT – nahrání dat na server,
- DELETE – smazání dat ze serveru,
- TRACE – odeslání kopie obdrženého požadavku zpět původnímu odesílateli,
- OPTIONS – dotázání serveru na jím podporované metody,
- CONNECT – navázání obousměrné komunikace s uvedeným zdrojem, používá se při HTTPS.

Odpověď se pak skládá ze stavového kódu, hlaviček a případně také dat. Stavový kód se nachází na prvním řádku. Stavové kódy se dělí do 5 hlavních kategorií podle počáteční číslice. Další dvě číslice pak slouží k určení konkrétního stavového kódu.

- 1XX – informační povaha
- 2XX – vyjadřuje kladné zpracování požadavku
- 3XX – obstarává přesměrování
- 4XX – chyba v požadavku klienta
- 5XX – chyba serveru

Nejčastěji se setkáváme se stavovými kódy 200 OK, 201 Created, 401 Unauthorized, 403 Forbidden, 404 Not Found, 500 Internal Server Error. Hlavičky obsahují informace o klientovi (verze prohlížeče, operační systém…), informace o znakové sadě, typ odesílaného obsahu, autentifikační údaje apod. Každá hlavička je na samostatném řádku. Případná data následují za poslední hlavičkou a jedním prázdným řádkem. [12]

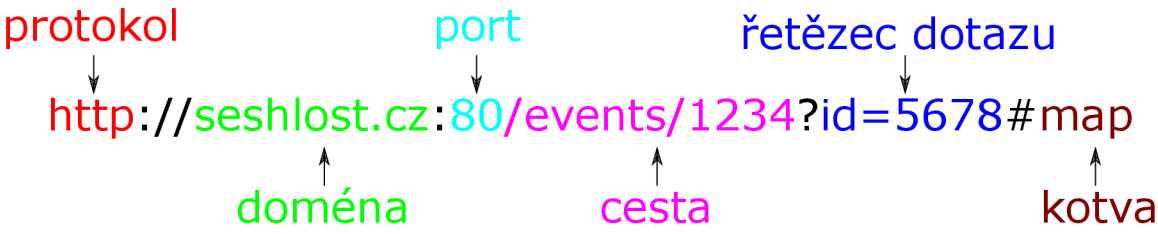


Obrázek 2.3: Diagram principu požadavek-odpověď

2.3 Uniform Resource Locator

Uniform Resource Locator (také URL, v češtině jednotná adresa zdroje) je sekvence znaků s předem danou strukturou (viz. obr. 2.4), která slouží k unikátnímu popisu umístění zdroje na webu. Zdroj může být HTML dokument, CSS dokument, obrázek, soubor atd. Často ale URL neukazuje na fyzický dokument, nýbrž slouží pouze k určení akce, kterou má server provést.

URL se skládá z několika částí, z nichž jen některé jsou povinné. URL začíná označením protokolu, které se ve webových prohlížečích většinou doplňuje automaticky na http nebo https, jiný protokol je třeba specifikovat. Dále následuje dvojtečka, dvě lomítka a poté doménové jméno, případně přímo IP adresa serveru. Následuje dvojtečka a číslo portu. To je ve webových prohlížečích rovněž doplněno automaticky, neboť standardní protokoly mají pevně dané číslo portu. Pokud se jedná o nestandardní port, je třeba jej uvést. Za číslem portu se pak nachází cesta ke zdroji na serveru. Poté může následovat otazník, za kterým se nachází řetězec dotazu – dvojice „klíč=hodnota“ v URL kódování. Jednotlivé dvojice jsou odděleny znakem &. Tímto způsobem lze předat serveru data přímo v URL. Na konci URL se pak může nacházet ještě znak #, za kterým následuje tzv. kotva, která většinou blíže specifikuje místo v cílovém dokumentu. Webový prohlížeč pak na toto místo může po načtení stránky, případně po kliknutí na odkaz, automaticky scrollovat. [13]



Obrázek 2.4: Jednotlivé části URL

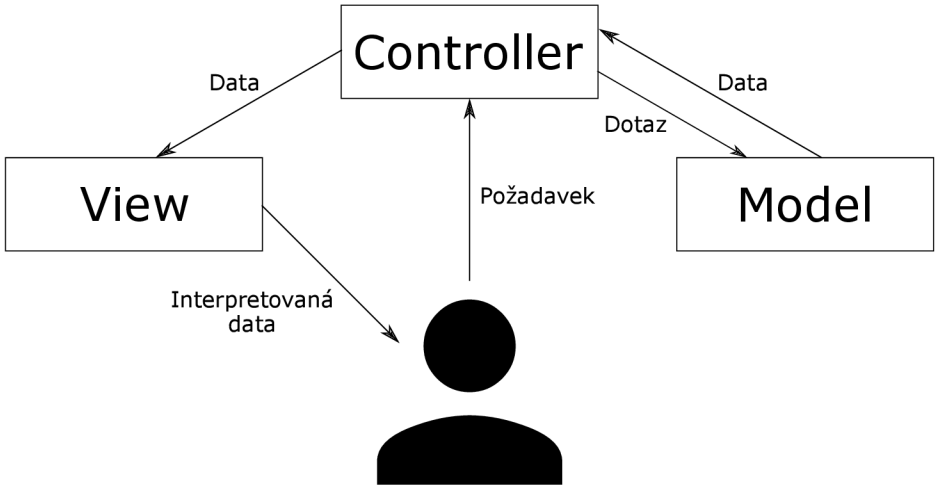
Kapitola 3

Technologie a tvorba webových aplikací

Webové aplikace jsou zpravidla rozděleny na dvě části – klientskou a serverovou. Už z názvů je patrné, jak toto rozdělení vzniklo. Jelikož každá část zastává jinou funkci, jsou s nimi také spojeny jiné technologie a aplikační rámce. V této kapitole jsou proto vysvětleny nejdůležitější technologie pro vývoj webových aplikací a nachází se zde také popis a srovnání populárních aplikačních rámců. Taktéž jsou zmíněny dva hlavní typy databází – relační a tzv. NoSQL, jelikož dlouhodobé ukládání dat je pro funkčnost webových aplikací téměř nepostradatelné.

3.1 Model View Controller

Model View Controller (zkráceně MVC) je architektonický vzor, kterým se řídí velké množství webových aplikací. Aplikace vytvářené s využitím tohoto vzoru se skládají ze tří komponent (viz. obr. 3.1). View neboli pohled prezentuje data uživateli a umožňuje mu s nimi interagovat. Controller reaguje na interakce uživatele, jeho požadavky zpracovává, zajišťuje změny v modelu a data k interpretaci předává do View. Pod pojmem Model pak chápeme datovou vrstvu celého systému, nejčastěji tedy databázi. [21]



Obrázek 3.1: Diagram architektonického vzoru MVC

3.2 Technologie klientské části

Úlohou klientské částí webové aplikace je zobrazovat data uživateli a reagovat na jeho akce. Klientská část je zkrátka to, co koncový uživatel vidí ve svém prohlížeči a s čím nějakým způsobem interaguje. Je to například stisknutí tlačítka, kliknutí na odkaz, vyplnění textového pole, zobrazení zpětné vazby pro uživatele jako je třeba načítání, chybové hlášky, potvrzení provedené akce atd. Pod pojmem technologie klientské části tedy rozumíme především programovací nebo značkovací jazyky, které je možno interpretovat ve webovém prohlížeči.

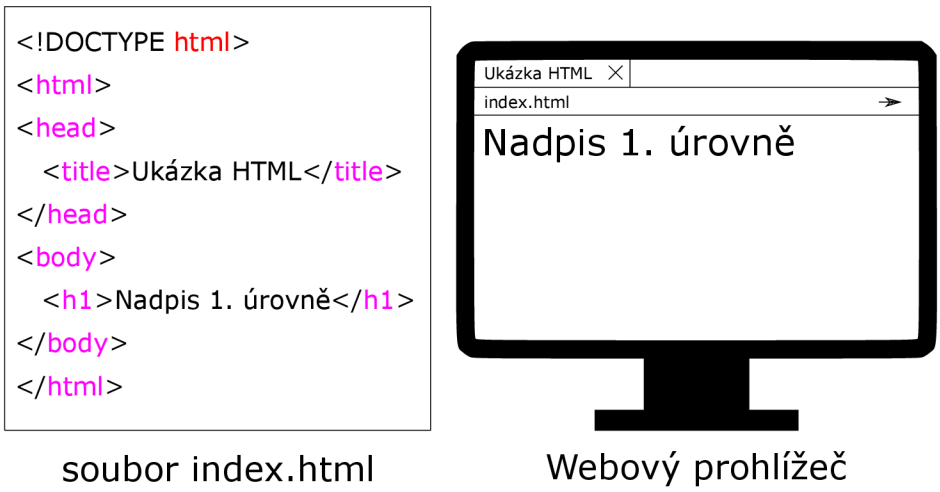
HyperText Markup Language

Hypertext Markup Language (zkráceně HTML) je značkovací jazyk používaný pro tvorbu webových (hypertextových) stránek. Vznikl v roce 1990 společně s protokolem HTTP pro vnitřní účely výzkumného střediska CERN. Jeho autory jsou Tim Berners-Lee a Robert Cailliau.

Jazyk se skládá z množiny značek a jejich vlastností (viz. obr. 3.2). Mezi značky jsou uzavírány jednotlivé elementy stránky, čímž získávají svůj význam. Názvy značek a jejich vlastnosti se uzavírají mezi úhlové závorky. Značky dělíme na párové a nepárové. Párové značky mají koncovou značku, která se od počáteční značky liší přidaným lomítkem před názvem značky. Nepárové značky se skládají pouze z jedné (počáteční) značky.

Dále se značky dělí na strukturální, popisné a stylistické. Strukturální rozvrhují strukturu dokumentu, jsou to například značky , <h1> a <p>. Popisné neboli sémantické značky popisují obsah prvků, který uzavírají, například <title> nebo <address>. Stylistické značky určují vzhled uzavřeného prvku. Od jejich použití se však upouští, přičemž jsou nahrazovány kaskádovými styly.

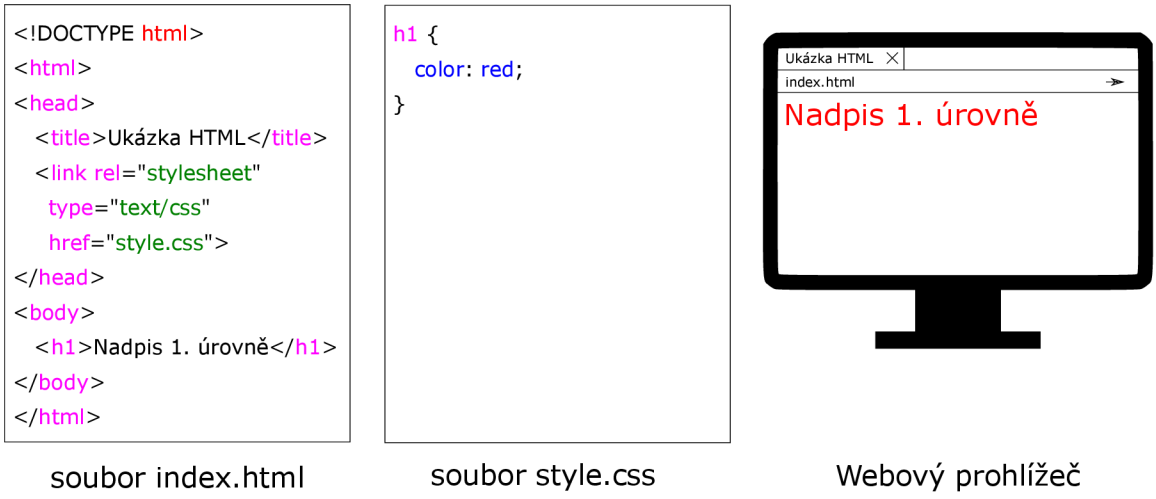
HTML dokument má danou strukturu. Nejprve je nutné pomocí značky <!DOCTYPE html> určit, že se jedná o dokument v jazyce HTML. Mezi párovými značkami <html> se poté nachází celý dokument, který obsahuje hlavičku uvnitř značek <head> a samotné tělo dokumentu mezi značkami <body>. [14]



Obrázek 3.2: HTML kód a jeho zobrazení ve webovém prohlížeči

Cascading Style Sheets

Cascading Style Sheets je jazyk pro popis způsobu zobrazení dokumentu napsaném ve značkovacím jazyce (HTML, XML…). Pomocí pravidel udává vzhled jednotlivých elementů (viz. obr. 3.3). Jazyk byl navržen organizací Word Wide Web Consortium a první verze byla zpřístupněna 17. prosince 1996. Hlavním smyslem CSS je možnost oddělit vzhled hypertextového dokumentu od jeho struktury a obsahu. Dokument v jazyce CSS se skládá z pravidel. Každé pravidlo obsahuje selektor a blok deklarací. Selektor udává element nebo množinu elementů, na které má být pravidlo aplikováno. Dva speciální selektory jsou třída a identifikátor, kterými je možno označit jakýkoliv HTML element. Blok deklarací pak obsahuje deklarace ve formátu „název-vlastnosti: hodnota“. Jednotlivé deklarace jsou odděleny středníkem. Vzhled dokumentu HTML lze popsat pomocí CSS buď přímo v rámci dokumentu, a to uzavřením pravidel mezi značky <style type="text/css»…</style>, nebo, a to je častější, jsou pravidla definována v externím souboru, na který je v hlavičce HTML dokumentu odkázáno pomocí značek <link>. [15]



Obrázek 3.3: Definování stylu nadpisu 1. úrovně (h1) v souboru style.css

JavaScript

JavaScript je vysokoúrovňový, dynamický, beztypový a interpretovaný programovací jazyk vyvinutý společností Netscape v roce 1995. Jako standardizovaný pod názvem ECMAScript je podporovaný ve všech moderních webových prohlížečích. Společně s HTML a CSS se tak JavaScript stal základní technologií pro tvorbu webových stránek. JavaScript je často využívaný pro obsluhu událostí a uživatelských akcí, dynamickou změnu obsahu stránky a v neposlední řadě také pro komunikaci klienta se serverem pomocí technologie AJAX[1]. [16]

Bootstrap

Bootstrap je knihovna v jazyce CSS a JavaScript, která umožňuje rychlé a snadné stylování HTML dokumentů. Díky celé škále předdefinovaných stylovacích pravidel umožňuje stylování HTML dokumentů pouhým přidáváním tříd a identifikátorů k jednotlivým elementům

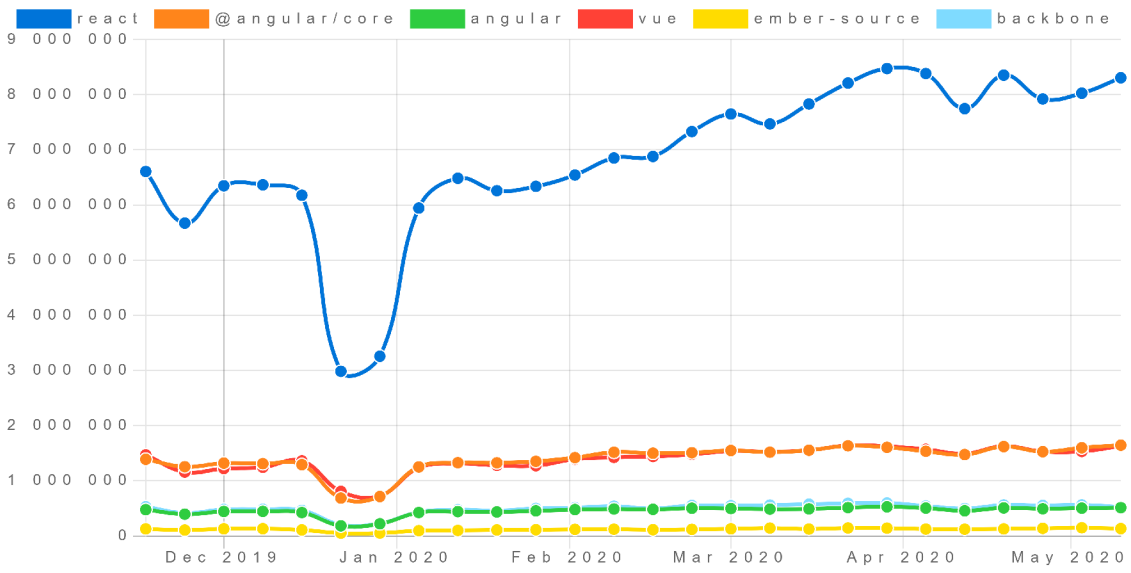
[1] AJAX – Asynchronous JavaScript and XML (Asynchronní JavaScript a XML) je soubor technologií používaných pro získání a zobrazení dat bez nutnosti stránku znovu načíst.

dokumentu. Tímto způsobem je možné rychle vytvořit rozložení stránky, navigaci, tlačítka, formuláře a další důležití prvky bez nutnosti ručně definovat CSS pravidla. Bootstrap taktéž používá tzv. grid systém, který umožňuje nastavit velikost jednotlivých elementů v závislosti na velikosti obrazovky zobrazovacího zařízení. Opět jen pomocí přidání třídy k elementu je možné zvolit, jak bude daný element vypadat na malé obrazovce mobilního telefonu a jak na velkém monitoru stolního počítače. Přidání podpory pro Bootstrap do projektu je velmi jednoduché a probíhá buď přímo stažením stylovacího a JavaScriptového souboru do příslušných adresářů, nebo odkázáním na CDN[2], kde se tyto soubory nachází.

3.3 Aplikační rámce klientské části

Aplikační rámce klientské části jsou zpravidla vystavěny nad jazykem JavaScript, který je podporován všemi moderními webovými prohlížeči. Aplikační rámec vývojáři přináší celou řadu předpřipravených funkcí, které by jinak musel sám implementovat, což by bylo časově náročné, nehledě na možné zanesení bezpečnostních a jiných chyb do aplikace. Důvod pro používání rámců v klientské části aplikace je tedy prostý – vývojáři stačí k implementaci klientské části aplikace mnohem méně kódu, než kdyby tutéž funkcionalitu implementoval přímo v jazyce JavaScript, navíc bude výsledné řešení pravděpodobně více odladěné a bezpečné. [5]

Na výběr je v dnešní době spousta aplikačních rámců, téměř všechny staví na jazyce JavaScript. Je běžné, že populární rámce zaštiťují některé z velkých technologických společností, čistě komunitní rámce ale nejsou výjimkou. Kromě podpory se pak dále liší složitostí na naučení, rozsahem nabízených funkcí a určením. Níže je výčet aktuálně nejpopulárnějších aplikačních rámců pro vývoj klientské časti (viz. obr. 3.4). [10]



Obrázek 3.4: Co se týče počtu stažení ze systému npm za posledních 6 měsíců, React nad konkurencí jasně vede. Jinak tomu není ani v ostatních dostupných statistikách. Převzato z: https://www.npmtrends.com/.

[2]CDN – Content Delivery Network (síť pro doručování obsahu)

React.js

React[3] je knihovna pro tvorbu uživatelských rozhraní v jazyce JavaScript. Je vyvíjena společností Facebook a komunitou vývojářů. Využívá se především pro tvorbu jednostránkových webových aplikací a mobilních aplikací. React využívá konceptu komponent, které představují zapouzdřené části rozhraní. React taktéž využívá virtuální DOM[4], který automaticky překresluje jen ty části uživatelského rozhraní, které se změnily. Tím je zajištěna jak rychlost výsledného rozhraní, tak i komfort při jeho vývoji.

Angular

Na rozdíl od většiny populárních klientských aplikačních rámců, které jsou založeny na JavaScriptu, je Angular[5] založený na jazyku TypeScript. Za jeho vývojem stojí společnost Google a podobně jako React je Angular založený na komponentách. Lze se také setkat s označením Angular 2, jelikož nahradil aplikační rámec AngularJS – rovněž vyvíjený společností Google. V původním AngularJS se nedoporučuje začínat vývoj, navíc tyto dva aplikační rámce nejsou navzájem kompatibilní.

Vue.js

Vue.js[6] je webový aplikační rámec založený na JavaScriptu. Oproti konkurenci je velmi snadný na naučení, přitom je vhodný pro vývoj složitějších aplikací. Stejně jako React využívá systému komponent a virtuální DOM. I přesto, že jeho vývoj není podpořený velkými IT společnostmi, jedná se o rychle rostoucí a stále oblíbenější klientský aplikační rámec.

Ember.js

Ember[7] je jeden z aplikačních rámců, které jsou obtížnější na naučení. Je založen na architektonickém vzoru Model-View-ViewModel, využívá komponenty a obousměrnou vazbu dat. Je vhodný pro komplexní aplikace. Nedostatkem ale může být poměrně malá komunita.

Backbone.js

Backbone[8] je jednoduchý a efektivní aplikační rámec v jazyce JavaScript. Neposkytuje vazbu dat. Díky své jednoduchosti je vhodný pro začátečníky a menší aplikace. Naopak pro komplexnější projekty se nehodí z důvodu nižšího výkonu.

3.4 Technologie serverové části

Serverová část je pravý opak klientské části. Koncový uživatel tuto část nevidí a ani s ní nijak neinteraguje. V této části je implementována veškerá vnitřní logika aplikace. Zpravidla

[3] https://reactjs.org/

[4] DOM – Document Object Model (objektový model dokumentu) je objektově orientovaná reprezentace XML nebo HTML dokumentu.

[5] https://angular.io/

[6] https://vuejs.org/

[7] https://emberjs.com/

[8] https://backbonejs.org/

je to přijímání požadavků a odesílání odpovědí klientské části, práce s databází, verifikace dat, autentifikace, správa sezení apod. Na rozdíl od klientské části není ta serverová omezena schopnostmi webových prohlížečů, a proto může být naprogramována za použití téměř jakéhokoliv programovacího jazyka, který podporuje síťové operace. Níže je uveden výčet základních technologií a nejpopulárnějších aplikačních rámců pro vývoj serverové části webových aplikací.

Node.js

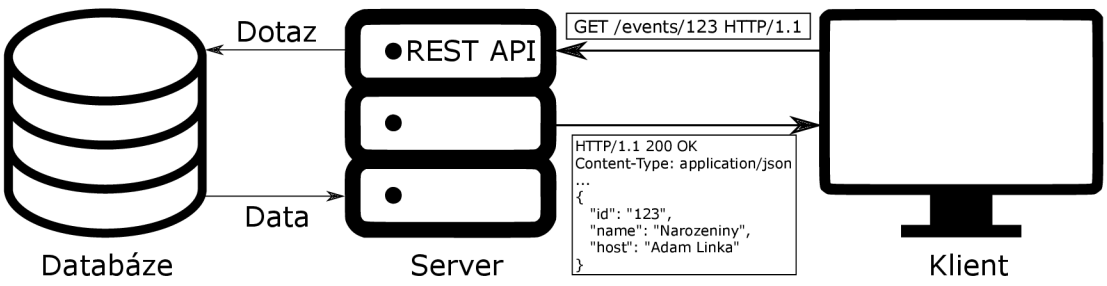
Node.js je multiplatformní prostředí pro běh JavaScriptu mimo webový prohlížeč. Využívá interpret V8, který byl původně vyvinut pro běh JavaScriptu v prohlížeči Google Chrome. I když je v tomto prostředí možno spouštět jakýkoliv kód napsaný v jazyce JavaScript, je primárně určeno a navrženo pro provoz serverové části webových aplikací. Z tohoto důvodu je vestavěný modul http, který umožňuje vytvořit webový server. Součástí Node.js je také balíčkovací systém npm, pomocí kterého lze snadno a rychle nainstalovat potřebné závislosti.

Representational State Transfer

Representational State Transfer je architektura rozhraní, která umožňuje manipulovat s daty na serveru pomocí standardních metod HTTP. REST byl představen a definován Royem Fieldingem, jež je jeden z autorů protokolu HTTP. REST definuje čtyři metody, které jsou implementovány pomocí odpovídajících metod HTTP a korespondují s CRUD operacemi.

- POST (Create) – vytvoření dat,
- GET (Retrieve) – získání dat,
- PUT (Update) – změna/úprava dat,
- DELETE (Delete) – smazání dat.

S využitím těchto metod jsou poté kontaktovány předem definované koncové body serveru, čímž na serveru dochází ke korespondující akci s daty (viz. obr. 3.5). Pro výměnu dat je nejčastěji používán formát JSON. REST je v dnešní době velmi rozšířený ve webových aplikacích/službách, kde se pomocí něj implementuje API[9]. K rozhraní implementovanému jako REST je poté možno přistupovat i bez nutnosti využít původní klientskou aplikaci. [9]



Obrázek 3.5: Princip fungováni REST API

[9]API – Application Programming Interface (rozhraní pro programováni aplikace) definuje, jaké požadavky, funkce atd. může programátor využívat, a jaké konvence při tom musí dodržovat.

3.5 Aplikační rámce serverové části

Serverové aplikační rámce jsou softwarové rámce usnadňující vývoj, údržbu a rozšiřování serverové části webových aplikací. Poskytují nástroje a knihovny, které zjednodušují vývoj běžně používaných funkcionalit. Mezi tyto funkcionality patří především práce s HTTP požadavky a odpověďmi, směrování požadavků na příslušné obslužné rutiny, práce s databázemi, podpora sezení a autentifikace. Webový server je sice možné vytvořit bez použití aplikačního rámce, vývoj se tím však výrazně prodlouží, neboť vývojář bude muset všechny součásti implementovat pouze za použití vestavěných funkci daného programovacího jazyka. Takto vytvořený webový server může také obsahovat bezpečnostní a jiné chyby.

V dnešní době je na výběr nespočet serverových aplikačních rámců, přičemž se liší především programovacím jazykem, nad kterým jsou implementovány, a celkovou filozofií. Některé rámce obsahují všechny potřebné funkce v základu, jiné jsou naopak v základu minimalistické a dodatečné funkce jsou dostupné ve formě knihoven. Níže je uvedeno několik populárních aplikačních rámců a jejich popis.

Django

Django[10] je vysokoúrovňový volně dostupný webový aplikační rámec využívající programovací jazyk Python. V základu obsahuje všechny důležité funkce pro vývoj serverové části webových aplikací, které navzájem dobře fungují a drží se jednotného návrhu. Jeho součástí jsou tedy např. objektově relační mapování, middleware, autentifikace, HTTP knihovny, admin panel a další. Oficiálně podporuje databáze PostgreSQL, MySQL a SQLite.

Django využívají například webové stránky Instagram, Mozilla, Nation Geographic, Pinterest a další.

Express

Express[11] je webový aplikační rámec pro běhové prostředí Node.js napsaný v jazyce JavaScript. Je navržen pro vývoj webových aplikací a API, proto v základu Express obsahuje sadu funkcí jako například směrování a podporu pro middleware. Narozdíl od Django ale neobsahuje všechny potřebné funkce. Ty jsou stejně jako samotný Express dostupné skrz npm jako nezávislé balíčky.

Jelikož většina klientských aplikačních rámců je založena na jazyce JavaScript, stává se Express velmi populární volbou pro vývoj serverové časti aplikace. Všechny součásti webové aplikace jsou tak napsány ve stejném jazyce, což vývojářům usnadňuje práci. Díky tomu je také součástí tzv. full stack aplikačních rámců jako jsou MEAN a MERN.

Express používá například Uber, IBM, Yandex a další.

Laravel

Laravel[12] je populární otevřený webový aplikační rámec založený na jazyku dobře známém ve webovém prostředí – PHP. Laravel využívá existujících komponent z jiných aplikačních rámců, čímž usnadňuje tvorbu webových aplikací.

[10] https://www.djangoproject.com/
[11] https://expressjs.com/
[12] https://laravel.com/

V základu podporuje směrování, autentifikaci, odesílání emailů, objektově relační mapování, tvorbu databázových schémat v jazyce PHP a další užitečné funkce. Laravel využívá nástroj Composer, který se stará o instalaci a aktualizaci všech závislostí a knihoven.

Ruby on Rails

Ruby on Rails (nebo také zkráceně Rails)[13] je otevřený webový aplikační rámec napsaný v jazyce Ruby. Má podobnou filozofii jako výše zmíněný Django, poskytuje tedy důležité funkce pro tvorbu webových aplikací již v základu. Řídí se návrhovým vzorem MVC a přístupem k programování DRY (Don't repeat yourself. – Neopakuj se.).

Rails je možno najít na pozadí stránek jako jsou GitHub, Airbnb, Twitch, SoundCloud, Hulu a další.

Spring Framework

Spring Framework[14] je serverový aplikační rámec postavený nad populárním programovacím jazykem Java. Implementuje architektonický vzor MVC. Pro některé vývojáře může výhodu představovat právě použitý jazyk Java díky svému silnému typování. Pro ty, kteří se v jazyku Java tolik neorientují, ale může být Spring Framework ze začátku obtížný na pochopení a naučení.

Spring Framework je možno najít na pozadí stránek Wix a TicketMaster.

3.6 Databáze

Tím, že uživatel používá webovou aplikaci, vznikají data. Velká část takto vytvořených dat musí být perzistentní a dostupná nezávisle na tom, jaké zařízení uživatel k přístupu k aplikaci používá. Obecně důležitá data tedy nelze ukládat do lokální paměti zařízení, ale musí být uložena centralizovaně na jednom či více serverech, kde budou skrz aplikaci, případně API, dostupná všem uživatelům. K tomuto účelu se využívá databáze, což je organizovaný soubor strukturovaných informací – dat. Vkládání, změny, odstranění a zobrazení dat pak provádí systém řízení báze dat (SŘBD) na základě dotazů serverové logiky případně samotného uživatele. Dohromady se databázi a SŘBD říká databázový systém.

V současné době je možné setkat se buď s relačními databázemi nebo nerelačními – tzv. NoSQL databázemi, které se od sebe výrazně liší způsobem ukládání dat.

Relační databáze

Základním prvkem relační databáze je relace, někdy také označována jako tabulka. Sloupce této pomyslné tabulky jsou označeny jako atributy, řádkům se říká záznamy. Atributy relace (tabulky) určují, jakého datového typu může být záznam v konkrétním sloupci a jakých hodnot může nabývat.

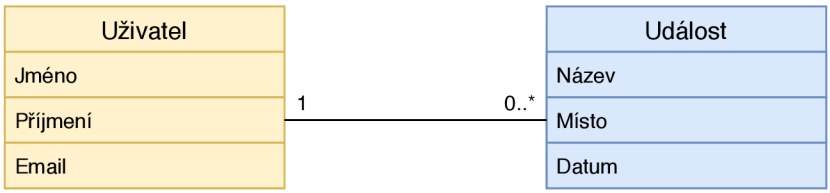
Existují také speciální atributy primární klíč a cizí klíč. Primární klíč představuje povinné unikátní označení záznamu v rámci jedné relace, podle kterého je možné záznam jednoznačně identifikovat. Primární klíč může být také složen z několika sloupců, pak se nazývá složený primární klíč. Cizí klíč je atribut jedné relace, který obsahuje primární klíče

[13] https://rubyonrails.org/
[14] https://spring.io/

druhé relace. Atributy primární a cizí klíč pak tvoří vazbu mezi danými relacemi (viz. obr. 3.6). [20]

Díky tomu, že jednotlivé relace mezi sebou tvoří vazby, nedochází v relačních databázích k tvorbě duplicitních dat. Relační databáze zaručují atomicitu, konzistenci, izolovanost a trvalost transakcí. Pro práci s daty v relačních databázích se používá strukturovaný dotazovací jazyk SQL.

Nejznámějšími relačními databázemi jsou Oracle Database, MySQL, PostgreeSQL, MariaDB, Microsoft SQL Server, IBM DB2.

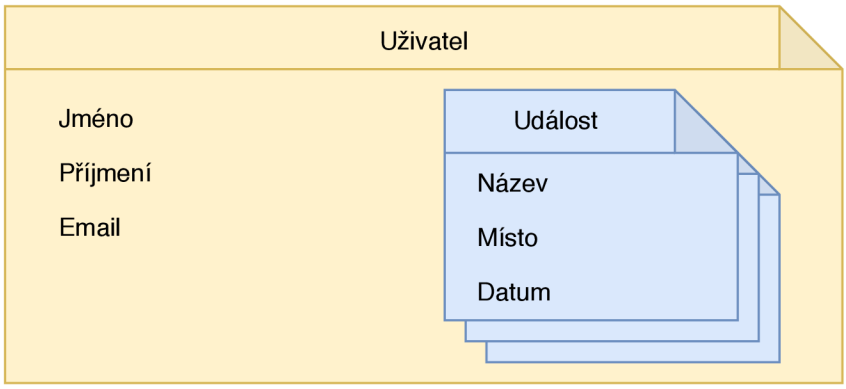


Obrázek 3.6: Ukázka dat v relační databázi

NoSQL databáze

NoSQL databáze představují moderní přístup ke způsobu práce s daty – nevyužívají relace. Přináší odpověď na stále zvyšující se nároky na správu a ukládání dat. Umožňují snazší decentralizaci úložiště dat, horizontální škálovatelnost, úmyslnou redundanci za účelem odolnosti proti výpadku a vyšší rychlosti, časté změny schématu nebo lepší podporu problematických datových typů – grafy, objekty atd. Existuje několik základních typů těchto databází. [1]

- Dokumentová databáze ukládá data do dokumentů v některém ze standardních formátů, například. XML, YAML, JSON nebo BSON (viz. obr. 3.7). Pravděpodobně nejznámější NoSQL databází tohoto typu je MongoDB
- Databáze typu klíč-hodnota nemají předem dané schéma a informace ukládají jako páry klíč-hodnota, kde hodnota může být prakticky cokoliv, například serializovaná data ve formátu JSON. Patří mezi ně například Oracle NoSQL nebo Dynamo.
- Grafové databáze uchovávají data v uzlech a hranách. Uzly většinou ukládají samotné informace, zatímco hrany vyjadřují vztahy mezi uzly. Mezi zástupce tohoto typu patří například Cassandra od společnosti Facebook nebo BigTable od Google.
- Sloupcové databáze jsou podobné jako klasické relační databáze – jsou založeny na tabulkách o řádcích a sloupcích. Na rozdíl od nich ale může mít každý řádek jedné tabulky jiné sloupce, případně i tzv. supersloupce, což jsou kolekce sloupců. Mezi tyto databáze řadíme například Neo4j nebo AllegroGraph.



Obrázek 3.7: Ukázka dat v dokumentové NoSQL databázi

3.7 Zabezpečení

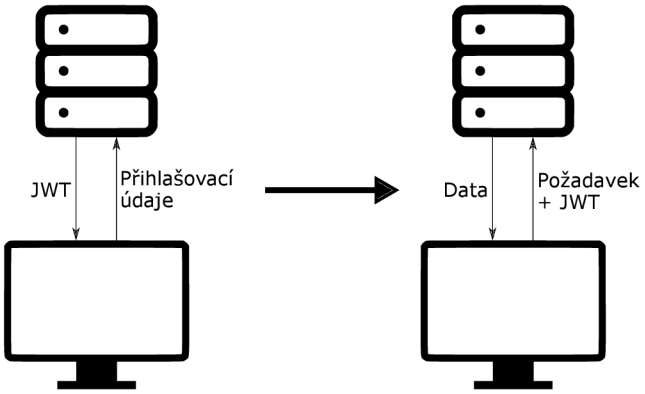
Se stále zvyšujícím se objemem zařízení, která mají možnost připojení k internetu, a růstem telekomunikačních sítí roste také počet uživatelů webu. V roce 2019 měl web celosvětově více než 4 miliardy uživatelů, což je více než polovina populace naší planety. Více uživatelů sdílí více dat, a právě ta jsou cílem útočníků. Takzvaní crackeři se snaží najít limity a chyby v systému za účelem jeho poškození a/nebo získání citlivých dat uživatelů. Mezi taková data mohou patřit údaje kreditních karet, hesla, emailové adresy, telefonní čísla nebo dokonce i tajné dokumenty organizací a států. Na webu se člověk může denně setkat s phishingem nebo reklamami lákajícími např. na výhru v loterii. Neméně časté jsou také podvodné webové stránky, které se od těch původních liší například jen změnou písmena v URL adrese, a samozřejmě také tím, že se chtějí zmocnit citlivých dat nic netušícího uživatele.

Jako obrana proti těmto útočníkům existuje řada technologií a principů, které je vhodné do webové aplikace implementovat, aby byla možnost úniku a zneužití uživatelských dat minimalizována. Níže se nachází jejich výčet včetně popisu a principu fungování.

JSON Web Token

JSON Web Token (zkráceně JWT) je otevřený internetový standard zveřejněný 28. prosince 2010 pro bezpečnou výměnu dat ve formátu JSON. Skládá se z hlavičky, přenášených dat a podpisu. V hlavičce se nachází údaje o typu tokenu a algoritmu, který byl použit pro podepsání. Druhá část obsahuje informace o vydavateli, dobu expirace a samotná data. Následuje digitální podpis, díky kterému je možné zaručit pravost tokenu. Je proto často využívaný webovými aplikacemi pro autorizaci požadavků uživatele (viz. obr. 3.8). Uživatel nejprve zašle své přihlašovací údaje, ty jsou na serveru ověřeny, a pokud jsou správné, server vygeneruje JWT a zašle jej klientovi. Klient (webový prohlížeč) si přijatý token uloží do lokálního úložiště. Při každém dalším požadavku pak místo uživatelského jména a hesla posílá tento token. Server tak při každém požadavku ověřuje pouze správnost a platnost přijatého tokenu.

Nevýhodou JWT je možnost jeho zcizení. Pokud získá útočník přístup k zařízení, kde je token uložený, může jej zkopírovat a pomocí něj pak autorizovat požadavky jako poškozený uživatel. Také pro tyto případy v sobě JWT nese informaci o vlastní expiraci, která se dá nastavit na libovolně dlouhou dobu. Po uplynutí expirační doby je token neplatný a útočník pomocí něj již nemůže autorizovat další požadavky. [2]



Obrázek 3.8: Autentifikace uživatele a následná autorizace požadavku pomocí JWT

reCAPTCHA

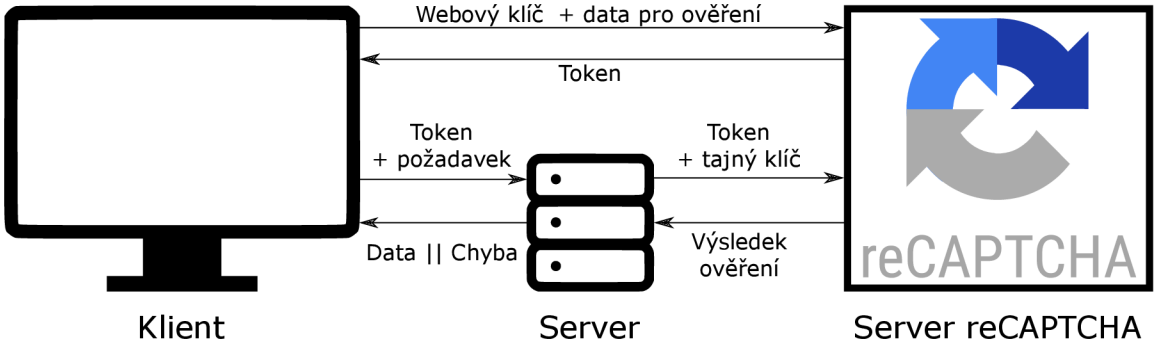
Systém reCAPTCHA vyvíjený společností Google umožňuje uživatelsky přívětivou ochranu webové aplikace proti spamu a jiným způsobům automatizovaného zneužití. Google reCAPTCHA pro svou funkčnost vyžaduje dvojici klíčů – webový klíč a tajný klíč, které je možné získat skrz administrátorský panel služby Google reCAPTCHA. Webový klíč je umístěný přímo v klientské části systému, je tedy veřejný. Tajný klíč je pak používaný v serverové části webové aplikace pro autorizaci komunikace se serverem služby reCAPTCHA (viz. obr. 3.9). Tento klíč, jak už název napovídá, musí být uchován na bezpečném místě, aby nemohlo dojít k jeho zcizení a zneužití.

V současné době je dostupné několik verzí, které se liší jak principem funkčnosti, tak způsobem interakce s uživatelem. Původní verze – reCAPTCHA verze 1 – již dnes není dostupná. Tato verze byla založena na principu opisování textu z obrázku.

Nahradila ji druhá verze, která se vyskytuje hned ve dvou formách. Asi nejznámější formou je ta se zaškrtávacím polem a popiskem „Nejsem robot“. Pokud chce uživatel ve webové aplikaci vykonat akci, která je tímto způsobem chráněna, musí kliknout na ono zaškrtávací políčko a tím potvrdit, že není robot. Následně je buď zobrazena zelená fajfka a uživatel může provést kýženou akci, nebo je uživatel vyzván, aby vyřešil pro člověka zpravidla snadný úkol. V tomto případě je uživateli prezentováno několik obrázků, z nichž musí vybrat ty, na kterých se nachází určitá věc. Pokud uživatel v této výzvě obstojí, může provést kýženou akci. I když není oficiálně přesně známo, jaký algoritmus společnost Google používá pro rozhodování, zda zobrazit zelenou fajfku nebo uživatele podrobit výzvě, předpokládá se, že se tak děje na základě předchozí aktivity uživatele na webových stránkách patřící společnosti Google nebo na stránkách, kde působí analytická služba Google Analytics. Spekuluje se také o tom, že má na rozhodování vliv i pohyb myši před a při kliknutí na zaškrtávací pole – robot dělá oproti člověku přímočaré pohyby myší a často kliká přesně do středu zaškrtávacího pole. Druhá forma této verze je obdobná s tím rozdílem, že původně předem dané zaškrtávací pole „Nejsem robot“ je možno nahradit libovolným prvkem stránky – třeba přímo tlačítkem „Odeslat formulář“.

Nejnovější verze systému reCAPTCHA, verze 3, se liší od předchozích verzí tím, že žádným způsobem neinteraguje s uživatelem. Místo toho běží na pozadí různých částí webové aplikace a chování uživatelů hodnotí na stupnici od 0,0 do 1,0. Pokud chce uživatel vykonat nějakou akci, je toto ohodnocení odesláno serverové části. Interní logika webové aplikace pak může na základě dosaženého skóre vyhodnotit, zda je příchozí požadavek podezřelý

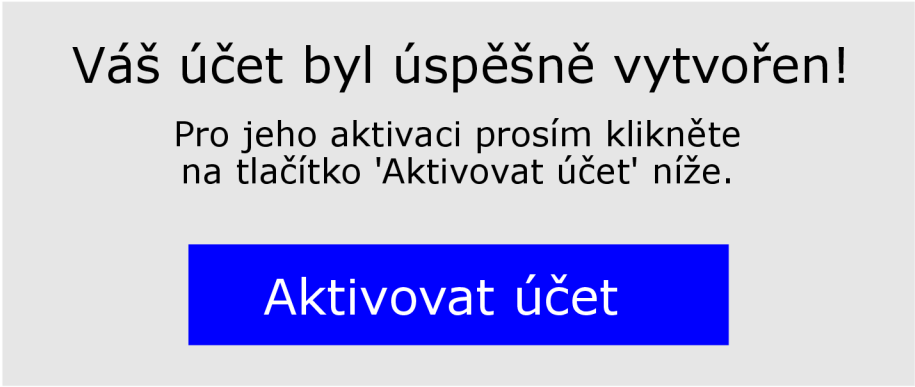
nebo ne, a případně provést náležitá opatření, např. při přijetí podezřelého požadavku na přihlášení bude požadováno potvrzení této akce pomocí odkazu v emailu. [11]



Obrázek 3.9: Princip činnosti Google reCAPTCHA

Ověření pomocí emailu

Další z populárních technik k omezení spamu a vydávání se pod falešnou identitou je ověření prostřednictvím emailu. Princip je prostý. Pokud chce uživatel vykonat určitou akci, je na jeho emailovou adresu odeslán hypertextový odkaz (viz. obr. 3.10). Tento odkaz zpravidla zahrnuje token nebo aktivační kód ve formátu application/x-www-form-urlencoded. Jakmile uživatel přistoupí na danou URL adresu, je uživatelova akce považována za ověřenou. Předpokladem pro efektivní fungování je dostatečné zabezpečení emailové schránky ze strany uživatele i poskytovatele. Tento princip se často používá pro aktivaci účtu po registraci nebo jako součást procesu obnovy uživatelského hesla.



Obrázek 3.10: Takto může vypadat obsah emailové zprávy sloužící k ověření.

Hesla

S rostoucím počtem webových služeb, na kterých si uživatel založí účet, se zvyšuje také pravděpodobnost úniku nebo prolomení uživatelova hesla. Jelikož používání stejného hesla ke všem službám je běžná (a špatná) praxe, může tak útočník prolomením jediného hesla ovládnout hned několik různých služeb daného uživatele. Šance na prolomení se ještě umocňuje, pokud uživatel používá příliš jednoduché nebo obecně známé heslo. S pomocí celých

databází hesel a metod typu brute force je prolomení takového hesla otázkou i jen několika málo vteřin. Na toto je třeba při vývoji webové služby myslet a vyžadovat po uživateli, aby si zvolil silné heslo – typicky kombinace čísel, malých i velkých písmen a speciálních symbolů o minimální délce 8 znaků. Je také vhodné implementovat adekvátně bezpečnou metodu pro obnovení hesla, která může zahrnovat již zmíněné ověření pomocí emailu.

Na provozovateli služby je pak zodpovědnost, aby uživatelovo heslo bezpečně uložil. V nedaleké minulosti se objevily případy, kdy neoprávněná osoba získala přístup do databáze, ve které se nacházely uživatelské údaje včetně hesel v čistě textové podobě. S případy vyloženě nebezpečného ukládání hesel se potýkala i společnost Facebook, kdy takto ukládala hesla několika stovek milionů svých uživatelů. Každé heslo, které má být uloženo do databáze, musí nejprve projít jednosměrnou hašovací funkcí, která v podstatě znemožní jeho zpětné zjištění – haše hesel jsou potencionálnímu útočníkovi k ničemu. Takovou hašovací funkcí je například bcrypt.

Kapitola 4

Požadavky na službu a její návrh

Analýza existujících řešení, a především jejich silných a slabých stránek je klíčová pro návrh lepšího řešení. V této kapitole je popsáno několik existujících služeb, které se zabývají sdílením informací o událostech a společenských akcích. Dále je pak podrobně rozepsán návrh architektury a jednotlivých částí vznikající služby.

4.1 Existující řešení

Na webu existuje řada služeb, které umožňují sdílet informace o událostech. Většina z nich je ovšem zaměřená na jejich veřejné sdílení, vyhledávání a propagaci. Několik služeb umožňuje i sdílení soukromých událostí nebo se k tomuto užití alespoň blíží. Často ale takové služby vyžadují registraci a/nebo jsou zpoplatněny.

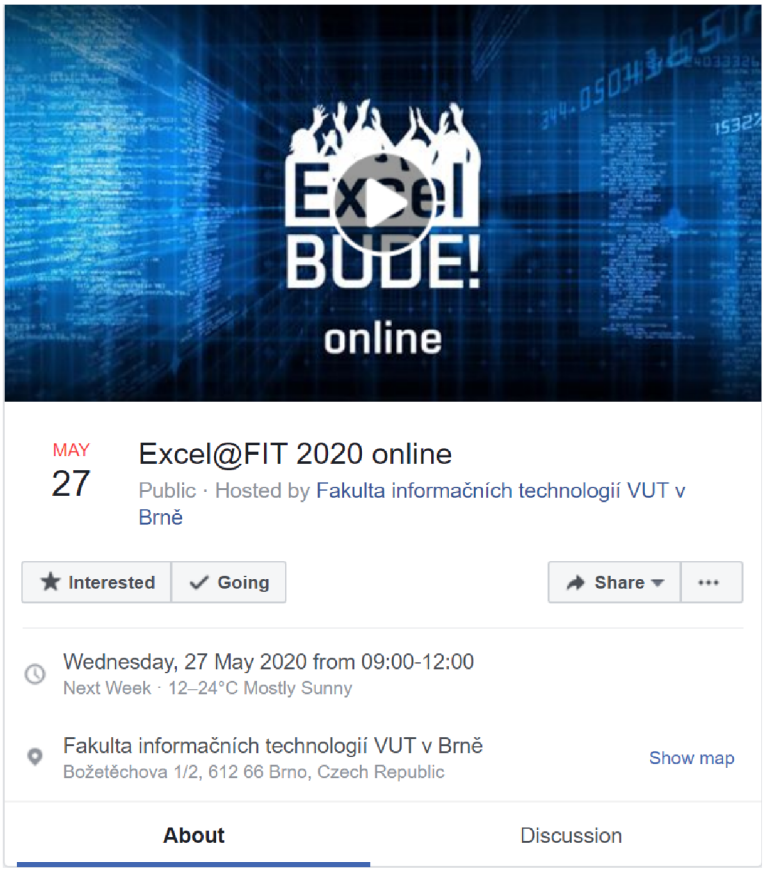
Facebook Události

Sociální síť společnosti Facebook byla založena v roce 2004 Markem Zuckerbergem, který je v současné době jejím výkonným ředitelem. Svým uživatelům umožňuje vzájemnou komunikaci, sdílení informací, odkazů a dalšího multimediálního obsahu, hraní různých her, účast ve skupinách, tvorbu osobních i firemních stránek a v neposlední řadě tvorbu událostí. Jelikož má tato síť téměř 2,5 miliardy aktivních uživatelů, má společnost Facebook k dispozici obrovské množství osobních údajů, které využívá k přesnému zacílení reklamy a různým návrhům pro své uživatele. Společnost je nechvalně proslulá kauzou „Facebook–Cambridge Analytica“, kdy byla data milionů uživatelů využita pro zacílení politické kampaně. [6] Společnost Facebook je také známá instantním komunikátorem Messenger, který je se sociální sítí úzce spjatý. Vlastní taktéž populární sociální síť Instagram.

Jedna součást sítě, Facebook Události, slouží ke sdílení informací o nadcházejících událostech a společenských setkáních. Pro vytvoření události je nutno mít založený účet na sociální síti Facebook a zadat následující informace: název události, místo konání, datum a čas konání, případně také podrobnější popis a úvodní fotografii. K událostem je možné přidávat příspěvky, které mohou obsahovat aktuální informace o události nebo organizační pokyny. Události mohou být buď veřejné nebo soukromé, přičemž k těm soukromým mají přístup pouze pozvaní uživatelé sociální sítě. Ty veřejné je možné najít prostřednictvím interního vyhledávače sociální sítě, případně je možno k nim přistoupit přes odkaz, aniž by byl uživatel na událost pozván. Neregistrovanému uživateli se v tomto případě zobrazí pouze základní informace, k dodatečným příspěvkům už přístup nemá.

Vytvořená stránka události přehledně zobrazuje zadané informace, místo konání ukazuje na mapě (viz. obr. 4.1). Přihlášenému uživateli je umožněno událost sdílet, odebírat aktualizace skrz notifikace v rámci sítě nebo informovat pořadatele o své účasti. V závislosti na nastavení také mohou uživatelé k události přidávat příspěvky a sdílet své fotografie nebo videa. Neregistrovaný uživatel má pak možnosti značně omezeny, a to pouze na zobrazení základních informací. Bez registrace na sociální síti nemůže zobrazovat příspěvky ani jakýmkoliv způsobem odebírat aktualizace. Funkce jako potvrzení účasti a přidávání příspěvků jsou pak logicky také vázány k uživatelskému účtu.

Jednoznačnou výhodou služby Facebook Události je její působení v rámci největší sociální sítě na světě. Služba je zdarma, nabízí jednoduché sdílení a pozvánky v rámci sítě. Nespornou výhodou může být také přidávání fotografií a videí přímo od účastníků události. Nevýhodou je pak nutnost registrace pro vytvoření události a s tím související poskytování osobních údajů společnosti Facebook. Jelikož událost může být buď soukromá nebo veřejná, musí si uživatel vybrat, zda chce, aby byla událost viditelná pouze pro jeho pozvané přátelé na sociální síti (soukromá) nebo aby byla viditelná úplně pro všechny a dala se vyhledat v interním vyhledávači sociální sítě (veřejná). Soukromou událost, kterou může pořadatel sdílet mimo sociální síť, tak vytvořit nelze.



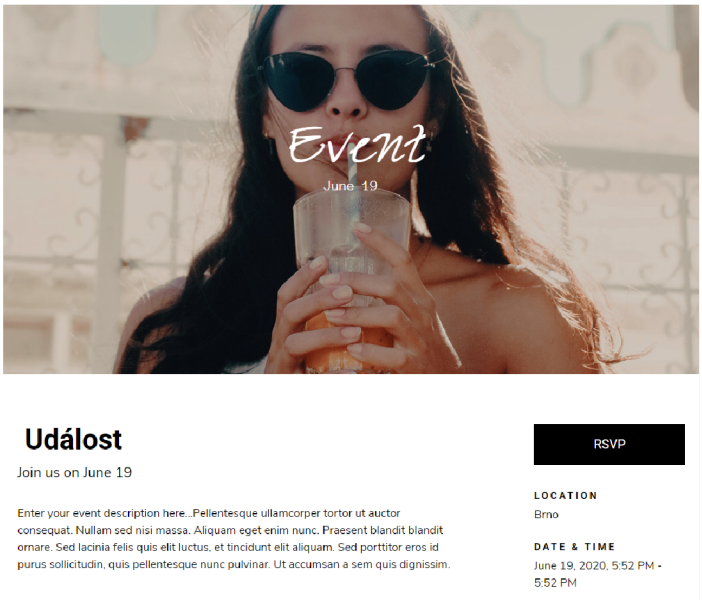
Obrázek 4.1: Detail události na sociální síti Facebook

EventCreate

EventCreate je služba zaměřená na vytváření webových stránek, které slouží jako pozvánky na události (viz. obr. 4.2). Pro vytvoření stránky je nutno se zaregistrovat. Kromě zadání základních informací je možné výslednou pozvánku různě upravovat – přemísťovat prvky, měnit font a velikost písma, nastavit různé akce na tlačítka apod. K tomuto účelu využívá WYSIWYG[1] editor.

Základní služba je zdarma, nabízí ale také prémiové funkce jako například zadání vlastní domény, rozeslání pozvánek emailem, odstranění loga EventCreate z pozvánky a jiné. Uživatelé mohou potvrdit svou účast na události, není přitom vyžadována registrace. Služba dále nabízí prodej a správu lístků, rozšířenou analýzu dosahu pozvánky, prodeje lístků, návštěvnosti apod. Vytvořená událost může být také veřejně propagována přímo mezi komunitou služby EventCreate.

Tato služba tedy nabízí obrovské možnosti přizpůsobení pozvánky, správu účastníků, prodej lístků, propagaci události a spoustu dalšího. Svým zaměřením je vhodná spíše pro veřejné a větší události. I když má pořadatel události za příplatek možnost poslat účastníkům email, služba neumožňuje k události přidávat příspěvky s aktualitami nebo automaticky upozorňovat účastníky na změny informací.



Obrázek 4.2: Stránka události vytvořená službou EventCreate

Evvnt

Evvnt je služba pro online propagaci různých společenských akcí a událostí. Služba deklaruje sdílení události až na 3500 stránkách s potenciálním dosahem až 40 milionů lidí. [7] Větší míra propagace je samozřejmě zpoplatněna. Pro vytvoření události je třeba zadat název, místo konání, datum a čas začátku, popis, kontakt na pořadatele a odkaz na webové stránky události. Služba sama o sobě neposkytuje informační stránku k vytvořené události.

[1] WYSIWYG – What You See Is What You Get (co vidíš, to dostaneš) označuje způsob editace dokumentu, kdy je jeho vzhled při úpravách v podstatě stejný jako vzhled výsledný.

Tato služba je tak primárně vhodná pro pořadatele velkých společenských akcí, kterým usnadňuje propagaci a zviditelnění dané události. Pro menší soukromé akce, kdy je třeba účastníky informovat o detailech konané události a případných změnách, se tedy nehodí.

4.2 Požadavky

Na existujících řešeních je možno najít silné i slabé stránky. Ty se ovšem projevují především v kontextu toho, jaké jsou na nově vyvíjenou službu kladeny požadavky. Cílem této bakalářské práce je vytvořit webovou službu, která bude umožňovat rychle a jednoduše vytvářet informační stránky k událostem. Vytvořené události bude moct pořadatel upravovat a přidávat k nim příspěvky s aktualitami. Konkurenční služby zpravidla vyžadují registraci již pro samotné vytvoření události, tento problém nová služba řeší – vytvoření i následná správa události bude umožněna bez registrace. Sdílení události bude probíhat výhradně formou stálé URL, což znamená, že se na stránku události dostane každý bez ohledu na to, zdali je zaregistrován nebo ne. Zároveň nebude možné události vyhledávat, čímž bude zachován jejich soukromý charakter. Velmi populární služba Facebook Události je v tomto ohledu značně omezující, viz poslední odstavec sekce 4.1. Další silnou stránkou nové služby bude možnost odebírat aktualizace v podobě emailových zpráv – opět bez nutnosti registrace. Konkurenční služby tuto možnost buď neumožňují vůbec nebo pouze skrze vlastní systém (notifikace na sociální síti Facebook je možné nechat zasílat na email). Od konkurence si služba vypůjčuje možnost nahrát úvodní fotografii, která stránce s událostí dodá originální vzhled. Zobrazení místa konání na mapě je funkce, kterou taktéž nabízí služba od společnosti Facebook. K událostem na sociální síti Facebook mohou uživatelé a samozřejmě pořadatel přidávat příspěvky – text, fotky, videa a ankety. Jelikož je nová služba zaměřena primárně na sdílení informací od pořadatele směrem k potenciálním účastníkům, může příspěvky přidávat pouze pořadatel, přičemž příspěvky jsou výhradně textové. Služba ale navíc umožňuje přidávat k události soubory jako jsou dokumenty, archivy a fotografie – může se jednat o program události ve formátu PDF, tabulka se zasedacím pořádkem apod. Služba tak bude zastávat komplexní nástroj pro sdílení informací, aktualit a souborů směrem k účastníkům události. Požadavky jsou modelovány pomocí diagramu případů užití (viz. obr. 4.3), a níže jsou jednotlivě rozepsány a objasněny.

Vytvoření a zobrazení události

Aplikace v prvé řadě musí podporovat vytváření událostí. To musí být realizováno jednoduchou formou se základními údaji jako jsou název události, datum a čas začátku, datum a čas konce, místo konání a popis události. Vytvářet událost musí být umožněno i neregistrovanému uživateli. Vytvořená stránka události musí být přístupná přes URL adresu, kterou tak lze snadno sdílet napříč různými komunikačními platformami.

Editace události, přidání příspěvku, úvodního obrázku a souborů

Registrovanému i neregistrovanému uživateli musí být umožněno vytvořenou událost upravit. Buď skrz vytvořený uživatelský účet nebo v případě neregistrovaného uživatele přes předem zvolené heslo k editaci. K události může její autor přidat úvodní obrázek, příspěvky (aktualizace) a soubory. Informace z vytvořené události lze taktéž rychle přenést do formuláře pro vytvoření nové události.

Odběr aktualizací

Registrovaný i neregistrovaný uživatel se může přihlásit k odběru aktualizací pomocí emailu. Pokud dojde u takto odebírané události ke změně informací, bude o tom uživatel informován prostřednictvím emailové zprávy. Taktéž bude uživatel upozorněn na přidání nového příspěvku u odebírané události. Notifikace opět v podobě emailu bude obsahovat kromě odkazu na událost také nadpis a obsah nově přidaného příspěvku. Odběr může neregistrovaný uživatel kdykoliv odhlásit kliknutím na odkaz v emailu, přihlášený uživatel pak i kliknutím na tlačítko „Odhlásit odběr“ přímo u události.

Registrace a správa účtu

Aplikace musí umožňovat založení uživatelského účtu. Po úspěšné registraci získá uživatel přístup i k těm událostem, které byly dříve založeny pod stejným emailem. Proto musí být přístup k účtu podmíněn jeho aktivací skrz kliknutí na aktivační odkaz v zaslaném emailu. Uživatel musí mít možnost upravit osobní údaje a v případě potřeby také obnovit a změnit své heslo.

Potvrzení účasti, zobrazení účastníků

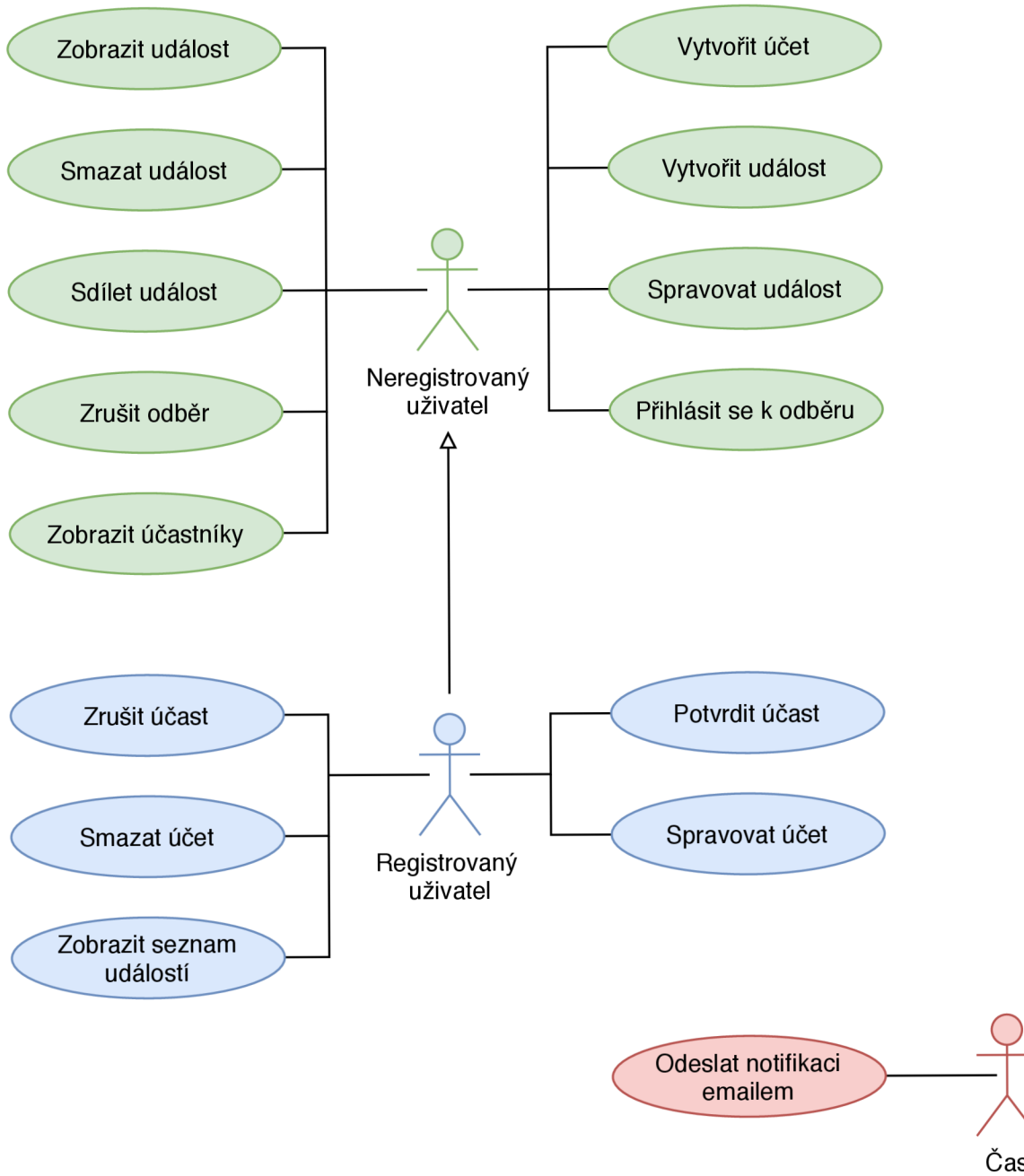
Registrovaný uživatel se může přihlásit na událost, čímž dá pořadateli události vědět o své účasti. Zároveň se takto označená událost bude zobrazovat v jeho profilu. Pořadatel má možnost zobrazit si jména uživatelů, kteří potvrdili svou účast.

Správa událostí

Registrovanému uživateli jsou po přihlášení zobrazeny jím vytvořené události, události, kterých se účastní, a události, u nichž odebírá aktualizace. U každé události je kromě názvu uvedeno také datum a místo konání. Události jsou seřazeny podle data konání, přičemž již proběhlé je možno skrýt. U každé události se také zobrazuje datum a čas poslední aktualizace.

Odstranění událostí a účtů

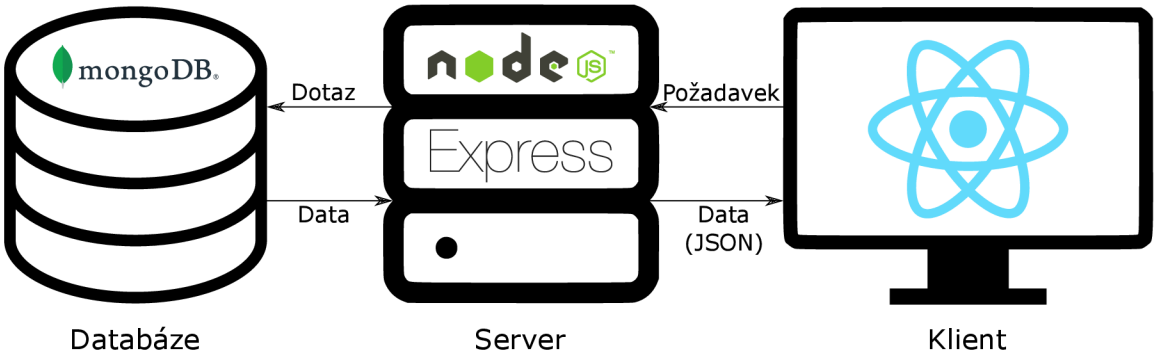
Každou událost je možné smazat. U neregistrovaného uživatele je k tomu vyžadováno heslo pro editaci, u přihlášeného uživatele je možné událost smazat přímo. Smazáním události dochází ke zrušení všech účastí a odběrů, které jsou k dané události přidruženy. Taktéž uživatelský účet je možno smazat. Tím jsou odstraněny také všechny události vytvořené tímto uživatelem a zrušeny jeho účasti i odběry na událostech vytvořených ostatními uživateli.



Obrázek 4.3: Diagram případů užití

4.3 Návrh architektury

Pro implementaci aplikace byl zvolený technologický balík složený z JavaScriptových aplikačních rámců a knihoven označovaný jako MERN (viz. obr. 4.4). Každé písmeno této zkratky označuje jednu komponentu balíku. Pro ukládání dat tedy byla zvolena NoSQL dokumentová databáze MongoDB. Jako serverový aplikační rámec byl zvolen Express, který poběží v běhovém prostředí Node.js. Klientská část aplikace pak bude využívat aplikační rámec React společně s jazykem HTML, CSS a knihovnou Bootstrap. Rozhraní pro komunikaci serverové a klientské části bude implementováno jako architektura REST.



Obrázek 4.4: Diagram architektury

React

Klientský aplikační rámec React byl zvolen především pro svou popularitu, které se v posledních několika letech těší. Mezi jeho nesporné výhody patří automatické překreslování jednotlivých elementů díky využití virtuálního DOMu. Při vývoji React aplikace je možné volit nebo dokonce kombinovat využití funkčních a třídních komponent. Výhodou je také velké množství knihoven a komponent, které byly vytvořeny přímo pro React a jsou dostupné jako balíčky skrze systém npm. [8]

Express

Serverový aplikační rámec Express byl zvolen především proto, že je postaven na programovacím jazyce JavaScript, tedy stejném jazyce, jaký je použit pro vývoj klientské části aplikace. I když je v základu minimalistický – především podpora směrování, middleware a práce s požadavkem/odpovědí, díky běhovému prostředí Node.js a jeho balíčkovacímu systému npm je na výběr nepřeberné množství knihoven. S použitím vhodných knihoven lze tak velmi rychle vytvořit komplexní serverovou část. [17]

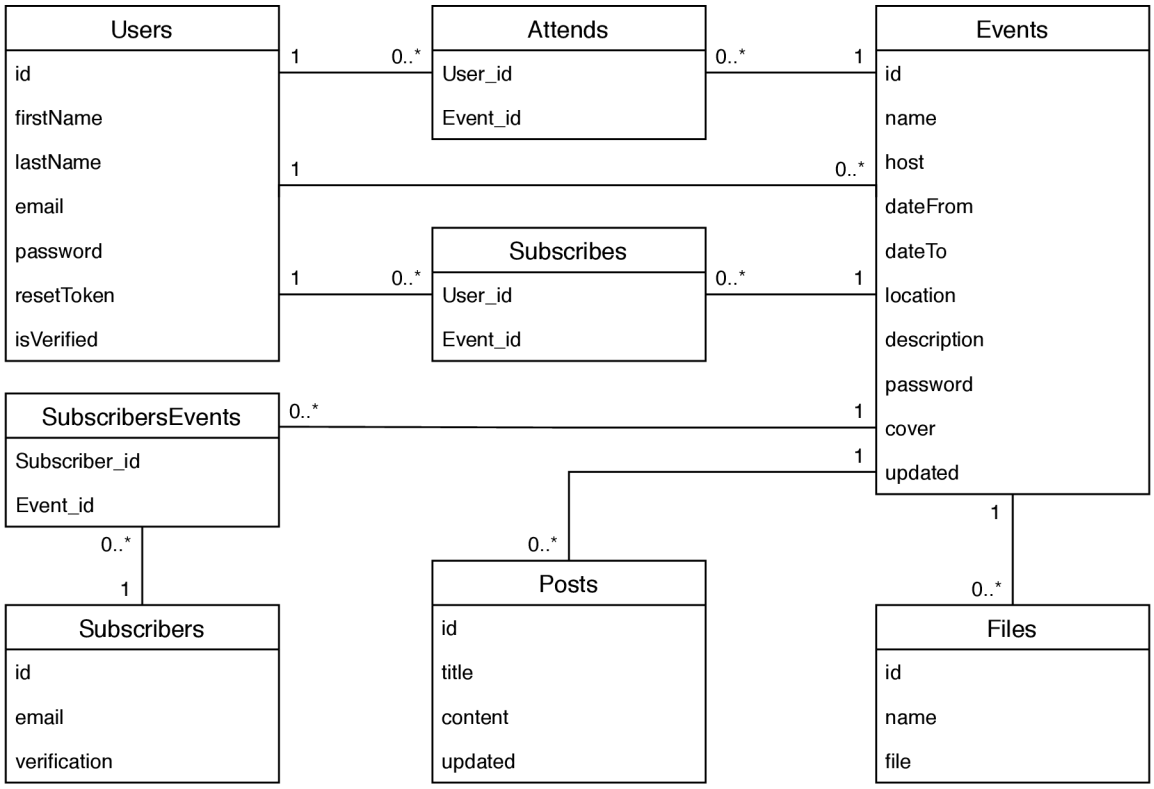
MongoDB

MongoDB je multiplatformní dokumentově orientovaná databáze. Řadí se mezi tzv. NoSQL databáze, což znamená, že její model není založený na relačních tabulkách. Místo toho jsou data ukládána v BSON dokumentech, které obsahují páry klíč – hodnota. Hodnotou může být jakýkoliv datový typ definovaný formátem BSON včetně jiných dokumentů, polí a polí dokumentů. Databáze MongoDB se vyznačuje dobrou škálovatelností a rychlostí. V případě správně navržené datové struktury je pak výhodou i přehlednost databáze, ze

které lze velmi jednoduše vyčíst informace bez nutnosti provádět spojování tabulek jako v případě SQL databází. Jelikož je formát BSON binárně kódovaná serializace dokumentů velice podobných dokumentům JSON, je práce s objekty databáze MongoDB v prostřední JavaScriptu velice přímočará. [19] Pro hostování databáze bude sloužit cloudová služba MongoDB Atlas.

4.4 Návrh struktury dat

Pro větší přehlednost byl návrh struktury dat modelován pomocí ER diagramu (viz. obr. 4.5). Pro potřeby dokumentové NoSQL databáze byl pak tento model převeden do podoby dokumentu a v něm vnořených subdokumentů. Jakou strukturu mají tyto dokumenty, respektive subdokumenty, lze nalézt v následujících sekcích.



Obrázek 4.5: ER diagram návrhu struktury dat

4.5 Uživatel

Ze struktury dat je patrné, že uživatel je základní částí celé aplikace. I v případě, že událost vytváří neregistrovaný uživatel, je do databáze nejprve uložena jeho emailová adresa a k ní je následně přiřazena vytvořená událost. Pokud je pod touto emailovou adresou vytvořena další událost, v databázi už je pouze přiřazena k dané existující emailové adrese. Událost je v databázi reprezentována subdokumentem „Event“. Každý uživatel má pole těchto subdokumentů (v databázi označeno jako „events“) (viz. obr. 4.6).

```
{
        _id: ObjectId
        firstName: String,
        lastName: String,
        email: String,
        password: String,
        resetToken: String,
        isVerified: Boolean,
        attends: [ ObjectId ],
        subscribes: [ ObjectId ],
        events: [ Event ]
}
```

Obrázek 4.6: Databázová reprezentace uživatele

Registrace

Registrace uživatelského účtu je rozdělena na tři fáze. Uživatel, který se rozhodne založit uživatelský účet musí nejprve vyplnit svou emailovou adresu, jméno a příjmení. Před odesláním registračního formuláře musí uživatel také úspěšně projít ověřením systému Google reCAPTCHA.

Po přijetí registračního formuláře serverem je ověřeno, zda v databází již existuje účet se stejnou emailovou adresu. Pokud je takový dokument nalezen, ale nereprezentuje uživatelský účet, je použit existující dokument. V opačném případě je vytvořen nový dokument obsahující jméno, příjmení a emailovou adresu uživatele. Databázový systém novému uživateli automaticky přidělí unikátní identifikátor (v databázi „_id“). Zároveň je vygenerován JWT a uložen do dokumentu jako „resetToken“. Poté je uživateli na jím zadaný email zaslána zpráva obsahující aktivační odkaz. Ten obsahuje zmíněný „resetToken“ ve formátu application/x-www-form-urlencoded. V této fázi je tedy ověřeno, zda má uživatel opravdu přístup k zadané emailové adrese.

Aktivační odkaz zaslaný na email uživatele po kliknutí přesměruje na stránku webové aplikace, kde si může zvolit své heslo a dokončit tak aktivaci účtu. Po úspěšném odeslání zvoleného hesla na server dojde k zašifrování a uložení hesla do databáze pod položkou „password“. Současně je vynulován „resetToken“ a položka „isVerified“ je nastavena na hodnotu „true“, což značí, že uživatelský účet byl ověřen a je aktivovaný.

Přihlášení, autorizace požadavků

Registrovaný uživatel, který má aktivovaný účet, se do webové aplikace může přihlásit zadáním emailové adresy, která byla použita při registraci, a jím zvoleného hesla. Pokud není uživatelský účet aktivovaný, není možné se k němu přihlásit.

Po ověření správnosti zadaných přihlašovacích údajů vygeneruje server JWT, který odešle klientovi (klientské aplikaci). Token, který obsahuje ID uživatele společně s jeho jménem a příjmením, si klient ukládá do svého lokálního úložiště. Token je podepsán hašem uživatelova hesla, což zabezpečuje okamžité zneplatnění tokenu, pokud si uživatel změní své heslo. Společně s každým požadavkem pak klient serveru odesílá tento token. Tímto způsobem

klient autorizuje požadavky, aniž by musel ukládat a pokaždé odesílat přihlašovací údaje uživatele.

Výpis událostí

Dokument reprezentující uživatele rovněž obsahuje pole „attends“ a „subscribes“. Tato pole slouží k ukládání identifikátorů událostí, kterých se uživatel účastní, resp. které uživatel odebírá. Při zobrazení uživatelského profilu jsou tato pole společně s polem událostí využívána pro rychlé vyhledání všech událostí, se kterými je uživatel nějakým způsobem spojený – pořádá je, účastní se jich nebo je odebírá.

Správa uživatelského účtu a obnovení hesla

Uživateli je umožněno změnit své jméno, příjmení a heslo. V případě, že uživatel zapomene své heslo, může jej v klientské aplikaci obnovit. K obnovení hesla je třeba zadat emailovou adresu použitou při registraci. Změna hesla probíhá na stejném principu jako prvotní aktivace účtu. Server tedy vygeneruje a do databáze uloží „resetToken“, který následně odešle na uživatelův email v podobě odkazu. Po kliknutí na odkaz si uživatel zvolí nové heslo, to je poté v zašifrované podobě uloženo do databáze. Díky tomu, že autorizační token je podepsán hašem uživatelova hesla, nemůže již uživatel použít existující token, a musí se tedy znovu přihlásit.

Uživatelský účet je možno odstranit, čímž dojde k odstranění všech uživatelem vytvořených událostí včetně případných nahraných souborů.

4.6 Událost

Jak už bylo zmíněno, každá událost je přirazena k emailové adrese, která je v systému unikátní. Díky tomuto přístupu k ukládání událostí je přechod z neregistrovaného uživatele na registrovaného velice snadný. Pokud v aplikaci již existují události vytvořené pod stejnou emailovou adresou, jsou automaticky zpřístupněny zaregistrovanému uživateli. Tímto vzniká určitý uživatelský komfort, kdy může uživatel vytvářet událost bez nutnosti registrovat se, a zároveň má kdykoliv možnost pomocí registrace sjednotit správu svých událostí pod jedním uživatelským účtem. Události jsou ukládány jako vnořené dokumenty (subdokumenty) (viz. obr. 4.7) do pole událostí v dokumentu reprezentujícím uživatele.

```
{
        _id: ObjectId,
        name: String,
        host: String,
        dateFrom: Date,
        dateTo: Date,
        location: String,
        description: String,
        password: String,
        subscribers: [ Subscriber ],
        attendees: [ ObjectId ],
        posts: [ Post ],
        cover: String,
        files: [ File ],
        updated: Date
}
```

Obrázek 4.7: Databázová reprezentace události

Vytvoření

Postup vytvoření události se liší podle toho, zda událost vytváří neregistrovaný nebo registrovaný (a přihlášený) uživatel. Pro oba postupy je společná povinnost zadat název události, jméno/název pořadatele, místo konání, datum a čas předpokládaného začátku a konce. Nepovinná položka je potom popis události. Neregistrovaný uživatel má navíc povinnost uvést svou emailovou adresu a heslo, které bude sloužit pro pozdější správu vytvořené události. Pokud událost vytváří neregistrovaný uživatel, je na straně serveru zkontrolováno, zda k zadané emailové adrese již neexistuje uživatelský účet, pokud ano, je na tuto skutečnost uživatel upozorněn. V případě vytváření přihlášeným uživatelem je správný uživatelský účet, ke kterému má být událost přiřazena, vyhledán podle identifikátoru v JWT. Každé vytvořené události je databázovým systémem automaticky přidělen unikátní identifikátor.

Po úspěšném vytvoření vrací server klientovi identifikátor nově vzniklé události a aplikace uživatele automaticky přesměruje na její detail.

Zobrazení

Zobrazení události je pravděpodobně nejvyužívanější část této webové aplikace. Na základě znalosti ID je možné zobrazit jakoukoliv událost. Tento princip je důležitý pro snadné sdílení pomocí URL adresy. Po zadání URL adresy ve tvaru „doména/events/id-události“ kontaktuje klientská aplikace serverovou část a vyžádá si od ní danou událost. Serverová část odesílá kýženou událost společně s údaji o tom, zda je přihlášený uživatel majitelem, účastníkem nebo odběratelem dané události. Klientská část pak na základě těchto údajů zobrazuje příslušná tlačítka a informace o události.

Uživateli je tedy zobrazen název události, jméno pořadatele, případný úvodní obrázek, datum a čas začátku a konce. Zobrazeno je taktéž místo konání, které je navíc vyznačeno na interaktivní mapě. Následuje případný detailní popis události, seznam příspěvků (aktualizací) a nahraných souborů.

Událost lze snadno sdílet pomocí dedikovaného tlačítka, které nabízí možnost rychlého zkopírování URL adresy do schránky a odkazy pro sdílení pomocí emailu a sociálních sítí. Taktéž je dostupné tlačítko pro rychlé přidání události do Google Kalendáře.

Úprava

Aby byly možnosti úpravy co nejméně rušivé, zobrazuje se tlačítko pro zobrazení úprav jen v případě, kdy je přihlášený pořadatel události nebo se jedná o událost, která není vázaná k uživatelskému účtu. Stisknutím tlačítka „Úpravy“ lze zobrazit několik dalších tlačítek, která slouží pro úpravu informací, změnu úvodní fotky, přidání/odebrání příspěvků a souborů a smazání celé události.

Informace o vytvořené události lze opravovat, a to tlačítkem „Upravit informace“. Upravit lze všechny informace kromě emailové adresy a hesla pro editaci. Neregistrovaný uživatel musí pro uložení úprav zadat heslo a projít ověřením reCAPTCHA.

Úvodní obrázek

Uživatel může k vytvořené události přidat úvodní obrázek, který dodá události originální vzhled. Obrázkem může být tematická fotografie, grafika, svatební oznámení atd. – fantazii se zde meze nekladou. Úvodní obrázek je nejprve nahrán na server, jeho umístění je poté uloženo do dokumentu události pod položku „cover“. Pokud již úvodní obrázek u události existuje a uživatel nahraje nový, ten původní je jím nahrazen. Úvodní obrázek lze taktéž odstranit. V případně neregistrovaného uživatele je nahrání i odstranění úvodního obrázku již klasicky podmíněno zadáním hesla a ověřením reCAPTCHA.

Příspěvky

Ke každé události má pořadatel možnost přidat libovolné množství příspěvků, z nichž každý má nadpis a obsah. Dokumenty reprezentující příspěvky (viz. obr. 4.8) jsou ukládány do pole „posts“ u události. Příspěvky slouží ke sdělení nových informací, organizačních pokynů, mohou být využity také pro poděkování účastníkům, ohlédnutím se za proběhlou událostí nebo např. pro sdílení odkazu na fotografie z události.

Příspěvky jsou řazeny sestupně podle data přidání, nahoře se tak nachází naposledy přidaný příspěvek. Přihlášený uživatel může příspěvky ke své události přidávat bez omezení, neregistrovaný uživatel musí pro přidání každého příspěvku zadat heslo pro úpravu události a projít ověřením reCAPTCHA.

Jelikož mají příspěvky primárně sloužit jako sdělení potenciálním účastníkům, není možné je po přidání upravovat. Pokud se některé informace z předchozích příspěvků změní, je nutno o této změně informovat přidáním nového příspěvku. Tím je zachován zpětně dohledatelný vývoj informací, přičemž nejaktuálnější informace jsou vždy v posledním přidaném příspěvku. V nejnutnějších případech je možné libovolný příspěvek smazat. Smazání příspěvku je u neregistrového pořadatel opět podmíněno zadáním hesla pro úpravu a úspěšným ověřením reCAPTCHA.

```
{
        _id: ObjectId,
        title: String,
        content: String,
        updated: Date
}
```

Obrázek 4.8: Databázová reprezentace příspěvku

Soubory

K vytvořené události je taktéž možno přidávat soubory. Nahrát je možné dokumenty, archivy nebo i fotografie. Uživatel může soubory výstižně pojmenovat, aby bylo na první pohled zřejmé, co se v nich nachází. Každý soubor je v databázi reprezentován jedním dokumentem (viz. obr. 4.9), kde položka „name“ značí zobrazovaný název souboru a „file“ značí adresářové umístění souboru v úložišti. Jednotlivé dokumenty se pak nacházejí v poli „files“ v dokumentu reprezentujícím událost. Přidané soubory je samozřejmě možné mazat.

```
{
        _id: ObjectId,
        name: String,
        file: String
}
```

Obrázek 4.9: Databázová reprezentace souboru

Odběr aktualizací

Klíčovou funkcionalitou této webové aplikace je schopnost notifikovat potenciální účastníky události o změnách, a to formou emailových zpráv. Při každé změně informací u události je uživatelům přihlášeným k odběru odeslána emailová zpráva, která obsahuje odkaz na událost, u které nastala změna, a také odkaz pro zrušení odběru. Podobná notifikace je odeslána také v případě přidání nového příspěvku u odebírané události. Tato zpráva kromě dvou výše zmíněných odkazů obsahuje také nadpis a obsah přidaného příspěvku.

Přihlášení k odběru notifikací se opět liší v závislosti na typu uživatele. Neregistrovaný uživatel je po kliknutí na tlačítko „Přihlásit odběr“ vyzván k zadání emailové adresy a ověření pomocí reCAPTCHA. Přihlášený uživatel je po kliknutí na stejné tlačítko ihned přihlášen k odběru, přičemž je použita emailová adresa uvedená při registraci. Tlačítko se dynamicky mění a u přihlášeného uživatele slouží zároveň ke zrušení odběru. Po přihlášení odběru je email uložen do databáze společně s vygenerovaným verifikačním kódem sloužícím k případnému odhlášení odběru. Každý odběratel je tedy v databázi reprezentován dokumentem (viz. obr. 4.10). Tyto dokumenty jsou uchovávány u události v poli „subscribers“. Pokud se k odběru přihlásí přihlášený uživatel, je navíc identifikátor dané události přidán do pole „subscribes“ v dokumentu reprezentujícím onoho uživatele.

Jak registrovaný, tak neregistrovaný uživatel může k odhlášení odběru využít odkaz, který je součástí každé emailové notifikace. Odkaz je tvořen URL adresou s řetězcem dotazu,

který obsahuje emailovou adresu a ověřovací kód. Registrovaný (a přihlášený) uživatel může odběr zrušit také tlačítkem „Odhlásit odběr“ přímo u detailu události.

```
{
        _id: ObjectId,
        email: String,
        verification: String
}
```

Obrázek 4.10: Databázová reprezentace odběratele

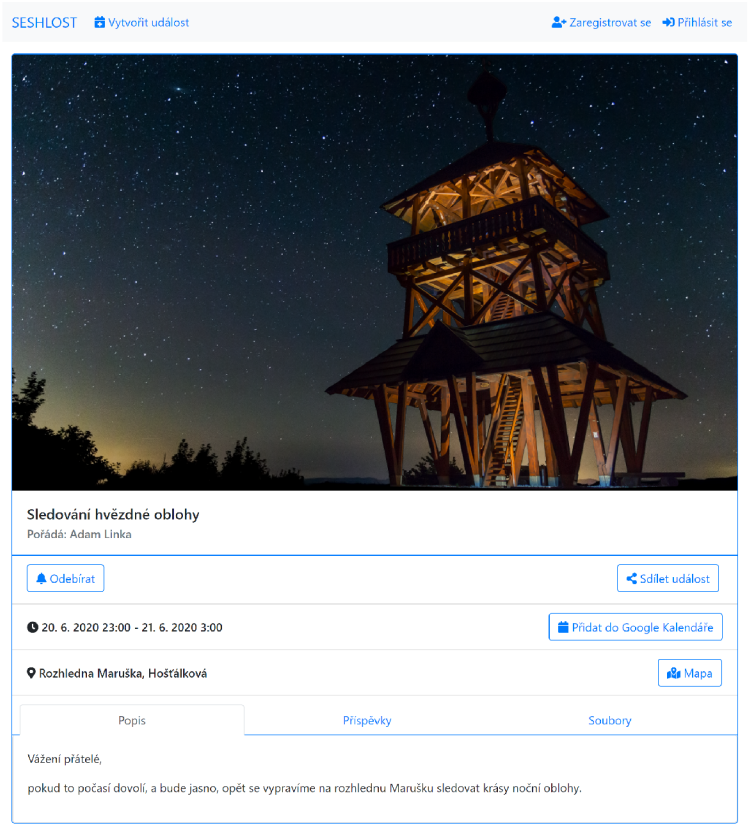
Potvrzení účasti

Přihlášený uživatel má možnost potvrdit účast na události. Tím je jeho identifikátor přidán do pole „attendees“ u dané události. Zároveň je identifikátor dané události přidán do pole „attends“ v dokumentu reprezentujícím onoho uživatele. Svou účast může uživatel kdykoliv zrušit kliknutím na tlačítko „Zrušit účast“.

Kapitola 5

Implementace

Dle návrhu uvedeném ve 4. kapitole byla implementována webový služba s pracovním názvem „SESHLOST“ (viz. obr. 5.1). Aplikační část byla implementována za použití programovacího jazyka JavaScript, aplikačního rámce React a vybraných knihoven dostupných skrze systém npm. Struktury a vzhledu stránky bylo dosaženo za pomocí HTML, CSS a knihovny Boostrap. Serverová část byla vytvořena s použitím aplikačního rámce Express a několika vhodných knihoven. Jako běhové prostředí pro serverovou část bylo použito Node.js. Jako datové úložiště byla použita NoSQL databáze MongoDB. Koncové body API lze nalézt v příloze A. V příloze B je k nahlédnutí struktura adresáře se zdrojovými soubory. Výsledná webová služba je plně funkční a splňuje požadavky definované v kapitole 4.2.



Obrázek 5.1: Screenshot z aplikace

5.1 Klientská část

Jak už bylo zmíněno, základ klientské části je vytvořen za pomocí aplikačního rámce React v jazyce JavaScript, jazyka HTML a knihovny Bootstrap. V klientské části je implementováno uživatelské prostředí včetně jeho vnitřní logiky. To zahrnuje zobrazování jednotlivých elementů uživatelského prostředí v závislosti na aktuálním stavu, filtrování dat, směrování v rámci aplikace, validaci dat a komunikaci se serverem. Komunikace je implementována jako asynchronní, díky čemuž jsou přechody mezi vykonáním uživatelské akce a reakcí serveru plynulé – aplikace během ukládání/načítání dat zobrazuje uživateli zpětnou vazbu v podobě animace. O výsledku akce, ať už úspěšném či nikoliv, je pak uživatel informován. Pro dosažení kýžené funkcionality klientské části aplikace je využíváno několik knihoven ze systému npm. Jejich popis a reálné využití v rámci aplikace je uvedeno níže.

Axios

Axios[1] je HTTP klient pro webové prohlížeče založený na příslibech (anglicky promises). Umožňuje webové aplikaci zasílat HTTP požadavky na koncové body serveru a zpracovávat odpovědi. Po odeslání požadavku je vrácen příslib, jakmile pak přijde odpověď ze serveru, může mít příslib dva výsledky. Buď je kladný, a v tom případě se vykonají příslušné akce, nebo je záporný a vykonají se akce jiné. Tímto způsobem je zajištěno, že komunikace klientské aplikace se serverem je asynchronní. V případě klienta Axios je za kladný výsledek považován stavový kód 2XX, vše ostatní je potom vyhodnoceno jako záporný výsledek příslibu – chyba.

V klientské části aplikace zastává Axios funkci asynchronní komunikace se serverem, kdy posílá požadavky na dané koncové body serveru a na odpovědi typu úspěch/neúspěch reaguje vykonáním předdefinovaných akcí. Axios také slouží jako tzv. interceptor a do hlavičky každého požadavku před odesláním doplňuje autorizační token z lokálního úložiště. Stejně tak přes Axios procházejí všechny příchozí odpovědi serveru, ve kterých detekuje stavový kód 401 označující chybu autorizace. Pokud je tento stavový kód detekován, dochází k odstranění případného tokenu z lokálního úložiště a k vynulování uživatelského kontextu – uživatel je odhlášen.

jwt-decode

Knihovna pro prohlížeče jwt-decode[2] dokáže dekódovat JSON Web Token a zpřístupnit tak jeho obsah. Neslouží ale pro ověření pravosti tokenu. Jelikož JWT zasílaný serverem po úspěšném přihlášení obsahuje také jméno a příjmení přihlášeného uživatele, je knihovna jwt-decode užita pro dekódování těchto údajů, které jsou následně nastaveny jako uživatelský kontext, který reprezentuje stav přihlášení uživatele. Na základě kontextu jsou zobrazena / skryta tlačítka a vypsáno jméno přihlášeného uživatele.

React Bootstrap

Knihovna React Bootstrap[3] je založena na Bootstrap verze 4. Všechny prvky, které originální knihovna nabízí, jsou přepsány do formy React komponent. Místo stylování jednotlivých elementů přidáváním tříd jsou pomocí Bootstrap stylovány celé komponenty.

[1] hhttps://www.npmjs.com/package/axios
[2] https://www.npmjs.com/package/jwt-decode
[3] https://react-bootstrap.github.io/

Pro správnou funkčnost také není třeba zahrnovat knihovnu jQuery jako v případě originální knihovny Bootstrap.

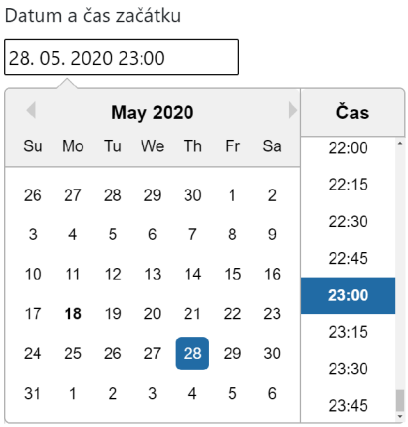
Téměř celá aplikace je tvořena komponenty právě z této knihovny. Navigační panel je vytvořen pomocí komponenty „Navbar“. Pro úpravu události, přidávání/mazání příspěvků, souborů a úvodního obrázku je využívána komponenta „Modal“ (viz. obr. 5.2). Správné formátování formulářů je dosaženo díky komponentě „Form“. Dále se pak používá „Button“ pro tlačítka, „Spinner“ pro animaci načítání, „Alert“ pro zobrazení upozornění, „Badge“ pro odznaky a „Card“ pro karty s událostmi.

Obrázek 5.2: Modální okno obsahující formulář a tlačítka z knihovny Bootstrap

React-Date-Picker

Knihovna React-Date-Picker[4] přidává do React aplikace další komponentu – kalendář pro výběr data a času. Knihovna má širokou škálu nastavení vzhledu, formátu data, zahrnutí času, výchozích hodnot apod.

V aplikaci je tato knihovna využívána pro výběr data a času začátku/konce události (viz. obr. 5.3). Ve formuláři pro vytváření události je kalendář pro výběr koncového data a času nastaven tak, aby se otevřel k datu, které bylo zvoleno jako začátek události, a zároveň nebylo možné vybrat dřívější datum. Čas lze volit v patnáctiminutových intervalech.



Obrázek 5.3: Výběr data a času s pomocí knihovny React-Date-Picker

[4] https://reactdatepicker.com/

react-google-recaptcha

Knihovna react-google-recaptcha[5] představuje Google reCAPTCHA v2 v podobě komponenty pro React aplikaci. Komponenta asynchronně načte skript pro Google reCAPTCHA a poté je instanciována samotná reCAPTCHA, se kterou může uživatel interagovat. Komponenta podporuje nastavení několika vlastností. Například „onChange“ umožňuje nastavit funkci, která má být zavolána, pokud uživatel úspěšně projde ověřením, onExpired potom volá danou funkci, pokud vyprší limit úspěšného ověření. Vlastnost „sitekey“ je určena pro nastavení webového klíče.

V aplikaci je tato komponenta hojně využívána jako ochrana proti spamu a zneužití různých uživatelských akcí. Především se vyskytuje u akcí neregistrovaného uživatele, které modifikují událost. Je použita také u registrace a samotného vytváření událostí.

google-calendar-url

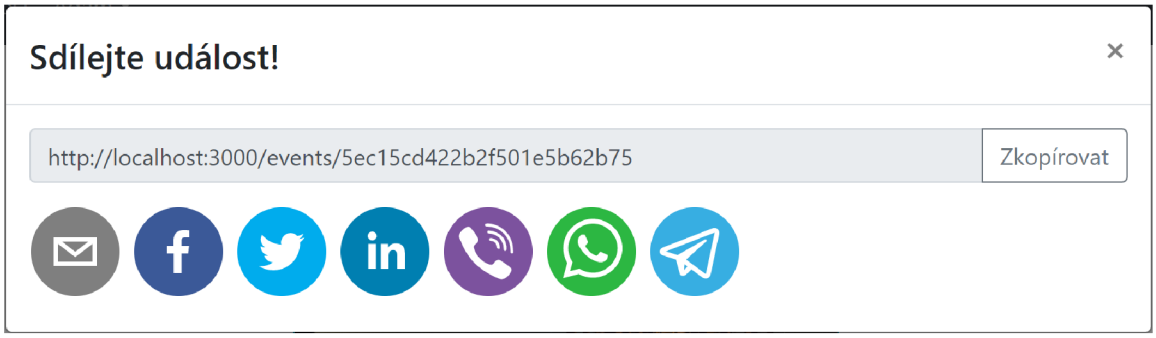
Knihovna google-calendar-url[6] na základě poskytnutých dat generuje odkaz pro přidání události do Google Kalendáře. Tato funkce v aplikaci generuje hypertextový odkaz pro tlačítko „Přidat do Google Kalendáře“.

query-string

Modul query-string[7] umožňuje číst hodnoty jednotlivých parametrů řetězce dotazu. V aplikaci je řetězec dotazu využíván pro předávání tokenu v rámci URL aktivačního odkazu, odkazu pro zrušení odběru aktualizací nebo pro předání informací při kliknutí na tlačítko „Duplikovat událost“.

react-share

Knihovna react-share[8] v podobě komponent kombinuje tlačítka pro rychlé sdílení URL adresy na populární internetové služby a ikonky s jejich logy. V aplikaci je přesně pro tento účel využívána v modálním okně „Sdílejte událost!“ (viz. obr. 5.4).



Obrázek 5.4: Odkazy pro sdílení události skrz populární sociální sítě a komunikátory

[5] https://www.npmjs.com/package/react-google-recaptcha
[6] https://www.npmjs.com/package/google-calendar-url
[7] https://www.npmjs.com/package/query-string
[8] https://www.npmjs.com/package/react-share

Google Maps Embed API

Google Maps Embed API[9] umožňuje na webovou stránku umístit interaktivní mapu pouhým vložením iframe elementu. V bezplatné základní verzi je možné toto API použít pro vyhledání a zobrazení konkrétního místa na mapě. V aplikaci je tak místo konání události předáno jako parametr pro toto API. Tímto parametrem může být název objektu, adresa nebo i souřadnice. Uživatel má tak při vytváření události celou škálu možností, jak místo konání specifikovat, aniž by se musel obávat o chybné zobrazení na mapě.

react-rte, react-render-html

Komponenta react-rte[10] implementuje WYSIWYG editor formátovaného textu. Formátování je dosaženo použitím HTML značek. Editor umožňuje mimo jiné použití tučného písma, kurzívy, číslování či odrážkového seznamu, vkládání odkazů, a dokonce i obrázků pomocí URL adresy. React-rte manipuluje s daty v rámci vlastního objektu, který je před odesláním na server převeden na řetězec. V aplikace je tato komponenta využívána pro úpravu popisu události a obsahu příspěvků. Aby při zobrazení události došlo ke správnému vykreslení textu formátovaného pomocí HTML, využívá aplikace taktéž knihovnu react-render-html[11].

5.2 Serverová část

Serverová část využívá jako základ běhové prostředí Node.js a aplikační rámec Express. Tento rámec v aplikaci slouží ke směrování požadavků na příslušné obslužné rutiny na základě definovaných koncových bodů REST API. Taktéž je hojně využívána možnost libovolného množství middleware, přes které požadavky procházejí, než se dostanou na koncovou obslužnou rutinu. Jako middleware jsou tak implementovány časti pro autorizaci a ověření reCAPTCHA. Odesílání odpovědí je taktéž řešeno skrze Express, který umožňuje jednoduše nastavit stavový kód odpovědi a připojit případná data. Serverová část taktéž slouží k validaci příchozích dat a k jejich manipulaci v rámci databáze MongoDB. Je také využíváno několik balíčků ze systému npm, jejichž popis a funkce v rámci aplikace jsou uvedeny níže.

Mongoose

Mongoose[12] je modul pro Node.js, který usnadňuje přístup k objektům v databázi MongoDB. Podporuje vytvoření schématu dat, CRUD operace a nabízí také validaci dat před jejich vložením do databáze. V serverové části webový služby SESHLOST je tento modul využíván pro vytvoření struktury dokumentů včetně definice datových typů jednotlivých polí. Dále jsou také využívány CRUD operace.

bcrypt.js

Bcrypt.js[13] je jednosměrná hašovací funkce bcrypt optimalizovaná pro jazyk JavaScript. Je založena na šifře Blowfish, ke které není dodnes známá efektivní metoda prolomení. Bcrypt

[9] https://developers.google.com/maps/documentation/embed/start
[10] https://www.npmjs.com/package/react-rte
[11] https://www.npmjs.com/package/react-render-html
[12] https://mongoosejs.com/
[13] https://www.npmjs.com/package/bcryptjs

v sobě také zahrnuje tzv. kryptografickou sůl, která při snaze o zpětné zjištění hesel ze zahašované podoby znemožňuje nebo minimálně velmi ztěžuje použití duhové tabulky. Ve webové službě je funkce bcryptjs používána pro hašování hesla k uživatelskému účtu a také hesla pro úpravy události.

jsonwebtoken

Knihovna jsonwebtoken[14] usnadňuje práci s JWT. Umožňuje vytvoření JWT podepsaného vlastním klíčem pomocí zvoleného hašovacího algoritmu. Dále také nabízí možnost ověření pravosti poskytnutého tokenu a jeho dekódování. Funkce vytvoření JWT je využívána při přihlášení uživatele, kdy je identifikátor, jméno a příjmení uživatele podepsáno hašem uživatelova hesla a odesláno klientovi. Dekódování a ověření pravosti JWT je pak využíváno v middleware, který slouží pro zabezpečení koncových bodů serveru.

express-recaptcha

Balíček express-recaptcha[15] poskytuje middleware, který se stará o vyhledání reCAPTCHA tokenu v příchozím požadavku. Middleware následně kontaktuje server služby Google reCAPTCHA, kde je token ověřen. Výsledek ověření je pak přidán do požadavku jako vlastnost „recaptcha“, který pokračuje k dalšímu zpracování. V aplikaci je pak v obslužné rutině koncového bodu serveru pouze kontrolováno, jestli se v požadavku ve vlastnosti „recaptcha“ nachází chyba. Pokud ano, ověření není úspěšné.

Nodemailer

Nodemailer[16] je modul pro Node.js umožňující snadné zasílání emailových zpráv. Podporuje odesílání čistého textu, HTML obsahu i příloh. Pro odesílání emailových zpráv primárně využívá protokol SMTP[17]. V základu obsahuje předdefinovaná nastavení SMTP pro nejznámější poskytovatele emailových služeb, je však možné definovat i vlastní nastavení.

V aplikaci je modul Nodemailer využíván pro odesílání aktivačního odkazu, potvrzení o vytvoření událostí a pro zasílání notifikací odběratelům. Při každé změně informací nebo při přidání příspěvku u události je na emailové adresy, které jsou přihlášeny u dané události k odběru, pomocí tohoto modulu odeslána notifikace.

Multer

Multer[18] je middleware pro Node.js sloužící pro práci s daty kódovanými ve formátu multipart/form-data. Využití najde především při zpracování nahrávaných souborů, kdy dokáže soubor z příchozího požadavku zpracovat, do požadavku přidat vlastnost „file“ nebo „files“ a požadavek poslat k dalšímu zpracování. V aplikaci je multer využíván pro ukládání úvodního obrázku a souborů k události. Multer příchozí soubor umístí do adresáře, jehož název odpovídá identifikátoru události, a k původnímu názvu souboru přidá datum a čas nahrání. Do požadavku je pak doplněna vlastnost „file“, jejíž součástí je mimo jiné také adresářová cesta uloženého souboru, která je uložena do databáze.

[14] https://www.npmjs.com/package/jsonwebtoken
[15] https://www.npmjs.com/package/express-recaptcha
[16] https://www.npmjs.com/package/nodemailer
[17] SMTP – Simple Mail Transfer Protocol je internetový protokol určený pro přenos zpráv elektronické pošty.
[18] https://www.npmjs.com/package/multer

fs-extra

Balíček fs-extra[19] přidává metody pro práci se souborovým systémem, které navíc podporují přísliby. V aplikaci jsou využívány metody „emptyDir“ a „mkdirs“ pro vytvoření adresářů sloužícího k ukládání úvodního obrázku a souborů dané události. Metoda „remove“ je pak využívána pro odstranění jednotlivých souborů dané události, v případě odstranění celé události pak také ke smazání příslušného adresáře.

5.3 Validace dat

Aby se zabránilo zanesení neplatných nebo špatně formátovaných dat do systému, probíhá jak v klientské, tak i v serverové části validace dat. V klientské části je využita nativní HTML5 validace formulářů, která na základě určení povinných polí a regulárních výrazů před odesláním vyhodnotí platnost zadaných dat, a případně uživatele upozorní na chyby. V serverové části je kontrolována přítomnost a délka povinných vstupních dat. Položky email a heslo jsou pak kontrolovány pomocí regulárních výrazů.

[19] https://www.npmjs.com/package/fs-extra

Kapitola 6

Testování

V rámci implementace byla průběžně testována funkčnost jednotlivých částí služby. Jednotlivé koncové body API byly vždy nejprve experimetálně otestovány pomocí programu Postman, který umožňuje přizpůsobení a zasílání HTTP požadavků všemi dostupnými metodami. Tímto způsobem bylo ověřeno správné zpracování požadavků včetně reakcí na chyby, a také byly kontrolovány odpovědi a případná data v nich obsažená. Také byl průběžně ověřován stav databáze. Jakmile byla funkčnost koncového bodu API ověřena, byla implementována část v klientské aplikaci, která na daný koncový bod zasílá požadavky. Implementovaná část klientské aplikace pak byla testována pomocí výpisu ladících informací ve vývojářských nástrojích webového prohlížeče Microsoft Edge. Na základě těchto informací a vizuální kontroly uživatelského rozhraní pak byly prováděny úpravy, které následně opět podléhaly testování. Po dokončení implementace byla ověřena funkčnost služby jako celku, a to na základě splnění následujících scénářů a kritérií.

Registrace

- Při zadání emailu v nesprávném formátu, nevyplnění povinných údajů nebo nesplnění ověření reCAPTCHA je uživatel po kliknutí na tlačítko „Zaregistrovat“ upozorněn a proces registrace nepokračuje.
- Pokud uživatel zadá email, ke kterému uživatelský účet již existuje, je uživatel po kliknutí na tlačítko „Zaregistrovat“ upozorněn a proces registrace nepokračuje.
- Pokud jsou zadané údaje validní, uživateli je po kliknutí na tlačítko „Zaregistrovat“ zobrazena informace o úspěšné registraci a o nutnosti aktivovat účet pomocí odkazu, který byl zaslán na zadaný email.
- Po kliknutí na aktivační odkaz v emailu je uživateli umožněno nastavit si heslo, a tím aktivovat svůj účet.
- Pokud zadané heslo neobsahuje minimálně jedno velké a malé písmo, minimálně jedno číslo nebo nemá minimálně 8 znaků, je na tuto skutečnost uživatel upozorněn a musí si zvolit silnější heslo.
- Pokud je zadané heslo validní, uživateli je zobrazena informace o úspěšném nastavení hesla a možnosti přihlásit se.

Přihlášení

- Pokud uživatel při přihlášení zadá nesprávný email nebo heslo, je na tuto skutečnost upozorněn.
- Pokud uživatel zadá správnou kombinaci emailové adresy a hesla, je přihlášen, přičemž se v navigační liště zobrazí tlačítko pro přístup k profilu se jménem uživatele a tlačítko „Odhlásit se“.
- Pokud uživatel zadá správnou kombinaci přihlašovacích údajů, ale účet ještě nebyl aktivován, je na tuto skutečnost upozorněn.

Profil

- Přihlášený uživatel může prohlížet svůj profil po kliknutí na tlačítko s jeho jménem v navigační liště.
- Pokud uživatel klikne na tlačítko „Změnit jméno“, je mu umožněno změnit jméno a příjmení, obě pole musí být vyplněna.
- Pokud uživatel klikne na tlačítko „Změnit heslo“, je mu umožněno nastavit nové heslo, přičemž pro provedení změny musí zadat taktéž své současné heslo. Pokud je zadané současné heslo nesprávné nebo je nové heslo příliš slabé, je uživatel upozorněn.
- Uživateli se v profilu zobrazují události, které pořádá, odebírané události, a události, kterých se účastní. Události jsou řazeny podle data konání a uživateli je umožněno skrýt již proběhlé.
- Uživateli je umožněno smazat svůj uživatelský účet a s ním také všechny vytvořené události, odběry a účasti na jiných událostech. Pro smazání účtu je nutno zadat uživatelské heslo.

Obnovení hesla

- Po kliknutí na tlačítko „Obnovit heslo“ v přihlašovacím dialogu je uživateli umožněno vyplnit emailovou adresu účtu a tím obnovit heslo.
- Při zadání emailové adresy, která patří k dosud neaktivovanému účtu, nebo nepatří k žádnému účtu, je o této skutečnosti uživatel informován.
- Pokud uživatel zadá validní emailovou adresu a splní ověření reCAPTCHA, je na danou emailovou adresu zaslán odkaz pro změnu hesla.
- Po kliknutí na odkaz pro změnu hesla v emailu je uživateli umožněno nastavit si nové heslo.
- Pokud zadané heslo neobsahuje minimálně jedno velké a malé písmo, minimálně jedno číslo nebo nemá minimálně 8 znaků, je na tuto skutečnost uživatel upozorněn a musí si zvolit silnější heslo.
- Pokud je zadané heslo validní, uživateli je zobrazena informace o úspěšném nastavení nového hesla a možnosti přihlásit se.

Vytvoření události

- Pokud při vytváření události nejsou vyplněna všechna povinná pole (název, pořadatel, místo konání, datum a čas začátku, datum a čas konce), je uživatel upozorněn. U nepřihlášeného uživatele je rovněž ověřena zadaná emailová adresa a heslo.
- Pokud je formulář vyplněný správně, uživatel je po jeho odeslání přesměrován na adresu vytvořené události, a je mu zobrazeno modální okno s možnostmi sdílení.

Zobrazení události

- Každému uživateli je v rámci zobrazení události kromě zobrazení základních informací umožněno: sdílet událost, přidat událost do Google Kalendáře, zobrazit mapu s vyznačeným místem konání, procházet příspěvky a stahovat soubory.
- Pokud si nepřihlášený uživatel zobrazí neregistrovanou událost, je mu umožněno odebírat aktualizace a událost po zadání hesla spravovat.
- Pokud si nepřihlášený uživatel zobrazí registrovanou událost, je mu umožněno odebírat aktualizace.
- Pokud si přihlášený uživatel zobrazí událost, kterou nepořádá, je mu umožněno odebírat aktualizace a události se zúčastnit.
- Pokud si přihlášený uživatel zobrazí událost, kterou pořádá, je mu umožněno událost spravovat.
- Pokud událost s daným identifikátorem neexistuje, je o tom uživatel informován.

Správa události

- Registrovanou událost může spravovat pouze její pořadatel skrz svůj uživatelský účet.
- Neregistrovanou událost může spravovat nepřihlášený uživatel po zadání hesla zvoleného při vytváření události a ověření pomocí reCAPTCHA.
- Uživateli je umožněno upravovat název, pořadatele, místo konání, datum a čas začátku a konce, a popis události.
- Uživateli je umožněno zobrazit účastníky události.
- Uživateli je umožněno událost duplikovat – otevřít formulář pro vytvoření nové události s předvyplněnými položkami.
- Uživateli je umožněno přidávat a mazat příspěvky. Každý příspěvek musí povinně obsahovat nadpis, jinak nelze přidat.
- Uživateli je umožněno nahrávat a mazat úvodní obrázek. Tento obrázek musí být ve formátu .gif, .jpg, .jpeg nebo .png. Jiné formáty nejsou akceptovány.
- Uživateli je umožněno nahrávat a mazat soubory. Soubory musí být v jednom z následujících formátů: .gif, .jpg, .jpeg, .png, .pdf, .doc, .docx, .xls, .xlsx, .zip, .rar.

- Uživateli je umožněno událost smazat, čímž dojde také ke zrušení všech účastí a odběrů, které jsou k dané události vázány.

Odběr aktualizací

- Pokud nepřihlášený uživatel klikne na tlačítko „Odebírat“, je mu umožněno vyplnit emailovou adresu, na kterou budou zasílány notifikace. Pokud je zadaná emailová adresa přiřazena k uživatelskému účtu, je na to uživatel upozorněn.
- Pokud přihlášený uživatel klikne na tlačítko „Odebírat“, zobrazí se vedle názvu události informace o odběru, a název onoho tlačítka se změní na „Zrušit odběr“. Po jeho stisknutí se vrací odběr notifikací do původního stavu – je zrušen.
- Při změně informací nebo přidání příspěvku u odebírané události je na uživatelem zadaný email odeslána notifikační zpráva.
- Pokud uživatel klikne v notifikační emailové zprávě k dané události na odkaz s názvem „Odhlásit odběr“, je dotázán na potvrzení o odhlášení odběru. Pokud odhlášení odběru potvrdí, je o tom informován.

Přihlášení účasti

- Pokud přihlášený uživatel klikne na tlačítko „Zúčastnit se“, zobrazí se vedle názvu události informace o účasti, a název onoho tlačítka se změní na „Zrušit účast“. Po jeho stisknutí se vrací účast do původního stavu – je zrušena.

Výsledky testování

Všechna výše uvedená kritéria a scénáře byly při testování naplněny. Službu taktéž otestoval nezávislý subjekt, který se s ní setkal poprvé. Při používání subjekt neodhalil žádné problémy a používání služby mu nečinilo potíže. Před případným veřejným spuštěním služby by ale bylo třeba udělat rozsáhlejší uživatelské testování.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo seznámit se s principy tvorby webových aplikací a s nimi souvisejícími technologiemi, a na základě těchto poznatků navrhnout a implementovat službu pro tvorbu informačních stránek k událostem. V práci byly nejprve popsány základní pojmy a technologie obecně týkající se webu, dále se pak práce zaměřila na technologie a aplikační rámce spojené s vývojem webových aplikací. Požadavky na samotnou službu vznikly na základě analýzy podobných existujících řešení a vlastních nápadů. Návrh služby byl pak modelován pomocí diagramu případů užití a ER diagramu. Návrh řešení každého požadavku byl detailně rozepsán. Dle návrhu byla implementována webová služba s názvem „SESHLOST“. Kromě značkovacího jazyka HTML a CSS pravidel byla celá služba napsána v programovacím jazyce JavaScript, přičemž využívá serverový aplikační rámec Express v běhovém prostředí Node.js, dokumentovou databázi MongoDB a klientský aplikační rámec React. Jak serverová, tak i klientská část jsou plně funkční a splňují předem dané požadavky. Služba svým uživatelům umožňuje vytvářet události, které mohou následně sdílet pomocí odkazu, upravovat je a přidávat k nim příspěvky či soubory. Služba taktéž umožňuje uživatelům potvrdit účast na události nebo odebírat aktualizace prostřednictvím emailových zpráv. Při vývoji služby bylo dbáno na řádné zabezpečení.

Byť minimalistické uživatelské prostředí dobře plní svůj účel, službě by do budoucna neuškodil líbivější a promyšlenější design. Taktéž revize a optimalizace serverové části by pravděpodobně prospěla. Co se týče další funkcionality, určitě by stála za zvážení větší míra přizpůsobení stránky s detaily události – různé barvy, fonty apod. Také filtrování nebo vyhledávání v seznamu událostí by mohlo uživateli zjednodušit práci, i když vzhledem k zaměření služby na soukromé události, kterých pravděpodobně nebude větší množství, je přínosnost takové funkce spíše diskutabilní. V kontextu dnešní doby se pak přímo nabízí implementace aplikace pro chytré mobilní telefony, která by webovou aplikaci doplnila a celou službu posunula o kus dále. Ostatně serverová část je na to částečně připravena díky využití REST API.

Kromě funkční webové služby přináší práce určitý rozhled ve světě webových aplikací a ukazuje sílu modulárních systémů a jejich knihoven. Jakožto nováček ve světě JavaScriptu jsem si v rámci tvorby této bakalářské práce osvojil principy tohoto programovacího jazyka. Taktéž jsem poprvé pracoval s NoSQL databází a velmi populárním aplikačním rámcem React. Neméně obohacující zkušeností byla tvorba jak serverové, tak i klientské části služby včetně snahy o vytvoření funkčního uživatelského prostředí. Pevně věřím, že zkušenosti nabyté v rámci tvorby své bakalářské práce v budoucnu zužitkuji.

Literatura

[1] Amazon. What is NoSQL? *Amazon Web Services (AWS) - Cloud Computing Services* [online]. 2020 [cit. 2020-02-21]. Dostupné z: https://aws.amazon.com/nosql/.

[2] auth0.com. Introduction to JSON Web Tokens. *JSON Web Tokens - jwt.io* [online]. 2020 [cit. 2020-03-04]. Dostupné z: https://jwt.io/introduction/.

[3] Berners Lee, T., Bray, T., Connolly, D. et al. Architecture of the World Wide Web, Volume One. *World Wide Web Consortium (W3C)* [online]. 2004 [cit. 2020-01-21]. Dostupné z: https://www.w3.org/TR/2004/REC-webarch-20041215.

[4] CERN. A short history of the Web. *Home | CERN* [online]. 2015 [cit. 2020-01-21]. Dostupné z: https://home.cern/science/computing/birth-web/short-history-web.

[5] Codesido, I. What is front-end development? *The Guardian* [online]. 2009 [cit. 2020-03-07]. Dostupné z: https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost.

[6] Confessore, N. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. *The New York Times* [online]. 2018 [cit. 2020-03-20]. Dostupné z: https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html.

[7] evvnt. *Evvnt - The Best Place To Promote Your Event Online* [online]. 2018. 2020 [cit. 2020-03-20]. Dostupné z: https://evvnt.com/.

[8] Facebook. *React – A JavaScript library for building user interfaces* [online]. 2020 [cit. 2020-03-21]. Dostupné z: https://reactjs.org/.

[9] Fielding, R. T. a Taylor, R. N. *Architectural Styles and the Design of Network-Based Software Architectures*. Irvine, California, USA, 2000. Disertační práce. University of California, Irvine. ISBN 0599871180.

[10] Goel, A. 10 Best Web Development Frameworks. *Hackr.io* [online]. 2020 [cit. 2020-03-11]. Dostupné z: https://hackr.io/blog/top-10-web-development-frameworks-in-2020.

[11] Google. Developer's Guide | reCAPTCHA. *Google Developers* [online]. 2019 [cit. 2020-04-15]. Dostupné z: https://developers.google.com/recaptcha/intro.

[12] MDN. An overview of HTTP. *MDN Web Docs* [online]. 2019 [cit. 2020-01-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

[13] MDN. What is a URL? *MDN Web Docs* [online]. 2019 [cit. 2020-02-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL.

[14] MDN. HTML basics. *MDN Web Docs* [online]. 2019 [cit. 2020-02-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics.

[15] MDN. Learn to style HTML using CSS. *MDN Web Docs* [online]. 2020 [cit. 2020-02-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS.

[16] MDN. What is JavaScript? *MDN Web Docs* [online]. 2020 [cit. 2020-02-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[17] MDN. Express/Node introduction. *MDN Web Docs* [online]. 2020 [cit. 2020-03-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

[18] Melendez, S. The Difference Between Dynamic & Static Web Pages. *Small Business - Chron.com<* [online]. 2018 [cit. 2020-01-21]. Dostupné z: https://smallbusiness.chron.com/difference-between-dynamic-static-pages-69951.html.

[19] MongoDB. *The most popular database for modern apps | MongoDB* [online]. 2020 [cit. 2020-03-22]. Dostupné z: https://www.mongodb.com/.

[20] Oracle. What Is a Relational Database? *Oracle | Integrated Cloud Applications and Platform Services* [online]. 2020 [cit. 2020-02-21]. Dostupné z: https://www.oracle.com/database/what-is-a-relational-database/.

[21] Reenskaug, T. a Coplien, J. O. The DCI Architecture: A New Vision of Object-Oriented Programming. *Artima* [online]. 2009 [cit. 2020-02-02]. Dostupné z: https://www.artima.com/articles/dci_vision.html.

Příloha A

Seznam koncových bodů API webové služby

Metoda	Koncový bod	Funkce
POST	api/users/create	Vytvoří uživatelský účet a na zadanou emailovou adresu zašle aktivační odkaz.
POST	api/users/login	V případě správných přihlašovacích údajů vrací JWT.
GET	api/users/profile	Vrací osobní údaje včetně seznamu relevantních událostí aktuálně přihlášeného uživatele.
PUT	api/users/profile	Upraví osobní údaje aktuálně přihlášeného uživatele.
DELETE	api/users/delete	Po ověření hesla odstraní uživatelský účet aktuálně přihlášeného uživatele včetně jeho událostí, účastí a odběrů.
PUT	api/users/password/change	Na základě zadání správného hesla změní uživatelovo heslo na jím požadované.
POST	api/users/password/new	Nastaví uživatelské heslo při aktivaci účtu nebo obnovení hesla.
POST	api/users/password/reset	Na zadanou emailové adresy zašle odkaz pro obnovení hesla.

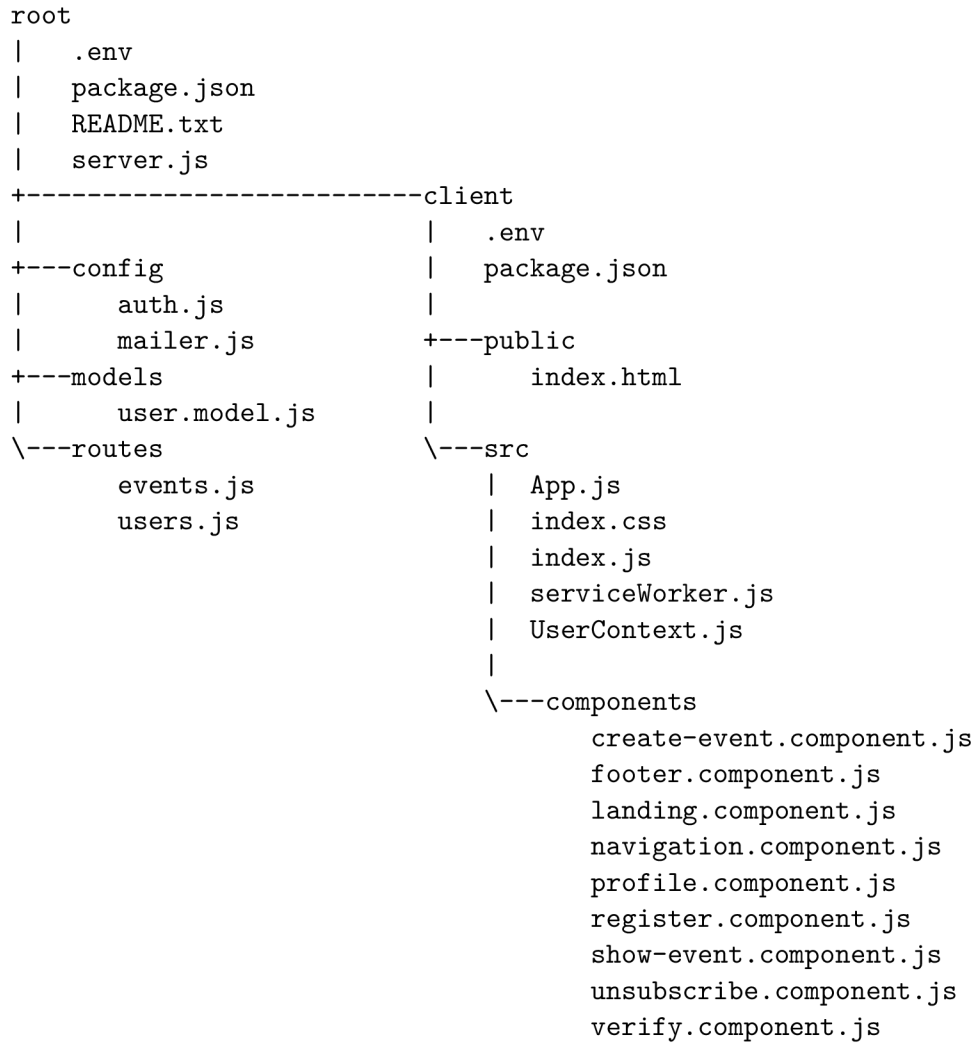
Tabulka A.1: Koncové body API pro požadavky týkající se uživatelů

Metoda	Koncový bod	Funkce
POST	api/events/create	Vytvoří událost se zadanými informacemi, vrací identifikátor vytvořené události.
PUT	api/events/:id	Upraví informace o události s daným identifikátorem.
GET	api/events/:id	Vrací informace o události s daným identifikátorem včetně informací o tom, zda je přihlášený uživatel vlastník, účastník či odběratel.
DELETE	api/events/:id	Odstraní událost s daným identifikátorem.
POST	api/events/:id/posts	K události s daným identifikátorem přidá příspěvek.
DELETE	api/events/:id/:post_id	Odstraní příspěvek s daným identifikátorem u události s daným indentifikátorem.
POST	api/events/:id/files	Nahraje soubor a přiřadí jej k událostí s daným identifikátorem.
DELETE	api/events/:id/files/	Odstraní soubor daný jeho cestou u události s daným identifikátorem.
POST	api/events/:id/cover	Nahraje úvodní obrázek a přiřadí jej k událostí s daným identifikátorem. Případný existující úvodní obrázek je tím novým nahrazen.
DELETE	api/events/:id/cover	Odstraní obrázek u události s daným identifikátorem.
POST	api/events/:id/subscribers	K události s daným identifikátorem přidá email odběratele. Pokud se jedná o přihlášeného uživatele, přidá k jeho uživatelskému účtu identifikátor události.
DELETE	api/events/:id/subscribers	Odstraní email u události s daným identifikátorem. V případě odběru přihlášeného uživatele také odstraní identifikátor události z jeho uživatelského účtu.
POST	api/events/:id/attendees	Přidá identifikátor aktuálně přihlášeného uživatele k události s daným identifikátorem. Identifikátor události je přidám k účtu aktuálně přihlášeného uživatele.
POST	api/events/:id/allattendees	Vrací seznam účastníků události s daným identifikátorem.
DELETE	api/events/:id/attendees	Odstraní identifikátor aktuálně přihlášeného uživatele u události s daným identifikátorem. Identifikátor události je rovněž odstraněn od účtu aktuálně přihlášeného uživatele.

Tabulka A.2: Koncové body API pro požadavky týkající se událostí

Příloha B

Struktura adresáře webové služby



Obrázek B.1: Struktura adresáře se zdrojovými soubory

V kořenovém adresáři (viz. obr. B.1) se nachází serverová část služby, přičemž v souboru server.js je implementováno nastavení serveru, připojení k databázi, směrování požadavků, přístup ke statickým souborům atd. V kořenovém adresáři se taktéž nachází soubor .env se systémovými proměnnými serveru a soubor README.txt s návodem pro konfiguraci a spuštění služby. Adresář config obsahuje middleware, který slouží k autorizaci uživatele, a konfiguraci modulu pro odesílání emailů. V adresáři models je definováno schéma databáze, routes pak obsahuje soubory s obslužnými rutinami koncových bodů a veškerou zbývající logiku.

Adresář client je vyhrazen pro klientskou část, soubor .env obsahuje systémové proměnné pro klientskou část. V adresáři public je umístěn jediný HTML dokument celé aplikace – veškeré zobrazení má pak na starosti React a jeho virtuální DOM. Adresář src obsahuje kaskádové styly a ostatní podpůrné soubory, App.js obstarává vložení autorizačního tokenu do hlavičky každého požadavku a zobrazování jednotlivých komponent v závislosti na aktuální URL. Adresář components obsahuje všechny komponenty, ve kterých je definována struktura, vzhled a logika klientské části.