ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
PROGRAM GEODÉZIE A KARTOGRAFIE
OBOR GEODÉZIE, KARTOGRAFIE A GEOINFORMATIKA

BAKALÁŘSKÁ PRÁCE

VIZUALIZACE METEOROLOGICKÝCH A ENVIRONMENTÁLNÍCH DAT GEODETICKÉ OBSERVATOŘE PECNÝ

Vedoucí práce: doc. Ing. Jakub Kostelecký, Ph.D.
Katedra geomatiky

červen 2020

Lukáš BĚLOCH

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Běloch Jméno: Lukáš Osobní číslo: 477134

Zadávající katedra: 155 - geomatika

Studijní program: geodézie a kartografie

Studijní obor: geodézie, kartografie a geoinformatika

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce: Vizualizace meteorologických a environmentálních dat Geodetické observatoře Pecný

Název bakalářské práce anglicky: Visualization of meteorological and environmental data of the Geodetic Observatory Pecný

Pokyny pro vypracování:
1. Nastudovat skriptovací jazyk pro tvorbu webových aplikací
2. Nastudovat tvorbu databází včetně načítání dat do databáze a vypisování dat z databáze
3. Vytvořit funkční webovou stránku pro zobrazení testovacího vzorku dat včetně vybraných statistických charakteristik

Seznam doporučené literatury:
WebStorm: The Smartest JavaScript IDE by JetBrains. [online] https://www.jetbrains.com/webstorm/
Express routing. Express - Node.js web application framework [online] https://expressjs.com/en/guide/routing.html
React – A JavaScript library for building user interfaces [online] https://reactjs.org/
Node.js. [online] https://nodejs.org/en/
Node.js Tutorial. [online] https://www.w3schools.com/nodejs/
Node.js MySQL. [online] https://www.w3schools.com/nodejs/nodejs_mysql.asp
Node.js. itnetwork.cz [online] https://www.itnetwork.cz/javascript/nodejs

Jméno vedoucího bakalářské práce: doc. Ing. Jakub Kostelecký, Ph.D.

Datum zadání bakalářské práce: 19.2.2020 Termín odevzdání bakalářské práce: 17.5.2020
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

Podpis vedoucího práce Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v bakalářské práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce" a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací".

19.2.2020
Datum převzetí zadání Podpis studenta(ky)

ABSTRAKT

Tato práce se věnuje návrhu a realizaci webové aplikace pro vizualizaci meteorologických a enviromentálních dat Geodetické observatoře Pecný. Úvodní část se zabývá návrhem aplikace, vymezením jednotlivých částí, včetně jejich charakteristik, a zdůvodněním jejich volby. Praktická část popisuje postup při vytváření této aplikace a následnou implementaci na server. Na závěr se tato práce zaobírá lokálním i uživatelským testováním a možným vylepšením aplikace.

KLÍČOVÁ SLOVA

webová aplikace, vizualizace dat, Node.js, React.js

ABSTRACT

This thesis is dedicated to design and implement web application for visualization of meteorolofical and environmental data of the Geodetic Observatory Pecný. The first part deals with the design of the application, definition of each part, including their atrributes, and explanation for their choice. The practical part describes the process of creating this application and continues by implementation on a server. At last concerns the thesis with local and user testing and with possible application improvement.

KEYWORDS

web application, data visualization, Node.js, React.js

PROHLÁŠENÍ

Prohlašuji, že bakalářskou práci na téma „Vizualizace meteorologických a environmentálních dat Geodetické observatoře Pecný“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Chtěl bych poděkovat doc.Ing. Jakubovi Kosteleckému, Ph.D. za vedení mé bakalářské práce a ochotu, která vedla až k její výsledné podobě. Dále děkuji panu Davidu Vurbsovi za poskytnuté rady a vysvětlení témat z oblasti programování.

V neposlední řadě bych chtěl poděkovat své rodině za podporu a pochopení, které mi bylo poskytnuto nejen při psaní této práce, ale i při studiu.

Obsah

Seznam obrázků

Úvod

Geodetická observatoř Pecný je experimentálním pracovištěm Výzkumného útvaru Geodézie a geodynamika spadající pod Výzkumný ústav geodetický, topografický a kartografický, v.v.i. (VÚGTK). Činnost útvaru je zaměřena na základní a aplikační výzkum v geodetických základech, kosmické a fyzikální geodézii a geodynamice. Observatoř je pracoviště určené pro experimentální geodetický výzkum. Je na ní umístěna geodetická referenční stanice České republiky včetně výškového a gravimetrického připojení. Její součástí je laboratoř, termokomora pro zkoušky gravimetrů a geodetických přístrojů a základna pro kalibraci GNSS techniky.

Hlavní náplní observatoře je provádění permanentní GPS observace, obsluha sítě VESOG a sledování změn vertikální složky tíhového zrychlení. Současně jsou zde prováděna další experimentální měření, zpracování GPS pozorování i teoretický výzkum v oblasti matematické a fyzikální geodézie, teorie tvaru Země, kvazigeoidu a geodynamiky.

Většina prováděných pozorování se neobejde bez oprav z prostředí. Měření je ovlivněno teplotou, tlakem, vlhkostí, výškou podzemní vody apod. Z tohoto důvodu byla v minulosti na observatoři zaváděna čidla, měřící například výšku hladiny vody ve studni nebo teplotu u GNSS antény, data z nich byla zaznamenána a archivována. Postupem času byla čidla instalována na další stanoviště, ať už k měřícím přístrojům, do země, do laboratoří či do kanceláří.

V současné době je na observatoři přibližně 82 čidel sdružených do několika měřících jednotek. Všechny měřící jednotky provádějí měření v pravidelných intervalech (nejčastěji v intervalu jedné minuty) a zaznamenávají je na server. Tyto údaje jsem použil při zpracování své bakalářské práce.

1 Cíle práce

Cílem práce je vizualizace časových řad měření z čidel pro sledování vývoje sledovaných charakteristik prostředí (včetně jejich extrémů) a jejich změn v průběhu let a umožnění sledování aktuálních dat prostřednictvím webové aplikace.

Aplikace by měla zprostředkovat vizualizaci dat, statistickou práci s daty a dát možnost stažení původních dat i jejich statistik (minim, maxim, průměrů, součtů) do počítače koncového uživatele.

V neposlední řadě je mým cílem vyzkoušení si návrhu a implementace takové aplikace. V textové části budeme pojednávat o zvolení programovacího jazyka, struktuře dat, aplikace i databáze.

2 Analýza a návrh

Tato kapitola se bude věnovat návrhu aplikace tak, aby vyhovovala zadaným požadavkům. Rozebereme zde známé informace o souborech s daty, jednotlivé části aplikace a zdůvodníme volbu pro použité části.

2.1 Informace o souborech s daty

2.1.1 Umístění souborů a jejich název

Jak už bylo zmíněno v úvodu, každá měřící jednotka ukládá své záznamy na server. Zde je ukládá do souboru, který nese jméno podle aktuálního dne ve formátu RRRRMMDD. Z toho plyne, že pro každý den, pro každou měřící jednotku vzniká jeden soubor. Všechny tyto soubory jsou textové, avšak může se lišit jejich přípona. Tato unikátnost nám umožňuje mít v jedné složce více souborů z téhož dne. V této práci používáme vzorek dat, který má tuto vlastnost, a tedy by nám stačilo pracovat pouze s jednou složkou. Budeme však uvažovat a pracovat s tím, že aplikace bude mít reálné využití a budeme muset pracovat i s více složkami.

2.1.2 Struktura dat

Všechny soubory jsou podobného formátu. Hodnoty jsou strukturovány do řádků a sloupců. Každý řádek obsahuje několik sloupců, prvních šest zahrnuje datum a za ním následují jednotlivé měřené hodnoty. Může se stát, že některé soubory obsahují hlavičku, z tohoto důvodu je nutné si namátkou projít několik desítek souborů, najít společné znaky a odfiltrovat tyto nežádoucí hodnoty.
Pokud čidlo je nefunkční, či nezměřilo v daný čas žádnou hodnotu, na tuto pozici v souboru se zapíše otazník. Jelikož budeme chtít pracovat s číselnými hodnotami, musíme i tento znak odstranit.

```
 -1 -56.030 0.013225  -0.010 0.002000 -25.200 0.012500
2009 01 09 00 00 00    -9.94   960.25    88.24    03485    05324    09075
2009 01 09 00 01 00    -9.94   960.25    88.25    03485    05324    09076
2009 01 09 00 02 00    -9.94   960.20    88.24    03485    05323    09075
2009 01 09 00 03 00    -9.95   960.20    88.25    03484    05323    09076
2009 01 09 00 04 00    -9.95   960.25    88.24    03484    05324    09075
2009 01 09 00 05 00    -9.95   960.20    88.25    03484    05323    09076
2009 01 09 00 06 00    -9.95   960.20    88.25    03484    05323    09076
2009 01 09 00 07 00    -9.97   960.14    88.25    03483    05322    09076
2009 01 09 00 08 00    -9.98   960.09    88.24    03482    05321    09075
2009 01 09 00 09 00    -9.98   960.09    88.25    03482    05321    09076
2009 01 09 00 10 00    -9.98   960.09    88.25    03482    05321    09076
```

Obrázek 2.1: Ukázka struktury souboru

2.2 Struktura aplikace

Naše webová aplikace se bude skládat ze tří základních částí. Části, která poběží na straně uživatele tzv. frontend, části, která poběží na straně serveru tzv. backend, a z databáze. Taktéž musíme zprostředkovat komunikaci mezi klientem a serverem pomocí veřejného rozhraní (API). Pro tyto účely použijeme rozhraní REST. [3, 16]

2.2.1 Databáze

V dnešní době se často pro ukládání informací využívají databáze. Nejpoužívanějším typem databází, jsou databáze relační. Tento typ databáze uchovává data v tabulkách a umožňuje nám manipulaci s nimi. Jejich vlastnosti, jako je hlavně rychlost přenosu a možnost dotazů, jsou nesporné, a proto tuto formu skladování a distribuci informací využijeme i u naší aplikace prostřednictvím relačního databázového systému MySQL.

Tento systém, jako řada dalších, umožňuje vytváření tabulek, ukládání dat a manipulaci s daty pomocí jazyka SQL. Jedná se o volně šiřitelný systém (tzv. open source) vyvíjený v jazyce C a C++. Lze s ním pracovat na většině operačních systémů a je vyvíjen především na rychlost dotazů. [18]

2.2.2 Backend

Backend je zjednodušeně ta část aplikace, kterou koncový uživatel nevidí. Tato část bude mít několik úkolů.

Prvním požadavkem bude, aby obsahoval kód, kterým bude možno vytvářet, upravovat, ale i mazat tabulky v databázi.

Jeho další, nejsložitější úkol bude spočívat v nahrávání dat. S tím souvisí čtení jmen a obsahu souborů, již zmíněná úprava dat (viz. kapitola 2.1.2) a načtení do databáze. Dále budeme chtít, aby data a soubory nebyly duplikovány.

Poslední součástí backendu bude REST API, která poslouží ke komunikaci mezi serverem a klientem a poskytne dotazovaná data. Tento typ rozhraní používá čtyři základní metody: GET, POST, PUT a DELETE. My ovšem použijeme první dvě. GET slouží k získání požadovaných dat klienta od serveru. POST taktéž slouží k získání dat, navíc je zde možné zadat informaci do těla požadavku. [34]

Pro backend byl zvolen jazyk JavaScript. Ten přednostně slouží k vytváření webových stránek, tedy program se spouští a běží na webovém prohlížeči klienta.

Díky této vlastnosti lze vytvářet dynamické stránky a program je spustitelný na všech operačních systémech. Sám jazyk je směsicí několika jazyků: Java, Scheme a Self. Název jazyka by mohl napovídat, že tento jazyk může mít mnoho společného s jazykem Java. Avšak opak je pravdou. Název JavaScript je pouze obchodní tah. V dnešní době je JavaScript velice používaný. Díky tomu existují různá prostředí a mnoho aplikačních rámců (frameworků). Framework je ucelený soubor tématicky zaměřených knihoven, je to kostra, na které může programátor vystavět své programy.[3, 16, 21]

Pro backendovou část zvolíme prostředí Node.js. Toto prostředí pracuje asynchronně (příkazy se nevykonávají popořadě) a umožňuje tvorbu části, která poběží na straně serveru. Pro instalaci a správu knihoven použijeme balíčkový systém npm. [3, 16, 19]

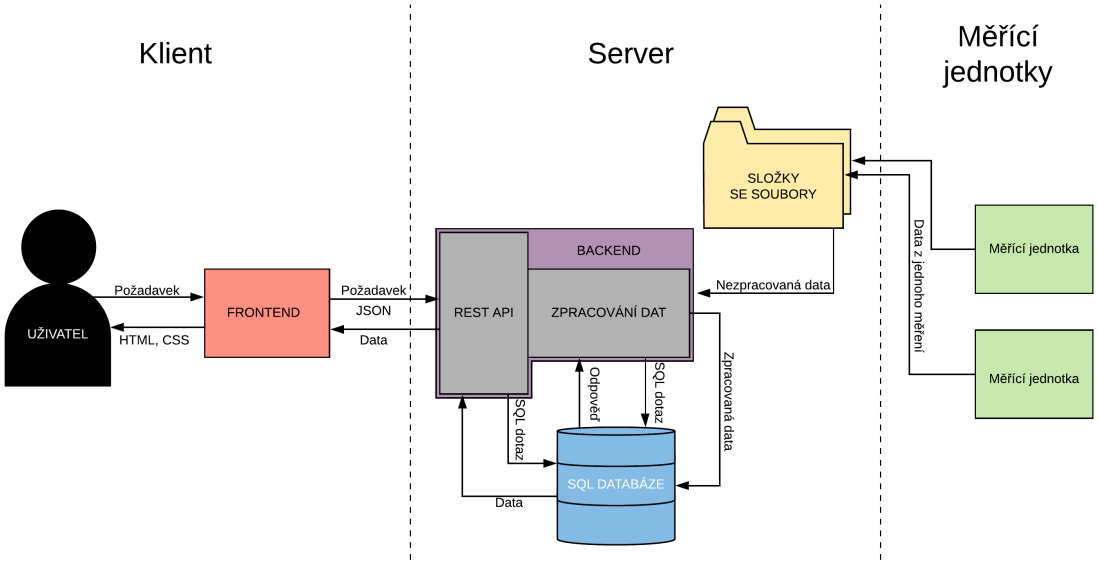
Poslední důležitou součástí bude framework Express. Tento jednoduchý a nenáročný framework poslouží k vytvoření API.[10]

2.2.3 Frontend

Frontend je část aplikace, která je viditelná koncovému uživateli. Jejím úkolem bude vizuálně a funkčně přívětivé prostředí, které zprostředkuje vizualizaci dat.

Jazyk programování použijeme opět JavaScript. Jak bylo zmíněno v kapitole 2.2.2, JavaScript je jazyk primárně určený pro tvorbu webových aplikací. Zde použijeme framework React. Tato open-source knihovna pro tvorbu uživatelského rozhraní je vyvinuta společností Facebook. V současné době patří mezi nejoblíbenější knihovny. Používají ji aplikace jako je Facebook, Dropbox, Airbnb, Tesla aj. Díky tomu existuje velké množství komponent pro tuto platformu. Komponenta je definována jako opakovatelně použitelný, předem vytvořený blok programu. O použitých knihovnách si povíme v kapitole 3.3 [1, 3, 24]

Jelikož se jedná o webové stránky, budeme (ve spojení s JavaScriptem) používat jazyk HTML. Pro to je nutné si předem osvojit základy v tomto jazyce i v jazyce CSS.



Obrázek 2.2: Schéma návrhu aplikace, autor: Lukáš Běloch

3 Realizace

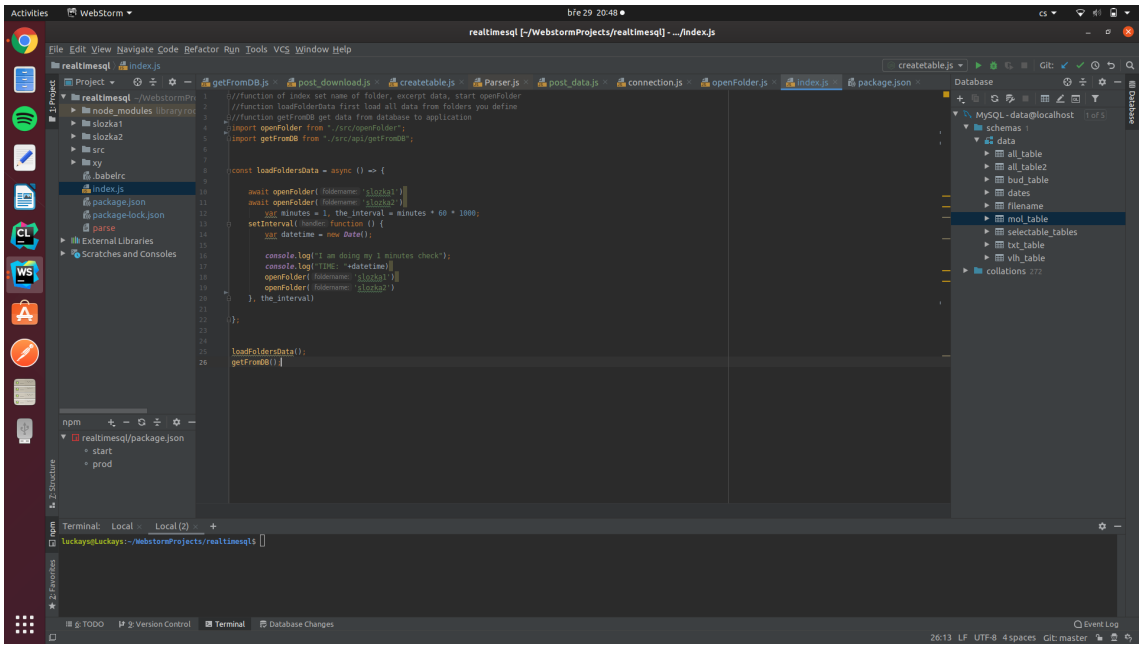
V této kapitole se budeme zabývat realizací webové aplikace nad testovacími daty. Popíšeme si použité frameworky, komponenty a postup práce při vytváření.

3.1 Vývojové prostředí

Aplikace byla vyvíjena v operačním systému Ubuntu v prostředí WebStorm od firmy JetBrains. Tento program nabízí velmi pohodlné uživatelské rozhraní. Do hlavního okna lze přidat několik menších oken, které nám usnadňují programování.

Prvním z nich je zobrazení složky aktuálního projektu (viz. obr. 3.1 vlevo). Díky tomu můžeme pohodlně otevírat jednotlivé části a funkce aplikace, máme o nich přehled a případně je zde možné tyto soubory editovat či mazat. Pod tímto oknem je další, které nám umožňuje spouštět tzv. npm skripty. Tyto funkce jsou definovány v souboru package.json, o kterém si povíme více v kapitole 3.2, a umožňují nám spouštět projekt.

V pravé části obrázku 3.1 je okno s databází. S jeho pomocí máme do databáze rychlý přístup a můžeme v ní prohlížet data a provádět atributové dotazy. Posledním nejdůležitějším oknem je okno terminálu umístěné dole. Zde můžeme zadávat různé příkazy, například pro spouštění projektu, instalace knihoven a frameworků, apod.



Obrázek 3.1: Prostředí WebStorm

3.2 Backend

V této části si popíšeme postup, který nám umožní nahrávání dat do databáze. Práce probíhala s testovacím vzorkem dat.

Nejprve si v programu WebStorm založíme Node.js projekt. Automaticky se nám vytvoří dva důležité soubory: index.js a package.json. První z nich bude sloužit k spuštění celé části backendu. Druhý obsahuje informace o nainstalovaných knihovnách a definice npm skriptů. Zde si tedy nadefinuje npm skript start, který nám bude spouštět soubor index.js.

3.2.1 Příprava databáze

Do počítače jsme nainstalovali databázový systém MySQL. V něm je nutné založit si uživatelský účet. Ostatní operace s databází budou prováděny v prostředí WebStorm. V našem projektu jsme si pro přehlednost vytvořili složku src (source), zde budou umístěny všechny námi vytvořené části. Do ní jsme umístili složku database. Ta bude obsahovat všechny JavaScript soubory, které budou pracovat s databází.

Nejprve musíme založit propojení s databází. To vytvoříme jednoduchým kódem pomocí knihovny mysql [17]. Definujeme zde uživatele, heslo a databázi, ke které se připojujeme, a zapíšeme příkaz k připojení. Další soubor, který vytvoříme, bude obsahovat SQL příkazy ke tvorbě tabulek a přidávání sloupců do databáze. Primárním klíčem tabulek bude datum.

Název hodnoty	Datový typ
txt_table	
datum	timestamp
den	varchar(255)
cas	varchar(255)
teplota_u_GPS_anteny	double
tlak_u_GPS_anteny	double
vlhkost_u_GPS_anteny	double
cteni_teploty	double
cteni_tlaku	double
cteni_vlhkosti	double

Název hodnoty	Datový typ
mol_table	
datum	timestamp
den	varchar(255)
cas	varchar(255)
vyska_hladiny_ve_studni	double
teplota_GPS_anteny	double
tlak_u_GPS_anteny	double
vlhkost_u_GPS_anteny	double
uhrn_srazek	double
tlak_v_supravodive_komore	double

Název hodnoty	Datový typ
filename	
jmeno_souboru	varchar(255)
velikost_souboru	double

Název hodnoty	Datový typ
selectable_tables	
jmeno_tabulky	varchar(255)
nazev	varchar(255)

Název hodnoty	Datový typ
bud_table	
datum	timestamp
den	varchar(255)
cas	varchar(255)
teplota_v_prizemi	double
teplota_v_pruzinove_komore	double
vlhkost_v_pruzinove_komore	double
teplota_v_supravodive_komore	double
vlhkost_v_supravodive_komore	double
vlhkost_pudy_MP_80cm_svisle	double
teplota_v_registracni_mistnosti	double
vlhkost_v_registracni_mistnosti	double
teplota_v_kompresovne	double
vlhkost_v_kompresovne	double
tlak_v_kompresovne	double
teplota_v_gravimetricke_laboratori	double
vlhkost_v_gravimetricke_laboratori	double
teplota_v_serverovne	double
teplota_v_UPS_mistnosti	double
teplota_v_GNNS_mistnosti	double
teplota_v_termokomore	double

Název hodnoty	Datový typ
vlh_table	
datum	timestamp
den	varchar(255)
cas	varchar(255)
vlhkost_pudy_MP_12cm	double
vlhkost_pudy_MP_31cm	double
vlhkost_pudy_MP_60cm	double
vlhkost_pudy_MP_86cm	double
vlhkost_pudy_MP_120cm	double
vlhkost půdy MP 80cm svisle	double
vlhkost_pudy_Z_15cm	double
vlhkost_pudy_Z_52cm	double
vlhkost_pudy_Z_91cm	double
vlhkost_pudy_S_14cm	double
vlhkost_pudy_S_56cm	double
vlhkost_pudy_S_91cm	double
vyska_hladiny_ve_studni	double
vyska_hladiny_ve_vrtu_29m	double
hloubka_vody_ve_studni	double
hloubka_vody_ve_vrtu_29m	double
vyska_hladiny_ve_vrtu_25m	double
hloubka_vody_ve_vrtu_25m	double

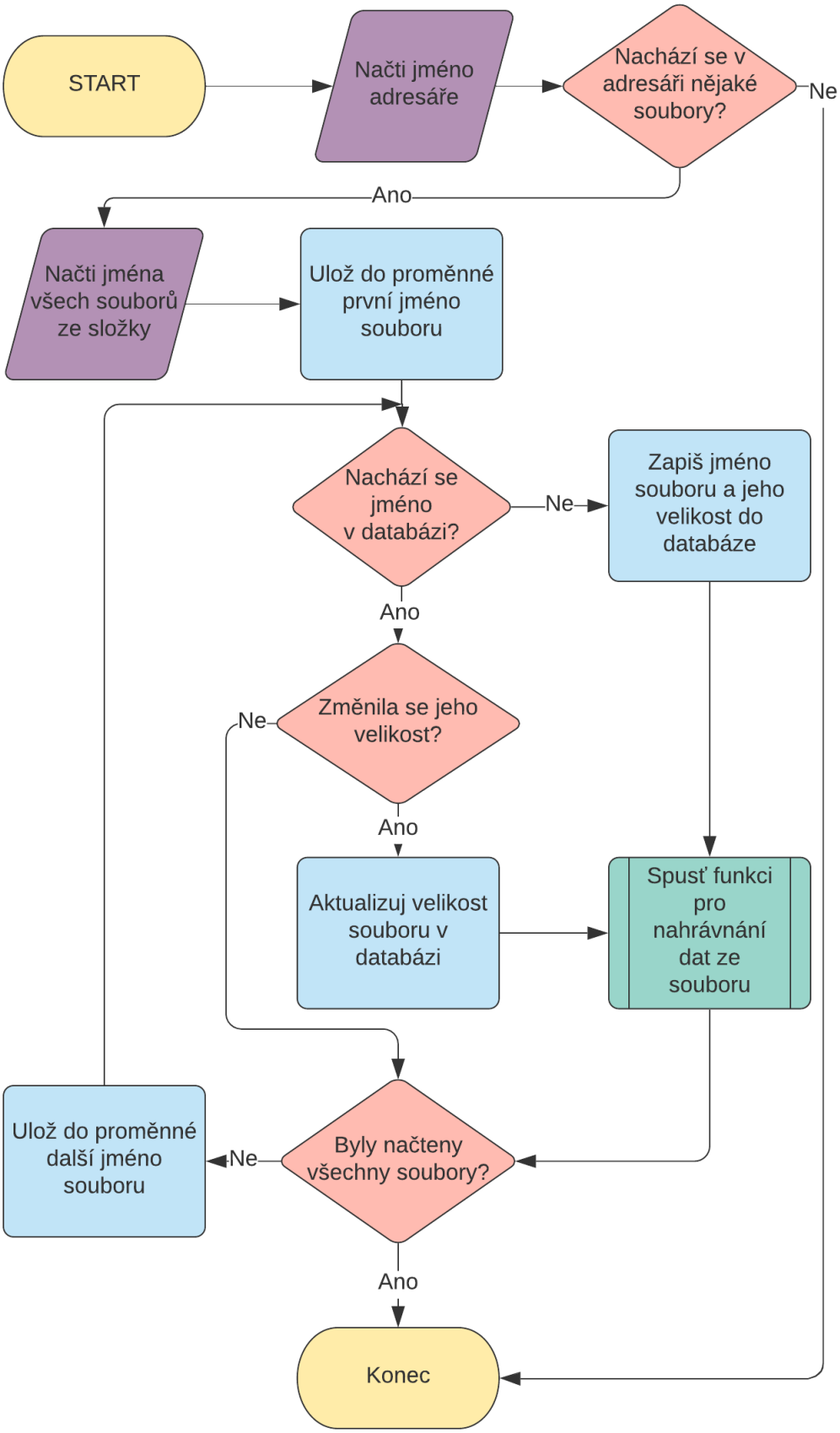
Název hodnoty	Datový typ
all_table	
datum	timestamp
den	varchar(255)
cas	varchar(255)
indikator_srazek	double
uhrn_srazek	double
teplota_u_GPS_anteny	double
tlak_u_GPS_anteny	double
vlhkost_u_GPS_anteny	double
vyska_hladiny_ve_studni	double
teplota_v_prizemi	double
tlak_u_GPS_anteny_kontrolni	double
teplota_u_GPS_anteny_kontrolni	double
vlhkost_u_GPS_anteny_kontrolni	double
vlhkost_v_registracni_mistnosti	double
teplota_v_kompresovne	double
vlhkost_v_kompresovne	double
tlak_v_kompresovne	double
vyska_hladiny_ve_vrtu_29m	double
hloubka_vody_ve_vrtu_29m	double
hloubka_vody_ve_studni	double
vyska_hladiny_ve_vrtu_25m	double
rychlost_vetru	double
smer_vetru	double
cteni_rychlosti_vetru	double

Obrázek 3.2: Struktura databáze pro testovací vzorek dat (modře znázorněn primární klíč)

3.2.2 Čtení dat ze složky

Prvním krokem k úspěšnému načtení dat ze souboru je zjištění jmen těchto souborů. Funkci budeme koncipovat tak, aby jejím vstupem byl název složky, která bude obsahovat soubory s daty. Pro čtení jmen souborů použijeme knihovnu filesystem [12]. Jména uložíme do proměnné a ta bude vstupem do další funkce ke čtení dat ze souboru.

Budeme chtít, aby naše aplikace při nahrávání neduplikovala data, přidávala nové soubory a při vložení řádku do jakéhokoliv souboru tento řádek nahrála. Z tohoto důvodu vytvoříme do databáze tabulku, do které budeme zaznamenávat jméno ve formátu "jméno složky/jméno souboru" a jeho velikost v bajtech. Dále vytvoříme funkci pro zaznamenávání do této databáze a funkci, která bude ověřovat, zda se již jméno souboru v databázi nachází a zda se nezměnila velikost již nahraného souboru.



Obrázek 3.3: Vývojový diagram pro čtení dat ze složky, autor: Lukáš Běloch

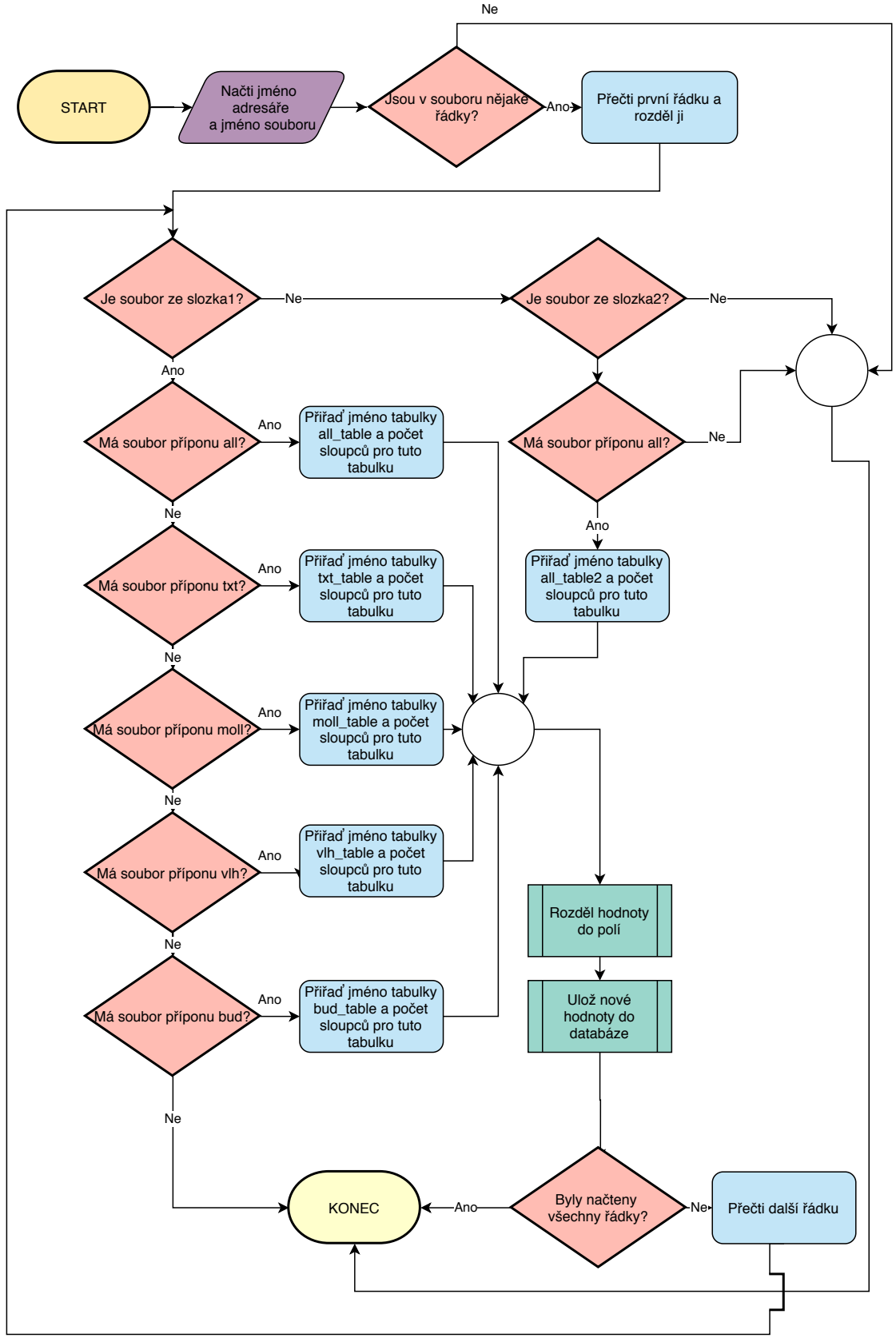
3.2.3 Čtení dat ze souboru a ukládání do databáze

Jak bylo zmíněno v předchozí části, funkce bude volána, pokud dojde k načtení nového souboru nebo souboru, ve kterém proběhla změna. Na vstupu bude jméno složky a souboru. Ze souboru budeme číst data pomocí knihoven file-system a readline [28]. Jak už název napovídá, data budeme číst po řádkách. Řádek se uloží do jedné proměnné a my tuto proměnnou musíme rozdělit po jednotlivých hodnotách. Budeme rozdělovat podle mezer. Na začátku této části je pro jednotlivé soubory nutné definovat, kolik sloupců ze souboru budeme chtít načíst. Toto číslo musí být o pět menší než počet sloupců v databázové tabulce (důvod bude vysvětlen dále).

Budeme chtít sloupce definovat, aby to nebyly jen samostatné hodnoty, a tak je rozdělíme do polí. Pro tento úkol budeme definovat novou funkci. Ta na vstupu bude mít již rozebranou řádku a počet sloupců, které chceme zapsat. Z prvních šesti sloupců vytvoří jednu hodnotu datum a ostatní sloupce zařadí do skupiny columns (sloupce). Dále ošetří hodnoty, které buďto obsahují otazník nebo neexistují (bylo definováno více sloupců, než soubor doopravdy obsahuje) a nahradí je hodnotou NULL. Funkce vrací hodnoty rozdělené do polí.

Nyní již budeme nahrávat toto pole do databáze. Vytvoříme si funkci, kde na vstupu bude jméno tabulky, do které budeme ukládat, a počet sloupců. V ní bude obsaženo spojení s databází a SQL příkaz pro uložení hodnot. Ten pomocí primárního klíče zajistí, že se řádky nebudou duplikovat.

Pro správné přiřazení hodnot do tabulek využijeme příkaz switch. První switch nám bude rozdělovat podle složky, ze kterého hodnoty pochází, a druhý podle přípony souboru. K jednotlivým řádkům tak přiřadíme jméno tabulky, do které se mají nahrávat.



Obrázek 3.4: Vývojový diagram pro čtení dat ze souboru pro testovací vzorek dat, autor: Lukáš Běloch

3.2.4 Čtení dat z databáze

Jelikož budeme chtít data zobrazit v naší webové aplikaci, je nutné nahraná data dostat z databáze. Vytvoříme si tedy soubor, který bude definovat naši API. K vytvoření budeme používat framework Express. Zde si nastavíme lokální adresu, na které bude náš výstup. Pro každý požadavek zvolíme jinou adresu.

Jelikož zatím nemáme k dispozici frontend, budeme vytvářet požadavky přes aplikaci Postman [23]. Budeme postupovat následovně:

1. Do aplikace zadáme, jakou metodou data poskytujeme (GET nebo POST).
2. Vyplníme kolonku s adresou výstupu.
3. Pro metodu POST vyplníme tělo (body).

Chceme zjistit, jaké tabulky a sloupce jsou v databázi. Tabulky vyhrajeme pomocí metody GET, neboť ta obsahuje dotaz SQL, který nepotřebuje žádné proměnné hodnoty. Na sloupce již musíme použít metodu POST. Vstupem bude jméno vybrané tabulky a obsahem SQL dotaz, jehož výsledek odešleme. Příkazy pro ostatní požadavky budou obdobné. Opět použijeme metodu POST, do které bude vstupovat tělo s hodnotami. Pomocí těchto hodnot provedeme opět SQL dotaz a výsledek odešleme. Jediným rozdílem je, v jakém formátu toto provedeme. Tabulky a sloupce budeme odesílat ve formátu JSON, data pro zobrazení pošleme v původním tvaru, tedy rozřazená do polí, a data pro stažení pošleme ve formátu CSV. Pro první dvě nám postačí používaná knihovna Express, pro CSV musíme nainstalovat knihovnu Csv-express [6]. Pokud máme kód napsaný, můžeme funkčnost všech adres vyzkoušet pomocí již zmíněné aplikace Postman. Určitě ji budeme používat i při vytváření frontendu, jelikož s její pomocí snadno najdeme, zda se eventuální chyba nachází na straně frontendu či backendu.

3.3 Frontend

V kapitole frontend se budeme zabývat vytvářením části aplikace, která bude přístupná koncovému uživateli.

V prostředí WebStorm si založíme projekt, tentokrát použijeme knihovnu React. Vytvoří se nám několik souborů, avšak většinu z nich nebudeme vůbec potřebovat.

Nejdůležitější pro nás bude:

1. index.js - Tento soubor upravíme tak, aby sloužil jen ke spuštění projektu.
2. App.js - V tomto souboru již můžeme vytvářet naši aplikaci. Pro její lepší přehlednost a srozumitelnost zde použijeme komponenty, které v dalších krocích vytvoříme.
3. package.json - Stejný soubor jako v kapitole 3.2. Npm skripty jsou již nadefinované, a tedy tento soubor nebudeme měnit.

K úpravě stylů stránky se nám vytvořil soubor App.css. Tento formát souboru nebudeme používat, proto soubor můžeme smazat. Místo něj budeme používat syntaxi SCSS, který nám rozšiřuje CSS. Pro tento účel je nutno si nainstalovat knihovnu Node-sass [20]. Pro všechny stránky, které vytvoříme, si založíme jeden nový soubor s příponou SCSS, kde budeme definovat styly pro jednotlivé sektory.

3.3.1 React Hooks

Hooky jsou prvky, které byly přidány do Reactu na začátku roku 2019. Ty nám dovolují pracovat se stavem komponent bez nutnosti použití tříd a metod. Tento rozdíl je znázorněn na webových stránkách Reactu v kapitole Hooks [31]. Také umožňují opakované použití logických částí komponent bez změny jejich hierarchie. V naší práci použijeme dva hooky:

1. **State Hook**
   Příklad kódu: const[value, setValue] = useState(0)
   Tento hook obsahuje hodnotu, která nám poskytuje současný stav (value) a funkci umožňující tento stav změnit. Funkce useState zde nastavuje počáteční stav hodnoty.
2. **Effect Hook**
   Příklad kódu: useEffect(() => api.post('/show-data'),[value])
   Tato funkce slouží k přerenderování komponenty při změně. Zapíšeme do ní kód, jenž se má při změně provést, a do hranatých závorek hodnoty, které při změně svého stavu spustí tuto funkci. Pokud závorky necháme

prázdné, spustí se funkce jen při prvním načtení, a pokud v tomto poli nebude nic, funkce se bude spouštět při jakékoliv změně v aplikaci. [2, 27, 31]

3.3.2 Stránka pro zobrazení dlouhodobých dat

1. **Kostra stránky**

Začneme se stránkou, která bude sloužit pro zobrazení dlouhodobých dat. Popíšeme si postup vytváření a tyto zkušenosti využijeme při tvorbě dalších stránek. Nejprve si však rozvrhneme stránku na jednotlivé sektory.

Obrázek 3.5: Návrh vzhledu webové aplikace

Podle návrhu (viz. obr. 3.5) si definujeme styly pro jednotlivé sektory a s využitím HTML tagů napíšeme základní prvky stránky(název, výběrová okna, tlačítka).

```
import image from "./logo.png";
import React from "react"
const HeadTitle =()=> {
return(
    <div className="container-preface">
        <div className="text-white">
            <div className="text-center"><h1> Vizualizace meteorologických a environmentálních dat Geodetické observatoře
                Pecný <img src={image} alt="Logo" height="95" width="95" align="top"/></h1></div>
        </div>
    </div>
)
}

export default HeadTitle;
```

Obrázek 3.6: Komponenta Hlavní titulek

2. **Vytvoření obsahu a propojení frontendu s backendem**
   Do kostry stránky doplníme příkazy, které odešlou backendu požadavek, přijmou data a ty zobrazí do grafu nebo zprostředkují jejich stažení.

   (a) Výběr tabulky
   Protože v databázi mohou tabulky přibývat, je nutné zajistit, aby byla možnost vybrat ty aktuální. Jména tabulek chceme získat při načtení stránky. Vytvoříme tedy Effect Hook s prázdnými závorkami (viz. 3.3.1). Jeho obsahem bude metoda GET, která tato jména získá.
   Pro zobrazení jmen ve výběrových oknech použijeme HTML tag <options> a funkci map.
   Vybraná tabulka se ukládá do proměnné definované State Hookem.

   (b) Výběr sloupce (zobrazované hodnoty)
   Každá tabulka obsahuje jiné sloupce. Opět použijeme Effect Hook, tentokrát se bude aktualizovat při změně vybrané tabulky. Uvnitř bude metoda POST. Ta odešle požadavek s vybraným jménem tabulky.
   Zobrazení a výběr sloupce je stejný jako u výběrového okna tabulky.

   (c) Výběr datumu
   Pro výběr časového rozsahu použijeme komponentu DateRangePicker z knihovny react-dates [25] od společnosti AirBnB. Upravíme zde názvy proměnných tak, aby byly shodné s našimi názvy. Dále povolíme výběr dat minulých a zakážeme výběr dat budoucích.
   Začátek a konec časového rozsahu se bude ukládat do proměnné definované State Hookem.

   (d) Analytické funkce
   Toto výběrové okno bude sloužit k volbě analytické funkce - průměr, minimum, maximum, součet.
   Vytvoříme výběrové okno, které již názvy těchto funkcí bude

obsahovat. Vybraná funkce se uloží do proměnné definované State Hookem.

(e) Tlačítko "Uložit"
Vytvoříme funkci, která se aktivuje po stisknutí tlačítka. Tato funkce bude obsahovat metodu POST, jejíž tělem budou proměnné definované výše. Budeme zde požadovat data po adrese, která nám vrátí data ve formátu CSV.
K uložení dat do počítače koncového uživatele použijeme knihovnu js-file-download [11].

(f) Zobrazení hodnot v grafu
Nejprve je nutné získat datumy a číselné hodnoty z databáze dle vybraných kriterií. K tomu poslouží Effect Hook, uvnitř kterého bude metoda POST. Tělo této metody bude stejné jako u tlačítka "Uložit", ale požadavek se bude odesílat na jinou adresu, která nám vrátí nenaformátovaná data.
Pro pole grafu zvolíme liniový graf z knihovny react-charts-2 a budeme postupovat pomocí dokumentace této knihovny [13].

Ověříme, zda nám stránka funguje, jestli správně odesílá požadavky, příjímá data a zobrazuje hodnoty. Pokud ne, použijeme program Postman nebo vývojářské prostředí v prohlížeči (aktivuje se ve webovém prohlížeči klávesou F12) k usnadnění nalezení chyby.
Pokud je vše v pořádku, upravíme jednotlivé komponenty tak, abychom je mohl použít i na jiných stránkách, převedeme je do samostatných souborů.

3.3.3 Stránka pro zobrazení denních dat

Pro ušetření času zkopírujeme stránku pro zobrazení dlouhodobých dat a tu budeme editovat. Jelikož budeme zobrazovat pouze surová data, odstraníme okno s analytickými funkcemi.

Místo kalendáře z react-dates použijeme kalendář z knihovny react-date-picker [25]. Ten umožňuje vybrat pouze jeden den. Těmto úpravám přizpůsobíme kód této stránky (úprava proměnných a těl požadavků).
Pro tuto stránku musíme upravit i backend. Vytvoříme novou adresu, která bude odesílat surová data podle požadavku vybraných kritérií.

3.3.4 Dokončení aplikace

Pro úplnost aplikace si sestavíme ještě dvě stránky.

1. **Úvod**

   Tato stránka bude obsahovat dvě pole s grafy. První bude zobrazovat minimum a maximum venkovní teploty za poslední týden, druhé venkovní teplotu aktuálního dne.

2. **O projektu**

   Dle názvu lze usoudit, že na této stránce bude informace o tom, kdo za aplikací stojí, proč byla vytvořena a k čemu slouží.

Jak už bylo řečeno na začátku kapitoly 3.3, v souboru App.js spojíme všechny stránky v jednu aplikaci. Využijeme komponent BrowserRouter, Switch, Route, z knihovny react-router-dom [7]. Tímto způsobem přiřadíme ke každé stránce její adresu (viz. obr. 3.7).

```
<BrowserRouter>
    <div>
        <Navigation/>
        <Switch>
            <Route path="/" component={Home} exact/>
            <Route path="/long_data_view" component={Long_data_view} exact/>
            <Route path="/short_data_view" component={Short_data_view} exact/>
            <Route path="/about_project" component={about} exact/>
            <Route component={Error}/>
        </Switch>
    </div>
</BrowserRouter>
```

Obrázek 3.7: Ukázka kódu ze souboru App.js

Ze stejné knihovny použijeme komponentu NavLink pro vytvoření naší komponenty, která bude sloužit k navigaci mezi stránkami. Tímto je naše aplikace hotová.

```
import React from 'react';

import { NavLink } from 'react-router-dom';

const Navigation = () => {
    return (
        <div>
            <NavLink to="/">Domů | </NavLink>
            <NavLink to="/long_data_view">Zobrazení dlouhodobých dat | </NavLink>
            <NavLink to="/short_data_view">Zobrazení krátkodobých dat | </NavLink>
            <NavLink to="/about_project">O projektu |</NavLink>
        </div>
    );
}

export default Navigation;
```

Obrázek 3.8: Komponenta navigace

4 Implementace

Aplikace bude umístěna na server **oko.pecny.cz**.

Nahrávání budeme provádět pomocí Průběžné integrace (Continuous Integration - CI). Tato metoda je souhrnem postupů a vývojářských nástrojů k urychlení vývoje softwaru. Umožňuje nahrání projektu umístěného v repozitáři (např. GitHub) na server. Hlavní výhodou je rychlé sestavení, otestování a aktualizace nových změn. Změníme-li kód aplikace a ten nahrajeme do repozitáře, CI nám projekt sestaví a otestuje. Pokud se vyskytnou v aplikaci chyby, nahrávání se zastaví a dostaneme výpis chyb. Předchozí verze aplikace nahraná na serveru zůstane zachována. Pokud test projde bez chyb, nahraje se na server nová verze. [4, 32]

4.1 Úprava aplikace

Jelikož jsme od začátku pracovali s tím, že aplikace může mít reálné využití, není třeba provádět velké změny.
Přesto znormalizujeme (přeorganizujeme) kód aplikace, neboť v průběhu její tvorby byly získány nemalé zkušenosti s programováním. Přepíšeme opakující se části do funkcí, ty vhodně pojmenujeme a umístíme do souboru a složky, dle jejich úlohy. Získáme tak lepší přehlednost kódu nejen pro nás, ale především pro budoucí editory aplikace.
Do aplikace navíc přidáme dvě zlepšení.

1. Aktualizace dat v databázi

   Zatím jsme vytvořili program, který data ze složky nahraje pouze jednou, a to při spuštění programu. Jelikož soubory a data budou přibývat, musíme zajistit, že se tato data také uloží do databáze. Proto v backendu nastavíme, že se data budou každou minutu aktualizovat. Jelikož se i data z měřících jednotek budou ukládat na server po minutě, toto nastavení postačuje.

2. Zobrazení dat v reálném čase

   Předchozí úprava nám umožňuje nahrávat data v reálném čase, a proto lze přidat i možnost jejich zobrazení. Upravíme tedy frontend, konkrétně stránku pro zobrazení denních dat. Do prázdného pole, které nám zde zbylo po odstranění analytických funkcí, přidáme tlačítko pro zapnutí a

vypnutí zobrazení dat v reálném čase. Vytvoříme funkci, která od chvíle aktivace bude požadovat po backendu data, a to v intervalu jedné minuty. Data se poté zobrazí v grafu.

4.2 Příprava serveru

U správce serveru si zažádáme o přístup na server a jeho přípravu.

Počáteční požadavky:

- Vytvoření uživatele s přístupem k serveru.
- Vytvoření uživatele s přístupem k serveru, který bude sloužit k spuštění a běhu aplikace.
- Instalace verzovacího systému GIT.
- Instalace softwaru pro kompilaci jazyka JavaScript na straně serveru - Node.js.
- Instalace softwaru pro správu balíčku - npm.
- Instalace softwaru pro správu databáze - MySQL.
  Byl nainstalován software MariaDB. Tyto dva systémy se od sebe moc neliší, a tedy lze MariaDB použít bez úprav aplikace.
- Instalace webového serveru - Nginx.
  Webové servery nám umožňují vystavení webu a nastavení jeho specifikací (např. jeho doménu a certifikát). Na serveru byl již nainstalován webový server Apache a s ním budeme dále pracovat.

Požadavky, které se vyskytly v průběhu implementace:

- Instalace pm2 a práva k němu.
  PM2 je správce procesů pro Node.js. Zajistí nám spuštění a udržení běhu aplikace na serveru. [22]
- Práva k npm.
- Práva pro zápis do složky, kde bude umístěna naše aplikace.

- Přesun databáze, kvůli nedostatku místa v jejím aktuálním umístění.
- Povolení portu pro vnitřní a vnější komunikaci.
- Oprávnění do složek s virtuálním serverem.
  Jedná se o složky: /etc/apache2/sites-available a /etc/apache2/sites-enabled.
- Práva k restartu Apache.
- Práva k příkazům k povolení nezbytných modulů Apache.
  Příkazy: sudo a2enmod proxy a sudo a2enmod proxy_http
- Zvětšení RAM paměti pro uživatele, který slouží k spuštění a běhu aplikace.
- Vytvoření domén pro backend a frontend
  Doména pro frontend bude sloužit pro přístup k aplikaci a doména pro backend pro komunikaci mezi těmito dvěma částmi.
  V této aplikaci použijeme doménu třetího řádu.
- Nastavení HTTPS konfigurace domén.

4.3 Příprava aplikace

Implementaci na server provedeme přes webovou službu GitHub. Proto pro náš backend a fronend vytvoříme repozitáře, uložíme do nich jednotlivé části a nahrajeme na GitHub. Doporučuji toto udělat již na začátku vytváření celé aplikace a průběžně projekt ukládat do repozitáře.

Náš projekt obsahuje citlivá data (např. údaje k databázi). Chceme, aby tato data nebyla v repozitáři veřejně přístupná. Do aplikace nainstalujeme knihovnu dotenv [8] a v hlavní složce vytvoříme soubor .ENV. V něm můžeme vytvořit jednotlivé proměnné specifické pro prostředí a jimi pak nahradit citlivé údaje v kódu. Tento soubor do repozitáře nenahráváme.

Abychom mohli použít metodu CI, musíme pro GitHub vytvořit konfigurační soubor s příponou .YML. Umístíme jej do složky "projekt/.github/workflows".
V tomto souboru zapíšeme, podle dokumentace GitHub Actions [33], příkazy, které se provedou po každém nahrání projektu do repozitáře na GitHubu. Definujeme zde například přístup k serveru, umístění projektu na něm, příkaz k instalaci npm

a příkaz pro spuštění aplikace.
Tento soubor lze vytvořit i na GitHabu v záložce "Actions"

4.4 Implementace

Pokud máme aplikaci a server připravené, můžeme přejít k samotné implementaci. V následujících krocích si popíšeme implementaci backendu. Postup pro implementaci frontendu bude totožný.

1. Přihlášení k serveru
   K serveru se přihlásíme pomocí SSH klienta PuTTY [9].

2. Vytvoření SSH klíče pro GitHub
   SSH klíč se skládá z klíče veřejného a privátního. Tento pár vytvoříme zadáním příkazu "ssh-keygen -t rsa -b 4096 -C “Deploy”" [14]. Příkaz po nás bude požadovat umístění na serveru (v našem případě klíč umístíme do "/home/metapp/.ssh/id_github_api") a heslo. To necháme prázdné. Vzniknou nám tak dva klíče: veřejný - "id_github_api.pub" a privátní - "id_github_api".

3. Přidání veřejného klíče do GitHubu
   Otevřeme náš repozitář a přejdeme do záložky "Settings". Zde vybereme možnost "Deploy keys" a přidáme nový. Klíč si nazveme dle libosti a vložíme sem obsah souboru "id_github_api.pub".
   Pro test připojení k repozitáři vložíme příkaz "ssh -i ./id_github_ap git@github.com"

4. Více účtů GitHub
   Jelikož ve složce máme klíče pro dva repozitáře, musíme je přiřadit. Učiníme to vytvořením souboru "config" a zapsáním kódu do souboru podle dokumentace[15]. V tomto souboru vytvoříme jméno pro naše části aplikace a ty použijeme dále.

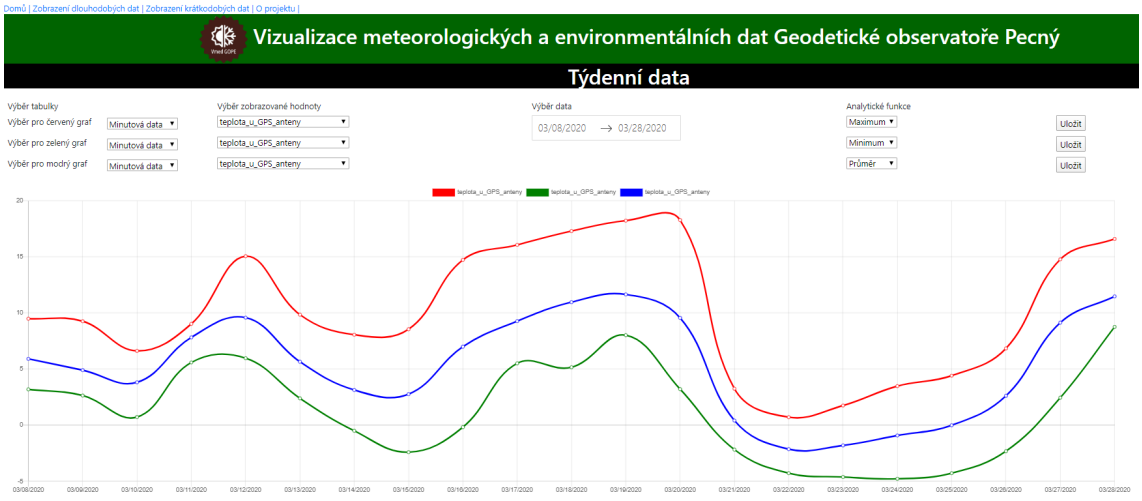
5. Vytvoření klíče pro připojení SSH
   Ve složce ".ssh" si vytvoříme ještě jeden pár klíčů.
   Ve stejné složce vytvoříme soubor "authorized_keys", do kterého vložíme veřejný klíč.

Privátní klíč vložíme do GitHubu. Tentokrát v záložce "Settings" zvolíme "Secrets" a vytvoříme zde nový klíč. Jméno klíče bude PRIVATE_KEY. Pro frontend nebudeme vytvářet nový klíč a použijeme tento.
Nyní by se měl projekt na GitHubu sestavit a neúspěšně se pokusit nahrát na server. Toto můžeme vidět v našem repozitáři v záložce "Actions".

6. Umístění repozitáře na server.
   Zkopírujeme URL adresu repozitáře pro SSH (Clone with SSH). Tu použijeme v dalším kroku. Na server ve složce pro backend napíšeme příkaz: "git clone git@jmeno-api.github.com: uzivatel/repozitar.git .", kde "jmeno-api" je jméno ze souboru "config". V našem případě tak bude celý příkaz: "git clone git@bernatd-api.github.com:Luckays/Vmed_API.git"

7. Konfigurace Apache
   Ve složce /etc/apache2/sites-available a /etc/apache2/sites-enabled vytvoříme soubor metapi.conf. V tomto souboru definujeme doménu pro backend, umístění složky s aplikací a ip adresu serveru. [29, 30]
   Po vytvoření spustíme v konzoli tyto dva příkazy: sudo a2enmod proxy a sudo a2enmod proxy_http. Tímto zapneme proxy server.

Všechny tyto kroky by měly vést k úspěšnému zveřejnění aplikace. Pokud se tak nepovedlo, je třeba hledat chyby na straně serveru, nejčastěji se jedná o špatnou konfiguraci či chybějící oprávnění.



Obrázek 4.1: Ukázka aplikace ve webovém prohlížeči

5 Testování

5.1 Lokální testování

Toto testování bylo provedeno nad testovacím vzorkem dat s použitím lokálních adres.

5.1.1 Backend

1. Nahrávání dat do databáze

   Do kódu byly zadány složky, které obsahovaly soubory s daty, a kód byl spuštěn. V průběhu nahrávání bylo v databázi kontrolováno, zda se do ní ukládají správné názvy nahraných složek a data z nich.

2. Aktualizace dat

   Pro ověření funkčnosti aktualizace dat, při uložení nového měření do souboru či vytvoření nového souboru, byl v jazyce C++ vytvořen projekt. Ten měl za úkol každou minutu ukládat do složky, ze které backend bral svoje data, soubory s příponou CSV. Tyto soubory měly stejnou strukturu jako soubory z testovacího vzorku: název odpovídal aktuálnímu dni (formát RRRRMMDD.CSV) a do prvních šesti sloupců se ukládal aktuální čas. Do sedmého sloupce se poté nahrávala náhodná číselná hodnota z intervalu (0;100).

   Tento test probíhal po dobu 48 hodin a během něho byly namátkově porovnávány hodnoty ze souboru s hodnotami z databáze.

3. API

   V programu Postman byl pro každou adresu vytvořen nový skript. Tam, kde byla použita metoda POST, bylo vyplněno tělo požadavku. Následně byly odeslány požadavky na backend a porovnáním s databází vyhodnocena přijatá data.

5.1.2 Frontend

V této části byla nejprve otestována funkčnost všech tlačítek, funkčnost výběrových oken a propojení s backendem. Bylo ověřeno, jestli se v grafu zobrazují data vybraných kritérií. Následně bude především tato část uživatelsky testována.

5.2 Uživatelské testování

Testování bylo provedeno u vzorku 6 lidí. Těm byla poskytnuta doména aplikace a formulář k ohodnocení.
Formulář obsahuje pět hodnotících prvků, se stupnicí 1-5, kde 1 je nejlepší. Jedno pole je k dispozici pro volný komentář.

5.3 Zhodnocení výsledků

5.3.1 Lokální testování

Testováním na lokálním počítači byly zjištěny tyto chyby:

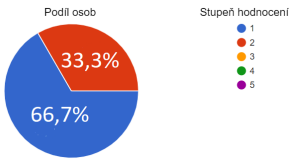
1. Nahrávání přestalo fungovat kvůli nedostatku paměti RAM
   Tato chyba vznikla díky asynchronicitě Node.js. Při malém množství souborů se do paměti nejdříve nahrála rozřazená data a až po načtení všech souborů se tato data uložila do databáze, což při velkém množství dat není možné. Chyba opravena příkazy async a await. Díky tomu bylo zajištěno, že každý soubor se nejprve nahraje do paměti počítače a po rozřazení všech řádků se data uloží do databáze, paměť se uvolní a spustí se nahrávání dalšího souboru.

2. Data se nenačítala do frontendu.
   Použitím aplikace Postman bylo zjištěno, že chyba není v nesprávném zpracování požadavku či v odesílání dat. Ve vývojářské konzoli webového prohlížeče se zobrazila hláška, že chybí "access-control-allow-origin".
   Zjednodušeně řečeno: není povolena zdrojová doména. Tento problém byl vyřešen instalací a použitím knihovny cors [5] v backendové části.

5.3.2 Uživatelské testování

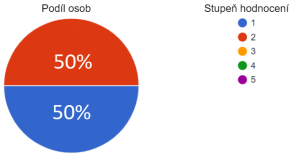
Tato část se zabývá vyhodnocením odpovědí z hodnotících formulářů.

- **Ovládání**



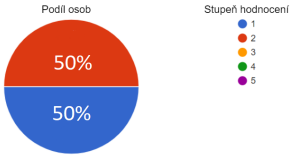
Obrázek 5.1: Graf hodnocení - ovládání

- **Přehlednost**



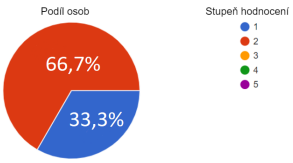
Obrázek 5.2: Graf hodnocení - přehlednost

- **Užitečnost**



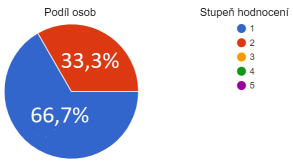
Obrázek 5.3: Graf hodnocení - užitečnost

- **Intuitivnost**



Obrázek 5.4: Graf hodnocení - intuitivnost

- **Funkčnost**



Obrázek 5.5: Graf hodnocení - funkčnost

- **Volné komentáře**

  Ve volných komentářích se vyskytovaly připomínky ke grafickému vzhledu. Zejména k velikosti navigace stránek a volbě barev.

Závěr

Cílem práce bylo vytvořit webovou aplikaci pro vizualizaci časových řad měření z čidel pro analýzu vývoje sledovaných charakteristik prostředí včetně jejich extrémů a jejich změn v průběhu let. Při realizaci se podařilo nejen splnit všechny zadané požadavky, ale byly přidány i další funkce a vylepšení. Nejdůležitějším z nich je aktualizace dat v databázi.
Kromě realizace aplikace byla provedena i její implementace na server a následné zveřejnění. Aplikace je dostupná na adrese: `https://metapp.pecny.cz/`.

Podle výsledků uživatelského testování je aplikace přehledná a srozumitelná, což považuji za velký úspěch. Navzdory tomu potřebuje aplikace opravu drobných chyb a je připravena k dalšímu vývoji. Hlavním tématem zlepšení by bylo uspořádání stránky při změně velikosti okna nebo při používání aplikace na mobilním telefonu. Mezi další úpravy bych zařadil grafickou úpravu stránky pro její lepší přehlednost a atraktivnější vzhled. V neposlední řadě by bylo na místě vytvoření záložky s aktuálním přehledem nejdůležitějších naměřených dat. Do aplikace bych navíc zařadil více dat z různých čidel. K tomuto již není potřeba vývoj aplikace, neboť je na to připravena.

Projekt splnil i můj osobní cíl, kterým bylo vyzkoušení si návrhu a implementace webové aplikace. Navíc jsem získal nemalé zkušenosti s jazykem JavaScript
a s programováním.

Tato bakalářská práce může posloužit jako zjednodušený návod pro tvorbu a implementaci webové aplikace. Pro osvojení použitých programovacích jazyků je ale zapotřebí velké množství času a trpělivosti.

Seznam zkratek

API	Application programming interface - Rozhraní pro programování aplikací
CI	Continuous Integration - Průběžné integrace
CSS	Cascading Style Sheets - Kaskádové styly
CSV	Comma-separated values - Hodnoty oddělené čárkami
ENV	Environment Variables - Proměnné prostředí
GNSS	Global Navigation Satellite Systems - Globální navigační družicové systémy
GPS	Global Positioning System - Globální polohový systém
HTTPS	Hypertext Transfer Protocol Secure - Protokol umožňující zabezpečenou komunikaci v počítačové síti
JS	JavaScript
JSON	JavaScript Object Notation - JavaScriptový objektový zápis
npm	Node.js package manager - Správce balíčků pro prostředí Node.js
RAM	Random Access Memory - Polovodičová pamět s přímým přístupem
REST	Representational state transfer
RRRRMMDD	rok, měsíc, den
SCSS	Sassy CSS
SSH	Secure Shell - Zabezpečený komunikační protokol v počítačových sítích
SQL	Structured Query Language - Strukturovaný dotazovací jazyk
VESOG	Výzkumná a experimentální síť pro observace s GNSS
VÚGTK	Výzkumný ústav geodetický, topografický a kartografický
YAML	Ain't Markup Language - Formát pro serializaci strukturovaných dat.

Literatura

[1] *AnyforSoft | Drupal Development Company* [online]. [cit. 21-03-2020]. Dostupné z: `https://anyforsoft.com/blog/10-famous-websites-built-react-js`.

[2] *Agentura pro vývoj mobilních a webových aplikací | Ackee* [online]. [cit. 17-04-2020]. Dostupné z: `https://www.ackee.cz/blog/react-hooks-2/`.

[3] *Ajťácká sociální síť a materiálová základna pro C, Java, PHP, HTML, CSS, JavaScript a další.* [online]. Copyright © 2020 itnetwork.cz. Veškerý obsah webu. [cit. 20-03-2020]. Dostupné z: `https://www.itnetwork.cz/`.

[4] *Continuous Integration: What is CI? Testing, Software and Process Tutorial. Continuous Integration, Deployment and Delivery with Codeship* [online]. [cit. 08-05-2020]. Dostupné z: `https://codeship.com/continuous-integration-essentials`.

[5] *cors - npm. npm | build amazing things* [online]. [cit. 08-05-2020]. Dostupné z: `https://www.npmjs.com/package/cors`.

[6] *csv-express - npm. npm | build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: `https://www.npmjs.com/package/csv-express`.

[7] *dom - npm. npm | build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: `https://www.npmjs.com/package/dom`.

[8] *dotenv - npm. npm | build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: `https://www.npmjs.com/package/dotenv`.

[9] *Download PuTTY - a free SSH and telnet client for Windows. Download PuTTY - a free SSH and telnet client for Windows* [online]. [cit. 08-05-2020]. Dostupné z: `https://www.putty.org/`.

[10] *Express - Node.js web application framework* [online]. Copyright © 2017 StrongLoop, IBM, and other expressjs.com contributors. [cit. 21-03-2020]. Dostupné z: `https://expressjs.com/`.

[11] *js-file-download - npm. npm / build amazing things* [online]. Copyright ©. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/js-file-download.

[12] *file-system - npm. npm / build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/file-system.

[13] *GitHub - jerairrest/react-chartjs-2: React wrapper for Chart.js* [online]. Copyright © 2020 GitHub, Inc.c. [cit. 17-04-2020]. Dostupné z: https://github.com/jerairrest/react-chartjs-2.

[14] *How To Set up SSH Keys on a Linux / Unix System - nixCraft. nixCraft - Linux Tips, Hacks, Tutorials, And Ideas In Blog* [online]. Copyright ©2000. [cit. 08-05-2020]. Dostupné z: https://www.cyberciti.biz/faq/how-to-set-up-ssh-keys-on-linux-unix/.

[15] *How to Work with GitHub and Multiple Accounts* [online]. Copyright © 2020 GitHub, Inc. [cit. 08-05-2020]. Dostupné z: https://gist.github.com/JoaquimLey/e6049a12c8fd2923611802384cd2fb4a.

[16] *IT Slovník.cz* [online]. [cit. 20-03-2020]. Dostupné z: https://it-slovnik.cz/.

[17] *mysql - npm. npm / build amazing things* [online]. [cit. 20-03-2020]. Dostupné z: https://en.wikipedia.org/wiki/MySQL.

[18] *MySQL - Wikipedia* [online]. [cit. 20-03-2020]. Dostupné z: https://www.npmjs.com/package/mysql.

[19] *Node.js.* [online]. [cit. 21-03-2020]. Dostupné z: https://nodejs.org/en/.

[20] *node-sass - npm. npm / build amazing things* [online]. Copyright ©. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/node-sass.

[21] *Peter Láng - web developer na volné noze* [online]. [cit. 21-03-2020]. Dostupné z: http://langi.cz/webarna/co-je-to-framework.

[22] *PM2 - Home.* [online]. Copyright © 2019 PM2. All Rights Reserved. [cit. 15-05-2020]. Dostupné z: https://pm2.keymetrics.io/.

[23] *Postman | The Collaboration Platform for API Development.* [online]. Copyright ©. [cit. 20-04-2020]. Dostupné z: Dostupnéz:https://www.postman.com/.

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

[25] *react-dates - npm. npm | build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/react-dates.

[26] *react-date-picker - npm. npm | build amazing things* [online]. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/react-date-picker.

[27] *React Hooks, které potřebujete znát - Zdroják. Zdroják - o tvorbě webových stránek a aplikací* [online]. [cit. 17-04-2020]. Dostupné z: https://www.zdrojak.cz/clanky/react-hooks-ktere-potrebujete-znat/.

[28] *readline - npm. npm | build amazing things* [online]. Copyright ©. [cit. 20-04-2020]. Dostupné z: https://www.npmjs.com/package/readline.

[29] *Seznámení a konfigurace s Apache | Interval.cz. Interval.cz | Svět Internetu, Technologií a Bezpečnosti* [online]. [cit. 15-05-2020]. Dostupné z: https://www.interval.cz/clanky/seznameni-a-konfigurace-s-apache/.

[30] *Správa linuxového serveru: Úvod do konfigurace Apache - Linux E X P R E S. Linux E X P R E S* [online]. Copyright © 2020 CCB, spol. s r. o., všechna práva vyhrazena. [cit. 15-05-2020]. Dostupné z: https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-uvod-do-konfigurace-apache.

[31] *Using the State Hook – React* [online]. Copyright © 2020 Facebook Inc. [cit. 17-04-2020]. Dostupné z: https://reactjs.org/docs/hooks-state.html.

[32] *Vy ještě nepoužíváte Continuous Integration? - Zdroják. Zdroják - o tvorbě webových stránek a aplikací* [online]. [cit. 08-05-2020]. Dostupné z: https://www.zdrojak.cz/clanky/vy-jeste-nepouzivate-continuous-integration/.

[33] *Workflow syntax for GitHub Actions - GitHub Help.* [online]. Copyright © 2020 GitHub, Inc. [cit. 08-05-2020]. Dostupné z: https://help.github.com/en/actions/reference/workflow-syntax-for-github-actions.

[34] *Zdroják - o tvorbě webových stránek a aplikací* [online]. [cit. 21-03-2020]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.

A Backend - GIT repozitář

Projekt je dostupný z Githubu z repozitáře BP_Vmed_API uživatele Luckays.
`https://github.com/Luckays/BP_Vmed_API`

B Frontend - GIT repozitář

Projekt je dostupný z Githubu z repozitáře BP_Vmed_APP uživatele Luckays.
`https://github.com/Luckays/BP_Vmed_APP`