TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Online zobrazení dat z terénního měření

Bakalářská práce

Studijní program: B0613A140005 Informační technologie
Autor práce: **Valentin Daitkhe**
Vedoucí práce: Ing. Lenka Kosková Třísková, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Online zobrazení dat z terénního měření

Jméno a příjmení: **Valentin Daitkhe**
Osobní číslo: M19000054
Studijní program: B0613A140005 Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se stávajícími vlastnostmi systému.
2. Analyzujte technologickou vyspělost stávajícího řešení a navrhněte případné úpravy.
3. Navrhněte architekturu pro nový zobrazovací frotend systému, jenž naváže na stávající řešení (databáze + datový model).
4. Navržené řešení implementujte.

Rozsah grafických prací: dle potřeby
Rozsah pracovní zprávy: 30-40 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] Mardan A.: React Quickly: Painless web apps with React, JSX, Redux, and GraphQL2017,ISBN: 1617293342.
[2] Gore A.: Full-Stack Vue.js 2 and Laravel 5: Bring the frontend and backend together with Vue, Vuex, and Laravel, Packt Publishing 2017, ISBN: 1788299582.

Vedoucí práce: Ing. Lenka Kosková Třísková, Ph.D.
Ústav nových technologií a aplikované informatiky

Datum zadání práce: 9. října 2019
Předpokládaný termín odevzdání: 3. ledna 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 17. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. června 2021 Valentin Daitkhe

Online zobrazení dat z terenního meření

Abstrakt

Tato bakalářská práce je zaměřena na analýzu a zlepšení existující webové aplikace, která nabízí běžnému uživateli pohodlné rozhraní s funkcemi určenými pro správu a analýzu dat naměřených skupinou čidel. Rešerše této práce se zabývá analýzou existujícího webového portálu, požadavky na funkcionalitu možného zlepšujícího řešení a možnostmi vylepšení aktuální webové aplikace. V závěrů rešerše je uveden návrh zlepšení webové aplikace, přímá implementace tohoto návrhu je popsána v kapitole s názvem ”Implementace”, ve které je detailně popsáno jak uživatelské rozhraní na klientské straně, tak i metody správy a proudu dat ze strany serveru. V kapitole ”Implementace” je popsaná architektura celého řešení této práce. Všechny použité technologie pro realizaci řešení této práce jsou popsány v kapitole ”Použité technologie”. Pro tvorbu klientské části byl použit programovací jazyk JavaScript, knihovna React JS a knihovna Matherial UI. Serverová část řešení je realizována pomocí programovacího jazyku PHP a frameworku Lumen. Pro zachráněni dat uživatelů je využito databáze MySQL s ohledem na to, že aktuální datový model zůstane beze změn. V závěrečné části práce je uveden souhrn cílů stanovený na začátku práce, ohodnocení dosažených výsledků a možnosti dalšího rozšíření této práce.

Klíčová slova: JavaScript, React JS, Redux, Matherial UI, JSX, CSS, SASS, LESS, PHP, Lumen, Laravel, MySQL, JWT, MVC, REST API.

Abstract

This bachelor thesis is focused on analysis and improvement of an existing web application, which allows users to use it as a tool for management and anasysis of data measured by a group of sensors. The research of this work deals with anylysis of the existing web portal, requirements for functionalities of improvement of a possible better solution and possibilities of improvement of the current web application. The conclusion presents a concept of application improving, an implementation of this concept is described in the ”Implementation” part, which tells particularly about both user interface on client side and methods of data flow and management on server side. Also this part describes an architecture of the whole work solution. All technologies used for implemention of the solution of this study are contained in ”Available Technologies” part. For client part of the solution were used JavaScript programming language, React JS and Matherial UI libraries. The server part of the solution is implemented using PHP programming language and Lumen framework. For user's data storage was applied MySQL database, with a condition that the current data model will not be changed. The conclusion summarizes the goals set at the beginning of the work, evaluation of expected results and possibilities for a further development.

Keywords: JavaScript, React JS, Redux, Matherial UI, JSX, CSS, SASS, LESS, PHP, Lumen, Laravel, MySQL, JWT, MVC, REST API.

Poděkování

Rad bych poděkoval vědoucí této bakalářské práce paní Ing. Lence Koskové Třískové Ph.D. za rady při psaní této práce.

Obsah

Seznam zkratek 11

1 Úvod **12**

2 Rešerše **13**
2.1 Problematika 13
2.2 Analýza stávajícího řešení 14
2.2.1 Analýza databázového modelu 14
2.2.2 Analýza serverové části 16
2.2.3 Analýza klientské části uživatelského rozhraní 17
2.3 Základní požadavky na funkčnost aplikace 19
2.4 Možnost vylepšení aktuální aplikace 21
2.4.1 Možnost vylepšení datového úložiště 22
2.4.2 Možnost vylepšení serverové části 24
2.4.3 Možnost vylepšení klientské části 25
2.4.4 Závěr 26

3 Použité technologie **27**
3.1 JavaScript knihovna React 27
3.1.1 Komponenty React a virtuální DOM 27
3.1.2 Problematika předávaní dat mezi komponentami 28
3.1.3 Řízení stavu aplikace 28
3.2 JavaScript knihovna Material UI 29
3.3 NodeJS, NPM a React Scripts 30
3.4 Framework Lumen a manager balíčků Composer 31
3.4.1 Práce s databází a Eloquent ORM 32
3.4.2 REST API ve frameworku Lumen 33
3.4.3 JWT ve frameworku Lumen 34
3.5 MySQL 34

4 Implementace **35**
4.1 Klient 35
4.1.1 Seznam experimentů 35
4.1.2 Seznam čidel 37
4.1.3 Grafické zobrazení dat a export 38
4.1.4 Výběr a porovnaní dat z různých experimentů 39

4.1.5 Profil uživatele 40
4.1.6 Přihlášení uživatelů 40
4.2 Server 41

5 Závěr **42**

Seznam zkratek

HTML	(Hypertext Markup Language) - strukturální jazyk pro tvorbu webových stránek.
XML	(eXtensible Markup Language) - rozšiřitelný strukturální jazyk.
JSON	(JavaScript Object Notation) - datový formát, určený pro přenos dat.
CSS	(Cascading Style Sheets) - jazyk popisující způsob zobrazení dokumentu tvořeného pomocí strukturálních jazyku.
CSV	(Comma-Separated Values) - datový formát, určený pro uložení dat v formě tabulek.
REST	(Representational State Transfer) - architektura, určena pro výměnu informací mezi klientem a serverem.
HTTP	(HyperText Transfer Protocol) - internetový protokol určený pro přenos hypertextových dokumentů.
SQL	(Structured Query Language) - jazyk pro manipulaci a správu dat uložených v databázi.
JWT	(JSON Web Token) - způsob pro bezpečnou komunikace mezi mezi dvěma stranami.
URL	(Uniform Resource Locator) - jednoznačný textový řádek, používá se pro přesnou identifikaci dokumentů na internetu.
UML	(Unified Modeling Language) - jazyk pro tvorbu diagramů, který se používá pří vývoji softwaru.
No-SQL	(not only SQL) - struktura datového úložiště nepoužívajícího SQL pro manipulace s data.
ORM	(Objektově–relační mapování) - objektový přístup k SQL databázi.
JSX	(JavaScript XML) - dává možnost zápisu XML (HTML) tagů přímo do JavaScriptového kódu.
MVVM	(Model–view–viewmodel) - návrhový vzor, odděluje data, stav aplikace a uživatelské rozhraní.
DOM	(Document Object Model) - objektový model dokumentu, umožňující pomocí JavaScriptu přistupovat k jednotlivým prvkům v HTML.

1 Úvod

Dnes je poměrně hodně webových aplikací zaměřených se na zjednodušení práce s nejrůznějšími typy informací. Existuje velká potřeba aplikací pro sledovaní stavu a aktivit nejrůznějších zdrojů a snímačů. Důraz je kladen na zobrazení dat ve vhodném a čitelném pro uživatele tvaru, stejně jako další správu či export do jiných tvarů. V jakékoliv velké laboratoři nebo při konání experimentu někde jinde je potřebný podobný systém. Ve světě existuje hodně řešení tohoto druhu, a zpravidla jsou některé z nich aktivně využívány. Avšak se často vyžaduje specifická funkcionalita pro konkretní systém, kterou stávající software neimplementuje. Může se stát opačná situace, když je hotové softwarové řešení tak obrovské a obsahuje v sobě příliš hodně funkcí, že jeho využití není optimální.

Pro tuto práci byl stanoven požadavek na zlepšení a modernizaci měřicího systému pro experimenty centra experimentální geotechniky. Tento požadavek určuje dva konkrétní cíle: analytický a praktický. První cíl spočívá v seznámení se s existujícím systémem a definici konkretní funkcionality pro jeho modernizaci a následujícího zlepšení. Součástí analýzy je taky zkoumání problematiky výběru technologií, tedy nástrojů a programovacích jazyků pro tvorbu efektivního a moderního řešení. Praktická část zahrnuje implementaci vlastního řešení.

Analytická část obsahuje seznámení se s existující aplikací, průzkum jejího kódu a určení kritických míst uživatelského rozhraní. Řešení analytické části dále spočívá ve vytvoření seznamu uživatelských funkcí a stanovení požadavků, ze kterých je pak vytvořen diagram základních požadavků pro konečnou funkcionalitu systému. Po stanovení konkretních požadavků byla potřeba prozkoumat aktuální instrumenty a technologie pro realizaci těchto požadavků a funkcí.

Řešení praktické části zahrnuje tvorbu nadhledů jednotlivých částí aplikace založených na diagramu základních požadavků systému a dosažení pozitivního hodnocení od zákazníka, tedy zástupce centra experimentální geotechniky. Dalším bodem praktického řešení je implementace funkčních nadhledů ve tvaru modulů výchozí aplikace, vytvořených na základě vybraných technologií.

Mojí motivací k vypracovaní této práce byly seznámení se s řadou moderních technologií a praxe, která byla získána během celého cyklu vývoje softwaru od obdržení požadavků na budoucí aplikaci do prezentace první verzi.

2 Rešerše

Tato kapitola se zabývá problematikou analýzy existující webové aplikace. Obsahuje průzkum zdrojového kódu a architektury aplikace a nalezení kritických míst, chyb a útoků. Dále se v této kapitole popisuje analýza základních komponentů uživatelského rozhraní aktuálního systému.

Po analýze měřicího systému jsou definovány požadavky na základní funkcionalitu. Výsledkem je diagram požadavků a funkčností existujícího řešení. Na základě diagramu se rozhodne o možnostech zlepšení, tedy modernizace aktuální webové aplikace včetně uživatelského rozhraní a serverové části. V závěru kapitoly je navržena vylepšená verze měřicího systému pro experimenty centra experimentální geotechniky.

2.1 Problematika

Problematika této práce je obecnou problematikou normálního cyklu vývoje libovolného softwaru, od obdržení požadavků zákazníka, přes jejich analýzu i návrh architektury softwaru až do vydání alfa verze. Po analýze problémů je možné stanovit několik nejčastějších:

- Disharmonie - programátor není vždy odborníkem v oblasti, kde bude program využíván, a zákazník nevyjadřuje své požadavky vždy jasně. Proto často dochází k neporozuměním kvůli rozdílným názorům na budoucí softwarový produkt.
- Nedostatek průhlednosti - v jakémkoliv okamžiku je obtížné říct, v jakém je projekt stavu a jaké je procento jeho dokončení. Tento problém vzniká při nedostatečném plánování struktury (nebo architektury) budoucího softwaru. Během plánování je třeba brát v úvahu, jak dlouho bude vývoj trvat, jaké jsou fáze vývoje.
- Ladění softwarů - nejtěžší z problémů je vyhledávání a opravy chyb softwaru, přičemž čim dříve vznikla chyba v procesu vývoje softwaru, tím více času je třeba na její opravu.

2.2 Analýza stávajícího řešení

Analýzou aplikací se lidé zabývají od doby vzniku počítače. V té době následkem této analýzy byla ruční optimalizace kódu na děrných štítcích. Dneska je to velká industrie a je jednou z nejdůležitějších oblastí IT trhu.

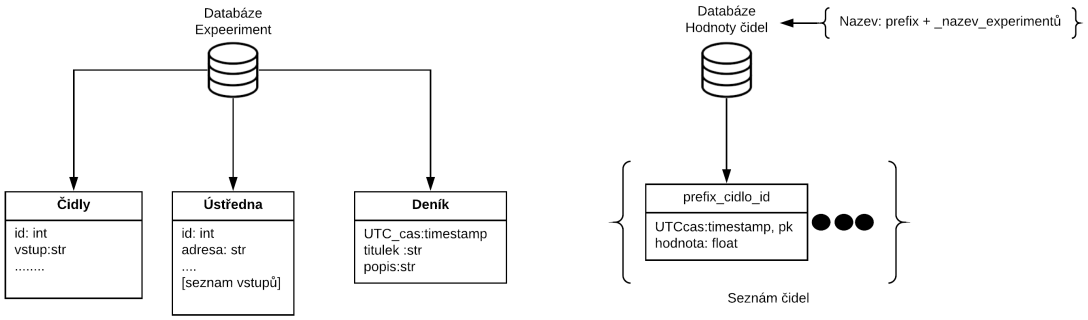
Aktuální webová aplikace je měřicím systémem pro experimenty centra experimentální geotechniky. Umožňuje výběr experimentů ze seznamu, každý z experimentů obsahuje vlastní seznam čidel a jejich technické parametry, pomocí kterých může uživatel vyhledávat jednotlivé čidly a přidávat je do grafu. Graf zobrazuje data čidel podle časového intervalu. Tato čidla jsou propojena s experimentem přes ústřednu, aplikace taky umožňuje zobrazit stav ústředny a připojených čidel. Měřicí systém umožňuje zobrazení 3D modelu experimentu, v němž lze vybrat jednotlivá čidla.

Problematika analýzy spočívá v ohodnocení kvality softwarového balíku. Příkladem metody vyhodnocení kvality softwaru je ideologie FURPS, která byla navržena firmou Hewlett-Packard. Zkratka názvu pochází z pojmů funkčnost, použitelnost, spolehlivost, výkon, podpora [1]. FURPS je použitelná pro hodnocení kvality libovolného softwaru.

2.2.1 Analýza databázového modelu

Analýza a optimalizace datového modelu je jedna z nejdůležitějších částí vývoje softwaru, protože neoptimální řešení může zvětšovat celkový výkon systému a zkomplikovávat další podporu, následně i zvyšovat složitost celého systému.

Aktuální webová aplikace pro uchovávání dat využívá relační databáze MySQL, jehož model je na obrázku 2.1. Hlavním problémem tohoto modelu je, že nejsou implementovány základní principy relačních databází. Experimenty jsou chráněny jako jednotlivé databáze, obsahující nenormalizované entity [5].



Obrázek 2.1: Aktuální model databáze

Měřicí systém přijímá velké množství dat z čidel v malých časových intervalech. Z tohoto důvodu je dobrým nápadem provést jednoduché testovaní relačních databází. Cílem je mít představu o tom, jak s touto úlohou pracují. Seznam testovaných relačních databází je: MySQL (samostatné implementace InnoDB a MyISAM), MariaDB, PostgreSQL, Oracle. Pro realizaci testu využijeme programovací jazyk Java a knihovnu JMH [9] pro měření času odezvy.

Obrázek 2.2: Tabulka Uživatel

V daném jednoduchém testu relačních databází hlavním cílem bylo ohodnocení časů odezvy na příslušné dotazy. V rámci této práce místo aktuálního datového modelu stačí tabulka "Uživatel" zahrnující standardní informace o uživateli 2.2. Propojení s knihovnou JMH lze snadno provést přes závislost "maven". Nastavení bylo provedeno v konfiguračním souboru pom.xml [9].

	Oracle	PostgreSQL	H2	MyISAM	MariaDB	InnoDB
100 řádků	499, 774	435, 367	124, 041	387, 613	487, 720	545, 770
1000 řádků	559, 887	714, 510	708, 488	1 441, 750	2 634, 031	2 611, 083
10 000 řádků	8 842, 546	4 236, 861	9 433, 653	10 359, 910	26 056, 574	25 008, 635
100 000 řádků	18 965, 460	36 947, 172	2 357 105, 910	99 816, 738	— *	— *

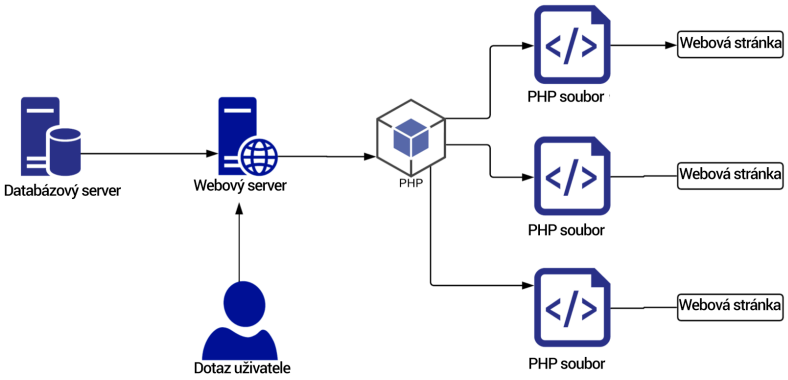
Obrázek 2.3: Výsledky testu v mikrosekundách

Na základě výsledků testu 2.3 relační databáze bylo rozhodnuto, že databáze tohoto typu nejsou optimální variantou kvůli tomu, že v tomto případě by zvětšovaly složitost celé aplikace. Hlavními problémy použití MySQL jsou zachráněni velkého objemu nestrukturovaných dat a rychlost zpracovávání velkého počtu dotazů. Jedním ze základních plusů relačních databází je dobrá struktura informace, která se tady však nevyužívá (2.1), a z toho vycházejí následující nedostatky datového úložiště:

- Složitost dotazovaní
- Složitost při implementaci principů ORM [2].
- Složitost rozšíření aplikace o další možnosti
- Složitost provedení libovolných změn.

2.2.2 Analýza serverové části

Severová část měřicího systému je realizována pomocí programovacího jazyku PHP a využití archaické procedurální metody programovaní. Slučuje logiku uživatelského rozhraní a základní kód aplikace 2.4.



Obrázek 2.4: Aktuální architektura serverové části

Pro připojení k databázi je v kódu aktuální webové aplikace využito zastaralé a nebezpečné knihovny msqlli. Korekce uživatelských vstupů se provádí pomocí funkce addslashes() 2.5, která nerozpoznává mnohabitové symboly, potřebné například při kódování GBK, což může teoreticky způsobit útok SQL injekce [3].

```
if (isset($_GET['cidlo'])) {
    $cc = addslashes($_GET["cidlo"]);/// Sql Inj
} else {
    die('# neni cidlo!');
}
mysqli_select_db($con, $databaze);

$sql = "SELECT * FROM cidla WHERE cislo = '$cc'

$result = mysqli_query($con, $sql);
```

Obrázek 2.5: Kód aktuální aplikace umožnující útok SQL injekce

2.2.3 Analýza klientské části uživatelského rozhraní

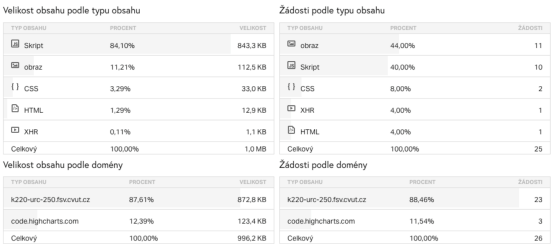
Daná podkapitola popisuje analýzu klientské části aplikace. Je rozdělena podle logických skupin analýzy struktury webových stránek a uživatelského rozhraní.

Analýza logiky klientské části

Klientská část měřicího systému je realizována pomocí standardních technologií, jako jsou knihovny JQuery jazyka JavaScript, HTML, CSS a PHP. Dále byly využity knihovna TreeJS pro realizaci 3D modelu experimentu s příslušnými čidly a doplňující moduly pro zobrazení dat jako HighlightJS a Datatables.

Kvůli tomu, že serverová logika je propojena přímo s konkretními částmi uživatelského rozhraní a navíc uživatelské rozhraní je realizováno jako jednotlivé stránky měřicího systému, nejde systém snadno rozšířit o další funkce, stejně jako použit znovu existující komponenty klientské části aplikace.

Základním bodem dobré webové aplikace je rychlost odezvy na dotaz. Za její rychlost odpovídá celkový rozměr aplikace a počet stažených závislostí, tedy zdrojového kódu, obrázků atd. Pomocí servisu tools.pingdom.com bylo provedeno testovaní aktuální webové aplikace.



Obrázek 2.6: Test rychlosti

Jak vidíme z obrázku 2.6, hlavní příčinou zatěžovaní je stažení JavaScriptu. Tato situace vzniká kvůli tomu, že měřicí systém využívá plné verze docela velkých knihoven, například JQuery a TreeJS. Všechny podobné zdroje nejsou stlačené třeba pomocí gzip, a proto potřebují velký objem paměti.

Druhou důležitou vlastností webových aplikací je možnost práce s různými hardwarovými systémy, aby funkcionalita v různých browserech a na různých typech zařízení byla stejná. Servisů umožnujících testování této funkcionality je poměrně dost. Libovolný browser poskytuje emulaci různých zařízení jako doplňkovou funkcionalitu pro vývojáře.

Na základě emulací různých zařízení a následujícího testování měřicího systému bylo zjištěno, že aktuální měřicí systém neposkytuje funkcionalitu represivního designu, tedy nemůže fungovat na různých typech zařízení, a že existující styly nejsou normalizovány. Kvůli tomu, že existuje hodně různých browserů a každý z nich je výrobkem odlišné IT společnosti, jednotlivý browser interpretuje stylizaci stránek trochu jinak. Pro řešení daného problému se

obvykle používají normalizační knihovny, které resetují styly browseru, aby nebyly změněny styly aplikace.

Analýza použitelnosti uživatelského rozhraní

Někdy narazíme na nejasné, nelogické aplikace, jejichž funkce a způsoby použití nejsou často srozumitelné. Po práci s podobnými aplikací se zřídka zachce použit je znovu, a tak vzniká potřeba najít vhodnější analog. Pro normální práci s aplikací nestačí jenom ten fakt, že umí plnit svoji funkci. Měla by být také pohodlná. Intuitivně pochopitelná aplikace šetří uživatelům nervy a čas, aby se s ní naučili pracovat.

Chceme-li zjistit, jestli je web nebo jiný softwarový produkt pohodlný pro uživatele, musíme se jich na to zeptat. Předpokládá se však, že testování aplikace reálnými uživateli vyžaduje hodně času a energie vývojáře. Pro tento případ existují speciální nástroje pro testování použitelnosti, které pomáhají zjistit, jak moc aplikace splňuje očekávání uživatelů a zároveň šetří čas a peníze.

Testování použitelnosti je rešerše, která může určit, zda je hromadný produkt, prototyp nebo koncept vhodný pro zamýšlené použití. Tento proces je založen na ergonomickém testování. Je to metoda pro určení toho, jak se bude aplikace chovat během použití na základě účasti testeru nebo softwaru, který simuluje činnost testeru.

Pomocí webového servisu heat-map.co, jenž simuluje reálné uživatele, se dá sestavit teplotní mapu skutečného systému. Existují i jiné servisy a softwary s takovou funkcionalitou, ale v daném případě servis heat-map.co je nejlepší variantou díky tomu, že podporuje přesně ty funkcionality, které jsou potřebné, a je zadarmo.

Obrázek 2.7: Teplotní mapa heat-map.co

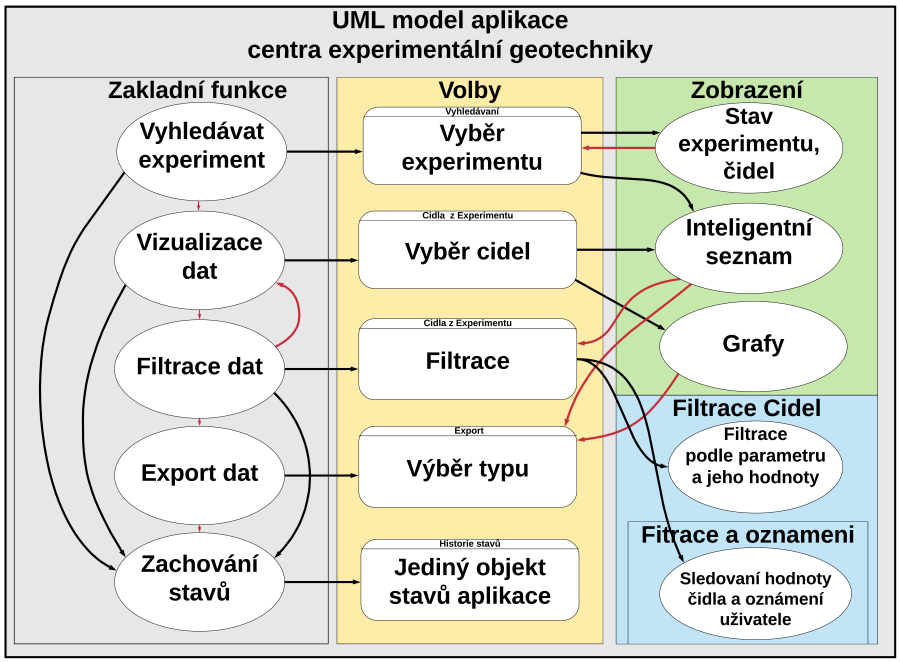
Na obrázku 2.7 je teplotní mapa sestavená po třiceti sekundách testu. Podle základních řídicích elementů je vidět, že mapa vypadá chaoticky. To znamená, že není komfortně používat tuto část aplikace. Uživatele by pak vyhledávali nápovědu na použití a přecházeli by do jiných položek aplikace.

2.3 Základní požadavky na funkčnost aplikace

Bez ohledu na metodiku vývoje, první fází vývoje je formulace požadavků pro software. Sada požadavků je technický úkol, přičemž jsou požadavky rozděleny na funkční (co systém umožňuje) a nefunkční (požadavky na zařízení, operační systém atd.). Na základě provedené analýzy měřicího systému pro experimenty centra experimentální geotechniky byl sestaven následující seznam požadavků:

- Inteligentní výběr čidel pro další zpracování
- Analýza a filtrování dat
- Grafická reprezentace dat
- Export dat grafické reprezentace
- Kontrola stavu – zda je vše v pořádku
- Uložení konfigurace do URL.

Z tohoto seznamu byl vytvořen UML diagram celé aplikace pro lepší přehlednost.



Obrázek 2.8: UML diagram

Diagram byl rozdělen na tři moduly. První je označen šedou barvou. Reprezentuje základní požadavky pro měřící systém, tedy co všechno aplikace dělá.

Druhý sloupec popisuje konkretní funkcionalitu aplikace podle provedené aktivity z prvního sloupce. Třetí modul je podmnožinou druhé třídy a reprezentuje možný výstup funkce aplikace po splnění akce z druhé třídy.

Díky provedené analýze lze snadno vytvořit stručný popis sekvence uživatelských aktivit během využití měřicího systému:

- Vyhledávaní experimentu - začátkem práce s měřicím systémem je výběr experimentu, pro který budou pak v aplikaci zobrazena data. V aktuální aplikaci se tento výběr provádí pomocí seznamu experimentů. Z pohledu využitelnosti tato funkcionalita funguje dobře, ale jenom díky tomu, že měřicí systém obsahuje současně jenom několik experimentů. V případě, že experimentů bude více, aplikace by musela mít možnost vyhledávaní experimentů třeba podle nazvu.

  Po provedení akce výběru potřebného experimentu aplikace musí zobrazit stav vybraného experimentu - ukázat, zda je všechno v pořádku a experiment probíhá normálně.

  Vyhledávaní experimentu a obdržení informací o jeho stavu bylo rozhodnuto umístit do jedné akce pro lepší použitelnost aplikace, aby uživatel mohl zjistit všechno o experimentu ještě v době výběru.

- Vizualizace dat - po výběru experimentu aplikace zobrazuje jeho data v jednoduše pochopitelné formě, třeba pomocí inteligentního seznamu či tabulky. Data konkretních čidel experimentu lze zobrazit i v grafické podobě podle zvoleného časového intervalu.

- Filtrace dat - zobrazená data lze snadno třídit a filtrovat pomocí aplikace filtru, který dává uživateli možnost filtrovat a třídit data podle jednoho parametru neboli skupiny parametru. Po výběru čidel musí aplikace poskytnout možnost sledování hodnoty jednotlivého čidla a filtrace podle časového intervalu, a taky oznámit uživatele, vyhovuje-li aktuální hodnota čidla podmínkám filtru.

- Export dat - data seznamu čidel pro příslušný experiment lze exportovat do různých typů souboru. Grafickou reprezentaci dat lze exportovat do URL, t.j. aplikace umožní vytvořit odkaz na graf reprezentující podmnožinu čidel z experimentu. Tento odkaz musí obsahovat i data o tvůrci tohoto experimentu a další informace.

- Zachování stavu - systém musí pamatovat stav aplikace, informace o uživateli a další nastavení. Tyto informace musí být přístupné v libovolné části aplikace z toho důvodu, že různé části měřicího systému pracují se stejnou množinou dat.

K seznamu požadavků pro měřicí systém centra geotechniky existují i technické požadavky tykající se jenom vývojáře a správce systému. Jsou globální pro celý systém a jsou uvedené níže:

- Minimalizace dotazů na server - HTTP dotazy jsou baličky dat, pomocí kterých aplikace pracují na internetu během interakce mezi klientem a serverem. Rychlé dodání výsledků internetové aplikace uživateli výrazně zlepšuje celkový výkon. Jedním z hlavních faktorů, které zpomalují proces načítání stránky, je nadměrný počet požadavků HTTP. Z tohoto důvodu je minimalizace požadavků důležitým bodem při návrhu moderní aplikace.

- Design musí fungovat na všech typech zařízení - podle studií Digital 2020 Reports od We Are Social Inc [7]. a Hootsuite Inc. se počet uživatelů internetu po celém světě zvyšuje o 9 lidí za sekundu. To znamená, že každý den se ke globální komunitě online, která používá stolní nebo mobilní zařízení, připojí více než 800 tisíc lidí. Je zajímavé, že druhá možnost se stává každým měsícem stále populárnější. Využití smartphonů v každodenním životě roste po celém světě. Očekává se, že do roku 2024 budou tři ze čtyř používaných mobilních telefonů chytré. Podle statistik StatCounter [8] klesl podíl uživatelů desktopů na 45,66%. Z výše uvedených informací vyplývá, že aktuální webová aplikace by měla poskytovat možnost podpory nejen různých browserů, ale i mobilních zařízení.

- Architektura aplikace musí být snadno rozšiřitelná - možnost rozšíření libovolné aplikace o další funkce je problém architektury dané aplikace. Definice architektury aplikace je uvedena v standartu IEEE 1471 [6]: „Architektura je základní organizace systému, která popisuje vztahy mezi komponentami tohoto systému (a vnějším prostředím) a také definuje zásady jeho návrhu a vývoje. Složitost aplikace zpravidla roste mnohem rychleji než její velikost. A pokud se o to nestará předem, pak docela rychle nastane okamžik, kdy bude obtížné jej ovládat. Správná architektura šetří spoustu času a úsilí.

- Kompatibilita s JavaScript knihovnou JQuery - JQuery je populární knihovna JavaScript. Byla vytvořena v roce 2006 s cílem usnadnit vývojářům používání JavaScriptu na webových stránkách se zaměřením na interakci JavaScriptu a HTML. Knihovna jQuery může snadno manipulovat s jakýmkoliv prvkem webové stránky, přistupovat k atributům a obsahu prvků webové stranky. Příčinou zahrnutí tohoto požadavku je, že aktuální měřicí systém je vytvořen pomocí této knihovny, a programátor by musel mít možnost použit existující moduly aplikace.

Požadavky popsané v této kapitole jsou základem návrhu vlastního řešení, tedy návrhu zlepšující verze měřícího systému.

2.4 Možnost vylepšení aktuální aplikace

Tato podkapitola se zabývá problematikou vylepšení aplikace měřícího systému na základě provedené analýzy. Díky této analýze bylo zjištěno, že nejlepším

řešením problému zlepšení měřicího systému je navržení nové verze aplikace s využitím moderních technologií vývoje webových aplikací.

2.4.1 Možnost vylepšení datového úložiště

Na základě provedené rešerše 2.2.1 bylo rozhodnuto, že použitý model relační databáze není optimálním řešením pro uchovávaní velkého počtu nestrukturovaných dat. Jakékoliv změny ve struktuře velké a načtené databáze jsou poměrně nákladné kvůli složitosti procesu migrace dat. Proto se nejprve musely zkusit možnosti optimalizace, které neovlivňují strukturu dat, ale jsou omezeny na kódu a SQL dotazech. S aktuální databází pracují i jiné softwarové systémy, úprava kódu kterých potřebuje dost času vývojáře. Změna databáze není v tomto případě kriticky potřebná. Z toho vyplývá, že nejoptimálnějším řešením problémy vylepšení datového úložiště je jeho optimalizace pro aktuální potřeby.

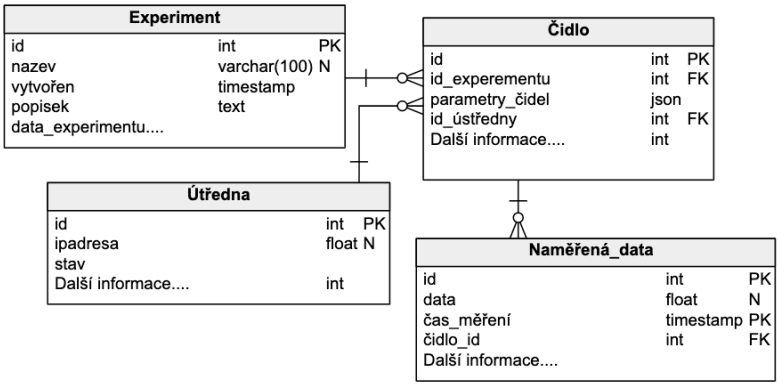
Prvním bodem optimalizace relačních databází v době vývoje architektury je jejích normalizace [5]. Pokyny pro správný návrh relačních databází jsou uvedeny v relačním datovém modelu. Jsou shromažďovány do 5 skupin, které se nazývají normální formy. První normální forma představuje nejnižší úroveň normalizace databáze, pátá představuje nejvyšší úroveň normalizace.

Normální formy jsou doporučení pro návrh databází. Při návrhu databází nemusí vývojář dodržovat všechny pět normální formuláře. Doporučováno je však [4] normalizovat databáze do určité míry, protože tento proces má řadu významných výhod:

- V normalizované struktuře databáze se dá vytvářet složité vzorky dat pomocí relativně jednoduchých dotazů SQL.
- Integrita dat. Normalizovaná databáze umožňuje spolehlivé ukládání dat.
- Normalizace zabraňuje redundanci uložených dat. Data jsou vždy uložena pouze v jednom konkrétním místě, což usnadňuje procesy vkládání, aktualizace a mazání dat. Z tohoto pravidla existuje výjimka. Samotné klíče jsou uloženy do několika míst, protože jsou zkopírovány jako cizí klíče do jiných tabulek.
- Škálovatelnost je schopnost systému vyrovnat se s budoucím růstem. Pro databáze to znamená, že by měly být schopné fungovat rychle, když se zvyšuje počet uživatelů a množství dat. Škálovatelnost je velmi důležitá charakteristika jakéhokoliv databázového modelu.

Na základě výše popsaných výhod normalizovaných relačních databází byla provedena částečná normalizace aktuálního databázového modelu. Výsledkem je relační diagram 2.9.

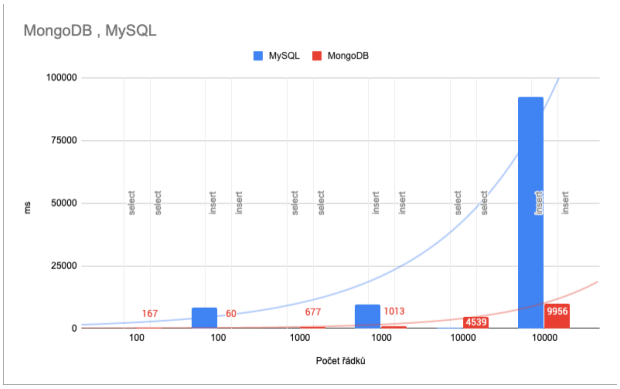
Relační diagram 2.9 popisuje jednotlivé relace mezi objekty. Podle tohoto diagramu lze říct, že jednotlivé čidlo může mít pravě jednu ústřednu, ke které



Obrázek 2.9: Příklad normalizace modelu aktuální databáze

je připojeno, a jeden přiřazený experiment, takže každé čidlo může mít celou řadu naměřených hodnot, stejný experiment a ústředna mohou mít množinu přiřazených čidel.

Popsaný diagram je součástí řešení problému optimalizace aktuální databáze. Avšak bylo rozhodnuto probrat situace, když by šlo použit nerelační databázi. Důvodem daného rozhodnutí bylo porovnaní relační MySQL a nerelační databáze MongoDB. Výsledky jsou uvedeny na obrázku níže 2.10. Pro provedení porovnaní byla použita tabulka uživatelů, která je zobrazena na obrázku 2.2.



Obrázek 2.10: Porovnaní datových úložišť

Z provedeného testu je vidět, že rychlejší datová záchranka je MongoDB, ale jenom v případě velkého počtu dotazů. Při využití malého počtu dotazů MySql je pomalejší jenom během vkládání prvků. Takže by mělo být bráno v úvahu, že výsledky budou různé na různých typech zařízení.

Druhou variantou zlepšení aktuálního datového úložiště je využití No-SQL databáze MongoDB, která je poměrně rychlá na základě výsledků testu rychlostí a umožňuje zachovat data ve tvaru dokumentů, navíc je chrání ve formátu JSON, což je vhodné pro práci s nestrukturovanými daty ze senzorů.

Třetí a nejtěžší z pohledu ztráty času na vývoj je možnost zlepšení aktuálního datového modelu použitím dvojici technologií: Redis a libovolnou relační databázi. Redis je nerelační, vysoce výkonná datová schránka. Redis ukládá všechna data do operační paměti, přístup k datům se provádí klíčem. Volitelně lze kopii dat uložit na disk. Tento přístup poskytuje výkon desetkrát větší než výkon relačních databází. Z toho důvodu Redis je dost výhodně používat pro rychle se měnící aktivně požadovaná data jako jsou data ze senzorů. Při použití Redisu jako mezipaměť, kterou lze znovu sestavit podle dat ze záložní databáze, lze dát aplikaci možnost zobrazovat data senzoru v reálném čase.

2.4.2 Možnost vylepšení serverové části

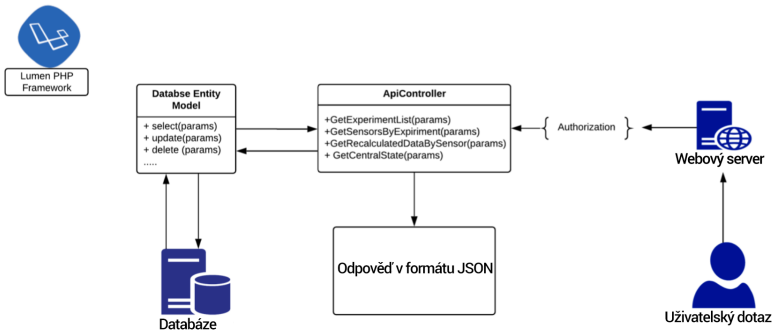
Hlavním nedostatkem aktuálního kódu serverové části je architektura, která nedává možnost rozšíření a jednoduché správy aplikace. Řešením tohoto problému je využití frameworku, který bude nabízet dobrou architekturu kódu a nebude příliš rozsáhlým na straně potřebné funkcionality.

Kvůli tomu, že množina PHP frameworků je poměrně velká, byl sestaven seznam požadavků pro toto řešení:

- Nemusí mít velký objem
- Je možná práce s databází s využitím ORM
- Možnost rozšíření o další funkcionalitu

Výše uvedeným požadavkům odpovídájí jenom tři frameworky. Jsou to: Fat-Free Framework, Lumen a Slim. Nevýhodnějším výběrem tady bude framework Lumen, protože dědí od významného frameworku Laravel a dá se jej rozšířit, což bude výhodou při možném růstu aplikace. Tento framework má velký počet uživatelů vývojářů. Nachází se v každodenním rozvoji. Na rozdíl od mnoha jiných microframeworků Lumen umožňuje používat funkce velkých analogů, jako jsou směrování, injekce závislostí, výměnný ORM, migrace, fronty a dokonce i naplánované příkazy. Díky použití jedinečného procesu inicializace je Lumen schopný poskytnout spolehlivou sadu funkcí a současně poskytovat neuvěřitelně vysoký výkon, což znamená ideální řešení pro webové aplikace. Za podmínky využití frameworku Lumen architektura serverové častí aplikace může vypadat následujícím způsobem.

Tento model představuje tzv. návrhový vzor Model-View-Controller (MVC). Umožňuje rozdělit aplikační logiku, data a uživatelské rozhraní. Návrhový vzor MVC je vzor softwarové architektury vytvořený na základě ukládání reprezentace dat odděleně od metod, které s daty interagují. Návrhový vzor MVC byl původně vyvinut pro osobní počítače. Weboví vývojáři jej začali široce používat díky přesné diferenciaci úkolů a možnosti opětovného použití kódu. Schéma stimuluje vývoj modulárních systémů, což umožňuje vývojářům rychle aktualizovat, přidávat nebo odebírat funkce.



Obrázek 2.11: Koncept architektury serverové části

2.4.3 Možnost vylepšení klientské části

Problematika zlepšení klientské části aplikace spočívá v návrhu nové verze uživatelského rozhraní s využitím moderních technologií a instrumentů. Jak bylo zjištěno provedenou analýzou, je lehčí vytvořit nové uživatelské rozhraní něž zlepšovat aktuální.

V dnešní době, stejně jako před 20 lety, existuje jenom jeden programovací jazyk, který může být spouštěn v browseru bez jiných plaginů. Díky tomu, že JavaScript nemá ani jednoho konkurenta, existuje nekonečně hodně knihoven a nástrojů zlehčujících návrh uživatelského rozhraní. Pro výběr nástroje byla provedena rešerše trhu JavaScript frameworků a vytvořena tabulka třech leaderů.

	Angular	React	Vue
Složitost	velká	střední	malá
Rozměr	143Kb	31.8Kb	20.9Kb
Github hodnocení (hvězdy)	61469	149345	164713
Datum výtvoření	2010	2013	2014
Používa	Google	Facebook, Uber	Alibaba, GitLab

Obrázek 2.12: Porovnání JavaScript Frameworků

Angular je velký JavaScript MVVM framework, založený v roce 2010. Je skvělý pro vytváření interaktivních a velkých webových aplikací. Množina různých struktur (injekční stříkačky, komponenty, potrubí, moduly atd.) komplikuje začátek práce ve srovnání s React a Vue, které mají pouze „komponentu“.

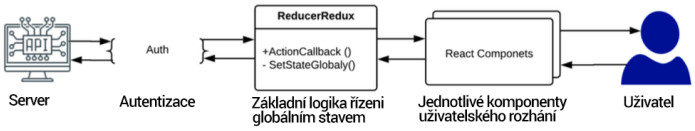
React je knihovna JavaScriptu vyvinutá společností Facebook v roce 2013. Hodí se pro vytváření moderních jednostránkových aplikací libovolné velikosti a měřítka. Díky jednoduchému designu, šablonám a velmi podrobné dokumentaci se lze snadno naučit JSX (syntaxe podobné HTML). Vývojář se může věnovat více času psaní moderního JavaScriptu a méně se obávat, že by byl kód specifický pro daný rámec.

Vue.js je framework JavaScriptu založený v roce 2013, který je ideální pro vytváření vysoce přizpůsobitelných uživatelských rozhraní a komplexních jed-

nostránkových aplikací. Vue.js má stále poměrně malý podíl na trhu ve srovnání s React nebo Angular, což znamená, že sdílení znalostí v tomto prostředí není stále jednoznačné.

Z výše uvedeného bylo řešeno, že React je nejvhodnějším frameworkem pro realizaci uživatelského rozhraní měřicího systému pro experimenty centra experimentální geotechniky. Hlavními výhodami využití knihovny React jsou dostatečná ekonomie času na vývoj a plná kompatibilita s knihovnou JQuery, pomocí které je realizována většina základních komponentů aktuální aplikace.

Na základě analýzy JavaScript technologií byl navržen koncept architektury klientské části aplikace, který řeší problém architektury a opětovného použití jednotlivých elementů uživatelského rozhraní.



Obrázek 2.13: Koncept architektury klientské části aplikace

2.4.4 Závěr

Základním cílem této rešerše bylo seznámení se s měřicím systémem pro experimenty centra experimentální geotechniky a provedení analýzy, výsledky které jsou nalezená kritická místa a chyby architektury jak serverové části, tak i uživatelského rozhraní. Dále byla provedena rešerše aktuálních nástrojů pro vývoj moderních webových aplikací. Na základě výsledků analýzy byly vybrány nejvhodnější technologie pro návrh moderního řešení této práce.

3 Použité technologie

Tato kapitola popisuje technologie vybrané na základě provedené analýzy a použité pro návrh vylepšené verze aplikace měřicího systému pro experimenty centra experimentální geotechniky.

3.1 JavaScript knihovna React

React je knihovna JavaScript pro vývoj uživatelských rozhraní. React byl původně navržen tak, aby mohl být implementován v libovolné aplikaci postupně. Jinými slovy, vývojář může začít pomalu používat pouze ty funkce React, které v tuto chvíli potřebuje.

3.1.1 Komponenty React a virtuální DOM

Architektura knihovny React je sestavena podle navrhovacího vzoru kompozice. Kompozice je strukturní návrhový vzor, který umožňuje seskupovat mnoho objektů do stromové struktury a poté s nimi pracovat, jako by to byl jediný objekt. Základní jednotkou knihovny React je třída "Component". Každý element uživatelského rozhraní dědí od třídy "Component" a nazývá se komponentem aplikace. Celou aplikaci reprezentuje strom komponentů. Komponenty vypadají podobně funkcím JavaScript. Komponenty přijímají libovolná data (nazývaná parametry) a vracejí prvky React, které popisují, co by se mělo na obrazovce objevit.

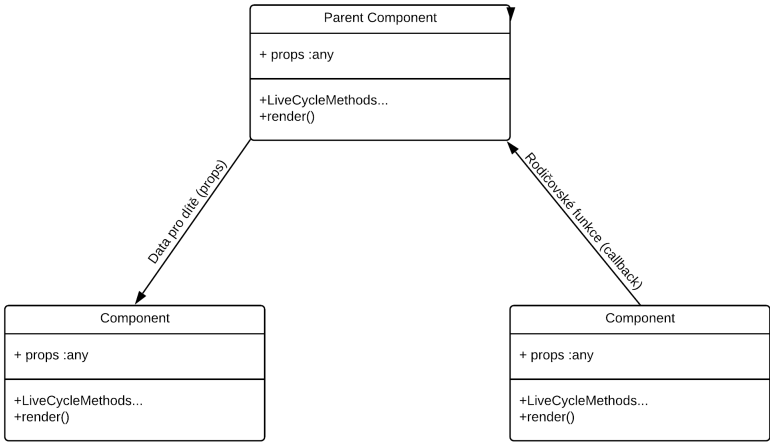
DOM je způsob, jak reprezentovat strukturální dokument pomocí objektů. Jedná se o dohodu napříč platformami a jazyky nezávislou na prezentaci a interakci s daty v HTML, XML atd. Webové prohlížeče zpracovávají komponenty DOM a dá se s nimi komunikovat pomocí JavaScriptu a CSS. Je možné pracovat s uzly dokumentu, měnit jejich data, mazat a vkládat nové uzly. V současné době je DOM API skoro multiplatformní. Hlavním problémem DOM je to, že nikdy nebyl navržen k vytvoření dynamického uživatelského rozhraní (UI). Pracuje se s ním pomocí JavaScriptu a knihoven jako jQuery, ale jejich použití neřeší problémy s výkonem.

Knihovna React řeší problém výkonu pomocí použití tzv. virtuálního DOM. Namísto přímé interakce s DOM funguje React s jeho lehkou kopií. React může na základě potřeby provést změny v kopii a poté použít změny ve skutečném DOM. V tomto případě je strom DOM porovnán s jeho virtuální

kopií, a stanoven rozdíl mezi nimi. Dále se překresluje to, co bylo změněno. Tento přístup funguje rychleji, protože nezahrnuje všechny těžké části reálného DOM.

3.1.2 Problematika předávaní dat mezi komponentami

Interakce komponent React je reprezentována ve dvou formách: tok dat z nadřazené do podřízené komponenty a tok dat z podřízené do nadřazené. Tok od rodičů k dítěti pomocí parametru byl již popsán. Pro úspěšné dosažení přenosu dat z dítěte na rodiče se v Reactu používá funkce předání prostřednictvím parametru:



Obrázek 3.1: Předávaní dat mezi komponenty

3.1.3 Řízení stavu aplikace

Komponenty React jsou rozděleny na dvě základní skupiny: ty co mají vnitřní stav a ty co nemají. Vnitřním stavem komponenty React se nazývají data potřebná pro práci komponenty. Může to být například stav tlačítka, je-li zapnuté či ne. React dává možnost pracovat se stavem komponenty. Podřízená komponenta se může pomocí parametru dozvědět o změně stavu rodiče. V případě, když komponenty nemají společného předka, nemůžou mezi sebou posílat data. Řešením je ukládat aplikační data v nějakém centralizovaném úložišti, aby požadované komponenty měly k nim přístup. Na to se využívá knihovna Redux. Redux je stavový manažer. Nejčastěji se používá spolu s React, ale jeho schopnosti nejsou omezeny pouze na tuto knihovnu. V Reduxu je obecný stav aplikace reprezentován jediným objektem JavaScript - stavem nebo stavovým stromem. Neměnný stavový strom je pouze pro čtení, nelze nic změnit přímo. Změny jsou možné pouze při odesílaní akce. Akce je objekt JavaScriptu, který stručně popisuje změnu stavu. Jediným požadavkem na objekt akce je přítomnost vlastnosti type, jejíž hodnota je obvykle řetězec. Při spuštění akce se změní

stav aplikace. Reduktor je čistá funkce, která vypočítá další stav komponenty na základě předchozího stavu a použité akce. Čistá funkce funguje bez ohledu na stav programu a produkuje výstupní hodnotu, přijímá vstup a nemění jeho strukturu. Reduktor vrací zcela nový objekt stavu, který nahrazuje předchozí stav. Knihovna Redux v tomto případě porovnává výsledek reduktoru a aktuální stav aplikace a mění jenom ty složky stavu, které se nerovnají předchozím.

3.2 JavaScript knihovna Material UI

Material UI je knihovna umožňující vytvářet aplikace ve stylu Google Material Design pomocí komponent React. Zjednodušuje vývoj webových aplikací, interaktivních uživatelských rozhraní a jednostránkových aplikací.

V předchozí kapitole byl popsán význam normalizačních stylů 2.2.3. Material UI obsahuje sadu stylů normalize.css pro normalizaci prvků HTML. Material UI má vlastní verzi normalize.css - CssBaseline, která poskytuje elegantní, konzistentní a jednoduchou sadu základních stylů. CSSBaseline provádí následující úpravy stylu:

- Úprava odsazení ve všech prohlížečích
- Nastavuje výchozí barvu pozadí
- Umožňuje vyhlazování písma pro lepší vykreslení písma Roboto základního pro Material UI
- Základní velikost písma není uvedena v HTML, ale předpokládá se, že je 16px (výchozí velikost prohlížeče).

Material UI obsahuje velké množství komponent uživatelského rozhraní přispívající k vytvoření aplikace React s využitím designu Material Design. Níže jsou uvedeny příklady některých z nich:

- Ikonky
- tlačítka
- navigace
- seznamy
- modální komponenty
- tabulky.

Tato knihovna ne jenom dává sadu nástrojů pro vytvoření vlastního webu. Nabízí seznam šablon, ze kterých si může vývojář vybrat. To může být velmi výhodné pro flexibilitu designu uživatelského rozhraní. Některé z nejlepších šablon bohužel nejsou zdarma.

3.3 NodeJS, NPM a React Scripts

Vysvětlení toho, co je Node.js, pomůže lépe porozumět NPM. Node.js je interpretem JavaScriptu. Samotná aplikace Node.js je aplikací C ++, která přijímá kód JavaScript na vstupu a spouští jej. Balíček v Node.js odkazuje na jeden nebo více souborů JavaScript, které jsou nějakou knihovnou nebo nástrojem.

NPM je standardní správce balíků, je automaticky nainstalován spolu s Node.js. Používá se pro stahování balíčků ze serveru NPM. Soubor package.json je konfiguračním souborem obsahujícím informace o aplikaci: název, verze, závislosti a podobně. Každý adresář, který tento soubor má, je interpretován jako balíček Node.js.

Před vydáním aplikace softwarového balíku React Scripts museli vývojáři při každém spuštění nové aplikace zkopírovat konfiguraci a další soubory z předchozích projektů nebo vše nakonfigurovat ručně, a to trvalo hodně času.

Naštěstí dnes existuje softwarový balík React Scripts. Praktický modul jde dohromady s vynikající konfigurací a sadou skriptů. Usnadňuje vytváření a podporu aplikací React. Základními příkazy balíku React Scripts jsou:

```
"scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject"
 }
```

Obrázek 3.2: Seznam základních příkazů

- Start - spouštějící skript umožňuje spustit server Node.js na adrese http: // localhost: 3000
- Test - React Scripts používá knihovnu Jest pro testy, příkaz umožňuje spustit test v režimu interaktivního zobrazení.
- Build - používá se pro zjištění, zda je dokončený projekt integrován, minimalizován a optimalizován do jediného malého souboru.
- Eject - zkopíruje konfigurační soubory a přechodné závislosti (např. Webpack, Babel atd.) jako závislosti do souboru package.json.

Softwarový balík React Scripts je k dipozici s některými příkazy zahrnujícími flexibilní parametry, které umožňují přizpůsobit skripty jedinečným potřebám projektu.

3.4 Framework Lumen a manager balíčků Composer

Lumen je PHP framework s otevřeným zdrojovým kódem sestavený na základě návrhového vzoru MVC [2]. Klíčové rysy práce s frameworkem Lumen jsou:

- Balíčky - umožňují vytvářet a propojovat moduly ve formátu balíků pomocí balíkového manageru Composer pro aplikace napsané v jazyku PHP.
- Logika aplikace - je striktně oddělena od ostatních částí aplikace.
- Reverzní směrování - propojuje mezi sebou vygenerované aplikací odkazy a trasy. Umožňuje uživateli změnit vygenerované trasy pomocí automatické aktualizace propojených odkazů. Při vytváření odkazů pomocí pojmenovaných tras Lumen automaticky vygeneruje konečné adresy URL.
- Automatické stažení tříd (autoloading) - mechanismus automatického načtení tříd PHP bez potřeby připojovat soubory pomocí importu. Stahování bez potřeby zabraňuje zbytečnému stahování komponent, tedy jsou načítány pouze ty, co jsou opravdu používány.
- Inverze kontroly - umožňuje přijímat instance objektů na principu zpětného řízení. Může také sloužit k vytváření a načítání objektů typu singleton.
- Migrace - systém pro správu verzí databázového modelu. Umožňuje spojit změny v kódu aplikace se změnami ve struktuře databáze, což zjednodušuje nasazení a aktualizaci aplikace.
- Testování - hraje velmi velkou roli v Lumenu, jelikož framework obsahuje vlastní knihovnu testů.

Základním nástrojem pro práci s libovolným frameworkem či balíčkem v programovacím jazyku PHP je manager balíků Composer. Pomocí Composeru lze popsat, na jakých knihovnách bude projekt záviset, a Composer nainstaluje potřebné knihovny sám. Navíc není Composer správcem balíků v klasickém pojetí. I když pracuje s entitami, které se nazývají balíčky nebo knihovny, jsou instalovány uvnitř každého projektu zvlášť, nikoliv globálně.

Instalaci frameworku Lumen lze provést pomocí balíkového manageru Composer a příkazu create-project, kde parametr ”prefer-dist” odpovídá za soubor, kam se aplikace uloží:

Obrázek 3.3: Instalace frameworku Lumen

3.4.1 Práce s databází a Eloquent ORM

Nastavení databáze pro aplikaci je uloženo v souboru .env. Je možné vytvořit konfigurační soubor config/database.php, a v něm určit všechna připojení k databázi, nebo nastavit výchozí připojení.

Hlavním prvkem práce s databází a implementace principu prvního kódu (code first) je migrace. Migrace je systém řízení verzí pro databázi. Umožňuje změnit strukturu databáze přímo z kódu a ovládat historii verzí. Migrace se vytvoří následujícím příkazem:

Obrázek 3.4: Vytvoření migrace tabulky uživatelů

Pro zahájení všech potřebných migrací existuje příkaz "php artisan migrate". Některé migrační operace jsou destruktivní, což znamená, že mohou vést ke ztrátě dat. Před jejich provedením je vyžadováno potvrzení, které slouží jako obrana proti náhodnému spuštění těchto příkazů v databázi. Pro spuštění příkazů bez potvrzení existuje parametr "force".

Systém Eloquent (ORM) je krásná a jednoduchá implementace návrhového vzoru ActiveRecord v Lumenu pro práci s databázemi. Každá tabulka má odpovídající třídu modelu, která se používá pro práci s touto tabulkou. Modely umožňují vyhledávat data z tabulek a vkládat nové záznamy. Pomocí následujícího příkazu lze vytvořit Model databázové entity. V případě potřeby migrace se musí použit doplněk -m.

Obrázek 3.5: Vytvoření modelu

Lumen poskytuje pohodlné a expresivní rozhraní pro vytváření a provádění databázových dotazů. Toto rozhraní může být použito k provádění většiny typů operací a pracuje se všemi podporovanými databázemi. Návrhář dotazů Lumenu používá vazby parametrů PDO pro ochranu aplikace před útokem SQL injekce. To znamená, že před předáním dotazu není třeba ověřovat vstup.

Obrázek 3.6: Příklad dotazovaní

3.4.2 REST API ve frameworku Lumen

REST je snadný způsob, jak uspořádat interakci mezi nezávislými systémy. Je populární již od roku 2005 a inspiruje design služeb, jako je Twitter API. Vzhledem k tomu, že REST zajišťuje interakci s tak rozmanitými klienty, jako jsou mobilní telefony a jiné webové stránky, pomocí protokolu HTTP. Výsledkem je, že REST lze použít všude tam, kde je využitý HTTP.

Webová služba je aplikace, která běží na webu a je přístupná prostřednictvím protokolu HTTP. Informace jsou vyměňovány pomocí formátu XML nebo JSON. Pro každou informační jednotku jsou definovány 4 akce, jsou to GET, PUT, POST, DELETE. Z pohledu architektury REST je operace (nebo servisní volání) idempotentní, tedy mohou klienti provést stejné volání více než jednou se stejným výsledkem na serveru. Jinými slovy, vytvoření velkého počtu identických dotazů má stejný účinek jako jediný dotaz. Je zajímavé, že zatímco operace idempotent vedou ke stejnému výsledku na serveru, samotná odpověď nemusí být stejná (například se stav prostředku může mezi požadavky měnit). Metody PUT a DELETE jsou podle definice idempotentní. Existuje však jedna výhrada s metodou DELETE. Problém spočívá v tom, že úspěšný požadavek na DELETE vrací stav 200 (OK) nebo 204 (bez obsahu), ale pro následné žádosti vždy vrací 404 (nenalezeno). Stav na serveru po každém volání DELETE je stejný, ale odpovědi jsou odlišné. Metoda GET je definována jako bezpečná. To znamená, že je určena pouze k získání informací a neměla by měnit stav serveru. Podle definice jsou bezpečné operace idempotentní, protože vedou ke stejnému výsledku na serveru. Bezpečné metody jsou implementovány jako operace pouze pro čtení. Zabezpečení však neznamená, že by server měl vždy vracet stejný výsledek.

Framework Lumen je pak ideální pro tvorbu servisů postavených na architektuře REST díky možnosti flexibilního či dynamického vytváření koncových bodů REST rozhraní.

```
$router->get('/commutationStation/', [
    'recalculated',
    'uses' => 'ExperimentController@getCommutationStationState'
]);

$router->post('/commutationStation/', [
    'recalculated',
    'uses' => 'ExperimentController@getCommutationStationState'
]);
$router->put('/commutationStation/', [
    'recalculated',
    'uses' => 'ExperimentController@getCommutationStationState'
]);

$router->delete('/commutationStation/', [
    'recalculated',
    'uses' => 'ExperimentController@getCommutationStationState'
]);
```

Obrázek 3.7: Vytváření koncových bodů REST rozhraní

3.4.3 JWT ve frameworku Lumen

JWT je otevřený standard (RFC 7519) [10] pro vytváření přístupových tokenů založených na formátu JSON. Obvykle se používá pro přenos dat pro ověření v aplikacích klient-server. Tokeny jsou vytvářeny serverem, podepsány tajným klíčem a přenášeny na klienta, který tento token dále používá k potvrzení své identity. Poté jak uživatel prošel autorizací v systému a uvedl své uživatelské jméno a heslo, systém mu dá 2 tokeny: přístupový a obnovovací. Obnovovací token je potřebný pro zjištění nové dvojici tokenů po vypršení přístupového tokenu. Když uživatel chce přijímat data ze serveru, například jeho profil, spolu s požadavkem předá přístupový token. Server zkontroluje jeho obdržení, zda je platný (více k tomu níže), odečte užitečná data a může tedy uživatele identifikovat.

Framework Lumen umožňuje rozdělit jednotlivé adresy URL, tzv. routy, do speciálních skupin, pak každé skupině lze přiřadit speciální třídu "Middleware", kód které bude volán po každém dotazu na skupinu adres. Tímto způsobem lze realizovat ověření přístupového tokenu pro identifikaci uživatele. Obvykle se generace tokenu nejprve provádí v uživatelském kontroléru, ve kterém musí byt realizována i jiná logika tykající se uživatele.

3.5 MySQL

Z výsledků provedené rešerše relačních databází, jako je například MySQL, bylo řešeno, že nejsou vhodnou variantou pro aplikaci měřicího systému, ale zadaní této bakalářské práce nezahrnuje změny datového modelu aplikace centra experimentální geotechniky, jelikož na něm běží i jiné služby. V rámci této práce bylo proto navrhnuto datové úložiště v MySQL.

MySQL je jedním z nejpopulárnějších a nejrozšířenějších systémů správy databází v internetu. Není určen pro práci s velkým objemem informací, ale jeho použití je ideální pro malé a velké internetové stránky. MySQL je velmi rychlý, spolehlivý a snadno použitelný. MySQL má řadu praktických funkcí vyvinutých za úzkého kontaktu s uživateli. Server MySQL byl původně navržen pro správu velkých databází, aby poskytoval vyšší rychlost ve srovnání se současnými prostředky v té době. Tento server se již několik let úspěšně používá v průmyslových podmínkách s vysokými požadavky. Přestože se MySQL neustále zlepšuje, dnes poskytuje širokou škálu užitečných funkcí. Díky své dostupnosti, rychlosti a bezpečnosti je MySQL velmi vhodný pro přístup k databázím přes internet. MySQL je systém typu klient-server obsahující vícevláknový server SQL, který poskytuje podporu pro různé databázové počítače, několik různých klientských programů a knihoven, nástroje pro správu a širokou škálu programovacích rozhraní (API). K dispozici je také velké množství softwarů pro systém MySQL vyvinutých vývojáři třetích stran.

4 Implementace

Tato kapitola přesně popisuje návrh zlepšení verze měřicího systému pro experimenty centra experimentální geotechniky. Na základě výsledků dosažených v teoretické části této práce byly navrženy koncepty serverové a klientské částí aplikace, které komunikují mezi sebou pomocí architektury REST. Data se posílají ve formátu JSON. Směrovaní mezi stránkami aplikace je realizováno pomocí JavaScriptu bez opětovného načítaní celé stránky, což dělá práci s aplikací pohodlnější pro uživatele.

4.1 Klient

Klientská část aplikace je navržena s pomocí knihovny React a pomocného balíku React Scripts. Pro správu globálního stavu aplikace byla využita knihovna Redux. Hlavním designem aplikace byl vybrán již předem vyzkoušený design od firmy Google, tzv. Material Design. Pro jeho implementaci v tomto projektu byla na základě analýzy zvolena knihovna komponent Material UI. Každá komponenta klientské části aplikace má proto nezávislé styly a muže být použita nezavile.

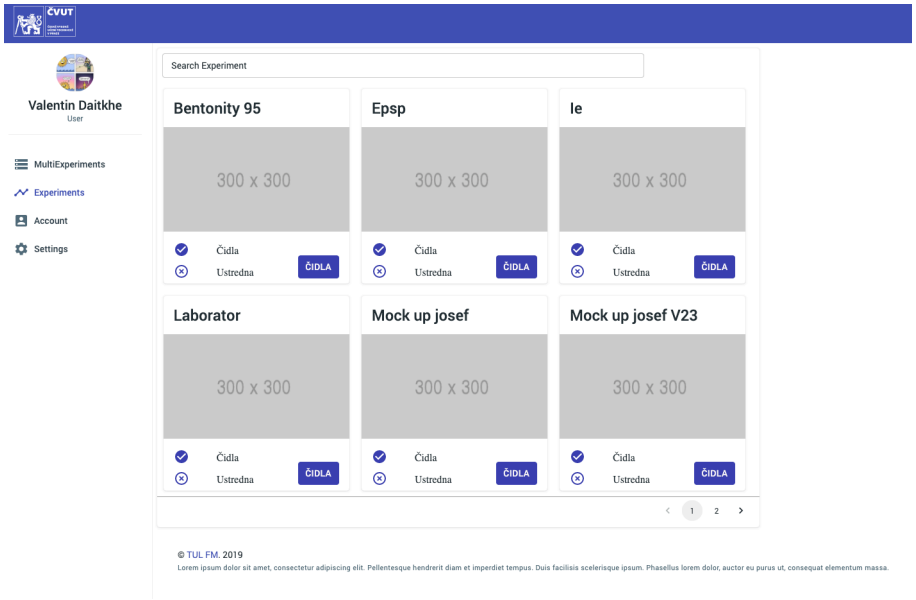
4.1.1 Seznam experimentů

Seznam experimentů 4.1 je nezavilá komponenta React. Bere za vstup počet zobrazovaných experimentů na jedné stránce. Jestliže vstup není zadaný, počet experimentů na stránce se vypočítá na základě šířky uživatelské obrazovky. Komponenta obsahuje funkci pro vyhledávání jednotlivých experimentů podle jejich názvů.

Komponenta Seznam experimentů závisí na globálním stavu Redux, který je použit pro asynchronní dotazovaní serverů a zobrazení stavů stažených dat.

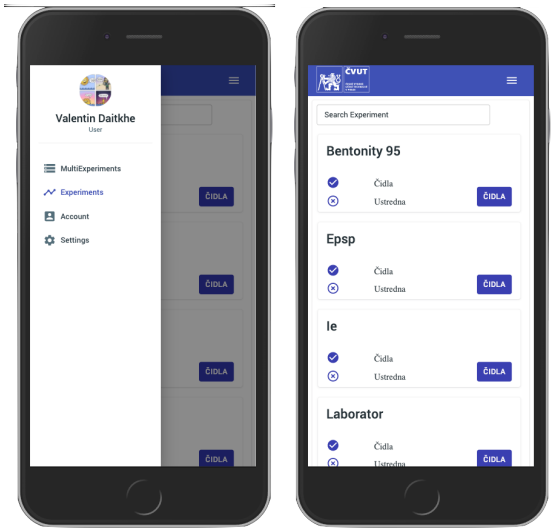
Dotazování se provádí s možností stránkovaní seznamů, t.j. server vygeneruje data pro každou stránku seznamu. Tato funkčnost byla implementována s ohledem na velký počet experimentů. Vyhledávání se provádí na základě pole názvů experimentů.

Jednotlivé experimenty jsou implementovány pomocí komponenty knihovny Material Design ”Card” obsahující informace o aktuálním stavu experimentu i čidel a odkaz na seznam čidel.



Obrázek 4.1: Seznam experimentů

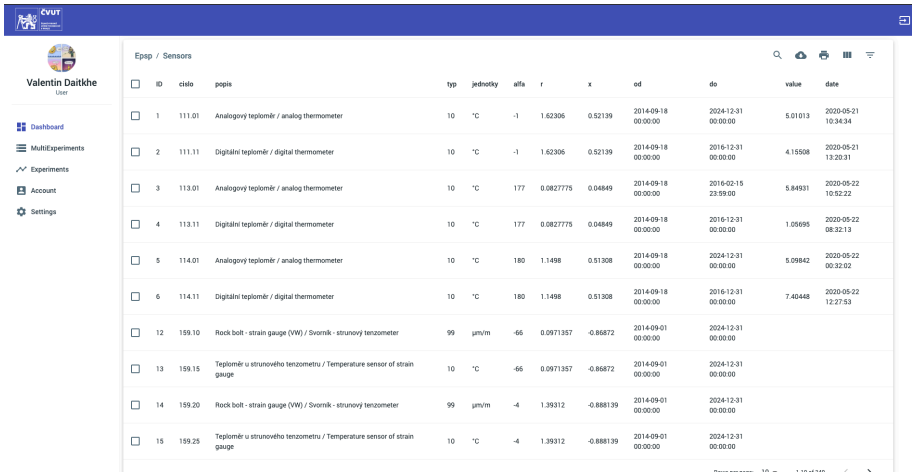
Komponenta Seznam experimentů má možnost zobrazení na jiných typech zařízení, například na smartphonech 4.2. Mobilní verze komponenty je samostatnou třídou, protože nejde umístit originální komponentu do malé obrazovky mobilu. Mobilní verze komponenty se zobrazí jenom v případě, když šířka obrazovky klesne pod 500 pixelů, změna šířky se vypočítá automaticky. Tato komponenta poskytuje všechny funkce verze pro desktop jako jsou vyhledávaní jednotlivých experimentů a zobrazení jejich stavů. Stránkovaní seznamu experimentů se v tomto případě provádí pomocí rolovaní dolu pro lepší použitelnost.



Obrázek 4.2: Mobilní verze seznamu experimentů

4.1.2 Seznam čidel

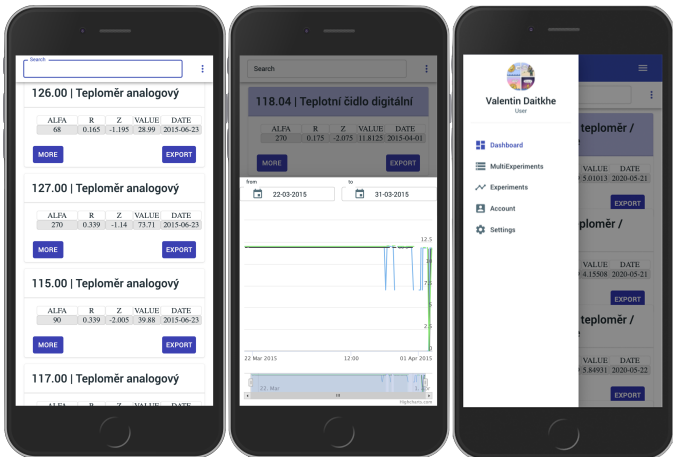
Seznam čidel je komponenta zobrazující tabulku čidel. Je postavena na Material-UI. Tato komponenta poskytuje funkce jako filtrování, změna velikosti zobrazení neboli skrytí vybraných sloupců, vyhledávání a export do CSV souboru, stažení, tisk, stránkování a třídění. Mobilní verze komponenty poskytuje



Obrázek 4.3: Seznam čidel

všechny funkce rodičovské komponenty, kromě filtrace čidel podle příslušného parametru. Kvůli tomu, že komponenta je realizována ve formě seznamu, tato funkcionalita by byla zbytečná, a navíc ji nahrazuje funkce vyhledávaní podle libovolného parametru.

Nejtěžším v návrhu této komponenty byl problém zatížení browseru při zobrazení velkého počtu čidel v mobilní verzi. Tento problém byl vyřešen tak, že komponenta spočítá plochu v reálném čase, bude tedy viditelná pro uživatele, a vykreslí jenom ta čidla, co do této plochy vejdou.



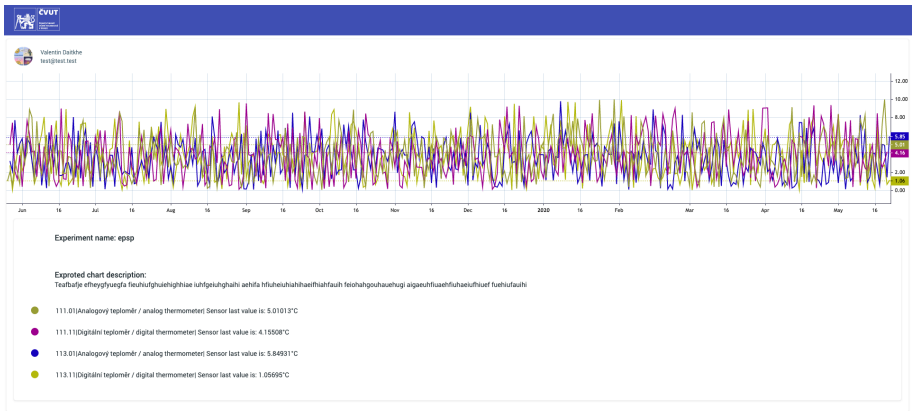
Obrázek 4.4: Mobilní verze seznamu čidel

4.1.3 Grafické zobrazení dat a export

Komponenta Grafické zobrazení dat a exportu je realizována pomocí dialogu. Dialog je typ modálního okna, které se objeví nad aplikací a poskytuje důležité informace a řešení. Dialogy deaktivují všechny funkce aplikace. Když se objeví, zůstanou na obrazovce, dokud nejsou potvrzeny, odmítnuty nebo dokud nejsou provedeny potřebné kroky pro odstranění. Tato komponenta je součástí seznamu čidel. Po výběru potřebných čidel z tabulky se v záhlaví seznamu čidel objeví tlačítko pro aktivaci této komponenty. V mobilní verzi se aktivace provádí příslušným bodem horního menu.

Obrázek 4.5: Grafické zobrazení dat

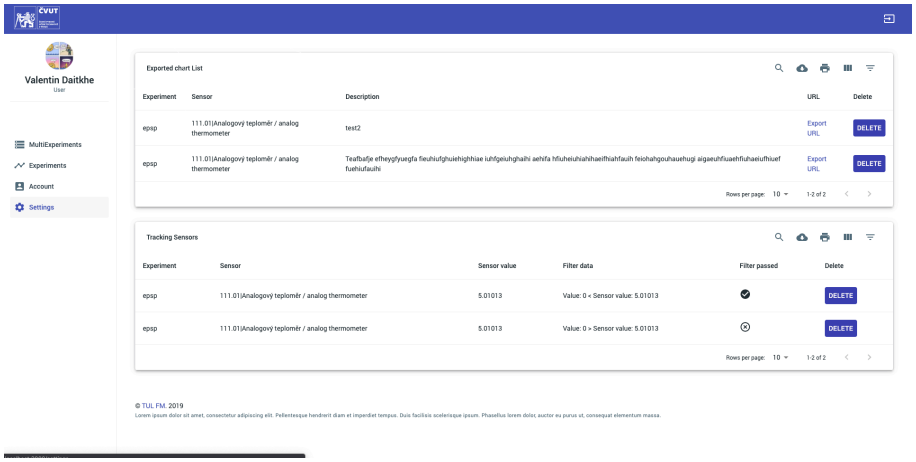
Komponenta bere za vstup vybraná čidla ze seznamu čidel a pak je přiřazuje do tabulky. Grafické zobrazení dat je realizováno pomocí technologie canvas. Graf se zobrazuje dole, a samotný graf, stejně jako tabulku vybraných čidel, lze skrýt stisknutím záhlaví pro lepší průhlednost.



Obrázek 4.6: Exportovaný graf

V mobilní verzi musí uživatel pro vykreslení grafu zmáčknout tlačítko. Graf zobrazuje naměřená data čidel podle vybraného časového intervalu s možností je exportovat. Modul poskytuje dvě možnosti exportu. První možností je jednoduchý export aktuálního stavu grafu do obrázku a následné stažení. Druhá možnost spočívá v ukládaní stavu grafu do databáze a vytvoření nezávislého odkazu na tento graf s příslušnými informacemi zadanými uživatelem, jako jsou popis exportu a kontakt na uživatele, který daný odkaz vytvořil 4.6.
Komponenta Grafické zobrazení dat obsahuje také aplikaci filtrů. Lze je nastavit pro vybraná data čidel z tabulky v horní části okna komponenty 4.5. Tato aplikace umožňuje uživateli nastavit speciální filtr, který záleží na aktuální hodnotě přiřazeného čidla. Po nastavení se filtr ukládá do databáze. Pro jednotlivé čidlo lze nastavit více filtrů, a tím se řeší problém sledovaní intervalu hodnot, kterého přiřazené čidlo muže dosáhnout.

Výsledky jednotlivých filtrů, stejně jako seznam odkazů exportovaných grafů, se zobrazují v jiné části aplikace v záložce ”settings” 4.7.



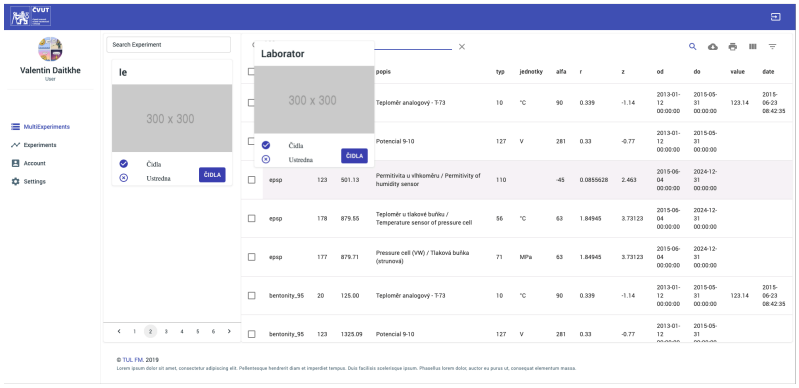
Obrázek 4.7: Zobrazení exportovaných grafů a vytvořených filtrů

Tyto seznamy jsou realizovány děděním od komponenty Seznam čidel, proto seznamy poskytují všechny funkce pro filtraci, vyhledávání a export jednotlivých záznamů. Mimo to jsou tyto komponenty rozšířeny o funkcionalitu smazaní záznamů po jednotlivých řádcích pro možnost snadného zrušení nepotřebných exportů a filtrů. Funkcionalita vymazaní celého seznamu není implementována z důvodu ochrany dat proti náhodnému vymazaní.

4.1.4 Výběr a porovnaní dat z různých experimentů

Komponenta pro výběr a porovnaní dat z různých experimentů byla v aplikaci implementována proto, že všechny realizované komponenty se mohou využívat nezavile. Tato komponenta umožňuje uživateli všechno to, co umožňuje skupina komponent pro zobrazení informací o čidlech experimentu. Komponenta pro grafické zobrazení, sledovaní a export se tedy liší jenom tím, že má možnost porovnaní a práce s daty z různých experimentů.

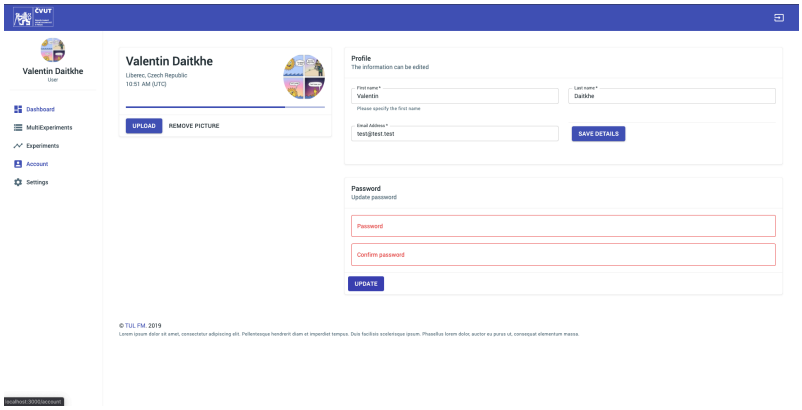
Pro interakci s uživateli využívá komponenta technologii dag & drop. Zobrazuje data různých experimentů pomocí komponenty pro grafické zobrazení dat a export 4.8.



Obrázek 4.8: Výběr a porovnání dat z různých experimentů

4.1.5 Profil uživatele

Profil uživatele reprezentuje množina nezávislých komponent propojených jak se serverem, tak i s globálním stavem, jenž chrání uživatelská data a dává možnost přístupu k nim z různých částí aplikace 4.9. Tyto komponenty jsou určeny pro snadnou změnu uživatelských údajů, například přihlašovacích údajů a dalších.

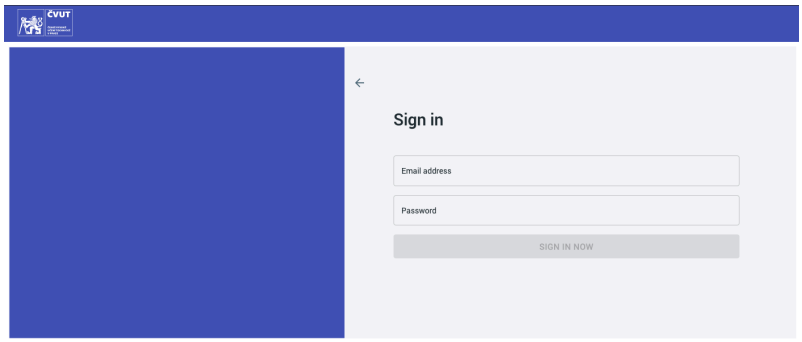


Obrázek 4.9: Profil uživatele

4.1.6 Přihlášení uživatelů

První stránka, kterou uživatel vidí, je stránka přihlášení uživatele, aplikace se nepřesměruje dále bez vyplnění uživatelských údajů. Komponenta ověřuje

správnost uvedených údajů pro minimalizaci dotazů na server. Pokud nemají přihlašovací údaje správnou formu nebo uživatel neexistuje, v odpovědi serveru přijde kód chyby, a na jeho základě komponenta vygeneruje oznámení pro uživatele **??**.



Obrázek 4.10: Přihlášení uživatele

4.2 Server

Serverová část aplikace je implementována pomocí PHP frameworku Lumen. Jeho základní možnosti jsou uvedeny v kapitolách 3.4, 3.4.1, 3.4.2. Architektura serverové části je založena na architektuře REST API 3.4.2. Z toho vyplývá docela jednoduchá možnost rozšíření o další funkcionality. Základními prvky serverové části aplikace jsou tzv. kontroléry:

- Třída AuthController - zodpovídá za aktivity registrace, přihlášení a autentizace uživatele. Pro autentizaci uživatelů se používá standart JWT popsaný v kapitole 3.4.3.
- Třída ExperimentController - pracuje s daty experimentů a čidly. Všechny metody pro obdržení libovolných informací tykajících se dat měřícího systému jsou implementovány v této třídě.
- Třída UserController - implementuje jednotlivé požadavky uživatele o obdržení informací profilu, uložení grafu a filtru hodnoty čidla, mazaní a další.

Za vstup jednotlivých metod kontroléru reprezentujících jednotlivé koncové body API se bere vlastní implementace třídy Request reprezentující jednotlivý dotaz klienta. Pro každou implementaci Requestu byla podle potřeby určena pravidla validace parametrů dotazu. V případě, že podmínky nejsou splněny, metoda vrací chybu. Každému kontroléru je přiřazen tzv. Middleware. Je to kód, který se spouští do okamžiku obdržení kontrolérem jednotlivých dotazů, a to je nutné pro ověření tokenů 3.4.3.

5 Závěr

Téma, které bylo zvoleno, se v dnešní době považuje za relevantní, protože miliony lidí používají internetové aplikace. V průběhu vypracovaní této práce se autor seznámil s celou řadou moderních technologií.

Počet uživatelů internetu roste každodenně a nepřetržitě. Webové aplikace značně zjednodušují mnoho procesů v našem životě a rozšiřují naše možnosti. Cíle této práce byly analýza a zlepšení stávajícího webového portálu pro experimenty centra geotechniky. Při vykonání zadání byly provedeny následující kroky:

- Průzkum a analýza existující aplikace
- Specifikace požadavků pro měřící systém
- Analýza aktuálních nástrojů a technologií pro vývoj webových aplikací
- Softwarová implementace moderní verze měřícího systému
- Psaní dokumentace pro závěrečnou práci.

Pro dosažení cílů této práce byla provedena hluboká analýza existujícího systému, na základě které byla nalezena kritická místa stávající aplikace. Byly zpřesněny požadavky na základní funkce aplikace, k tomuto účelu byl zpracován UML diagram a podrobně popsány případy použití.

Praktickým výsledkem rešerše této práce je návrh jak serverové, tak i klientské části aplikace, který je založen na moderních technologiích. Webová aplikace byla vyvinuta s ohledem na všechny požadavky, má snadno použitelné a intuitivně pohodlné rozhraní.

Dalším krokem rozvinutí tohoto tématu by mohla být budoucí optimalizace datového úložiště, jejíž problematika je popsána v kapitole 2.2.1. Aktuální architektura aplikace umožňuje jednoduché přidaní dalších funkcí, jako jsou například vizualizace dat senzoru v reálném čase a 3D vizualizace experimentů. V rámci použité knihovny React existuje velký doplněk pro tvorbu interaktivních 3 aplikace se nazývá React360.

Literatura

[1] Matera M., Rizzo F., Carughi G.T. (2006) Web Usability: Principles and Evaluation Methods. In: Mendes E., Mosley N. (eds) Web Engineering. Springer, Berlin, Heidelberg

[2] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley, c2003. The Addison-Wesley Signature Series. ISBN 0-321--12742-0.)

[3] Gollmann D. (2011) Problems with Same Origin Policy (Transcript of Discussion). In: Christianson B., Malcolm J.A., Matyas V., Roe M. (eds) Security Protocols XVI. Security Protocols 2008. Lecture Notes in Computer Science, vol 6615. Springer, Berlin, Heidelberg

[4] HAMER, Malcolm. RELATIONAL DATABASE PRACTICES: BRIDGING THE GAP BETWEEN THE THEORY OF DATABASE DESIGN AND REAL-WORLD PRACTICES. 2017. ISBN 978-0998964607.

[5] Normalizace databáze. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-[cit. 2020-05-13]. Dostupné z: **https://cs.wikipedia.org/w/index.php?title=Normalizace_datab%C3%A1ze&oldid=18093305**

[6] IEEE 1471. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 27 May 2020 22:38 UTC [cit. 2020-05-28]. Dostupné z: **https://en.wikipedia.org/w/index.php?title=IEEE_1471&oldid=860662513**

[7] DIGITAL 2020: GLOBAL DIGITAL OVERVIEW [online]. DataReportal [cit. 2020-05-28]. Dostupné z: **https://datareportal.com/reports/digital-2020-global-digital-overview**

[8] DIGITAL 2020: GLOBAL DIGITAL OVERVIEW: Desktop vs Mobile vs Tablet Market Share Worldwide [online]. [cit. 2020-05-28]. Dostupné z: **https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet**

[9] Code Tools: jmh. In: Openjdk.java.net [online]. [cit. 2020-05-01]. Dostupné z: **https://openjdk.java.net/projects/code-tools/jmh/**

[10] JSON Web Token - JSON Web Token [online]. [cit. 2020-06-01]. Dostupné z: https://cs.qwe.wiki/wiki/JSON_Web_Token