FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Eco web
Student:	Roman Soběslav
Vedoucí:	Mgr. Tomáš Karella
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem této práce analyzovat, navrhnout webový portál, který bude vizualizovat možné teplotní změny v ČR.
Součástí práce je vytvoření datasetu z otevřených dat Českého hydrometeorologického ústavu. Důraz bude kladen na jednoduchost ovládání a pochopitelnost.
Student navrhne softwarové řešení, implementuje webový portál a aplikaci pro stahování, zdokumentuje a otestuje vzniklý kód.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 20. listopadu 2020 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Eco web

Roman Soběslav

Katedra softwarového inženýrství
Vedoucí práce: Mgr. Tomáš Karella

27. června 2021

Poděkování

Předně bych rád chtěl poděkovat Mgr. Tomáši Karellovi a Ing. Marku Sušickému za vedení mé bakalářské práce, velice cenné rady, věcné připomínky a vstřícnost při konzultacích. Dále bych chtěl poděkovat rodině a přátelům za podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 27. června 2021 . . . . . .

Odkaz na tuto práci

Soběslav, Roman. *Eco web*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Cílem této práce je analyzovat veřejně dostupná ekologická data a zvolit vhodná pro vizualizaci dat. Dále se zabývá analýzou již existujících řešení. Práce se následně zabývá návrhem a implementací jednotlivých částí pro získání dat, databázi, backend a frontend. Část pro získání dat se zabývá získáváním dat ze stránek ČHMÚ. Backend je navrhnut a implementován pomocí jazyka JavaScript s využitím technologie GraphQL pro tvorbu API. Frontend je navrhnut jako webová aplikace za pomoci designového jazyka Material Design. Pro implementaci webové aplikace byl použit JavaScript s knihovnou ReactJS. Následně se zabýváme uživatelským testováním samotné webové aplikace. Na závěr práce jsou uvedeny možná rozšíření implementace do budoucna.

Klíčová slova webová aplikace, ekologická data, ČHMÚ historická data, vizualizace dat, ČHMÚ, eko web, Material Design, ReactJS, GraphQL

Abstract

The purpose of this bachelor thesis is to analyze available ecological data for public use, which are appropriate for visualization. This thesis also focuses on analysis of the existing solutions. Then the focus shifts to the design and implementation of separate parts, which consist of a tool for downloading data, database, backend and frontend. The download tool is focused on downloading data that are available on the CHMI website. The backend is designed and implemented in JavaScript in combination with GraphQL technology for creating APIs. The frontend is designed as a web application in combination with a design language called Material Design. For implementation of the web application, JavaScript was used in combination with the ReactJS library. Upon completion, the web application was tested using a usability testing method. At the end, this bachelor thesis also covers plans for future extensions.

Keywords web application, ecological data, CHMI historical data, data visualisation, CHMI, eco web, Material Design, ReactJS, GraphQL

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Když se podíváme do novin, časopisu nebo si otevřeme web, určitě najdeme nějaký článek nebo text týkající se ekologie. Ekologie – toto téma rezonuje společností posledních pár let. Avšak tato problematika není nová a lidstvo se jí zabývá již pár desítek let. Jelikož se jedná o velmi široký vědní obor s velkým přesahem i do dalších vědních disciplín, můžeme zjistit, že počátky této vědy sahají až do antiky. Avšak dnes, i díky neodborné veřejnosti, se lze dočíst, že pod ekologii nepatří jen nauka o vztazích mezi živočichy a prostředím, ale i například recyklace materiálů, úprava životního prostředí, ve kterém žijeme nebo globální změna klimatu. Toto téma se díky medializaci v posledních letech stalo natolik politizované, že můžeme slýchat názory nejen vědecké obce. [5, 6] Tyto názory zabírají velmi široké spektrum – od absolutního odmítání, že něco takového existuje až po šiřitele nevyhnutelné zkázy.

Zveřejněné názory je nejlépe si nejprve ověřit, zde narážíme na problém, že odpůrci se často odkazují na zkreslené grafy a nepodložená data. Naopak vědecká obec se odkazuje na své výzkumy, které jsou pro neznalé z oboru velmi těžké na pochopení. Proto jednou z motivací pro vznik webu je myšlenka vytvořit jeden portál, na kterém je možné procházet data spojená jakkoliv s ekologií. Aktuálně se zaměřujeme na omezený rozsah dat, ale v rámci návrhu se počítá s přidáváním dalších datových zdrojů do budoucna. Dále jsme chtěli umožnit uživatelům sledovat vývoj různých dat v čase, což dostupné weby neumí.

Cílem této práce je vytvořit web, který bude určený v podstatě komukoliv zajímajícího se o ekologická historická data. Ať už se jedná o laika, který se chce podívat na historický vývoj dat nebo někoho, kdo hledá nějaké vhodné místo, které splňuje daná kriteria (kriterii je zde myšleno nějaká ekologická měřitelná a dostupná data). Například, když si chceme ověřit, že v dané lokalitě má smysl si pořídit fotovoltaické panely na střechu.

Jelikož práce nenavazuje na již existující řešení, v rámci této práce se zaměříme na analýzu dat, požadavků na web, návrh jednotlivých částí spolu

s volbou technologií, popsání kritických částí implementace a následného testování. Primárně se zaměříme na datové zdroje poskytované Českým hydrometeorologickým ústavem, konkrétně se jedná o veškerá denní meteorologická a hydrologická historická data, poskytovaná dle zákona 123/1998 Sb.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je analyzovat, navrhnout, implementovat a otestovat webový portál pro vizualizaci dat spojených s ekologií. Vzniklý portál musí být uživatelsky přívětivý, intuitivní na ovládání, přehledný a srozumitelný. Po technické stránce musí podporovat responzivní design a musí umožnit zobrazit data pomocí grafů nebo mapy. Uživatel bude mít možnost zobrazit související veličiny v jednom grafu a díky tomu sledovat například vývoj minimální, průměrné a maximální teploty zároveň. Následně má možnost přejít do porovnávací funkce, ve které lze sledovat historický vývoj vybraných veličin, která mohou být i z jiných datových typů. Lze tady porovnat například historický vývoj průměrné teploty a celkového úhrnu srážek. Mezi hlavní cíle této práce patří analýza dat a volba vhodných dat, která jsou k dispozici. Následně je potřeba navrhnout databázi, backend a frontend v závislosti na zvolených datech s důrazem na rozšiřitelnost do budoucna o další data.

Přínos této práce spočívá ve vytvoření Open-Source řešení vizualizace různých dat. Na rozdíl od existujících řešení bude nabízet více typů dat, jelikož většina dostupných portálů se zaměřuje na vývoj pouze jedné veličiny a to v blízkém čase. Další výhodou je možnost porovnat různá data.

Hlavním cílem kapitoly 2 je zanalyzovat dostupná data a zvolit vhodná data. Zároveň je třeba si v rámci této kapitoly zadefinovat funkční a nefunkční požadavky, které musí finální řešení splňovat. Dále je třeba zjistit, jestli existují weby s podobným zaměřením v ČR a zahraničí a případně provést jejich analýzu a vyhodnotit, zdali splňují požadavky definované v rámci této bakalářské práce. Následně je nutné definovat případy užití tohoto webu.

Co se týče kapitoly 3, tak zde jsou cíle stanoveny následovně. Nejprve je potřeba navrhnout způsob, pomocí kterého lze získat zvolená data. Následně navrhnout architekturu celé praktické části a provést analýzu technologií, které jsou dostupné a vhodné pro implementaci našeho webu. Následně je potřeba zvolit technologie pro jednotlivé části a odůvodnit jejich zvolení. Posledním cílem této kapitoly je představit principy Material Designu, který je zvolen jakožto designový jazyk pro uživatelské rozhraní webu. Tedy stručně

shrnout základní principy, které tento designový jazyk stanovuje pro potřeby této bakalářské práce. V souladu s těmito pravidly je představen i návrh uživatelského rozhraní.

Kapitola 4 má za cíl představit vývojové prostředí. Dále také seznámí čtenáře se strukturou a technologií použitou pro verzování kódu. Představíme použité knihovny, zdůvodníme, proč jsme si je vybrali a přiblížíme jejich důležité součásti.

Kapitola 5 se věnuje testování. Jejím cílem je navrhnout uživatelské testování, které se zaměří na uživatelského rozhraní (intuitivní ovládání, srozumitelnost a přehlednost) a následně vyhodnocení našeho web podle těchto kritérií. Dále se budeme zabývat programátorským testováním vzniklého softwaru.

Kapitola 2

Analýza

V této kapitole si představíme základní potřebné informace, které použijeme ve zbytku práce. Nejprve si v sekci 2.1 projdeme veřejně dostupná data týkající se ekologie a zvolíme si vhodné datové zdroje. Následně si v sekci 2.2 zadefinujeme funkční a nefunkční požadavky, které požadujeme od finálního řešení. Ty nám zároveň poslouží i jako hodnotící kritéria již existujících řešení, které si zanalyzujeme v sekci 2.3. Poté, co si představíme webové aplikace, které se zabývají stejnou tématikou a zhodnotíme je, definujeme si případy užití námi vypracovaného řešení v následující sekci 2.4.

2.1 Analýza dat

Prvním krokem této kapitoly, je analýza dostupných dat. V rámci této práce vybíráme pouze veřejně dostupné datové sady, které jsou relevantní pro ekologii. Dalším omezením, je nutnost poskytnutí datových sad ve strukturovaném souboru. Dále se zajímáme jen o oficiální zdroje dat, tedy provádíme analýzu dat ze subjektů, které jsou schváleny vládou České republiky jakožto ověřené zdroje určené pro poskytování těchto dat. Následující vybrané zdroje jsou seřazeny abecedně.

2.1.1 ČHMÚ

Dle zákona 123/1998 Sb. musí Český hydrometeorologický ústav poskytnout přístup veřejnosti k historickým datům související s ekologií. Konkrétní datové sady dostupné na stránkách [7] tvoří:

- Denní historická meteorologická data
- Měsíční historická meteorologická data
- Denní hydrologická data
- Kvalita ovzduší

Meteorologická i hydrologická data tvoří soubory denních měření, která jsou strukturovaná do kategorií. Jedná se o výsledky měření, která probíhala mezi roky 1961 a 2019 včetně. Celkově tyto datové sady jsou velmi přehledné a vhodné pro zobrazování pomocí grafů a mapy a jsou srozumitelné pro osoby neznalé této problematiky. Jedná se o velmi vhodný zdroj dat pro naši aplikaci.

Druhý typ dat jsou data týkající se kvality ovzduší. Sada se opět skládá z jednotlivých měření pro jednotlivé stanice. Zároveň každý sledovaný prvek v ovzduší má vlastní datovou sadu. Přestože výsledná zpracovaná sada se jeví jako vhodná pro zobrazení pomocí grafů, pro běžného člověka bez větší znalosti jsou tyto čísla nesrozumitelná. Přesto tento zdroj dat lze pro naše účely využít.

2.1.2 Golemio

Golemio je webový portál poskytující otevřená data pro území Prahy. Mezi zajímavé datové sady patří například zelené střechy [8], kvalita ovzduší [9] nebo odpadové hospodářství [10].

Zelené střechy tvoří velmi zajímavou datovou sadu, která obsahuje místa, vhodná pro pokrytí střech vegetací. Lze je tedy zobrazit do mapy. Jediná nevýhoda této sady, ale i ostatních na Golemio dostupných, je dostupnost dat pouze pro území Hlavního města Prahy.

Kvalita ovzduší navíc obsahuje odkaz na API poskytované ČHMÚ, které vrací aktuální hodnoty. Přestože se zde nachází i soubor s měsíčním znečištění míst poskytované ve spolupráci s ČHMÚ, opět narazíme na problém, že data se týkají pouze Prahy.

Poslední statistiku však tvoří pouze zaznamenané odpady na území Prahy v CSV souboru.

Celkově lze Golemio označit za nevhodný zdroj dat, z důvodu omezenosti dat a překrývání se s jinými zdroji.

2.1.3 Český statistický úřad

Český statistický úřad, dostupný na url [11], nabízí velké množství různých statistik, týkající se České republiky. Pokud se omezíme na data vhodná pro naši aplikaci, lze zmínit například zemědělství, lesnictví, životní prostředí nebo bilance elektrické energie.

Všechny výše zmíněné datové sady tvoří roční data pro celé území ČR. S výjimkou bilance elektrické energie jsou však dostupná i data pro jednotlivé kraje. Veškerá data jsou uložitelná do formátu XML, který je vhodný pro následné automatické zpracování. Vyjmenované datové sady jsou srozumitelné, vhodné pro grafické zobrazení a lze je tedy označit za vhodný zdroj.

2.1.4 Otevřená data rezortu životního prostředí

Na tomto webovém portálu [12] lze najít celkem 88 datových sad týkajících se ekologie. Mezi vhodné volby lze zařadit například chráněná území, památné stromy, národní parky nebo biosférické rezervace. Přestože se jedná o zajímavé statistiky, nejsou vhodná pro naši práci. Jedná se totiž pouze o data vhodná pro mapové zobrazení. Přesto stojí za zmínění, jelikož jsou vhodná pro pozdější rozšíření.

2.1.5 Shrnutí a volba dat

Po zhodnocení dostupných zdrojů jsme se rozhodli pro ČHMÚ, konkrétně pro denní historická hydrologická a meteorologická data. Jedná se o přehledné datové sady, které jsou srozumitelné pro běžného uživatele. Velikost a rozsah dat je pro naši práci dostatečný. Zároveň se jedná o ideální zdroj pro zobrazení pomocí map a grafů.

Přestože kvalita ovzduší je zajímavá a bylo by rozumné ji zvolit, z důvodu rozsahu a náročnosti pro zpracování těchto dat jsme se rozhodli ji nepoužít pro tuto práci. Jelikož chceme pokrýt větší rozsah dat, dáme přednost většímu počtu jednodušeji dostupných datových sad. Přesto se jedná o velmi vhodnou statistiku pro přidání do budoucna.

Český statistický úřad jsme se rozhodli nepoužít, jelikož se jedná o malé datové sady pro samostatné použití. Pokud bychom se rozhodli pro kombinování s dalším zdrojem dat, museli bychom zvolit vhodná data, která se doplňují pro zachování smyslu aplikace. Přesto je tento zdroj vhodný pro budoucí rozšíření našeho portálu.

Otevřená data rezortu životního prostředí nabízí vhodná data, avšak pouze pro zobrazení pomocí mapy. Pro tuto bakalářskou práci jsou tedy nevhodná, lze jej na druhou stranu přidat později.

Golemio jsme se rozhodli nepoužít jako zdroj dat, jelikož obsahuje data dostupná z ostatních zdrojů. Další důvod je, že obsahuje data pouze pro území Prahy. Jedním z našich cílů je vizualizace historických dat pro celé území České republiky, což je přímo v rozporu s poskytovaným rozsahem dat.

Podobných zdrojů, zaměřujících se jen na určitou lokalitu, lze najít mnoho. V práci jsme je již nezmínili, jelikož je nelze použít, stejně jako Golemio.

2.2 Analýza uživatelských požadavků

Tato sekce pojednává o funkčních a nefunkčních požadavcích na finální řešení. Nejprve si v podsekci 2.2.1 zadefinujeme funkční požadavky na systém. Jedná se o požadavky funkcí, které systém musí bezpodmínečně obsahovat. Oproti tomu nefunkční požadavky nám definují obecné vlastnosti systému. Těmi se zabýváme v podsekci 2.2.2.

2.2.1 Funkční požadavky

Jak již bylo zmíněno, funkční požadavky nám definují důležité funkce systému, které výsledné řešení musí obsahovat. Ze zadání a konzultací nám vyplynuli následující funkční požadavky:

FP1 Uživatel má možnost zvolit si data, která chce zobrazit.

FP2 Uživatel si může zvolit úroveň, na jaké se mají zobrazovat data (stanice, kraj, celá Česká republika).

FP3 Uživatel si může zvolit granularitu zobrazovaných dat (Zda chce roční, měsíční nebo denní statistiku).

FP4 Uživatel má možnost porovnávat data z různých zdrojů.

FP5 Uživatel má možnost porovnávat data napříč různými kategoriemi.

FP6 Uživatel si zobrazuje data bez nutnosti registrace účtu.

2.2.2 Nefunkční požadavky

Z funkčních požadavků, konzultací a zadání nám vyšli následující nefunkční požadavky:

NP1 Systém musí být navržen tak, aby jej bylo možné provozovat na zařízení s operačním systémem Linux, MacOS a Windows 10. Vizualizační část musí jít zobrazit na jakémkoliv zařízení, které má nainstalovaný Interner Explorer (verze 11), Edge (verze 14 a novější), Firefox (verze 52 a novější), Chrome (verze 49 a novější) nebo Safari (verze 10 a novější).

NP2 Zvoleným klientem je tenký webový klient.

NP3 Systém musí být navržen tak, aby bylo možné jednoduše přidat další datové typy.

NP4 Systém musí podporovat různé jazykové lokalizace.

NP5 Systém musí podporovat tmavý a světlý režim uživatelského rozhraní

NP6 Výsledná webová aplikace je přehledná, moderně vypadající a podporuje responzivní design.

2.3 Analýza existujících řešení

Nyní, když máme zadefinované požadavky na finální řešení, se můžeme podívat na již existující řešení a zhodnotit, jak moc odpovídají našim požadavkům. Analyzujeme webové portály týkající se ekologie a životního prostředí jak na území České republiky, tak i zahraniční weby. Webové portály jsou seřazeny v abecedním pořadí.

2.3.1 Agentura ochrany přírody a krajiny České republiky

Agentura ochrany přírody a krajiny České republiky je agentura pro ochranu a péči chráněných území. Portál je dostupný na adrese [13], a lze se na webu podívat na mapy, které zobrazují biotopy, chráněná území nebo místa aplikované ochrany přírody.

Design celého webového portálu je přehledný a moderní. Mapy jsou interaktivní a lze si zapínat a vypínat jednotlivé datové vrstvy.

Z našich požadavků portál splňuje FP1 a FP6. Data jsou přístupná bez registrace a lze si vybrat kategorii, která uživatele zajímá.

2.3.2 Climate.gov

Climate.gov je americký webový portál zaměřující se na informace ohledně klimatu a počasí. Samotný portál je dostupný na url [14].

Co se týče vizualizace a dat, k dispozici je historický vývoj teploty, oxidu uhličitého nebo například výšky hladiny moří. Vizualizace dat je dostupná formou grafů nebo pomocí mapy. Hlavní oblastí, pro kterou jsou data určena, tvoří Spojené státy americké.

Pokud provedeme analýzu našich požadavků, web splňuje FP1, částečně FP5 a FP6. Web tedy dovoluje uživateli si zvolit data, která jej zajímají, na druhou stranu může porovnávat data pouze ve třech kategoriích zároveň. Podmínka přístupu k datům bez registrace je tedy také splněna.

Co se týče designu, web je přehledný, ale kombinuje moderní design se zastaralým návrhem stránek. Další negativum tvoří chybějící podpora responzivního designu.

2.3.3 Česká geologická služba

Česká geologická služba je dostupná na adrese [15]. Jedná se o veřejný portál týkající se geologie. Na stránce si lze zobrazit různá geologická data pomocí map. Tyto mapy jsou interaktivní a přehledné.

Samotný webový portál je moderně vypadající, přehledný a podporuje responzivní design. Z našich funkčních požadavků web splňuje FP1 a FP6.

2.3.4 ČHMÚ

Web ČHMÚ jsme si již představili dříve. Nachází se na url [7] a nyní se zaměříme na možnosti vizualizace statistik. Na stránkách si lze prohlídnout jak aktuální data, tak historická.

Aktuální data si lze zobrazit pomocí mapy, avšak vzhledově se jedná celkově o zastaralý design portálu, který není responzivní. Po kliknutí na stanici se zobrazí aktuální data na samostatné stránce pomocí tabulky, pod níž je umístěn graf. Aktuální data o ovzduší jsou poskytnutá pouze pomocí dlouhé tabulky. Pokud tedy nehledáme konkrétní hodnotu, jedná se o velmi

nepřehledný způsob. Historická data jako taková jsou zobrazovány jako seznam roků. Po zvolení si konkrétního, dostaneme tabulku s kraji a hodnotami pro jednotlivé měsíce.

Co se týče vizualizace pomocí map, k dispozici jsou pěkné a přehledné statické mapy. Rádi bychom zmínili, že v době psaní této části (25. 4. 2021) prochází portál postupnou modernizací. Nově vzniklé části již hodnotíme jako přehledné.

Při analyzování webu, jsme došli ke zjištění, že portál splňuje FP1 a FP6. Základní vizualizace je k dispozici bez přihlášení. Také uživatel si může vybrat data jaké si chce zobrazit, avšak množina dat je mnohem omezenější než množina dat ke stažení.

2.3.5 Český statistický úřad

Český statistický úřad provozuje veřejnou databázi data na adrese [16]. Vizualizace však není dostupná u každého datového zdroje. Pokud je, tak je možnost zvolit náhled pomocí tabulky, popřípadě grafu nebo mapy. Je důležité zmínit, že vizualizace dat není primární účel tohoto webu.

Samotný design vypadá moderně a přehledně, avšak nepodporuje responzivní design. Co se týče vizualizace pomocí mapy, nejedná se o interaktivní mapu.

Z našich požadavků splňují FP6 a částečně FP1, FP2 a FP3. Uživatel pro přístup k datům nepotřebuje přihlášení, avšak zobrazení dat, stejně jako granularita a úroveň se liší sada od sady a také nemusí být vůbec k dispozici.

2.3.6 Golemio

Jak jsme již dříve zjistili, Golemio je portál Hlavního města Prahy pro poskytování veřejných dat, dostupný na url [17]. Vizualizace dat je dostupná pomocí grafů.

Web jako takový vypadá moderně, přehledně a podporuje responzivní design. Z našich požadavků splňuje FP1 a FP6.

2.3.7 Informační systém statistiky a reportingu v životním prostředí

ISSaR je web ministerstva životního prostředí, určený pro vizualizace ekologických dat. Portál je dostupná na adrese [18].

Jedná se o moderní přehledný web. Data jsou zobrazována pomocí tabulek, map a grafů. Z našich požadavků splňuje FP1,

2.3.8 In–Počasí

In–Počasí je webový portál, primárně k zobrazení aktuálního počasí. K tomu lze zobrazit i historická data až do 18. století. Web je dostupný na adrese [19].

Web je moderní, přehledný a responzivní. Data jsou zobrazována jak pomocí statické, tak i pomocí interaktivní mapy. Následuje přehledná tabulka s hodnotami. Je potřeba však zmínit, že mapa nemusí být vždy dostupná u historických dat.

Web splňuje FP1, FP6 a částečně FP2, FP3 a FP5. Uživatel může zvolit úroveň dat, granulitu nebo další statistiku pouze u vybraných kategorií.

2.4 Případy užití

Pro naši aplikaci má smysl se zabývat pouze třemi případy užití, a to interakcí uživatele se systémem pro zobrazení požadovaných dat, úprava parametrů zobrazených dat a přepnutí se do porovnávacího režimu.

2.4.1 Zobrazení dat

Aktéři: Aktéři tohoto případu užití jsou uživatel a systém.

Scénář: Systém zobrazí uživateli webovou stránku. Uživatel si následně zvolí kategorii dat, která ho zajímá. Následně mu systém zobrazí formulář, ve kterém si uživatel vybere úroveň dat (stanice, kraj, stát), zdroje dat, časový rozsah, granulitu dat a konkretní data a formu zobrazení. Po zvolení dat uživatel odešle požadavek systému, ten jej zpracuje, najde požadovaná data v databázi, zpracuje je a zobrazí uživateli.

2.4.2 Úprava parametrů zobrazených dat

Aktéři: Aktéři tohoto případu užití jsou uživatel a systém.

Scénář: Výchozí stav tohoto scénáře je takový, kdy uživatel vidí na obrazovce webovou stánku, na které jsou zobrazená data s požadovanými parametry. Následně se uživatel rozhodne upravit parametry zobrazovaných dat, jelikož chce přidat nějaká další data pro zobrazení (upravení časového rozpětí, změna měřítka, přidání další stanice atd.). Uživatel tedy klikne na příslušné tlačítko a systém mu následně zobrazí příslušný formulář, který je předvyplněný dříve aktuálně zvolenými parametry. Uživatel následně upraví parametry dle potřeby a potvrdí své změny. Systém následně tyto změny zpracuje, v případě potřeby nalezne požadovaná data v databázi a následně je znovu zobrazí.

2.4.3 Zobrazení dat

Aktéři: Aktéři tohoto případu užití jsou uživatel a systém.

Scénář: Výchozí stav tohoto scénáře je stejný, jako ve scénáři 2.4.2, kdy uživatel vidí zobrazená data na obrazovce pomocí webového portálu.

Uživatel se následně rozhodne porovnat data s daty jiné kategorie. Toho docílí kliknutím na tlačítko pro přepnutí systému do porovnávacího režimu. Systém na to zareaguje zobrazením příslušného formuláře, který je předvyplněný dle aktuálně platných parametrů. Uživatel následně upraví parametry dle potřeby a klikne na tlačítko zobrazit data. Tím zadá požadavek systému, který jej zpracuje, v databázi dohledá potřebná data a následně je zobrazí uživateli.

Kapitola 3

Návrh

V této sekci se zabýváme návrhem praktické části této práce. V sekci 3.1 si zvolíme a popíšeme architekturu výsledného řešení.

Následně v sekci 3.2 si podrobněji rozebereme zvolené datové sady. Projdeme si jejich strukturu a zdroj, odkud je získáme. Za použití těchto znalostí navrhneme konkrétní způsob, jakým je získáme a zpracujeme. Také se věnujeme technologiím, které potřebujeme pro následnou implementaci našeho návrhu.

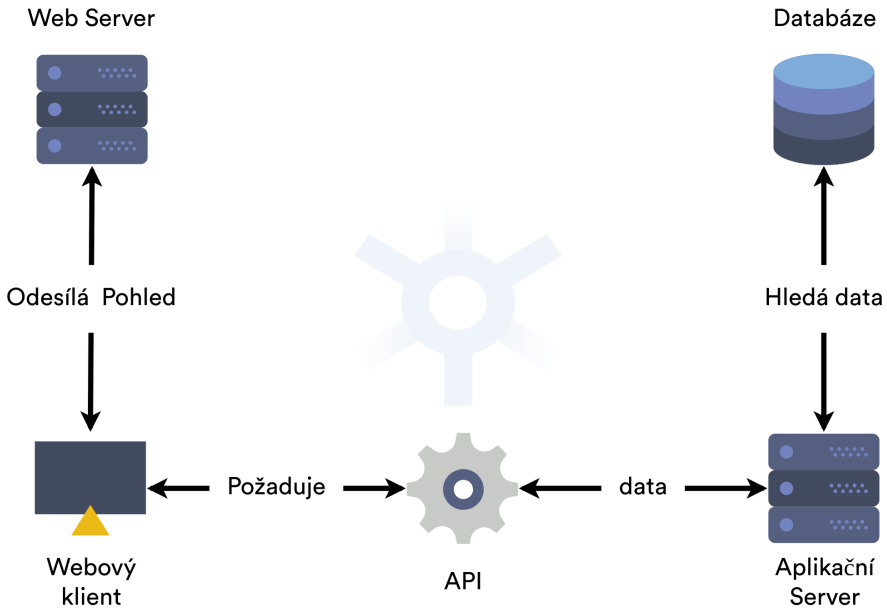
V sekci 3.3 se věnujeme návrhu databáze, jelikož je potřeba získaná data strukturovaně uložit pro jejich následné použití. Také si zde projdeme dostupné databáze a vybereme pro nás tu nejvhodnější.

Pro přístup k datům uložených v databázi je potřeba navrhnout vhodné API. V sekci 3.4 se proto věnujeme návrhu samotného backendového serveru, který nám umožní přístup k těmto datům. Provedeme zde obecný návrh backendu, probereme si dostupné technologie vhodné pro implementaci našeho řešení. Také si představíme technologie dostupné pro tvorbu API a pomocí zvolené technologie GraphQL navrhneme konkrétní API pro naší aplikaci.

V poslední části navrhneme webový portál. Frontendové části je věnována celá sekce 3.5. Nejprve si shrneme základní informace o designovém jazyce Material Design v podsekci 3.5.1. Na základě těchto pravidel následně provedeme obecný návrh webového rozhraní. Po jeho návrhu, je potřeba si zvolit jeden z dostupných JavaScriptových frameworků, ve kterém implementujeme následně finální návrh.

3.1 Architektura

Protože navrhujeme celý systém od základu, v této kapitole se zabýváme obecným návrhem celého praktického řešení. Samotné návrhy dílčích částí jsou probírány v následujících sekcích. Zároveň prezentovaný návrh v této sekci je nezávislý na výsledně zvolených technologiích.



Obrázek 3.1: Diagram zachycující obecný návrh architektury.

Když se podíváme na obrázek 3.1, můžeme vidět, že návrh architektury jsme rozdělili do 5 částí. Jednotlivé části popisujeme směrem od uživatele.

První část tvoří tenký webový klient. Z pohledu třívrstvé architektury se jedná o prezenční vrstvu, která zobrazuje data a reaguje na pokyny ze strany uživatele. Bude to jediná vrstva interagující přímo s uživatelem. V našem případě se jedná o webový portál, který uživatel uvidí ve svém prohlížeči. V naší aplikaci bude zobrazovat uživateli zvolená data v mapě a grafech. Mezi výhody webového klienta patří:

- *Multiplatformovost* – uživatel má možnost spustit klienta na jakémkoli zařízení s libovolným operačním systémem. Jediné co k tomu potřebuje je moderní webový prohlížeč podporující ES7 (ECMA starndar pro JavaScript ve verzi 7. Dokumentace k dispozici na url [20].). V našem případě je ještě potřeba navíc mít povolený JavaScript v prohlížeči.

- Uživatel nemusí nic instalovat, vše se stáhne z internetu pomocí HTTP protokolu.

- Uživatelovo zařízení není tolik vytěžované, jelikož výpočetně náročné operace probíhají na serveru a uživatelské zařízení se stará jen o zobrazování dat a zpracovává požadavky uživatele.

Další část, kterou na obrázku 3.1 vidíme, je Web server. Jedná se o server, který se stará o odesílání webového klienta uživateli. Dále reaguje na příchozí požadavky z klienta tak, že odesílá jednotlivé webové stránky, které má uživatel vidět. Spolu s webovým klientem v našem návrhu tvoří frontendovou část.

V rámci části API si definujeme rozhraní pro komunikaci frontendové a backendové části. Jedná se o sadu pravidel, pomocí které mohou komunikovat dvě a více částí systému, či různé systémy navzájem. Hlavním důvodem pro tvorbu API je vytvoření rozhraní pro získávání dat pro webového klienta z databáze.

Předposlední částí návrhu architektury tvoří Aplikační server. Pokud opět použijeme terminologii třívrstvé architektury, jedná se o Aplikační vrstvu a zde se nachází naše byznys logika. Hlavním úkolem aplikačního serveru je hostování backendové služby, která zprostředkovává komunikaci mezi frontendem a databází. mezi další úlohy backendu patří předzpracování dat. Jelikož v databázi si ukládáme denní statistiky, pro potřeby zobrazení měsíčních a ročních statistik je potřeba tyto data vytvořit z denních. Abychom nezatěžovali síť a zařízení uživatele, toto zpracování dat probíhá právě zde, na backendu.

Poslední část tvoří samotná databáze. Jedná se o perzistentní úložiště dat, se kterými pracujeme. V třívrstvé architektuře se označuje jako Datová vrstva.

Jednotlivé vrstvy spolu spolupracují a tvoří dohromady celistvý systém, který má za úkol zobrazit uživateli požadovaná data. Od databáze, která uchovává data, přes backend, který předzpracovává data a následně je přes API pošle frontendu, který je zobrazí. Jedná se o standardní architekturu webové aplikace.

3.2 Získávání dat

Tato sekce se zabývá analýzou stránek ČHMÚ, jelikož jsme si jej zvolili jako zdroj dat. Následně provedeme rozbor struktury dat, která získáme z webu ČHMÚ. Na základě poznatků z této části následně navrhneme způsob, pomocí kterého budeme získávat námi zvolená historická data. Následně se zabýváme výběrem technologie pro následnou implementaci této části.

3.2.1 ČHMÚ a data

Nejprve si probereme denní historická hydrologická data, jelikož se jedná o menší datový soubor. Data lze získat z portálu ČHMÚ z webové adresy [21]. Po otevření url, vidíme jedinou kategorii – průměrný denní průtok. Po kliknutí na tuto kategorii se nám zobrazí nabídka krajů, která se po zvolení jednoho kraje mění na výpis stanic pro daný kraj. Kliknutím na zvolenou stanici si stáhneme příslušný zip soubor, který obsahuje data.

Před popisem struktury zip souboru si nejprve rozebereme názvy stanic na webu, jelikož se nám to hodí později při návrhu získávání dat. Při pozorném sledování seznamu stanic si můžeme všimnout vzorce pojmenování odkazu pro získání datového souboru. Ten je totiž tvořen třemi částmi – číselným kódem stanice, názvem řeky a nakonec obec, ve které se stanice nachází.

Nyní se můžeme podívat na strukturu souboru, který jsme si stáhli. Již na první pohled nás může zajmout název souboru, který je tvořen číselným kódem stanice, kterému je dán prefix *QD_*.

Po otevření zipu vidíme tři CSV soubory. Soubor s příponou *_data* obsahuje jednotlivá denní měření. Každý řádek tohoto souboru tvoří jedno samostatné měření a strukturu řádku nám zachycuje následující tabulka 3.1.

kód stanice	typ dat (QD)	rok	měsíc	den	naměřená hodnota
int	string	int	int	int	double

Tabulka 3.1: Struktura jednotlivých řádků hydrologických dat.

Zbylé dva soubory obsahují dodatečné informace o jednotlivých sloupcích datového souboru a podrobnější informace o stanici. Tyto metadata tvoří poloha stanice, název toku a pobočka pod kterou stanice spadá. Důvod existence dvou souborů metadat je jazyková mutace – česká a anglická.

Pokud se podíváme blíže na meteorologická data, tak po otevření webové stránky nás přivítá celkem jedenáct kategorií dat:

1. průměrná denní teplota vzduchu
2. maximální denní teplota vzduchu
3. minimální denní teplota vzduchu
4. průměrná denní relativní vlhkost vzduchu
5. denní úhrn srážek
6. výška nově napadlého sněhu
7. celková výška sněhové pokrývky
8. denní úhrn doby trvání slunečního svitu

9. průměrný denní tlak vzduchu
10. průměrná denní rychlost větru
11. maximální rychlost větru

Rádi bychom Vás upozornili, že zbytek podsekce reflektuje návrh získávání dat z webu ve verzi, která byla dostupná na přelomu prosince roku 2020 a ledna roku 2021. Text rešerše a návrhu totiž vznikal v této době. Z důvodu funkčnosti této části v době psaní zbytku práce (platné k datu 20. 4. 2021) nebyla implementace aktualizována, přestože je na webu ČHMÚ nasazena nová verze.

Aktuální verze vypadá tak, že po zvolení kategorie se nám zobrazí tabulka, která obsahuje jednotlivé kraje. Kliknutím na námi zvolený kraj dojde k zobrazení nové tabulky, jejíž sloupce tvoří název stanice, název okresu, datové období a název souboru.

Původně se zobrazil seznam stanic ve formátu název obce, (volitelné) název městské části, (volitelné) okres a datový rozsah.

Zároveň url pro odpověď API se liší od původní, pro kterou bylo navrhované řešení. Pokud pošleme dotaz na původní url, API nám stále odpoví. Kliknutím na řádek dojde ke stažení datového souboru.

Datový soubor je opět zip archiv a název tohoto souboru tvoří kód stanice, k němu přidaná zkratka kategorie a přípona *_N.csv*. Obsah archivu tvoří jediný CSV soubor, který obsahuje na začátku metadata stanice (tabulka 3.2), metadata přístroje pro měření (tabulka 3.3), jednotky, popis dat a seznam příznaků dat. Dále následují samotná data, kdy řádek tvoří jednotlivá měření ve formátu 3.4

ID stanice	jméno stanice	poloha stanice	začátek a konec měření
string	string	double	date

Tabulka 3.2: Struktura metadat meteorologické stanice.

typ přístroje	začátek a konec měření	výška umístění
string	date	double

Tabulka 3.3: Struktura metadat přistroje meteorologické stanice.

rok	měsíc	den	hodnota	příznak
int	int	int	double	string

Tabulka 3.4: Struktura jednotlivých řádků meteorologických dat.

Nyní, když jsme si popsali strukturu dat, můžeme navrhnout způsob, jakým tyto data získáme. Vhodné je zmínit, že web ČHMÚ využívá JavaScript

a jedná se tak o dynamicky generovanou webovou stránku. Jelikož jsme si zvolili pro získání dat framework Scrapy, je potřeba zmínit, že neumí stahovat soubory, které nejsou dostupné přes href atribut. My jej využijeme k získání seznamů stanic pro jednotlivé kraje spolu s kódy stanic. Následně si napíšeme skript, kterému předáme tyto seznamy, a následně stáhneme jednotlivé soubory. Dalším problémem frameworku Scrapy je, že bez použití dalších knihoven si neporadí s dynamicky generovanou stránkou. Jelikož kódy stanic jsou skryté na stránce a stahování souboru je provedeno pomocí funkce v JavaScriptu, obejdeme toto pomocí scrapovaní HTML souboru, který vrací API při volání souboru pro zobrazení seznamu stanic daného kraje a typu dat. Požadovanou adresu získáme spojením společné části URL, kterou nám zachycuje tabulka 3.5 s relativními URL, které nám zachycují následující tabulky.

Jednotná část URL:
https://www.chmi.cz/files/portal/docs/

Tabulka 3.5: Tabulka zachycující jednotnou části URL.

Adresu, ze které získáme potřebná HTML, získáme spojením společné části URL z tabulky 3.5 s částmi pro meteorologická a hydrologická data z tabulky 3.6. Název stanice získáme z textu HTML prvku seznamu a kód příslušné stanice z jeho href atributu. V případě hydrologických stanic, již samotný text obsahuje kód stanice, a proto jej vezmeme odsud. Takto vytvoříme celkem dvanáct seznamů pro každý datový typ jeden.

Relativní URL pro meteorologické stanice ve zvoleném kraj:
../meteo/ok/denni_data/{typ_dat}/HTML/{kraj}.html
Relativní URL pro hydrologické stanice ve zvoleném kraj:
../hydro/denni_data/QD/HTML/{kraj}.html

Tabulka 3.6: Tabulka obsahující relativní URL pro získání HTML se seznamem stanic.

Následně tyto seznamy využijeme jako vstup pro stahovací skript, který stáhne data ze serveru ČHMÚ. Zároveň využijeme faktu, že také url pro stažení dat je standardizovaná, a tak lze stahovat data pomocí API požadavků. Cílová url vznikne dosazením kódu stanice, kraje a případně datového typu do url, kterou získáme spojením společné části URL z tabulky 3.5 s relativní částí URL, která je uvedena v tabulce 3.6.

Stažený zip archiv stačí rozbalit a rozdělit obsah do dvou souborů: data.csv a stanice.csv pro informace o stanici. Abychom mohli vložit data pomocí skriptu do databáze, musíme převést stažená data na jednotný formát. Také je potřeba sloučit datové typy, které spolu souvisí. Například spojit minimální, průměrnou a maximální teplotu v jeden celek.

Relativní URL pro data konkrétní meteorologickou stanici:
../hydro/denni_data/QD/{kraj}/{ID_stanice}
Relativní URL pro data konkrétní hydrologickou stanici:
../meteo/ok/denni_data/{datovy_typ}/{kraj}/{ID_stanice}

Tabulka 3.7: Tabulka obsahující relativní URL pro získání dat.

3.2.2 Technologie

Tato podsekce se věnuje popisu programovacího jazyka Python, ve kterém je následně implementovaná celá část zabývající se získáváním dat. Následně je představen Scrapy, scrapovací framework pro získání dat.

3.2.2.1 Python

Python je vysokoúrovňový skriptovací jazyk. Autorem je Guido van Rossum, který vytvořil tento jazyk na začátku 90. let 20. století v Nizozemsku. Jedná se o nástupce jazyka ABC. [22]

Mezi jeho vlastnosti patří specifické formátování kódu. Nepíše se zde na konci řádku středník a cyklus se neuzavírá do složených závorek. Místo toho se vše řídí odsazením. Díky tomu je kód snadno čitelný a jazyk jako takový snadný na naučení. Částečně mezi výhody patří dynamické typování proměnných. To nám umožňuje řešit datové typy až za běhu programu, pokud je neznáme dopředu. Následek této vlastnosti je však nutnost ověřování si vstupu funkcí, jelikož Python nám neumí zařídit, aby do námi napsané funkce, která akceptuje pouze číselný vstup, nešel předat textový řetězec.

Mezi další výhody patří, že vzniklý kód je multiplatformní, tedy jej lze spustit na jakémkoliv operačním systému. Dále mezi výhody patří také typ licence, jelikož se jedná o Open-Source.

Přestože lze tuto část práce implementovat i v jiných programovacích jazycích jako například Java nebo C#, použijeme právě Python ve verzi 3. Hlavní výhodou pro použití tohoto jazyka tvoří knihovna Pandas, která slouží ke zpracování rozsáhlých dat. Dále využijeme tento programovací jazyk k implementaci scraperu pomocí frameworku Scrapy.

3.2.2.2 Scrapy

Scrapovaní je technika získávání dat z webové stránky nebo API pomocí webových crawlerů, kteří prochází HTML stránky. Tuto stánku lze stáhnout napřímo, nebo předat přístup k prohlížeči, odkud si stránku crawler přebere. Požadovanou stránku crawler dostane jako parametr a následně pomocí definovaných pravidel crawler extrahuje požadovaná data, která ukládá do zvolené struktury, například JSON nebo CSV.

Pro tuto činnost lze využít scrapovací framework Scrapy, který postavený na programovacím jazyce Python. Uživatel si vytvoří novou třídu, která dědí jedno z před připravených spiderů. Následně jako parametr se předá URL požadovaného webu a pomocí Pythonu se napíší pravidla pro navigaci a extrakci dat. Následně je možné upravit formátování výstupu. Při spouštění spideru se jako parametr předá typ výstupu.

Jelikož píšeme celou tuto část v Pythonu, tento framework je jasnou volbou. Spousta frameworků jsou komerční nástroje, kdy uživatel interaktivně zadá prvek na stránce, ze kterého chce získat data a vše proběhne na pozadí. Uživatel tak dostane jen požadovaný výstup. Zároveň nemá smysl znovu vymýšlet kolo, tedy psát si vlastní scraper. Jediná nevýhoda Scrapyho je, že neumí stahovat soubory, které nejsou dostupné pomocí url. To je způsobeno tím, že Scrapy pracuje se statickou stránkou a neumí vykreslit změny HTML souboru pomocí JavaScriptu. Proto jej využijeme pouze pro získání identifikátorů stanic, názvu souborů a datových typů.

3.3 Databáze

V této sekci se zabýváme návrhem modelu databáze pro uložení dat, která následně zobrazujeme uživateli ve webové aplikaci. Samotnému návrhu modelu se věnuje podsekce 3.3.1 a v podsekci 3.3.2 si projdeme dostupné technologie pro realizaci databáze.

3.3.1 Databázový model

Z analýzy dat jsme sestavili vhodný návrh pro databázi, který je zachycen na obrázku 3.2. Výsledný návrh reflektuje podmínku jednoduchosti přidání dalších datových sad, jelikož tabulku *Measurements* jsme navrhli co nejvíce obecně. Zároveň lze přidávat i mezinárodní data, jelikož návrh je brán univerzálně a stačí tak vložit data do již existujících tabulek. V případě potřeby přidání speciálních záznamů, kterým tabulka *Measurements* nevyhovuje, stačí vytvořit novou tabulku pro tato měření a napojit je na již existující tabulku stanice nebo kraj. Nyní si popíšeme jednotlivé tabulky.

Tabulka *region* slouží pro ukládání jednotlivých krajů. Její atributy jsou:

id umělý identifikátor

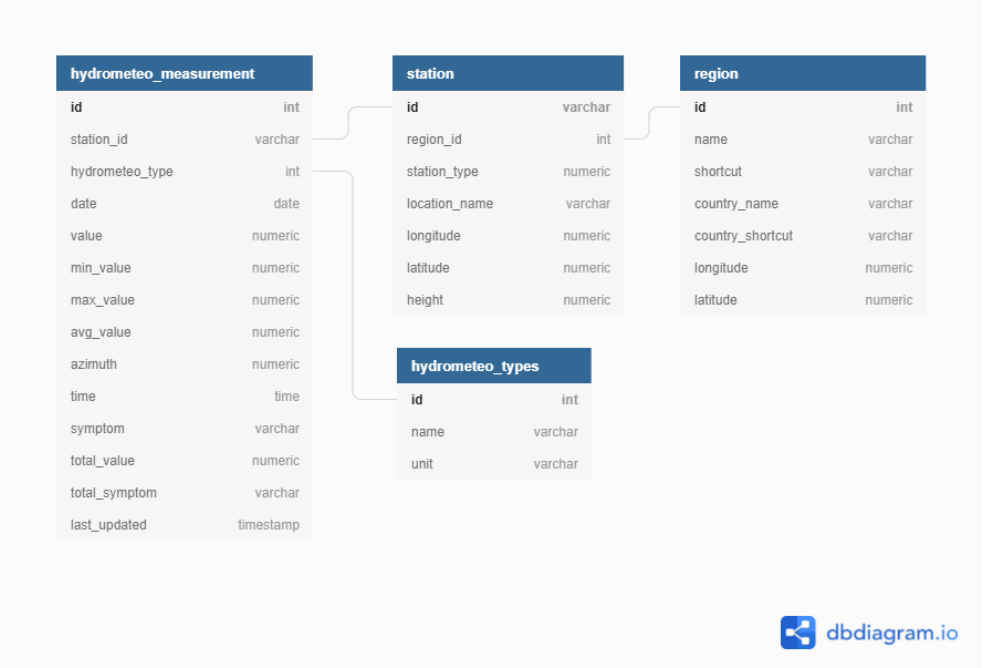
name název kraje

shortcut oficiální zkratka kraje

country_name název státu, kterému kraj patří

country_shortcut dvouznaková zkratka státu

latitude zeměpisná šířka



Obrázek 3.2: Diagram návrhu databáze (datové typy platné pro Postgres).

longitude zeměpisná délka

Tabulka *stanice* slouží k uchování si údajů o stanici, ze které pochází uložená měření. Relace mezi tabulkami stanice a region vyjadřuje vztah 1 : N a určuje, která stanice přísluší jednotlivým krajům. Stanice je ve vztahu s právě jedním krajem. Atributy stanice jsou:

id umělý identifikátor stanice převzatý z databáze ČHMÚ.

region_id umělý identifikátor regionu, do kterého stanice patří

station_type bitová maska určující druhy měření touto stanicí

location_name název lokace, kde se stanice nachází

latitude zeměpisná šířka

longitude zeměpisná délka

height nadmořská výška

Další tabulku tvoří *hydrometeo_types*, která definuje typ měření. Atributy tvoří:

id umělý identifikátor

name název měření

unit fyzikální jednotka/y měření

Poslední tabulku tvoří *hydrometeo_measurement*. Jedná se o tabulku pro uchování data samotných měření. Relace se stanicí nám říká, odkud měření pochází. Vztah je 1 : N, kdy měření má právě jednu stanici. Relace s hydrometeo_types nám říká, o jaký druh měření se jedná a relace je opět 1 : N, kdy měření má právě jeden typ. Atributy jsou:

id umělý identifikátor

station_id umělý identifikátor stanice

hydrometeo_type umělý identifikátor typu měření

date datum měření

value naměřená hodnota, pokud existuje

min_value naměřená minimální hodnota, pokud existuje

max_value naměřená maximální hodnota, pokud existuje

avg_value průměrná hodnota měření, pokud existuje

azimuth směr měření (použito pro uchování si směru maximálního větru), pokud existuje

time čas měření (použito pro uchování si času maximálního větru), pokud existuje

symptom příznak měření (textový popis), pokud existuje

total_value celková naměřená hodnota, pokud existuje

total_symptom příznak celkové hodnoty měření (textový popis), pokud existuje

last_updated časové razítko poslední úpravy měření

3.3.2 Technologie

Z důvodu nutnosti vyhledávat a filtrovat data, které se zobrazují, je vhodné zvolit relační databáze, takže NoSQL databáze zastoupené například MongoDB nepřichází v úvahu.

Pro připomenutí si představíme stručně co je to relační databáze a SQL. Relační databáze uchovává data uložená v tabulkách. Tyto tabulky mají definované sloupce reprezentující atributy, které si chceme uchovat a jednotlivé

řádky tvoří samotné záznamy. Mezi vytvořenými tabulkami lze vytvářet relace, které definují vztahy mezi tabulkami. Kardinalita relace nám říká kolik záznamů z jedné tabulky může být v relaci se záznamem z druhé tabulky a může být typu 1 : 1, 1 : N nebo M : N. Následně si u relace definujeme povinnost prvku býti v relaci.

SQL je dotazovací jazyk pro komunikaci s relační databází. Jedná se o standard, který si implementuje každý dodavatel databází po svém, přesto základní příkazy jsou všude implementovány stejně. SQL nám dovoluje vytvářet dotazy pro tvorbu a smazání tabulek, vkládání, upravování a odebírání záznamů a nebo vyhledávání v databázi. Zároveň obsahuje základní agregační funkce, které si dodavatelé databází mohou rozšířit i o další.

Následující technologie jsou všechny relační databáze implementující SQL jazyk.

3.3.2.1 MySQL

Jedná se o nejpopulárnější technologii pro databáze, která vznikla původně jako open-source projekt švédské společnosti MySQL AB. Nyní jej od roku 2010 vlastní Oracle, jelikož odkoupil firmu Sun Microsystems, která dříve odkoupila společnost MySQL AB. [23, 24]

Mezi výhody MySQL patří možnost hostovat databázi v cloudovém řešení, jelikož je kompatibilní s mnoha poskytovateli cloudových služeb po celém světě. Další výhoda je multiplatformovost. Databázi je možné nainstalovat a používat na jakémkoliv operačním systému. V našem případě také lze mezi výhody zařadit i bezplatnou verzi, která je k dispozici pro jednotlivce a malé firmy. [23]

Naopak, pokud bychom byli velká firma hledající enterprise řešení, máme smůlu s bezplatnou licencí a musíme si zvolit jeden z před připravených platebních modelů. Další nevýhodou je i lehce zaostávající kvalita dokumentace oproti konkurenci. Co se týče implementace SQL jazyka, tak ta se zde liší o proti specifikaci. Najdeme zde užitečné funkce navíc, avšak některé standardní funkce jazyka SQL jsou implementovány odlišně. To může působit problém lidem, kteří mají zkušenosti s konkurenčním řešením. Jako poslední nevýhodu MySQL lze označit nedostatečnou možnost pro škálovatelnost. Pokud neplánujeme dopředu, je velice obtížné později rozšířit databázi. [23]

3.3.2.2 MS-SQL

MS-SQL je databázové řešení od společnosti Microsoft. Jedná se primárně o komerční řešení s využitím pro velké firmy, díky optimalizaci a integraci nástrojů vhodné pro firmy. Lze tak jednoduše provádět analýzy nad daty a generovat reporty. Mezi další výhodu patří široká komunita, kvalitní dokumentace a podpora cloudového řešení.

Velkou nevýhodu tvoří cena. Přestože je k dispozici verze zdarma, jedná se o ořezanou verzi vhodnou pro testování a učení se. Pro větší použití je potřeba zakoupit licenci. S tím souvisí nepřehledné podmínky licencí pro správné zvolení. [23]

3.3.2.3 Oracle

Jedná se o databázový systém vyvinutý společností Oracle. Mezi výhody tohoto řešení patří snadná škálovatelnost databáze, velmi kvalitní dokumentace, profesionální a rozsáhlá technická podpora nebo podpora cloudových výpočtů. Velkou výhodou je možnost synchronního zpracování dat a agregace více databází v rámci jedné technologie. [23]

Mezi hlavní nevýhodu patří cena. Ta je proti konkurenci značně vyšší, ať už za základní edici, nebo enterprise řešení. I když existuje bezplatná verze, pro rozumné použití má velice omezené funkce. Další nevýhodu tvoří zvýšené náklady na infrastrukturu, jelikož databáze vyžaduje značný výkon zařízení, na kterém běží. [23]

3.3.2.4 PostgreSQL

PostgreSQL je open-source databázový systém společnosti PostgreSQL Global Development Group. Mezi výhody patří škálovatelnost, podpora různých datových formátů, uživatelská podpora, rozsáhlá komunita, jednoduchá integrace nástrojů třetích stran, nebo samotný fakt, že se jedná o open-source projekt, který je vyvíjen ve spolupráci s komunitou a uživateli.

Mezi nevýhody naopak patří nekonzistentní kvalita napříč dokumentací a chybějící systém pro oznámení problému. [23]

3.3.2.5 Finální volba

Poté, co jsme vyhodnotili vyjmenované nástroje, rozhodli jsme se použít PostgresSQL. Důvodem byla široká podpora a bezplatná licence, která dovoluje plně využívat tuto technologii. Z důvodu nutnosti pořízení si licence jsme se rozhodli nezvolit řešení od Oracle a MS-SQL. Vzhledem k plánům škálovat aplikaci do budoucna, rozhodli jsme se nepoužít MySQL z důvodu vyhnutí se problémům a následné nutnosti znovu navrhování databáze.

3.4 Backend

V této sekci nejprve porovnáme dva přístupy pro tvorbu API, jeden zvolíme a následně navrhneme samotný backend.

3.4.1 Nástroje pro tvorbu API

3.4.1.1 REST

Dříve než můžeme porovnat a rozhodnout se, který přístup pro tvorbu API zvolíme, musíme si je představit. Začneme tedy RESTem. REST je zkratka pro representational state transfer. Jedná se o sadu architektonických stylů pro tvorbu API. Pokud API dodržuje tyto styly, mluvíme o něm jako o RESTful API. Tyto pravidla jsou:

Klient–server Rozdělení aplikace na server, který poskytuje data, a na klienty, kteří zobrazují příchozí data. Díky tomu lze docílit multiplatformovosti a škálovatelnosti aplikace. Zároveň lze rozvíjet server a klienta nezávisle na sobě.

Bezstavová komunikace Server si neukládá data o klientovi mezi dotazy. Každý dotaz je tedy separátní a po odeslání odpovědi je následně spojení ukončeno.

Kešovatelnost dat Každá odpověď od serveru musí být označena, zda ji lze či nelze uložit do keše. Pokud ano, data jsou uložena do keše a nedochází tak znovu ke stahování dat, jelikož je klient může znovu použít.

Jednotné rozhraní Nutnost zavést jednotné rozhraní pro výměnu dat. Díky tomu dosáhneme většího přehledu v komunikaci mezi komponentami systému. Toho dosáhneme pomocí jednoznačného označení zdroje pomocí URI, dovolíme klientovi manipulovat s daty, jelikož dostane dostatečné množství informací k tomu. Každá zpráva vyměněná mezi klientem a serverem je samopopisná, oba mají dostatek informací pro zpracování přijaté zprávy. Poslední omezení nám říká, že hypertext je stále dostupný po přístupu ke zdroji. Klient je tak schopný provést veškeré dostupné akce.

Systém organizovaný do vrstev Všechny komponenty serveru umístíme do vlastní vrstvy. Jednotlivé komponenty tak nevidí mimo komponenty, se kterými komunikují a dochází k zapouzdření. Servery tak tvoří hierarchii, která je pro klienta neviditelná.

Kód na vyžádání Možnost serveru poslat spustitelný kód klientovi na jeho vyžádání. Díky tomu lze rozšířit klientovu funkčnost.

Samotná komunikace je poté realizovaná pomocí HTTP protokolu. Z klienta je poslán dotaz na konkrétní URI, který je zpracován a následně je zaslána odpověď. Data jsou poslána pomocí těla dotazu a mohou být ve formátu JSON nebo XML. [25, 26]

3.4.1.2 GraphQL

GraphQL je dotazovací jazyk, který přenáší veškerou výkonnou část na server. Byl vytvořen v roce 2015 Facebookem jako alternativa k RESTu pro jejich specifické potřeby. Aktuálně se jedná o open-source technologii spravovanou The GraphQL Foundation.

Jak již název napovídá, GraphQL nám dovoluje strukturovat data do grafové struktury. Nejprve si zadefinujeme datové typy a vztahy mezi nimi. Díky tomu lze pomocí jednoho dotazu získat data, pro která bychom museli provést mnoho dotazů na jednotlivé příslušné zdroje. Zároveň nemůže dojí k nedorozumění mezi klientem a serverem, jelikož se jedná o silně typovaný jazyk.

Další vlastností GraphQL je, že klient může specifikovat jaká konkrétní data potřebuje. Pokud nás zajímá pouze jeden atribut třídy, GraphQL nám dovolí tento atribut získat bez nutnosti předání celého objektu. Na serveru sice dojde k zadefinování datových typů, avšak klient při tvorbě dotazu může specifikovat jaké atributy jednotlivých objetů požaduje.

Další výhodou je, že lze provozovat GraphQL API paralelně s REST API. Nejedná se totiž o řešení, které má REST nahradit, ale je to alternativa.

Nevýhoda tohoto řešení je, že převádí velké množství práce na server. Nejprve musí zpracovat požadavek, zjistit jaké data klient požaduje, zvolit způsob jak je získá a následně vrátit jen požadovanou množinu. [27, 28]

3.4.1.3 Finální volba

Jelikož mezi klientem a serverem bude docházet k velké komunikaci, rozhodli jsme se pro použití GraphQL. Díky možnosti specifikovat požadavek na potřebná data klientem, dojde ke snížení počtu dotazů na backend. Například všechny stanice dostupné ve všech regionech lze získat pomocí jednoho dotazu místo patnácti separátních na dvě URI. Přestože nedojde ke snížení přenesených dat, nebude docházet k častému využití sítě tak, jako v případě RESTu.

3.4.2 API rozhraní

Samotné GraphQL poslouchá na URI s cestou /graphql a má povolený grafický klient. Díky tomu může uživatel psát dotazy na API. V rámci naší práce si vystačíme pouze s jedním typem dotazu, a to query. Úlohou backendového serveru v této aplikaci je pouze poskytování dat klientům z databáze. Jelikož klient nemůže přidat data do databáze, je zbytečné vytvářet mutace (Mutace v QraphQL označuje typ dotazu, který neslouží k zobrazení dat, ale k manipulaci s nimi na perzistentním uložišti).

Všechny námi vytvořené query pro získání dat jsou potomci root query, což je v souladu s dokumentací GraphQL. Jelikož jednotlivé query pro samotné druhy měření jsou principiálně stejné a liší se pouze atributy ohledně

měření (maximální hodnota, průměrná hodnota...) shrneme je obecně. Dostupné query jsou:

- **region (id:int)** slouží k získání konkrétního kraje
- **regions ()** vrací všechny dostupné regiony
- **station (id:string)** slouží pro získání konkrétní stanice
- **stations ()** vrací všechny stanice
- **stationsInRegion (regionID:int)** vrací stanice dostupné pro zvolený kraj
- **stationsInRegionForDataType (regionID:int, dataType:string)** vrací stanice konkrétního typu (teplota, tlak...) pro zvolený kraj
- **hydrometeoType (id:int)** vrací konkrétní hydrometeorologický typ
- **hydrometeoTypes ()** vrací všechny hydrometeorologické typy
- **measurements** slouží pro získání množiny měření. Každý datový typ má devět variant. Varianty se liší úrovní zobrazení a periodicitou (denní, měsíční, roční). Argumenty tvoří seznam ID krajů/stanic, nebo zkratek států. Poslední argumenty tvoří data pro začátek a konec měření.

3.4.3 Doménový model

Obrázek 3.3 zachycuje návrh backendu pomocí doménového modelu. Výsledný návrh je v souladu s dokumentací GraphQL. Jednotlivé třídy jsou:

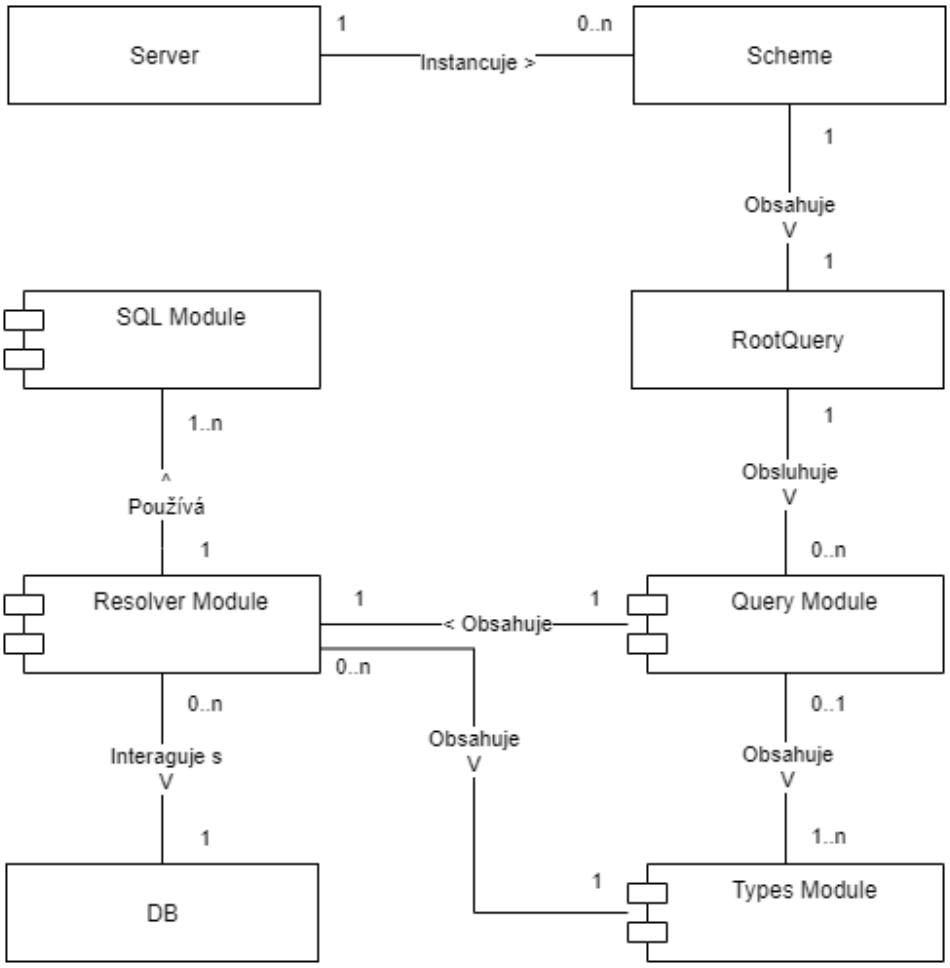
Server jedná se o samotnou instanci serveru. V rámci této třídy nastavíme port a URI na které bude backend poslouchat. Následně instancuje třídu Schéma.

Schéma tvoří samotné GraphQL. V rámci této třídy se vytváří celé schéma, které tvoří jednotlivé query a námi definované typy.

RootQuery nemá žádný jiný význam, než vytvořit hlavní query, která obsahuje všechny ostatní námi definované query.

DB je třída, která zprostředkovává komunikaci s databází.

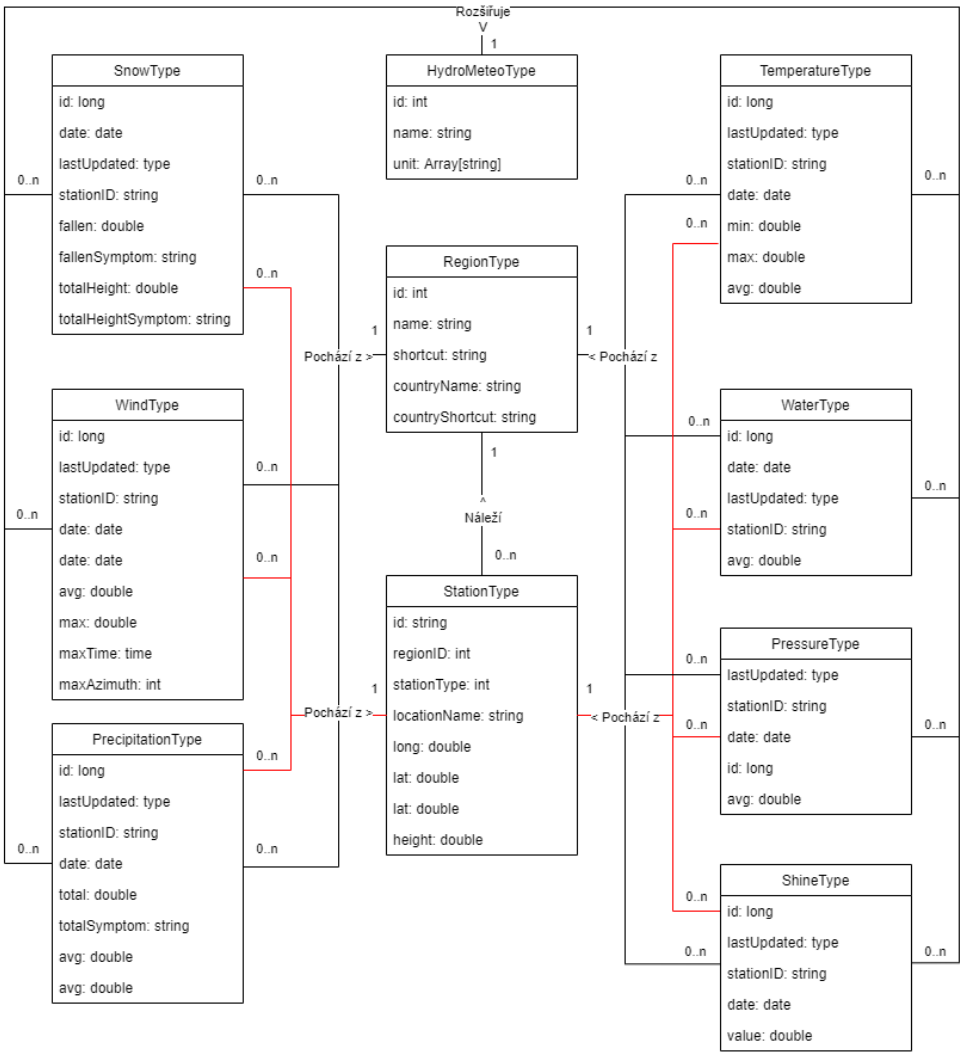
SQL Modul obsahuje veškeré dotazy pro získání dat z PostgreSQL databáze. Jedná se o sadu jednotlivých dotazů ve formě skriptů, které jsou následně předávány jako argument společně s parametry funkci volané z instance třídy DB.



Obrázek 3.3: Doménový model návrhu backendu.

Types modul tvoří definice jednotlivých námi definovaných typů a vazeb mezi nimi. Samotné typy jsou definovány pomocí atributů daných tříd s příslušnými datovými typy. Pomocí toho je GraphQL schopné získaná data převést na tyto typy, které vrací následně klientovi. Zároveň se zde vytváří vazby mezi jednotlivými typy. Doménový model pro *Types modul* je zobrazen na obrázku 3.4. Zde jsou zachyceny pouze vztahy mezi jednotlivými typy. Každý typ má vazbu na *query modul*, kdy jednotlivé query používají definice těchto typů. Dále se zde nenachází vazba na *resolver modul*, která je opět mezi všemi typy a tímto modulem. Tato vazba je potřeba vždy, když jsou dva typy v relaci. Pomocí resolverů jsme schopni získat data pro druhý typ ve vazbě.

Resolver modul tvoří funkce, které jsou volány pro získání dat z databáze,



Obrázek 3.4: Doménový model návrhu backendu.

aby následně mohla být převedena na námi definované GraphQL typy.

Query modul tvoří instance samotných query pro konkrétní datový typ. Jedná se o sadu pravidel, pomocí kterých GraphQL identifikuje datové typy argumentů, pozná co má vrátit a který resolver použít pro získání a následné převedení dat.

3.4.4 Technologie

Backend lze implementovat v mnoha technologiích. Přestože jsme si zvolili GraphQL technologii pro rozhraní API, tak na stránce dokumentace [29] vidíme, že veškeré velké programovací a skriptovací jazyky jsou podporované. Tuto část práce lze tedy implementovat v jazyku dle preference.

Jelikož backend není nijak rozsáhlý (zprostředkovává pouze komunikaci mezi backendem a frontendem) a není potřeba řešit autentizaci uživatele, rozhodli jsme se implementovat backend v jazyce JavaScript pomocí technologie NodeJS.

NodeJS umožňuje spouštět kód napsaný v jazyce JavaScript mimo webový prohlížeč. Díky tomu lze spustit server na zařízení, které má tento software nainstalovaný. Bez NodeJS by nebylo možné zvolit JavaScript jakožto jazyk pro implementaci této části.

3.5 Frontend

V rámci této podkapitoly se zaměříme na návrh frontendové části. Jelikož je potřeba vytvořit webový klient, který zobrazuje mapy a grafy, rozhodli jsme se využít jako programovací jazyk JavaScript. Z tohoto důvodu se v sekci 3.5.3 zabýváme pouze frameworky určenými pro tvorbu webové aplikace v JavaScriptu.

Nejprve si povíme základní principy Material designu v následující sekci 3.5.1, jelikož jsme si jej zvolili jakožto designový jazyk pro naši aplikaci. Jedná se o velmi rozšířený moderní designový jazyk, který klade důraz na přehlednost a srozumitelnost. Toho je docíleno pravidly, které minimalizují rušivé elementy a zaměřují se na to podstatné. Tyto vlastnosti splňují naše požadavky na UI. Dodržením těchto pravidel tak získáme velmi přívětivé GUI, které bude vypadat moderně a uživatel jej bude schopen intuitivně ovládat. Zároveň uživatel rychle na webu nalezne požadovanou informaci. Není tedy třeba vymýšlet si vlastní designový jazyk, ale použijeme dlouhodobě osvědčený.

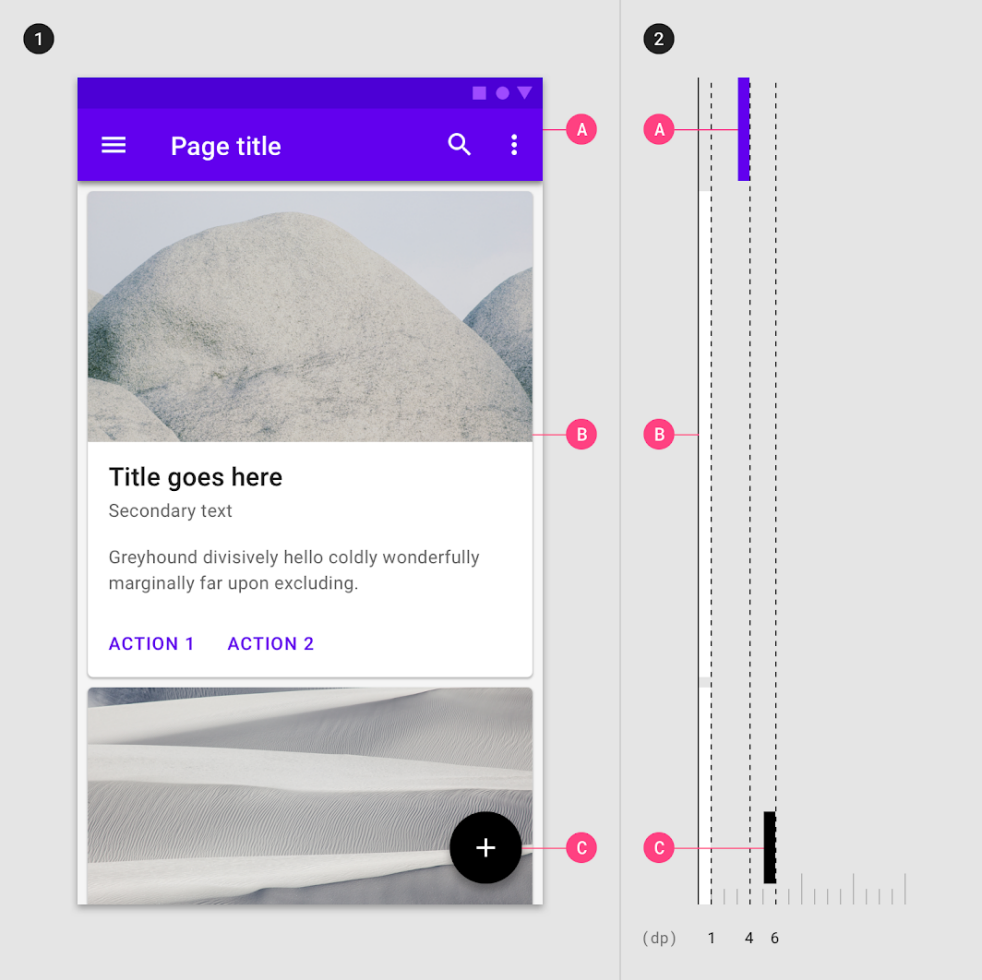
Tyto poznatky následně aplikujeme v sekci 3.5.2, kde navrhneme dle těchto pravidel samotné UI celého našeho webu. Jednotlivé vzniklé WireFramy si lze prohlídnout přiložené na médiu. Vzhledem k jejich velkému množství, zabýváme se v této sekci omezeným výběrem WF pro obecný popis aplikace v počítačovém režimu.

3.5.1 Úvod do Material Design

Material design je designový jazyk určený pro tvorbu GUI. Nejedná se o programovací jazyk, ale soubor pravidel. Díky tomu je nezávislý na použité technologii. Vytvořený návrh tak lze použít pro implementaci v jakémkoliv programovacím jazyce a typu klienta. Jelikož chceme vytvořit moderní, intuitivní a přehledný design, je vhodné se držet ověřených a otestovaných možností, mezi než Material Design určitě patří. Jelikož jsme si zvolili tento designový jazyk pro návrh a implementaci webového rozhraní, shrneme si podstatná pravidla pro návrh GUI v této sekci. Samotná dokumentace a další podrobné informace si lze přečíst na této URL [30].

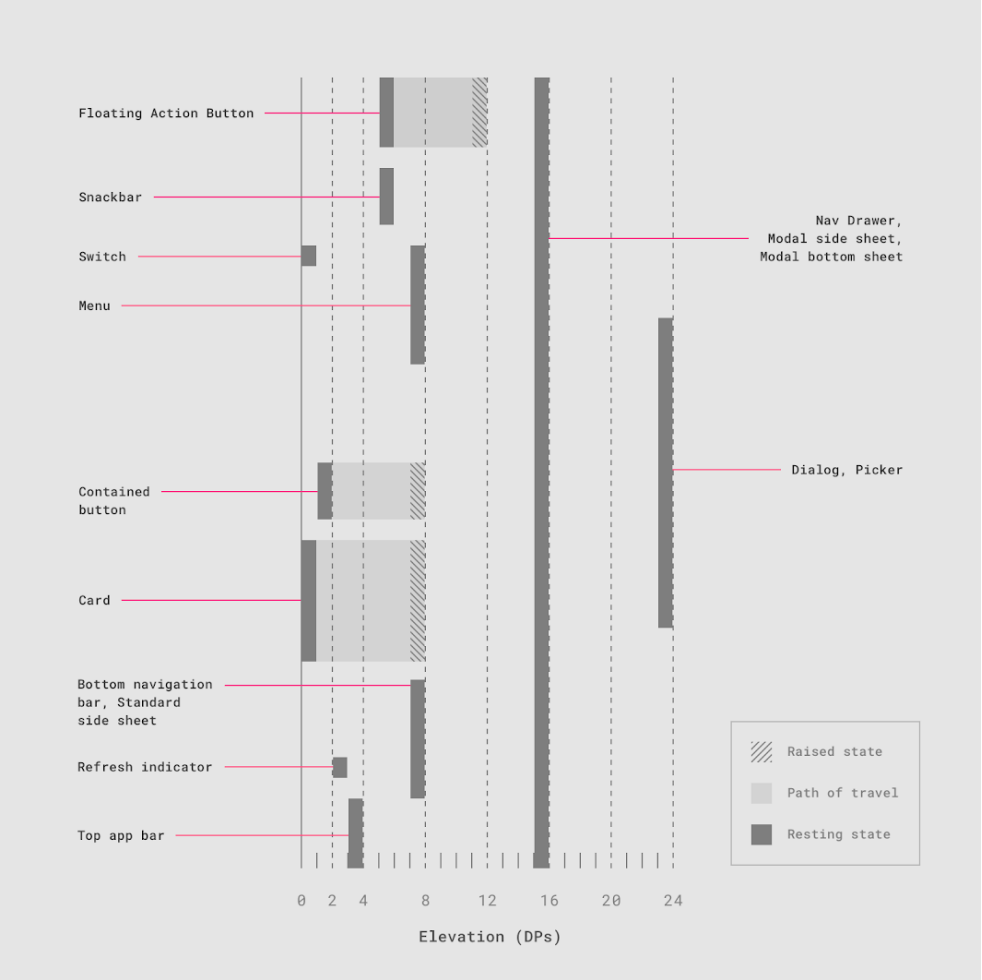
Historie Nejprve si však projdeme stručně historii. Material Design byl představen veřejnosti Googlem 25. června 2014 na konferenci Google I/O. Hlavní motivací pro vznik nového designu bylo zlepšení uživatelského zážitku na mobilních telefonech s operačním systémem Android. Toho bylo dosaženo sjednocením UI a ovládání zařízení napříč celým systémem, včetně aplikací třetích stran. Zároveň Google chtěl sjednotit ovládání a design vlastních služeb i napříč různými platformami tak, aby se vše ovládalo stejně nezávisle na operačním systému nebo druhu zařízení, který uživatel používá. Proto byly uveřejněny doporučení pro tvorbu nových Android aplikací. Avšak tyto pravidla lze použít i pro tvorbu aplikací na iOS, Windows nebo webové stránky. První verzí Androidu, ve které se tato úprava grafické nadstavby objevila, byl Android verze 5.0 neboli L, též známy jako Lollipop. K jeho představení došlo na stejné konferenci, spolu s přepracovanými Google službami jako například Google Drive nebo Gmail. Od této doby se s tímto designovým jazykem lze setkat na jakémkoliv telefonu s Android verzí 5 a vyšší, v Google službách a nebo na spoustě webových stránkách či mobilních aplikací, jelikož je velmi přehledný a intuitivní na ovládání. Toho Google docílil hlavně inspirací se v interakci uživatelů svými zařízeními. [31, 32]

Použití materiálů Jak již název napovídá, hlavní myšlenkou Material designu je inspirace v reálném světě, kde je vše tvořeno z materiálů. Ty mají své vlastnosti a nejinak je tomu i zde. Každý prvek má zde kromě své výšky a šířky také hloubku, přesně 1 dp. Přestože zobrazovací plocha je dvou rozměrná rovina, ve světě Material designu se pohybujeme ve trojrozměrném prostoru. Tedy každému prvku UI lze přiřadit z souřadnici, která určuje výšku umístnění prvku v prostoru. Jelikož se díváme na tento prostor z ptačí perspektivy (kamera je umístěna nad zobrazovanou plochu a natočena tak, že je kolmá k osám x a y a pohled směřuje do záporné části osy z), výšku objektu v prostoru můžeme pozorovat pomocí stínu, který vrhá a jeho velikost je přímo úměrná jeho výšce v prostoru. Toto nám ilustruje obrázek 3.5. Mezi klíčové vlastnosti materiálů patří i pevnost. Nemůže tak nastat situace, kdy se dva prvky prolínají. Mohou být vedle sebe, ale v případě, že jeden prvek by měl pro-



Obrázek 3.5: Ukázka výškového umístění prvků v aplikaci.[1]

tnout druhý, musí se jeden nacházet výše než ten druhý. Pokud tedy máme textové pole a tlačítko, které vyvolá dialogové okno, toto dialogové okno musí být výše než textové pole a tlačítko, které jej vyvolalo. Pravidlo nám říká, že prvek, který je interaktivní je umístěn nad prvek, který není. Důležitost prvku zde hraje také roli. Významnější prvky se nacházejí ve vyšší vrstvě, než méně významné. Například vysunutý boční ovládací panel se ve scéně nachází nejvýše, a tak vrhá na ostatní prvky stín. Pokud s prvkem probíhá interakce, například tlačítko je zmáčknuté, dočasně vystoupí ze své vrstvy do vyšší, a po uvolnění se opět vrací do své vrstvy. Tím uživatel dostane vizuální odezvu. Vhodná výšková umístění, včetně dynamického posunu lze vidět na tomto diagramu 3.6. [33, 34, 35]



Obrázek 3.6: Diagram vhodného umístění prvků v aplikaci, včetně dynamického posunu.[2]

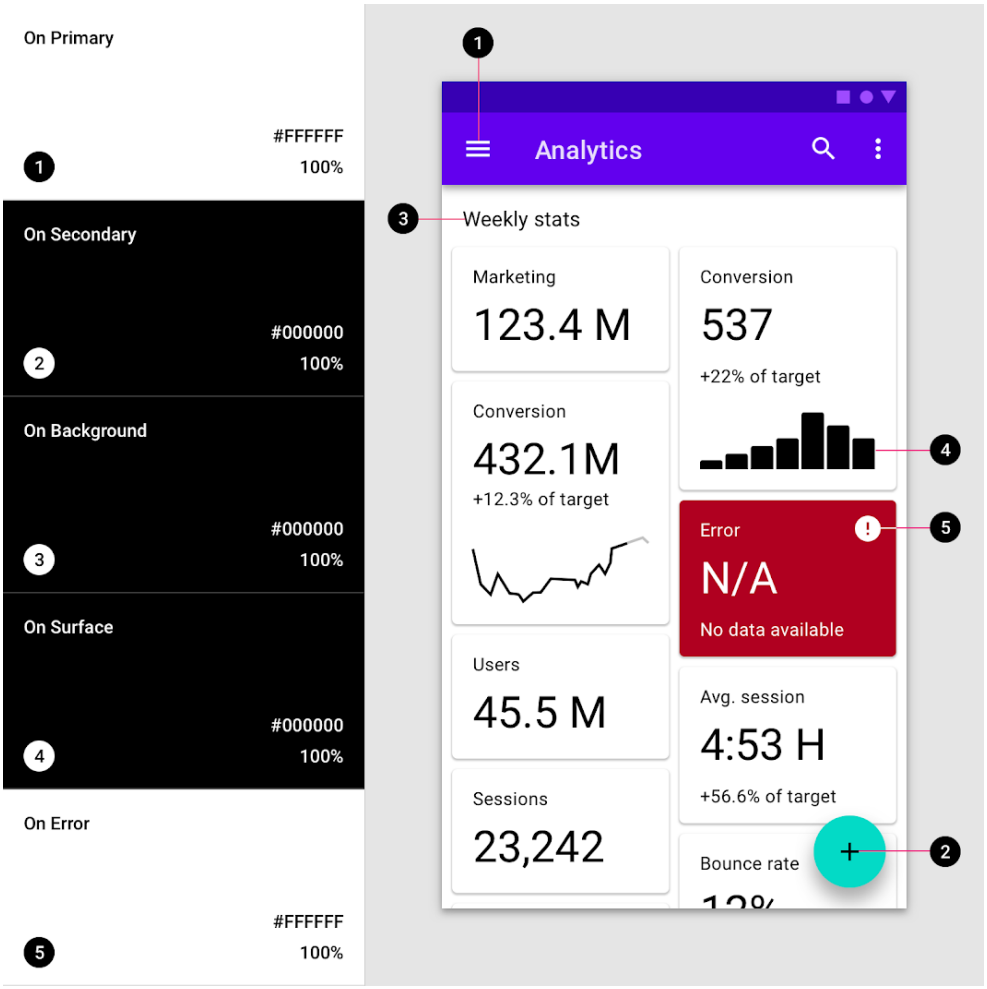
Myšlenka papíru Velmi důležitou myšlenkou je také to, že každý prvek UI je ve své podstatě papír. A však na rozdíl od reálného papíru lze tento virtuální zmenšovat, zvětšovat, ale i měnit jeho tvar. Díky tomu lze vytvářet jednoduché animace nebo upravovat rozměry v závislosti na akcích nebo velikosti obrazovky. [36]

Téma aplikace Dalším velmi důležitým principem, je požití témat. Místo toho, abychom každému prvku UI přiřazovali vlastní barvy, tvary a další vlastnosti, je vhodné použít témata. Díky tomu lze jednoduše udržet jednolitý design celého UI. Zároveň je následně jednodušší upravit aplikaci na tmavý režim, jelikož stačí globálně změnit téma a ostatní prvky se automaticky přizpůsobí. Také lze, díky oddělení barev jednotlivých prvků do témat,

jednoduše měnit barevnou paletu aplikace na základě uživatelské preference. [36, 37]

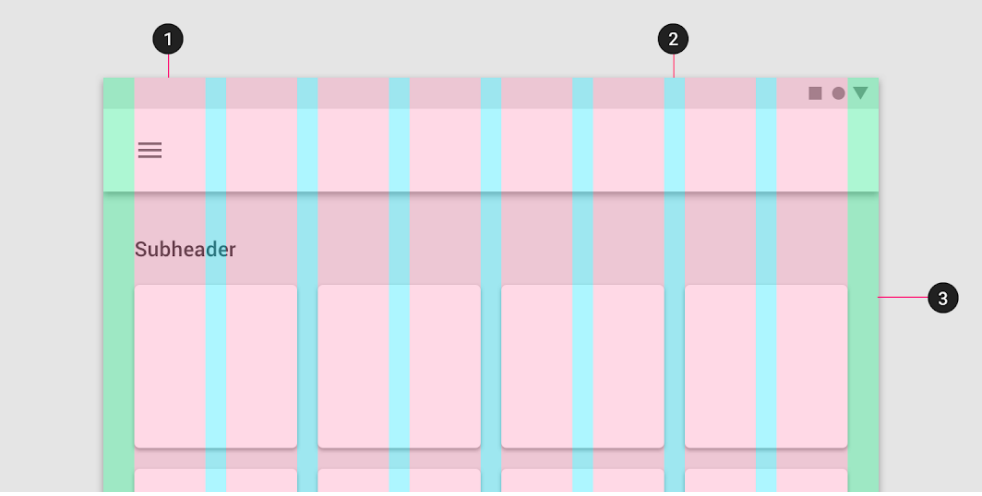
Barevná paleta Velmi významnou roli v UI hrají barvy. Nejinak je tomu i zde, v Material Designu. Kromě vizuálního oživení vzhledu stránky a aplikace, je hlavním smyslem barev ukázat, jaký význam mají jednotlivé elementy. Pokud se jedná o důležitý prvek, měl by být více zřetelný, a tedy je vhodné zvolit výraznější barvu na rozdíl od okolí nebo méně významných prvků. Pomocí barev tak můžeme vyjádřit hierarchii prvků a vztahy mezi nimi navzájem. Také pomocí barev dáváme uživateli jasně najevo, které prvky UI jsou interaktivní, a které nejsou. I když pomocí barev můžeme definovat vlastní značku, existují zde nějaká pravidla. V rámci těchto pravidel se nám může na první pohled zdát, že barevná paleta je prázdná. Ta se totiž v základu skládá pouze ze dvou barev – primární a sekundární. Tyto barvy by měly být rozdílné, avšak neměly by si oponovat navzájem. Úkolem primární barvy je totiž odlišit interaktivní prvky GUI od textu a pozadí. Naopak, sekundární barvu můžeme úplně vynechat, jelikož slouží hlavně k odlišení významu prvků. Primární barva je sice dominantní a vyskytuje se na většině prvků, avšak některé elementy UI mají menší význam, přestože jsou interaktivní. V případě potřeby dalších barev si však musíme vystačit pouze se světlejší a tmavší variantou primární a sekundární barvy. Pro ilustraci barevné palety se můžeme kouknout na obrázek 3.7. [36, 38]

Typografie Pokud se podíváme na typografii, tak zjistíme, že pravidla jsou zde volnější. Google doporučuje používání jejich vlastního fontu Roboto, avšak pravidla jsou natolik obecná, že je lze využít pro jakýkoliv font. Hlavním pravidlem je použití na první pohled zřetelně rozlišitelnou velikost písma pro jednotlivé úrovně nadpisu, podnadpisu a textu. Díky tomu uživatel jasně rozezná od sebe nadpis, popisek nebo prostý text. Zároveň správné použití velikosti písma a fontu značně zlepšuje čitelnost a přehlednost webové stránky či aplikace. Pokud používáme expresivní fonty, doporučuje se značná obezřetnost. Například, použití takovéhoto fontu nejspíše nebude vadit v nadpisu, naopak pro prostý text je absolutně nevhodný. Pokud jej použijeme jako podnadpis, velmi jednoduše se může stát, že font bude špatně čitelný a nebo se ztratí v okolním textu a designu. Pokud se bavíme o barvách fontu, určitě je nevhodné zvolit barvu textu tlačítka jako běžného textu nebo grafiky pozadí. Pokud totiž tlačítko nemá ohraničení či jiné zvýraznění, uživatel jej snadno přehlédne. Další pravidlo nám říká, abychom použili kontrastní barvu písma vůči pozadí. Pokud máme bílé pozadí, černý text je jasná volba. Pro černé nebo tmavě šedé pozadí se zase hodí bílé písmo. Pokud budeme chtít barevné písmo, tmavě modrý text na žlutém pozadí bude ještě relativně čitelný, avšak světle šedý text na červeném pozadí již nikoliv. [39, 38]



Obrázek 3.7: Ukázka barevné palety. [3]

Ikonografie Nedílnou součástí každého GUI jsou ikony. Jestliže se rozhodneme použít již existující ikony, měli bychom primárně používat oficiální Material Design ikony. Ty vznikly tak, aby byli minimalistické, moderní, přehledné a srozumitelné. Z tohoto důvodu bychom neměli používat jinou ikonu pro danou činnost, než již existující. Také je potřeba se vyhnout zavádějícímu použití ikon, kdy uživatel očekává nějakou akci a místo toho proběhne jiná. Pokud se naopak rozhodneme vytvořit si vlastní ikony, je potřeba dodržet pár zásad. Hlavní zásadou je, že danou ikonu musíme být schopni vytvořit z papíru. Správné stínování je též potřeba dodržet a také není vhodné jakkoliv přerušovat ikonu. Celistvá jednolitá ikona dodržující geometrické principy je proto ideální volbou. Zvolením jednoduché ikony člověk nic nezkazí, naopak natáčením obsahu ikony již ano. Ikony musí být vždy orientovány čelem k uživateli. [40, 41]



Obrázek 3.8: Ukázka rozložení prvků na stránce. 1 – sloupec, 2 – odsazení mezi sloupci, 3 – odsazení od krajů obrazovky [4].

Rozmístění Rozmístění prvků je vhodné logicky uspořádat. Pro uživatele je vždy lepší předvídatelné chování a rozložení jednotlivých prvků. Pokud se podíváme opět na obrázek 3.7, intuitivně si můžeme domyslet chování aplikace po stisknutí tlačítek na horní ovládací liště. Stisknutím levého tlačítka (tři vertikálně umístěné čárky) docílíme vysunutí bočního menu z levé části obrazovky. Naopak, tlačítko lupy se změní na textový vstup a skryje nadpis. Poslední tlačítko vpravo vyvolá kontextové menu, které se objeví na jeho místě a expanduje směrem dolů. Pokud toto pravidlo spojíme i s tvorbou uživatelského rozhraní do jednotlivých sloupců s konzistentním rozestupem mezi sloupci a prvky, snadno docílíme jednotného vzhledu napříč aplikací. Velmi vhodným vedlejším produktem dodržovaní těchto pravidel je přirozený responzivní design. Jednotlivé zobrazené prvky jsou tak schopné se sami intuitivně přeskládat v závislosti na velikosti obrazovky tak, aby zůstali zarovnané, přehledné a čitelné. Vhodné je také zvolit odsazení prvků od krajů obrazovky, avšak pozor na velikost tohoto odsazení. Pokud nezvolíme dynamické odsazení, nebo naopak zvolíme příliš velké statické, může se nám stát, že obsah bude nečitelný. Správnost rozložení prvků na stránce nám ilustruje obrázek 3.8. Počet sloupců a jejich šíře je přímo úměrná velikosti obrazovky. Pokud se rozhodneme pro horizontální rozložení stránky místo vertikálního (například na dotykové obrazovce se jedná o lepší rozložení pro danou aplikaci), platí stejná pravidla, akorát sloupce jsou otočeny o 90°. [42, 43]

Závěr Všechny výše zmíněné principy jsou univerzálně platné. Platí tedy nejen pro tvorbu Android aplikace, ale i pro tvorbu webového UI. Velmi vhodný příklad použití Material Designu si můžeme ukázat na domovské

stránce letecké společnosti Lufthansa, který lze shlédnout na následujícím odkaze [44]. Proto se těmito pravidly řídíme při tvorbě našeho webového portálu, který si popíšeme v následující sekci 3.5.2.

3.5.2 Návrh uživatelského rozhraní

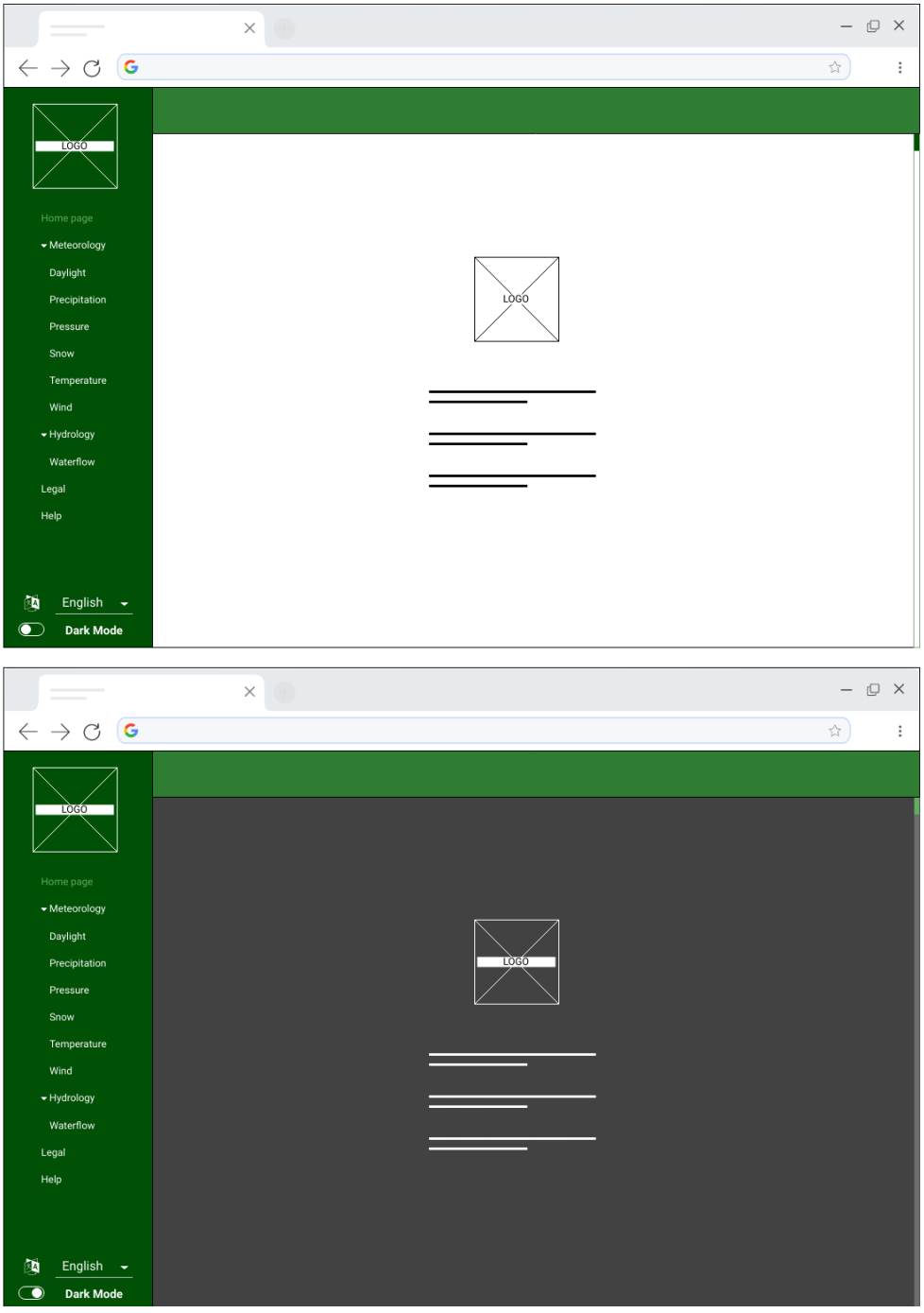
Nyní, když víme, jaká pravidla používat při návrhu UI, můžeme si probrat námi navržené wireframy. V této sekci si probereme pět obrazovek – domovskou stránku, stránku pro výběr parametrů, stránku pro vizualizaci dat, stánku pro nastavení parametrů porovnávacího režimu a samotnou stránku zobrazení grafu v porovnávacím režimu. Zbylé navržené stránky včetně verzí pro mobilní telefon si lze prohlédnout v příloze na přiloženém médiu. Vybrali jsme těchto pět obrazovek, jelikož názorně ilustrují obecný design aplikace.

Domovskou stánku nám zachycuje obrázek 3.9. Pokud se na něj podíváme, vidíme základní rozložení aplikace, které nám tvoří tři samostatné části. První částí je samotný navigační panel umístěný vlevo. Na něm se nachází od shora dolů logo, jednotlivé stránky a kategorie dat a jednoduché nastavení aplikace. Navigační panel se skládá z nabídky domácí stránky, práva, nápovědy a samotných kategorií dat rozdělených do dvou úrovní. První úroveň obsahuje obecný název kategorie a jednotlivé data pro zobrazení tvoří až podkategorie. Pokud prvek obsahuje jednotlivé podkategorie, nachází se před textem ikonka šipky orientované směrem dolů pro rozbalení a směrem nahoru pro sbalení nabídky. Kliknutím na tento řádek dojde k rozbalení obsahu nabídky, který se objeví pod názvem kategorie a pro přehlednost jsou názvy podkategorií zleva odsazeny. Jednoduché nastavení obsahuje nabídku pro zvolení si jazyka aplikace (podporované jazyky jsou zatím pouze čeština a angličtina) a přepínač pro volbu mezi světlým a tmavým režimem aplikace.

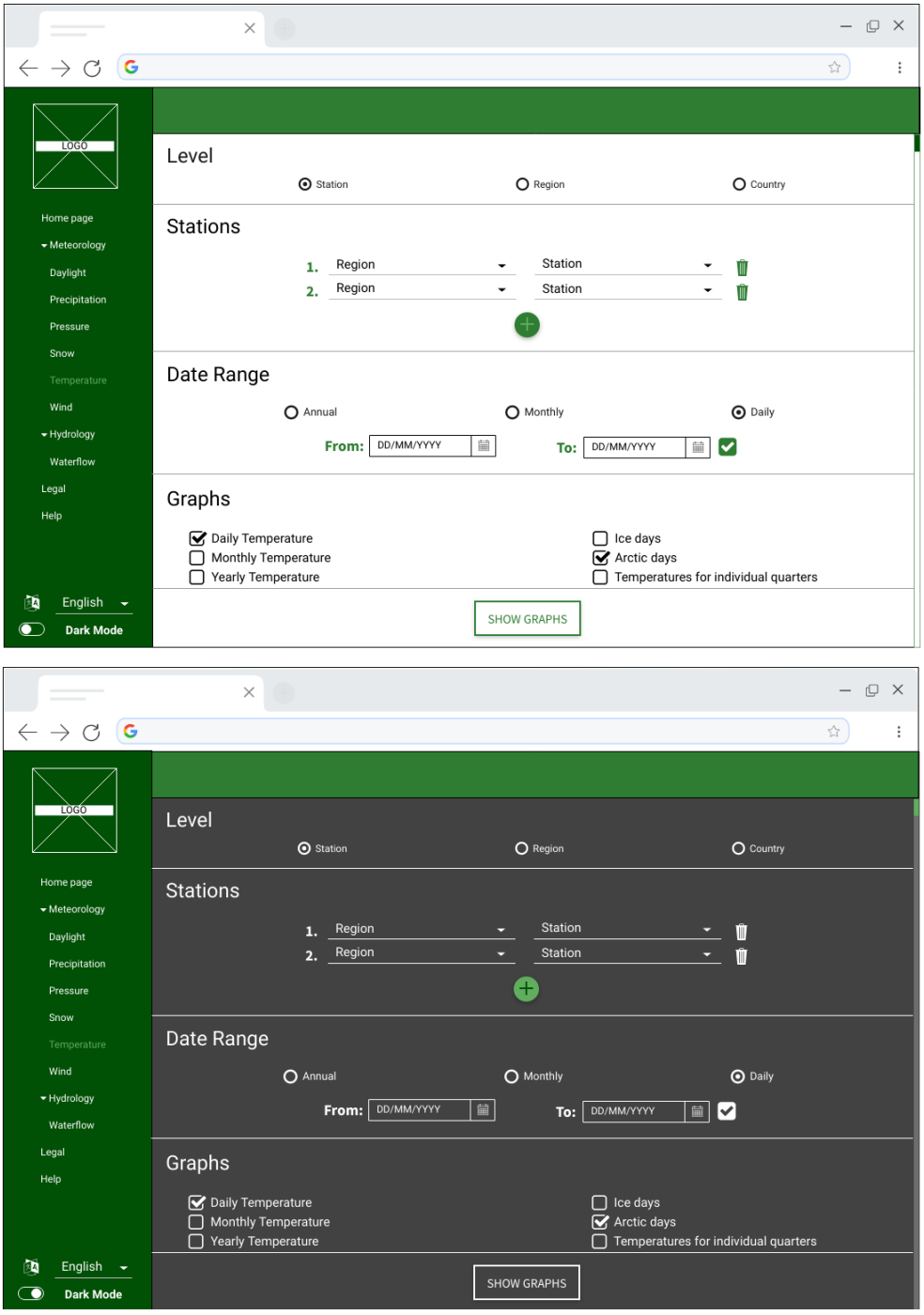
Další prvek tvoří horní ovládací lišta, která slouží k ovládání aplikace. Ta je přítomna po celou dobu a v základním stavu je prázdná. Pokud dojde ke zmenšení aplikace, vlevo se objeví ikonka tří čárek nad sebou, která slouží k vysunutí nebo skrytí navigační lišty. Pokud se přesuneme do vizualizační části aplikace, zobrazí se vpravo další tlačítka, která si popíšeme později.

Poslední část tvoří hlavní okno aplikace, kde se zobrazuje samotný obsah stránky. V případě domovské obrazovky se jedná o logo a název webu umístěné do středu obrazovky. Následně může být vložen nějaký text, avšak v rámci této práce si vystačíme s dočasným logem a názvem. Veškeré textové obrazovky jsou podobného stylu.

Pokud se podíváme na obrázek 3.10, uvidíme obrazovku, která slouží pro nastavení parametrů dat, které zajímají uživatele. Na tuto obrazovku se uživatel přesune po kliknutí na konkrétní kategorii dat, kterou si chce zobrazit. Stránka se skládá ze čtyř samostatných modulů pro výběr parametrů a tlačítka pro potvrzení výběru a následné přesunutí se na stránku pro vizualizaci. V levém horním rohu každého modulu se nachází jeho název a jednotlivé moduly jsou od sebe vizuálně odděleny pomocí horizontální linky.



Obrázek 3.9: Wireframe zachycující návrh domovské stránky.



Obrázek 3.10: Wireframe zachycující návrh stánky pro výběr parametrů.

Pokud si popíšeme jednotlivé moduly shora dolů, první který vidíme, slouží k výběru úrovně zobrazení dat. Přepínání mezi úrovněmi jsme vyřešili pomocí tří přepínačů, kdy zvolený je vždy právě jeden. Vedle každého přepínače vpravo se nachází jeho popisek.

Další modul slouží k výběru zdrojů dat. Pokud je zvolený stát jako úroveň, tento modul není vykreslován vůbec. V ostatních případech se jedná o modul, který tvoří v dolní části tlačítko plus. To přidává jednotlivé řádky, které tvoří zprava číslo řádku, nabídka krajů, pokud je zvolená úroveň stanice, nachází se zde nabídka stanic pro zvolený kraj. Poslední prvek na řádku je ikonka pro smazání řádku.

Následuje modul pro volbu časového rozsahu. Nejprve si uživatel zvolí periodicitu dat pomocí přepínačů, a následně si zvolí data pro začátek a konec měření pomocí kalendáře.

Poslední modul tvoří nabídka dostupných statistik pro zvolenou kategorii. Ta je navržena jakožto seznam se zaškrtávacími tlačítky. Chceme upozornit, že zobrazený seznam statistky je pouze ilustrační a nereprezentuje reálné dostupné kategorie.

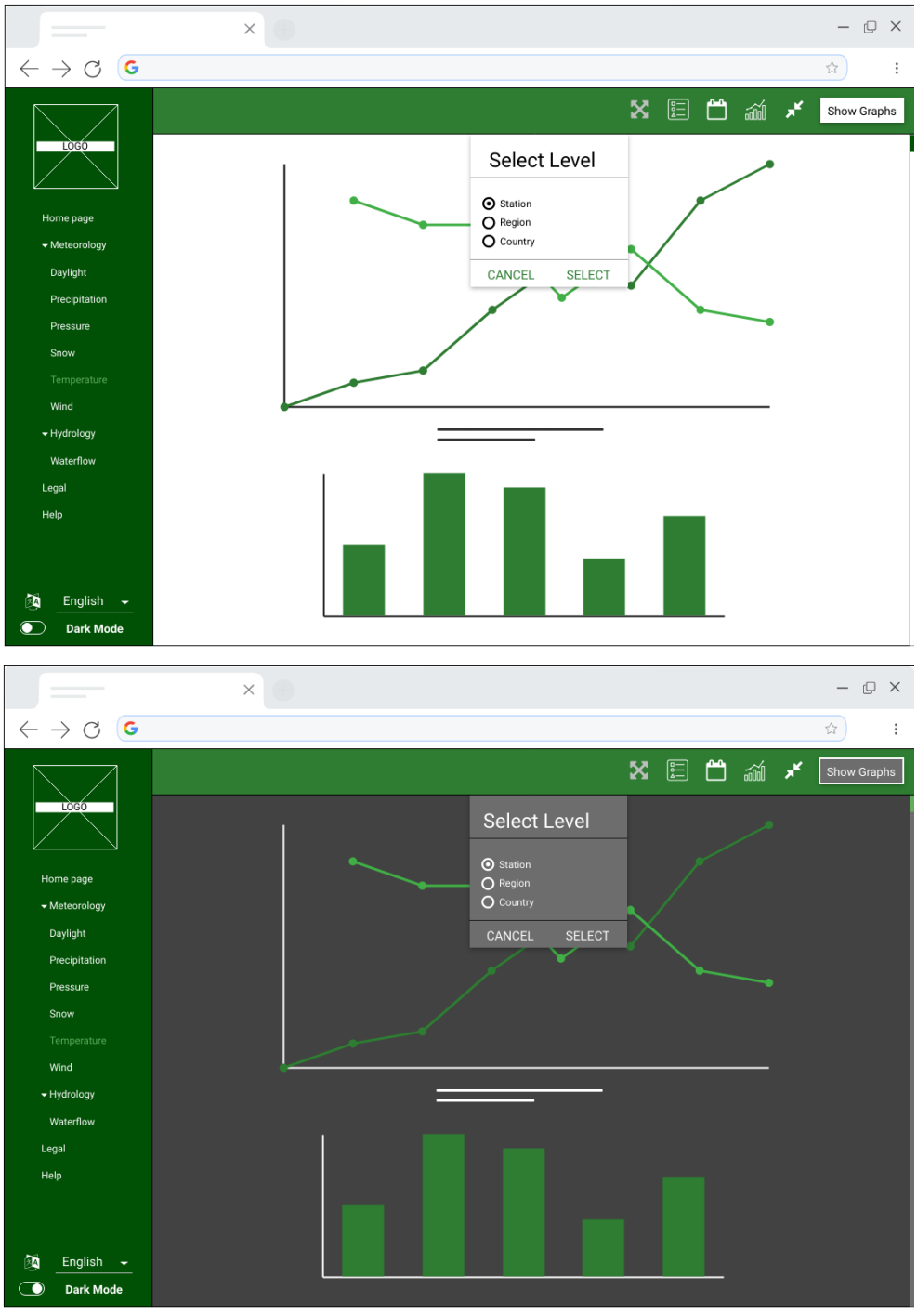
Třetí obrazovku, kterou si popíšeme tvoří samotná stránka pro vizualizaci dat, která je zachycená na obrázku 3.11. Pokud se podíváme na horní ovládací lištu, můžeme vidět nová tlačítka vpravo. Jedná se o ikonky, které zobrazí příslušné moduly pro nastavení parametrů pomocí dialogového okna. Jednotlivé kategorie zleva doprava odpovídají pořadí modulů na stránce pro nastavení parametrů. Poté následuje tlačítko s ikonou dvou šipek mířící k sobě, které slouží pro přepnutí se do porovnávacího režimu. Úplně vpravo se pak nachází tlačítko pro znovu načtení stránky s grafy poté, co jsme upravili parametry.

Čtvrtou obrazovku, zachycenou na obrázku 3.12, je obrazovka pro nastavení parametrů porovnávacího režimu. Rozloženi stránky je velmi podobné stránce pro nastavení parametru. Na této stránce se nachází výpis všech možných datových sad, které lze zobrazit rozdělených do příslušných kategorií. Každý záznam tohoto výpisu tvoří název a zaškrtávací pole, které značí zda uživatel si zvolil tento typ dat pro zobrazení či nikoliv. Dole se poté nachází tlačítko pro zrušení porovnávacího režimu a tlačítko pro potvrzení parametrů a následné zobrazení dat.

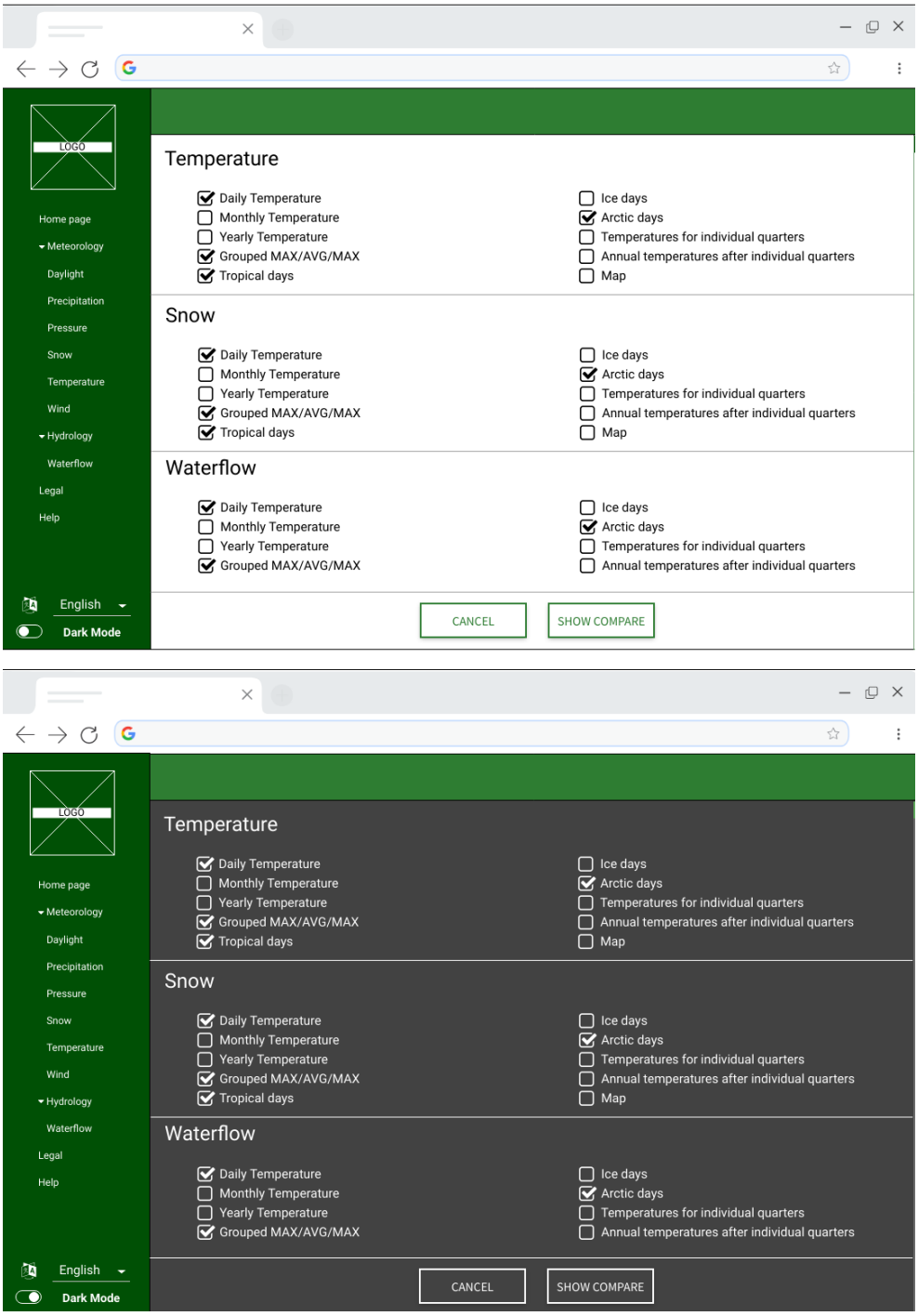
Samotné zobrazení dat pomocí porovnávacího režimu nám zachycuje poslední obrázek 3.13. Na této obrazovce máme opět zobrazená ovládací tlačítka v horní liště aplikace, která jsou shodná s tlačítky zobrazená na obrazovce grafů.

V části pro zobrazení obsahu se nachází grafy, které kombinují veškeré zvolené datové sady. Vpravo vedle grafu se nachází výpis těchto datových sad se zaškrtávacími poli, pomocí kterých lze vypnout a zapnou jejich zobrazení v grafu.

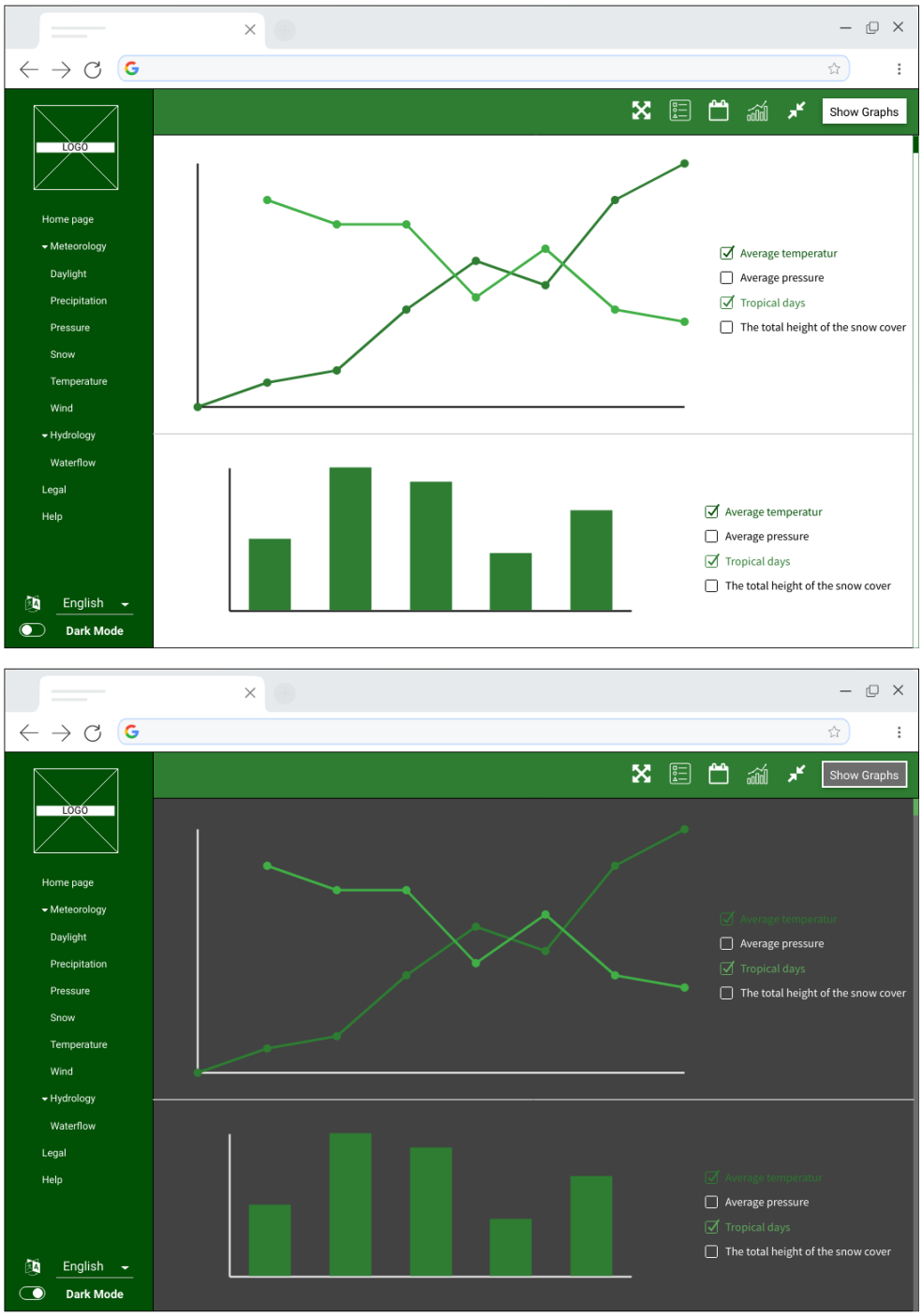
Samotné okno aplikace se skládá z jednotlivých grafů a může být doplněno o vysvětlující text.



Obrázek 3.11: Wireframe zachycující návrh zobrazení dat pomocí grafů.



Obrázek 3.12: Wireframe zachycující návrh stránky pro zadání parametrů při přepnutí do porovnávacího režimu.



Obrázek 3.13: Wireframe zachycující návrh zobrazení dat pomocí grafu v porovnávacím režimu.

Jako barevnou paletu jsme zvolili kombinaci světlé a tmavé zelené. Text je primárně černý. Po přepnutí do tmavého režimu navigační panel zůstává nezměněn. V horní liště dojde ke změně tlačítka z bílé na šedou. V hlavním okně dojde ke změně pozadí na tmavě šedou a černé prvky jsou přebarveny na bílo.

Aplikace díky tomuto návrhu by měla být responzivní, jelikož návrh počítá s podporou menších obrazovek a mobilních telefonů.

3.5.3 Technologie

V této sekci se zabýváme výběrem vhodné technologie pro tvorbu webové aplikace, která slouží jako frontendová část projektu. Jedná se o část projektu, s kterou interaguje uživatel. Jak již bylo zmíněno, jazyk pro vývoj této části jsme si zvolili JavaScript. Z tohoto důvodu se zabýváme pouze dostupnými frameworky pro JavaScript. Tento jazyk jsme si zvolili, kvůli své jednoduchosti a možnosti implementace grafů a mapových služeb pomocí existujících knihoven. Určitě lze použít i jiné technologie pro tvorbu této webové aplikace, například PHP nebo framework pro čistý HTLM a CSS. Pro projekt takovéhoto rozsahu je vhodnější použít již JavaScript. Jelikož chceme mít podporu kvalitních knihoven pro usnadnění práce, zvolili jsme větší frameworky, které jsou k dispozici, právě pro jejich rozsáhlou podporu a kvalitní dokumentaci. Určitě stojí za zmínku následující frameworky: Backbone.js, EmberJS, Foundation, Preact nebo Svelte. Avšak tyto jsme rovnou vyřadili pro jejich velkou jednoduchost, malou podporu, málo aktivní komunitu a tedy malý výběr dostupných knihoven a nebo pro nedostatečnou a málo kvalitní dokumentaci.

3.5.3.1 Vue.js

Vue.js je jeden z nejpopulárnějších frameworků pro tvorbu webové aplikace v JavaScriptu. Původně vznikl v roce 2014 v Googlu jakožto nástroj pro prototypování. Nyní se jedná o komunitou spravovaný open-source famework. [45]

Mezi výhody frameworku patří široká komunita, velká podpora, kvalitní rozsáhlá dokumentace, jednoduchá syntaxe a obousměrná datová vazba (anglicky two–way data binding). [45] Nevýhodou je jazyková bariéra, jelikož framework je velmi populární v Číně díky původu autora. Další nevýhodou je nízká stabilita komponent. [46, 45]

3.5.3.2 Angular2

Nejprve bychom rádi upozornili, že se v této části zabýváme Angularem ve verzi 2, tedy postavenou nad TypeScriptem, který je nadstavbou nad JS [47]. AngularJS se nezabýváme, jelikož v době psaní této práce je tento framework již nepodporován ze strany Googlu kvůli problémům s návrhem samotného

frameworku. Samotná logika a architektura frameworku byla přepracována a plně nahrazena Angular2, která je nadále podporován a vyvíjen. [48, 49]

Angular2 je frontendový framework vytvořený Googlem v roce 2016. Jak již bylo zmíněno, je postavený na jazyce TypeScript. [45]

Výhodou Angularu je podpora obousměrné datové vazby mezi modelem a pohledem. Další výhodu tvoří znovupoužitelnost komponent, jelikož jsou na sobě nezávislé a vyžívá techniku vkládání závislostí (dependency injection). [45]

Mezi nevýhody patří velká komplexita aplikace. Díky tomu lze narazit na problémy s optimalizací aplikace. Díky svojí velikosti může dojít k nízkému výkonu. Zároveň se Angular řadí velikostí mezi největší knihovny. Díky tomu uživatel musí stahovat objemnou stránku. Přestože Angular stále patří mezi nejoblíbenější frameworky, jeho popularita poslední roky konstantě klesá. [45]

3.5.3.3 React.js

React.js je open-source framework vytvořený Facebookem v roce 2013.

Hlavní výhodou Reactu je využívání virtuálních DOMů. Při vykreslování se každý virtuální DOM zkontroluje, zda došlo ke změně. Díky tomu dochází k překreslování pouze změněných prvků. Výsledkem toho je vysoký výkon aplikace, která je stabilní a rychlá i přes velkou komplexitu rozsáhlé aplikace. Jedná se o ideální framework pro jednostránkovou webovou aplikaci. Mezi další výhody patří jednoduchá možnost znovupoužití komponent, možnost psaní komponent jako funkce místo tříd, nízká náročnost pro naučení se používání frameworku, široká podpora a kvalitní dokumentace. Také mezi výhodu lze označit podobnost s knihovnou React Native pro tvorbu mobilních aplikací, která staví na stejných principech.

Nevýhodou je často měnící se dokumentace díky velkému množství aktualizací knihovny. Další nevýhodou je pouze jednosměrné předávání dat z rodiče do potomků. [45]

3.5.3.4 Finální volba

Po zhodnocení vybraných knihoven jsme se rozhodli pro React.js. Důvodem je široká podpora komunity, rychlost výsledné aplikace. Jelikož chceme aplikaci rozšiřovat do budoucna, React.js nám to usnadní díky znovupoužitelnosti komponent. Přestože Vue.js je vhodnou alternativou k React.js, z důvodu jazykové bariéry a nízké stability jsme dali přednost Reactu. Angular jsme nezvolili pro svoji komplexitu.

Kapitola 4

Implementace

Tato kapitola pojednává o implementaci navrhovaného řešení. Nejprve si projdeme použité nástroje pro vývoj a spravování kódu (sekce 4.1 a 4.2), následně si představíme ve zkratce nástroj Docker [50] (sekce 4.3), jelikož naše řešení je připraveno pro nasazení pomocí tohoto nástroje. Následně si zmíníme důležité knihovny použité při implementaci v sekci 4.4. Zbytek kapitoly pak stručně pojednává implementaci jednotlivých částí.

4.1 Vývojové prostředí

Pro vývoj celého řešení jsme použili nástroj Microsoft Visual Studio Code. Jde o velmi univerzální vývojové prostředí, které podporuje velké množství jazyků a frameworků. Pomocí tohoto nástroje lze programovat ve všech hlavních jazycích. Nechybí ani podpora všech operačních systémů.

Microsoft Visual Studio Code [51] se díky tomu těší velké popularitě, avšak nemalý podíl na tom má široká podpora pluginů a možnost přizpůsobeni si nástroje podle potřeby.

Velmi užitečnými pluginy při vývoji byly:

- Docker – modul pro správu Dockeru přímo z vývojového prostředí. [52]
- ES7 React snippets – plugin pro našeptávání a generování základních struktur pro React.js. [53]
- Python – oficiální modul pro IntelliSense, nástroj pro kontrolu kódu a našeptávání. [54]

4.2 Verzování kódu

Pro verzování kódu jsme použili nástroj GitHub, což je open-source webová služba pro správu kódu pomocí nástroje GIT. Veškerý kód je dostupný na url `https://github.com/opendatalabcz/eco-web`.

Co se týče nástroje GIT, jedná se o distribuovaný verzovací systém. Autorem je Linus Torvalds. Velkou výhodou distibuovaného systému oproti centralizovanému je, že každý uživatel má kopii repositáře uloženou lokálně. Díky tomu lze spolupracovat na jednom souboru současně s dalšími programátory. Následná synchronizace uživatelů probíhá pomocí patchů.

4.3 Docker

Docker je open-source nástroj pro automatizované nasazení aplikací založený na virtualizaci. Hlavní výhodou použití Dockeru pro nasazení je oddělení aplikace od infrastruktury. Díky tomu lze jednoduše předávat, testovat a nasazovat aplikaci. Zároveň lze na jednom zařízení hostovat více aplikací či částí aplikace, jelikož jednotlivé služby jsou od sebe izolované. [55]

Základní jednotkou Dockeru tvoří *Docker image*. Jedná se o strukturu pouze pro čtení, která obsahuje návod pro vytvoření kontejneru. Samotný image může být založený na jiném image. [55]

Pokud chceme vytvoř vlastní Docker image, je potřeba vytvořit *Dockerfile* v kořenovém adresáři aplikace, kterou chceme dockerizovat. Dockerfile je soubor, ve kterém popíšeme jednotlivé nutné kroky pro vytvoření image v pořadí jak mají jít za sebou. Je velice běžné definovat na prvním řádku, jaký existující image má být použit při tvorbě námi definovaného. [55]

Druhá velmi důležitá jednotka je samotný *kontejner*. Jedná se o spustitelnou instanci image. Samotný kontejner je izolovaný od ostatních, pokud nevytvoříme sít mezi kontejnery. Pokud kontejneru nepředáme lokální úložiště pomocí volume, veškerá data uvnitř kontejneru jsou ztracena po jeho smazání. [55]

Pokud chceme provozovat více kontejnerů zároveň, které jsou na sobě závislé, je vhodné použít nástroj *Docker Compose*. Mezi kontejnery je vytvořena také síť, takže mohou mezi sebou komunikovat. Toho lze docílit pomocí docker–compose.yml souboru, ve kterém definujeme pravidla pro tvorbu kontejnerů. [56]

V rámci této práce využíváme Docker pro zjednodušení nasazení celé aplikace. Pro databázi používáme oficiální PostgreSQL image. Backend a frontend tvoří vlastní námi vytvořené image založené na oficiálním Node.JS image. Přesto je backend a frontend navržen tak, aby jej bylo možné spustit lokálně bez Dockeru. Stačí vyplnit přiložené konfigurační soubory. Pro maximální usnadnění nasazení je přiložen i docker–compose.yml. Díky tomu je usnadněné celé nasazení, jelikož stačí vyplnit konfigurační soubor v kořenovém adresáři celé implementace našeho řešení a použít příkaz *docker compose up*. Tím dojde ke stažení potřebných Docker image, sestavení námi definovaných image a následnému spuštění databáze, backendu a frontendu.

4.4 Důležité použité knihovny

V této sekci zmíníme důležité knihovny pro implementaci. Jde o knihovny, bez kterých nelze získat finální řešení. Zároveň veškeré použité knihovny jsou vydány pod licencemi, které nám dovolují tyto knihovny použít jakožto svobodný software (GNU [57], MIT [58], BSD 3-Clause [59]).

4.4.1 Získávání dat

Pandas je knihovna pro práci s rozsáhlými strukturovanými daty. Využíváme ji pro práci se seznamy stanic a zpracování jednotlivých dat při vkládání do databáze. [60]

Psycopg2 je knihovna pro komunikaci a práci s PostgreSQL databází. [61]

python-dotenv je knihovna pro práci s proměnnýma prostředí a .env soubory. [62]

UTM je knihovna pro převod souřadnic z WGS84 systému na univerzální zeměpisné souřadnice. Potřeba z důvodu nekonzistence souřadnic u hydrologických stanic, které používají pravě WGS84 systém. [63]

4.4.2 Backend

CORS jedná se o knihovnu pro povolení sdílení zdrojů mezi doménami. Standardně je toto webovým prohlížečem zakázáno. [64]

Dotenv je knihovna pro práci s proměnnýma prostředí a .env soubory. [65]

Express je framework pro tvorbu webových a mobilních aplikací. Používáme jej pro vytvoření instance serveru, kterému předáme port na kterém má poslouchat. [66]

Express-GraphQL je knihovna, která nám dovoluje používat GraphQL knihovnu s Express knihovnou. [67]

GraphQL je knihovna pro implementaci GraphQL API v JavaScriptu. Obsahuje i podporované datové typy, které GraphQL potřebuje, jelikož JavaScript je dynamicky typovaný jazyk. [68]

GraphQL-iso-date je knihovna pro práci si datem a časem. Samotná GraphQL knihovna podle dokumentace nepodporuje tyto datové typy a pro práci s nimi je potřeba si napsat vlastní řešení. Jelikož existuje tato knihovna, není potřeba si psát vlastní řešení. [69]

PG je knihovna pro práci s PostgreSQL databází. [70]

4.4.3 Frontend

Následující knihovny uvádíme jako obecné. Téměř všechny uvedené knihovny potřebují dostáhnutí React verze knihovny pro umožnění plného používání v rámci Reactu. Jako příklad uvedeme první jmenovaný, Apollo client. Tato knihovna obsahuje veškeré potřebné funkcionality, ale pro použití funkcionalit v kódu musíme dostáhnout modul React-Apollo, který nám tuto knihovnu zpřístupní uvnitř frameworku.

Je to způsobeno tím, že React používá takzvané hooky a JSX. JSX je syntaktycké rozšíření JavaScriptu pro popisování UI. Na první pohled vypadá podobně jako HTML tagy, avšak jedná se o velmi silný nástroj. [71]

Následující knihony jsou použity pro realizaci frontendu:

Apollo client je knihovna pro práci s GraphQL API v React.js. [72]

Material-ui je knihovna pro práci s Material Designem v Reactu. Knihovna implementuje veškeré UI prvky do grafického stylu a chování v souladu s principy tohoto jazyka. [73]

GraphQL je knihovna pro práci s GraphQL API v JavaScriptu. [68]

I18Next je knihovna pro implementaci přepínání jazyka aplikace. Jedná se o jednoduchý nástroj pro docílení více jazyčnosti. Uživatel definuje soubory pro jednotlivé jazyky pomocí JSON notace, kdy konkrétní označení následně předává funkci této knihovny jako parametr, která použije přiřazený text ve zvoleném jazyce. [74]

Leaflet je knihovna pro práci s geografickými mapami. Jedná se o bezplatné open-source řešení. [75]

Plotly.js je knihovna pro zobrazení dat pomocí grafů. [76]

4.5 Realizace Downloaderu a Scraperu

Pro realizaci downloanderu a scraperu jsme se rozhodli z důvodu velkého rozsahu dat, který by trvalo dlouho ručně zpracovat. Implementací této části tak získáme nástroj, který nám dovolí automaticky stahovat data z portálu ČHMÚ a následně provedení předzpracování dat pro následné automatizované vkládání dat do databáze. Dle licence ČHMÚ [77] můžeme stažená data použít pro naše účely, avšak nesmíme je modifikovat a využívat ke komerčním účelům. Rádi bychom také uvedli, že ČHMÚ se nepodílí na této práci a není s ní jakkoliv spojená. Tato data používáme pouze z naší iniciativy pod výše zmíněnou licencí.

4.5.1 Scraper

Prvním potřebným krokem pro scrapování dat, byla nutnost provést analýzu cílové stránky. Jak jsme si již popsali v sekci 3.2, stránky ČHMÚ používají JavaScript. Díky tomu nelze použít Scrapy na získání dat přímo ze stránky, jelikož námi požadovaná data jsou zaslány a zobrazeny později. Tento problém jsme vyřešili scrapováním jednotlivých odpovědí serveru, který posílá dynamicky tyto záznamy na vyžádání. Tím, že uživatel zvolí kategorii a kraj, dostane JS funkce dostatek informací a provede dotaz na API, které zašle HTML soubor obsahující seznam jednotlivých stanic. Tuto URL jsme získali pomocí vývojářských nástrojů prohlížeče Google Chrome v sekci sítě.

Nyní, když máme potřebné zdroje dat, musíme identifikovat prvky, které chceme na této stránce získat. Tyto prvky tvoří jednotlivé položky seznamu stanic pro jednotlivé kraje. Abychom tyto prvky získaly, musíme vytvořit vhodný XPath. XPath je výraz, který slouží pro zvolení uzlu HTML nebo XML souboru.

Následně ve Scrapy vytvoříme pro každý hydrometeorologický typ vlastní crawler. Crawler je samotná jednotka, která pomocí námi definovaných pravidel prochází stránku, ke které mu předáme URL. Pomocí crawlerů tak vytvoříme samotné seznamy stanic pro jednotlivé kraje, které později použijeme jako vstup pro downloader.

Samotná implementace crawlerů je krátká. Jako parametr mu předáme seznam URL pro jednotlivé kraje a následně pomocí námi definovaného parseru získáme data za použití XPath. Výstup uložíme jako CSV soubor.

4.5.2 Downloader

Druhou částí pro získání dat tvoří samotný nástroj Downloader. Na svém vstupu očekává soubory ze scraperu ve složce InputFiles. Princip Downladeru je takový, že pro každý soubor vygeneruje příslušné názvy souborů stanic a URL k jejich získání. Následně tato data stáhne ve formě zip archivu, který otevře a soubory uvnitř rozdělí na dva soubory – data a informace o stanici.

Konkrétně jsme implementaci této části rozdělili do dvou souborů. První soubor obsahuje samotné tělo skriptu, druhý veškeré pomocné funkce.

Samotné tělo skriptu funguje tak, že nejprve zadefinujeme argumenty, které uživatel může použít pro alterování chování skriptu. Tyto argumenty jsou:

debug slouží k zapnutí výpisu debugovacích hlášek do konzole

list–regions slouží k vypsání seznamu dostupných krajů, pro která lze data stáhnout

list–types slouží k vypsání seznamu hydrometeorologických typů, pro která lze data stáhnout

output je argument pro předání cesty, kam se mají uložit stažená data

region argument na specifikování konkrétního kraje, pro něž se mají data stáhnout

silent je přepínač pro vypnutí veškerých výpisů

type je argument pro volbu konkrétního typu, který se má stáhnout

V samotném těle skriptu následně parsujeme vstupní data pomocí Pandas a voláme pomocné funkce definované v separátním souboru ve správném pořadí.

Pomocný soubor tvoří následující sada funkcí. První funkcí je funkce pro vygenerování názvu souboru pro konkrétní datový typ a název stanice. Tento výstup slouží jako vstup další pomocné funkce, která na základě toho vygeneruje příslušnou URL pro stažení. Pomocí další funkce provedeme stažení souboru a uložení do před připravené struktury. Následně podle toho, zda se jedná o hydrologická nebo meteorologická data zavoláme funkci, která zpracuje zip soubor na dva výsledné soubory.

Samotnou URL pro stažení dat jsme získaly opět pomocí vývojářských nástrojů webového prohlížeče Google Chrome v záložce sítě po zaznamenání odchozího dotazu z prohlížeče na server.

4.6 Realizace Databáze

4.6.1 Implementace databáze

Realizace samotné databáze není nijak složitá, jelikož jsme použili oficiální Docker image dostupný pro PostgreSQL. Z důvodu použití on-line nástroje dbdiagram.io [78] pro návrh databáze, můžeme návrh exportovat přímo do SQL create skriptu. Návrh databáze jsme totiž psali pomocí speciálního objektového programovacího jazyka, kde jsme definovali samostatné tabulky, vztahy mezi nimi a následně sloupce tabulek. Následně jsme upravili create skript tak, aby tabulka *hydrometeo_measurement* podporovala partitioning. Pomocí toho tak vytvoříme tabulky pro jednotlivé typy měření, které dědí parametry z této tabulky. Jako parametr, podle kterého se má volit příslušná tabulka zvolíme sloupec *hydrometeo_type*. Následně vytvoříme index pro *region_id* v tabulce *station* a druhý index pro sloupec *date* v tabulce *hydrometeo_measurement*. Samotný insert script jsme v databázi spustili pomocí grafického klienta pgAdmin4. [79]

4.6.2 Vkládání dat

Pro vložení dat do databáze musíme napsat další skript, jelikož soubor je velmi rozsáhlý a kvůli vazbám je potřeba získat dynamicky ID vložených dat. Tento skript, pojmenovaný jako *DB_Data_Inserter* jsme opět implementovali v Pythonu. Struktura je opět stejná, tedy soubor obsahující hlavní tělo skriptu

a pomocný soubor s funkcemi. Navíc jsme však přidali soubor .env pro definování parametrů pro připojení k databázi. Poslední soubor navíc tvoří třída Station.

V hlavním těle skriptu opět zadefinujeme dostupné přepínače. Následně ověříme, že uživatel zadal existující cestu, která obsahuje výstupní strukturu Downloaderu. Pokud ano, dochází k předzpracování dat. Vstupní struktura totiž není vhodná pro vkládání do databáze. Konkrétněji tuto transformaci rozebereme v části věnující se pomocným funkcím. Poté co máme připravená data pro vkládání, zahájíme samotné vkládání dat do databáze. Nejprve je potřeba uložit informace o krajích a hydrometeorologických typech. Vrácené ID těchto dat si uložíme, jelikož je použijeme při vkládání samotných stanic a měření, které následují hned potom. Nakonec po sobě uklidíme pomocnou strukturu, pokud uživatel nezvolil přepínač pro její zachování. Pokud při vkládání došlo k chybě, uživatel je upozorněn.

Třída *Station* slouží k naformátování informací o stanici a vytvoření bitové masky měření stanice. Jelikož do databáze nelze vložit stanici se stejným ID, je potřeba sjednotit duplicity, jelikož jedna stanice může sloužit například jak k měření teploty, tak i k měření vlhkosti a rychlosti větru.

Soubor s pomocnými funkcemi obsahuje pro nás mnoho užitečných funkcí. Kromě funkcí na vkládání dat do databáze, také funkce k ověření datové struktury, uklizení pomocné struktury, tak i funkce pro transformaci dat. Varianta pro stanice funguje tak, že projde všechny dostupné stanice a vytvoří jeden soubor, který obsahuje na řádcích jednotlivé stanice. Před tím, než může tento soubor vytvořit, musí si do paměti ukládat tyto stanice, a pokud narazí na již existující, musí upravit její bitovou masku zavoláním příslušné funkce definované ve třídě *Station*.

Spojování hydrometeorologických dat do jednotlivých kategorií, které jsou použity v naší aplikaci, provádíme pomocí spojování datových rámců, které jsme získali čtením dat pomocí Pandas. Výsledný datový rámec uložíme do nově vzniklé struktury, která organizuje data po nových kategoriích.

Vkládání nově vzniklé struktury probíhá pomocí načtení jednotlivých souboru do datového rámce pomocí Pandas a následně vytvoření SQL příkazu pro vložení pro každý řádek.

4.7 Realizace Backendu

V rámci třídy Server instancujeme express, kterému předáme parametr, aby očekával data ve formátu JSON. Následně mu předáme middleware CORS a námi vytvořené schéma s URI, na kterém provozujeme GraphQL API. Nyní mu můžeme předat port, na kterém má poslouchat.

Samotné Schéma tvoří pouze RootQuery, která je tvořena seznamem námi definovaných query. Samotné Query se skládají z názvu query, popisku, námi

definovaných typů a resolverů, který slouží jakožto návod jak má query získat požadovaná data.

Jelikož mezi datovými typy máme i cyklické vazby, musíme využít programovací techniku vkládání závislostí (Dependency Injection).

Samotné typy obsahují název, popisek, atributy a resolvery pro získání dat, se kterými jsou v relaci. Veškeré resolvery jsou implementovány zvlášť a mají vazbu na modul db, který zprostředkovává připojení na databázi. Dále importují jednotlivé SQL dotazy.

Získávání ročních, měsíčních, nebo i agregovaných statistik pro kraj a stát získáváme z databáze pomocí SQL příkazu.

Jelikož implementace knihovny PG automaticky převádí data a časy na lokální, bylo potřeba změnit toto chování. Řešením je přepsání převodníku pro datum tak, aby vracel nezměněné datum.

Poslední modul je pomocný modul obsahující funkci pro spojení a naformátování data, jelikož z databáze pro měsíční statistiky dostáváme rok a měsíc zvlášť. Druhá funkce nám slouží pro vracení bitové masky pro jednotlivé datové typy, se kterými pracujeme v rámci celé aplikace.

Nakonec bychom zmínili řešení ochrany před SQL Injetion. První ochranu tvoří samotné GraphQL, které přijímá query a parametry zvlášť. Pokud by však někdo vložil přímo parametry do query, stále dochází k ověřování, zda datové typy sedí. Pokud GraphQL detekuje, že nesedí, zahazuje tento dotaz a vrací chybu, jelikož neví co má dělat. Přesto však datový typ String může být zranitelný, proto nevkládáme parametry přímo do SQL, ale používáme parametrizovaný dotaz, kdy SQL dotaz a parametry jsou odeslány zvlášť. Díky tomu lze detekovat pokusy o SQL Injection a zahazovat takovéto dotazy.

4.8 Realizace Frontendu

Hlavní komponentu celé aplikace tvoří komponenta App. Zde dochází k inicializaci všech kontextů, které používáme v rámci celého frontendu. Dále zde dochází k vytvoření samotného klienta pro komunikaci s backendem skrze Apollo Client. Kromě inicializace kontextů také slouží k vytvoření hlavního rozložení celé aplikace do tří částí.

4.8.1 Postranní nabídka

První část tvoří komponenta *SideBar*. Ta je umístěna vlevo. Jedná se o přímou implementaci návrhu tohoto panelu. Samotná komponenta je tvořena kontextem, který udává viditelnost tohoto modulu. Ta je řešena tak, že pokud nemá být navigační lišta vidět, je odebrán celý modul. Samotný modul tvoří logo nahoře, nabídka jazyků, která mění překlad jazyka pomocí změny jazyka v knihovně i18next a přepínače tmavého režimu. Samotnou nabídku tvoří komponenta *NavBar*, která vykresluje tuto nabíku převzatou ze souboru *NavBarContent*. V něm jsou definovány veškeré možnosti, které má *SideBar* nabízet.

Pokud se má jednat o rozbalovací nabídku, je zde definováno jaké podnabídky má obsahovat.

4.8.2 Aplikační lišta

Horní aplikační lišta je implementována pomocí třídy *TopBar*. V základním stavu obsahuje pouze tlačítko pro zobrazení a schování bočního panelu. Podle kontextu dále může zobrazovat další tlačítka pro ovládání aplikace. Základní nastavení kontextu je skryto. Pokud je zvolena obrazovka pro zobrazení aplikace, dochází k vložení a vykreslení tlačítek pro editaci zvolených parametrů. Stisknutím tlačítka dojde k vyvolání dialogového okna, kterému je předán příslušný modul. Po těchto tlačítkách následuje tlačítko na přepnutí se do porovnávacího režimu. Poslední přidaný prvek tvoří tlačítko, které překresluje celou obrazovku dle upravených parametrů.

4.8.3 Okno pro obsah

Třetí část rozhraní aplikace tvoří samotné okno pro obsah. Zde je integrován modul, který přepíná zobrazované komponenty v závislosti na zvolené kategorii v navigačním panelu. Ve výchozím stavu zobrazuje domovskou stránku, kterou tvoří komponenta s nadpisem a logem.

Komunikace frontendu a backendu je realizovaná pomocí hooku useQuery, kterému je předán GraphQL tag a proměnné pro argumenty. Samotný GraphQL tag slouží k vytvoření query, jelikož jako parametr akceptuje string s napsanou query.

Co se týká nabídky parametrů, tak stránka s nimi je realizovaná pomocí vykreslení všech komponent pro jednotlivé možnosti. Těmto modulům jsou předány parametry a funkce pro změnu parametru. Následně se nachází dole tlačítko pro potvrzení volby a vykreslení dat. Samotné parametry jsou průběžně validovány.

Zobrazování dat pomocí grafů je implementováno pomocí modulů *Graph* a *GraphContainer*. Modul *Graph* přebírá jako parametr kolekci, která obsahuje data pro jednotlivé datové řady, které má zobrazit pomocí knihovny Plotly.js. Modul *GraphContainer* slouží k transformaci dat získaných z GraphQL API na jednotlivé datové řady pro knihovnu Plotly.JS. Tyto data následně předává jednotlivým modulům *Graph*, pro vykreslení dat. Zároveň obsahuje přepínač, pomocí kterého si může uživatel zvolit, zda chce zobrazit data pomocí jednoho grafu nebo ve více grafech po jednotlivých datových typech či zdrojů data.

Zobrazení dat pomocí mapy je realizováno modulem *GeoMap*. Ten stejně jako *GraphContainer* dostane z rodičovského modulu data jako parametr. Následně si tyto data transformuje pro vykreslení. Co se týká mapy, jedná se o open-source Leaflet mapy. Pro jejich zprovoznění je potřeba vložit skript tag do *index.html* souboru, který stáhne webovému klientovi potřebná data. Následně lze vytvořit samotný mapový modul, kterému musíme nastavit ve-

likost. Následně předáme sadu dlaždic pro vykreslování mapy jako takové. Poslední věcí je předání GeoJSON souboru, který obsahuje definice polygonů krajů a České republiky, která jsme získaly transformací dat z ČÚZK. Pro změnu data, pro který má mapa data zobrazovat slouží posuvník nad mapou.

Porovnávací režim je implementován podobně. Při kliknutí na příslušné tlačítko dojde k změně modulu na *CompareSelecPage*. Tento modul implementuje stránku pro nastavení parametrů dat, které chce uživatel zobrazit. Tato stránka se skládá ze stejných modulů jako stránka pro nastavení parametrů zobrazovaných dat. Jediný rozdíl tvoří modul pro zvolení si grafů pro zobrazení. Tento modul je zduplikován tak, aby každý ukazoval možnosti pro jednotlivé typy měření.

Kliknutím na tlačítko zobrazit dojde ke změně modulu na *CompareGraphPage*, který obsahuje modul na získání dat z backendu pomocí GraphQL. Tato data jsou následně upravena tak, aby vznikly jednotlivé datové řady, které jsou následně předány modulu *Graph*.

Vypínání a zapíná zobrazení jednotlivých řad je řešeno pomocí Plotly.JS, které toto podporuje pomocí kliknutí na danou řadu v legendě.

Kapitola 5

Testování

V této kapitole se zaměříme na průběh testování vzniklého kódu a samotné implementace. Nejprve v sekci 5.1 si popíšeme způsob testování vzniklého kódu. Následně v sekci 5.2 se podíváme na průběh uživatelského testování, které si také vyhodnotíme a navrhneme případné vhodné úpravy nalezených problémů.

5.1 Programátorské testování

Veškerý vzniklý kód je potřeba otestovat. Kromě testování funkčnosti kódu, musíme ověřit schopnost detekovat nevalidní vstupy a případy s následnou reakci na ně. Je potřeba ověřit, že na tyto případy, včetně mezních, reagujeme dle očekávání. Z tohoto důvodu jsme zvolili 2 způsoby testování kódu na úrovni programátora – uživatelské testování a unit testy.

Pomocí unit testů jsme částečně testovali jednotlivé funkce napsané v části získávání dat. Takto jsme otestovali některé funkcionality jednotlivých částí, ze kterých se skládá byznysová logika. Každý unit test volal příslušnou funkcionalitu s připravenými parametry, a následně zkontroloval výsledek, zda odpovídá očekávání. Pokud nebylo dosaženo očekávaných výsledků, provedli jsme hledání a následné odstraněni chyby v kódu tak, aby vše fungovalo dle očekávání a požadavků.

Druhý typ testování, který jsme převážně prováděli, bylo uživatelské testování. Dle předem připraveného scénáře byla testována každá funkcionalita zvlášť tak, aby jsme simulovali reálného uživatele. Nejprve jsme se snažili používat testovaný prvek dle návrhu a sledovali jsme, zda výsledek odpovídá našemu očekávání. Následně jsme se snažili používat prvek jinak, než bylo zamýšleno a zjišťovali jsme, jak se bude chovat. Pokud jsme narazili na nechtěné chování, upravili jsme vlastnosti testovaného prvku. Takto jsme testovali veškeré části implementace průběžně.

5.2 Uživatelské testování

Na závěr jsme se rozhodli podrobit vzniklý webový portál uživatelskému testování pomocí dalších lidí. Cílem tohoto testování bylo hlavně otestovat vhodnost návrhu uživatelského rozhraní, srozumitelnost a přehlednost. Následně vyhodnotit toto uživatelské testování, zhodnotit kvalitu návrhu a navrhnout případné úpravy pro budoucí rozšíření UI.

5.2.1 Metodika

Průběh uživatelského testování byl rozdělen do čtyř hlavních částí. Nejprve uživatel vyplnil dotazník (dotazník je přiložen jako příloha C), pomocí kterého jsme zjišťovali jaké má uživatel znalosti PC, zda někdy používal nějakou takovouto aplikaci a co od ní očekává.

Následně jsme provedli uživatelské testování podle předem připravených scénářů, které měly za cíl zjistit, jakým způsobem se bude orientovat uživatel na webové stránce při jeho první návštěvě. Scénáře byly vytvořeny dle námi očekávaného způsobu používání vzniklého webového portálu.

První scénář byl zaměřen na zobrazení dat pomocí mapy a grafů. Na začátku testu se uživatel nacházel na domovské stránce. Nejprve musel zvolit příslušnou kategorii a nastavit parametry pro zobrazení dat. Nakonec musel najít v grafu a na mapě požadovanou hodnotu pro příslušný den.

Ve druhém scénáři měl uživatel za úkol zvolit jinou kategorii dat, nastavit jiné parametry a zvolit kombinaci dvou veličin z vícero krajů. Následně musel najít požadovanou hodnotu. Cílem tohoto scénáře bylo otestovat přehlednost zobrazení vícero dat pomocí grafů.

Třetí scénář jsme zaměřili na úpravu zobrazených dat. Po splnění druhého scénáře měl uživatel za úkol upravit zobrazená data tak, aby změnil granulitu dat, časový rozsah a přidal třetí statistiku.

Poslední scénář testoval porovnávací režim. Uživatel byl vyzván k přepnutí se do porovnávacího režimu, nastavení parametrů a nalezení požadované hodnoty.

Po testování pomocí scénářů, dostal uživatel čas, aby si mohl projít portál dle vlastního uvážení. Když usoudil, že si prohlédl web dostatečně, vyplnil následně druhý dotazník (příloha D). Zde jsme se ptali na dojmy z používání, přehlednost a srozumitelnost, co bylo dobré na aplikaci a co naopak nebylo. Nakonec testovaný uživatel udělil známku 1 až 5, kdy 1 je nejlepší a 5 nejhorší.

5.2.2 Výsledky

Uživatelského testování se zúčastnili celkem čtyři lidé. Cílem nebylo primárně testovat funkčnost aplikace, ale návrh uživatelského rozhraní. Již na základě takto malé testovací skupiny však lze vyvodit obecné závěry a doporučení pro budoucí úpravy.

Nejprve si lehce představíme testovací skupinu. Co se týče znalosti PC, lze skupinu rozdělit na dvě poloviny, kdy jedna zahrnuje běžné uživatele PC a druhá pokročilé uživatele PC. Všichni uživatele již někdy používali aplikaci na vizualizaci dat, avšak až na jednu osobu se nejednalo o aplikaci zaměřenou na ekologická data. Přesto však všichni vyjádřili zájem o použití takovéto aplikace. Co se týče očekávání od aplikace, uživatelé uváděli shodné návrhy, které jsme si stanovili na začátku v rámci kapitoly 2.

Co se týče výsledků samotného testování dle scénářů, proběhlo vše úspěšně. Přesto však lze najít místa v samotném návrhu, která by šla vylepšit v budoucích verzích, avšak nejedná se o žádné zásadní problémy. Z pozorovaní řešení scénářů bylo vidět, že obecný návrh je dobrý, ale uživatelům tvořily menší problémy úkoly spojené s úpravou parametrů zobrazovaných dat. Pár uživatelů nejprve volilo možnost znovu otevření kategorie s kompletním nastavením nového zobrazení a ručně znovu vyplňovali parametry. Až při dalším požadavku na úpravu dat jim došlo, že zde musí být lehčí způsob na úpravu a začali hledat tato tlačítka. Poté co je nalezli již vše probíhalo dle očekávání. Jako nedostatek se jeví nedostatečná indikace zvolené úrovně. Jelikož výchozí stav zvolené úrovně je stanice, uživatelé se nejdříve snažili nastavit jako zdroj dat jen kraj, bez přepnutí úrovně. Dostávali se tedy do situace, kdy nebyli parametry správně zvolené a chvíli tápali, než změnili úroveň. Další problémy již nebyli pozorovány.

Výsledky získané z druhého dotazníku nám ukazují, že návrh uživatelského rozhraní se osvědčil a splnil naše očekávání a požadavky. Průměrná známka, udělená účastníky testování, je dvojka. Všichni uživatelé hlásili příjemné pocity z používání aplikace a vyslovili se, že by aplikaci používali i v budoucnu. Všem uživatelům se líbili funkcionality a nikomu nechyběla žádná funkcionalita. Všichni uživatelé uvedli, že aplikace byla přehledná, srozumitelná a dobře vypadající. Přesto však uvedli, že měli problém s tlačítky v horním panelu pro změnu parametrů dat a přepnutí se do porovnávacího režimu. Jako důvod uváděli, že tlačítka působí jakožto nastavení na vyšší úrovni i díky grafickému oddělení. Na druhou stranu, kromě tohoto problému si nikdo na nic jiného nestěžoval. Avšak někteří uživatelé hlásili problém s plynulostí aplikace při zobrazování většího množství dat.

Problém s rychlostí aplikace jsme se rozhodli řešit pomocí omezení rozsahu dat, které uživatel může mít v jednu chvíli lokálně k dispozici. Následně jsme snížili počet zdrojů dat na maximálně tři z původních pěti pro stanici a čtrnácti pro kraj. Do budoucna by bylo vhodné navrhnout zlepšení v této části, jelikož pro účely této práce je aktuální omezení dostačující. Toto by šlo řešit pomocí shlukovaní dat do bloků či jiné metody optimalizace dat na straně backendu. Další možnost je vykreslovat stránku na straně serveru a následně ji odeslat na klienta, avšak toto nese sebou nevýhodu, kdy uživatel ztratí možnost interagovat dynamicky se zobrazenými grafy.

Mezi návrhy úprav UI do budoucích aktualizací lze zařadit úpravu horních tlačítek. Z dotazníku a následné diskuse po testování nám vyplývá, že uživatelé

vidí primárně tlačítka v aplikační liště jakožto nastavení vyššího řádu. Na základě tohoto navrhujeme přesunutí těchto tlačítek na úroveň stránky okna se zobrazeným obsahem a případném odebrání aplikační lišty úplně. Také lze změnit tlačítka z ikon na klasická s nápisem.

Dalším návrhem na rozšíření UI je úprava nastavení parametrů. Jelikož uživatelé při testování nevyplňovali parametry od shora dolů, ale přeskakovali nabídky, došlo párkrát ke zmatení uživatele, který nepřepnul přepínač v jiné nabídce. K zamezením možnosti dezorientace uživatele tak navrhujeme změnit nabídku pro nastavení parametrů do vícekrokové nabídky. Toto řešení však přináší nevýhodu, kdy nastavit parametry bude trvat déle a může to tak uživatele zdržovat při častém vyhledávání.

Pokud tedy shrneme výsledky uživatelského testování, aktuální návrh se ukázal jako vhodný. Samotné uživatelské rozhraní splnilo očekávání a námi stanovené podmínky na přehledný, intuitivní a moderně vypadající systém. Zároveň jsme uvedli možnosti upravení návrhu v místech, které nám přijdou, že by šlo ještě vylepšit pro maximální uživatelský komfort.

Závěr

Cílem této práce je analyzovat, navrhnout, implementovat a otestovat webový portál pro vizualizaci dat spojených s ekologií. Jako dílčí cíle jsme zvolili analýzu vhodných dat, analýzu požadavků na systém, návrh jednotlivých částí s jejich následnou implementací a nakonec uživatelské testování pro otestování správnosti návrhu.

Samotný cíl práce, analýzu, jsme splnili v kapitole 2, v rámci které jsme zanalyzovali dostupná data a zvolili si konkrétní data. Dále jsme si zadefinovali požadavky na výsledné řešení, které tvoří hlavně požadavky na srozumitelnost a přehlednost. Nakonec jsme provedli analýzu již existujících řešení.

Poznatky z kapitoly 2 jsme následně využili ke splnění cíle návrhu architektury aplikace, získání dat, databáze, backendu a samotného webového portálu. Tento cíl jsme splnili v kapitole 3. Přestože jsme se věnovali všem částem práce, hlavní důraz jsme kladli na návrh UI. Výsledkem je komplexní návrh všech uživatelských obrazovek jak pro PC, tak i pro mobilní zařízeni. Zmíněné WF si lze prohlédnout v příloze na médiu v sekci obrázky.

Dle návrhu, z kapitoly 3, jsme implementovali jednotlivé části praktické části práce. Pomocí scraperu jsme získali potřebné informace, které jsme následně použili pro stažení dat z ČHMÚ. Také jsme implementovali databázi za použití metody partitioningu a vložili do ní data. Následně jsme vytvořili backend, který zašle požadavek na databázi, aby nalezl data, které odešle webovému klientu pomocí API implementovaného v GraphQL. Dále jsme implementovali webového klienta pomocí React.JS, který zobrazí data pomocí mapy a grafů. Dále umožňuje kombinovat různé datové typy pro zobrazení pomocí grafů. Implementačním detailům jsme se věnovali v kapitole 4.

Po dokončení implementace jsme provedli uživatelské testování webové aplikace. Výsledky testování jsme shrnuli v kapitole 5. Návrh UI se při tomto testování osvědčil. Kromě splnění našich požadavků všichni účastníci chválili aplikaci a UI. Dále také projevili zájem o takovouto aplikaci a v dotazníku uvedli, že mají zájem v budoucnu takovouto aplikaci znovu použít.

Co se týká návrhů pro budoucí rozšíření, lze v oblasti UI implementovat

námi navržené úpravy v sekci 5.2. Také lze rozšířit portál o další zajímavé statistiky nad již existujícími daty, jako je například počet tropických dnů či arktických dnů a podobně. Další možností pro rozšíření je přidání dalších datových sad. Vhodné datové sady jsme zmínili v sekci 2.1. Dále lze rozšířit portál i o informační text. Portál tak nemusí sloužit pouze pro vizualizaci dat, ale i jako informační centrum pro zájemce o tuto problematiku. Jako poslední možnost na rozšíření lze provést úpravu práce s daty tak, aby bylo možné zobrazit mnohem více dat v jeden okamžik, jelikož aktuální návrh má omezení.

Nakonec bychom rádi zmínili, že nedávno došlo k dokončení úprav portálu ČHMÚ a tedy implementovaná část pro získávání dat již není aktuální ke dni 25. 6. 2021. Bohužel toto má za následek i to, že již nelze získat data pomocí dowloaderu. Řešením tohoto problému je napsání nového scraperu, pomocí kterého lze již přímo stáhnout data (nový vzhled portálu toto již umožňuje bez problémů). Následně stačí přesunout část downloaderu, která předzpracovává data přímo do inserteru.

Literatura

[1] Google LLC: Ukázka výškového umístění prvků v aplikaci. [online], [09.05.2018], [cit. 2021-04-14]. Dostupné z: https://lh3.googleusercontent.com/IGup0QloUkkozMfTPbOYOCjUZrfaeNUMLKRBJ6fLBgSUj0_WnzNNy1GsyUijEJyEqDYGqbEW1G3UFO3XOLdZcfd278WCK01TWMOlWQ=w1064-v0

[2] Google LLC: Diagram umístění prvků v aplikaci, včetně dynamického posunu. [online], [09.05.2018], [cit. 2021-04-14]. Dostupné z: https://lh3.googleusercontent.com/pEk_CLv7V6MroLmKY5rA5nCknpKI0ltFZn5yypHiAt0zJzWd5frkMaj2VXWpQmcPtoA_8P-2n0wg9I4JgCWSOKN5nUSgN7IActWvZw=w1064-v0

[3] Google LLC: A UI showcases the baseline colors for text and iconography. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://lh3.googleusercontent.com/gPYvk6uMwqQaFsrdSLIN5w0sZrMAkXy8jtVLJR-uYPnw_OvA5Kbu56GXg2xb9oKd8VK03EMtelb9Gt5FWXO8fBm7xitlutKqY6gpYQ=w1064-v0

[4] Google LLC: Layout grid. [online], [09.05.2018], [cit. 2021-04-14]. Dostupné z: https://lh3.googleusercontent.com/r6fKrpfXlzXvGokFAgYHp_FE3JRXT4GLoSMPqzQqB4R-wMHYqc94xyHnF0IE4Ndtfmc6BvDBockiOSVnbdTohlj6ftAlV2mLjevy=w1064-v0

[5] Bova, S.; Rosenthal, Y.; Liu, Z.; aj.: Seasonal origin of the thermal maxima at the Holocene and the last interglacial. *nature*, ročník 589, č. 7843, 2021: s. 548–553.

[6] Walker, P. A.: Political ecology: where is the ecology? *Progress in human geography*, ročník 29, č. 1, 2005: s. 73–82.

[7] Český hydrometeorologický ústav: Historická data - meteorologie a klimatologie. [online], [1997], [cit. 2021-04-25]. Dostupné z: https://www.chmi.cz/historicka-data/pocasi/zakladni-informace

[8] Operátor ICT, a. s.: Zelené střechy. [online], ©2018, [cit. 2021-05-11]. Dostupné z: https://golemio.cz/cs/node/1443

[9] Operátor ICT, a. s.: Kvalita ovzduší. [online], ©2018, [cit. 2021-05-11]. Dostupné z: https://golemio.cz/cs/node/229

[10] Operátor ICT, a. s.: Odpadové hospodářství. [online], ©2018, [cit. 2021-05-11]. Dostupné z: https://golemio.cz/cs/node/230

[11] Český statistický úřad: Statistiky. [online], Vygenerováno 11.05.2021, [cit. 2021-05-11]. Dostupné z: https://vdb.czso.cz/vdbvo2/faces/cs/index.jsf?page=statistiky

[12] Otevřená data resortu životního prostředí: Datové sady. [online], [2017], [cit. 2021-05-11]. Dostupné z: https://opendata.mzp.cz/dataset

[13] Agentura ochrany přírody a krajiny České republiky: Domů. [online], [cit. 2021-04-25]. Dostupné z: https://aopkcr.maps.arcgis.com/home/index.html

[14] NOAA: Homepage. [online], 2010, [cit. 2021-04-25]. Dostupné z: https://www.climate.gov/

[15] Česká geologická služba: On-line aplikace. [online], [2007], [cit. 2021-04-25]. Dostupné z: http://www.geology.cz/extranet/sluzby/aplikace/

[16] Český statistický úřad: Statistiky. [online], Vygenerováno 25.04.2021, [cit. 2021-04-25]. Dostupné z: https://vdb.czso.cz/vdbvo2/faces/cs/index.jsf?page=statistiky

[17] Golemio: Homepage. [online], ©2018, [cit. 2021-04-25]. Dostupné z: https://golemio.cz/

[18] CENIA: NAVIGACE. [online], [2011], [cit. 2021-04-25]. Dostupné z: https://issar.cenia.cz/#zp-cr

[19] InMeteo, s.r.o.: Archiv počasí. [online], ©2021, [cit. 2021-04-25]. Dostupné z: https://www.in-pocasi.cz/archiv/

[20] Ecma International: ECMAScript® 2016 Language Specification. [online], ©2016, [cit. 2021-06-24]. Dostupné z: https://262.ecma-international.org/7.0/

[21] Český hydrometeorologický ústav: Denní data dle zákona 123/1998 Sb. [online], [1997], [cit. 2021-04-25]. Dostupné z: https://www.chmi.cz/historicka-data/hydrologie/denni_data/denni-data-dle-z.-123-1998-Sb

[22] Python Software Foundation: History and License. [online], ©2001-2021, [cit. 2021-04-20]. Dostupné z: https://docs.python.org/3/license.html

[23] Altexsoft: Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch and others. [online], 2019-06-20, [cit. 2021-04-29]. Dostupné z: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/

[24] Schutzberg, A.: Oracle Buys Sun: What Happens to MySQL? [online], 2009-04-24, [cit. 2021-04-29]. Dostupné z: https://www.directionsmag.com/article/2357

[25] Red Hat, Inc.: What is a REST API? [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[26] Fielding, R. T.: *Architectural styles and the design of network-based software architectures*, ročník 7. University of California, Irvine Irvine, 2000, 76–86 s.

[27] The GraphQL Foundation: Introduction to GraphQL. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://graphql.org/learn/

[28] Red Hat, Inc.: What is GraphQL? [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-graphql

[29] The GraphQL Foundation: Code using GraphQL. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://graphql.org/code/

[30] Google LLC: Design. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design

[31] Brian, M.: Google's new 'Material Design' UI coming to Android, Chrome OS and the web. [online], 25.06.2014, [cit. 2021-04-11]. Dostupné z: https://www.engadget.com/2014-06-25-googles-new-design-language-is-called-material-design.html

[32] Wilson, M.: Google's New, Improved Android Will Deliver A Unified Design Language. [online], 25.06.2014, [cit. 2021-04-11]. Dostupné z: https://www.fastcompany.com/3032378/googles-new-improved-android-will-deliver-a-unified-design-language

[33] Google LLC: Elevation. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/environment/elevation.html

[34] Google LLC: Surfaces. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/environment/surfaces.html#material-environment

[35] Google LLC: Light and shadows. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/environment/light-shadows.html#light

[36] Google LLC: Introduction. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/introduction

[37] Google LLC: Material Theming. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/material-theming/overview.html#material-theming

[38] Google LLC: The color system. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/color/the-color-system.html#color-usage-and-palettes

[39] Google LLC: The type system. [online], [09.05.2018], [cit. 2021-04-08]. Dostupné z: https://material.io/design/typography/the-type-system.html#type-scale

[40] Google LLC: System icons. [online], [09.05.2018], [cit. 2021-04-12]. Dostupné z: https://material.io/design/iconography/system-icons.html#system-icon-metrics

[41] Google LLC: Product icons. [online], [09.05.2018], [cit. 2021-04-12]. Dostupné z: https://material.io/design/iconography/product-icons.html#design-principleshttps://material.io/design/iconography/product-icons.html#design-principles

[42] Google LLC: Understanding layout. [online], [09.05.2018], [cit. 2021-04-14]. Dostupné z: https://material.io/design/layout/understanding-layout.html#

[43] Google LLC: Responsive layout grid. [online], [09.05.2018], [cit. 2021-04-14]. Dostupné z: https://material.io/design/layout/responsive-layout-grid.html#columns-gutters-and-margins

[44] Lufthansa: Homepage. [online], [2010], [cit. 2021-04-13]. Dostupné z: https://www.lufthansa.com/cz/en/homepage

[45] Dhaduk, H.: Best Frontend Frameworks of 2021 for Web Development. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://www.simform.com/best-frontend-frameworks

[46] You, E.: The Progressive JavaScript Framework. [online], ©2014-2021, [cit. 2021-05-12]. Dostupné z: https://vuejs.org/

[47] Microsoft: TypeScript for the New Programmer. [online], ©2012-2021, [cit. 2021-04-11]. Dostupné z: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

[48] Google: Version Support Status. [online], ©2010-2020, [cit. 2021-04-11]. Dostupné z: https://code.angularjs.org/snapshot/docs/misc/version-support-status

[49] Eidnes, L.: AngularJS: The Bad Parts. [online], 5.11.2014, [cit. 2021-04-12]. Dostupné z: https://larseidnes.com/2014/11/05/angularjs-the-bad-parts/

[50] Docker Inc: Homepage. [online], ©2013-2021, [cit. 2021-05-12]. Dostupné z: https://www.docker.com/

[51] Microsoft: Homepage. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://code.visualstudio.com/

[52] Microsoft: Docker. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker

[53] dsznajder: ES7 React/Redux/GraphQL/React-Native snippets. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets

[54] Microsoft: Python. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://marketplace.visualstudio.com/items?itemName=ms-python.python

[55] Docker Inc: Docker overview. [online], ©2013-2021, [cit. 2021-05-12]. Dostupné z: https://docs.docker.com/get-started/overview/

[56] Docker Inc: Overview of Docker Compose. [online], ©2013-2021, [cit. 2021-05-12]. Dostupné z: https://docs.docker.com/compose/

[57] Free Software Foundation, Inc.: Licenses. [online], ©2014-2021, [cit. 2021-05-12]. Dostupné z: http://www.gnu.org/licenses/

[58] Massachusetts Institute of Technology: The MIT License. [online], [2011], [cit. 2021-05-12]. Dostupné z: https://opensource.org/licenses/MIT

[59] Berkeley Software Distribution: The 3-Clause BSD License. [online], [2011], [cit. 2021-05-12]. Dostupné z: https://opensource.org/licenses/BSD-3-Clause

[60] The pandas development team: Homepage. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://pandas.pydata.org/

[61] The Psycopg Team: Homepage. [online], ©2010—2021, [cit. 2021-05-12]. Dostupné z: https://www.psycopg.org/

[62] Kumar, S.: python-dotenv 0.18.0. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://pypi.org/project/python-dotenv/

[63] Bieniek, T.: utm 0.7.0. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://pypi.org/project/utm/

[64] Goode, T.: expressjs/cors. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/expressjs/cors

[65] Motte, S.: motdotla/dotenv. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/motdotla/dotenv#readme

[66] OpenJS Foundation: Express. [online], ©2017, [cit. 2021-05-12]. Dostupné z: http://expressjs.com/

[67] The GraphQL Foundation: graphql/express-graphql. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/graphql/express-graphql

[68] The GraphQL Foundation: graphql/graphql-js. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/graphql/graphql-js

[69] Rutten, D.-J.: excitement-engineer/graphql-iso-date. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/excitement-engineer/graphql-iso-date

[70] Carlson, B.: brianc/node-postgres. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://github.com/brianc/node-postgres

[71] Facebook Inc: Introducing JSX. [online], ©2021, [cit. 2021-05-13]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[72] Apollo Graph Inc.: Introduction to Apollo Client. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://www.apollographql.com/docs/react/

[73] Material-UI: MATERIAL-UI. [online], ©2021, [cit. 2021-05-12]. Dostupné z: http://expressjs.com/

[74] i18next: Introduction. [online], [2017], [cit. 2021-05-12]. Dostupné z: https://react.i18next.com/

[75] Agafonkin, V.: Leaflet. [online], ©2010–2021, [cit. 2021-05-12]. Dostupné z: https://leafletjs.com/

[76] Plotly: Plotly JavaScript Open Source Graphing Library. [online], ©2021, [cit. 2021-05-12]. Dostupné z: https://plotly.com/javascript/

[77] Český hydrometeorologický ústav: Podmínky užití. [online], [1997], [cit. 2021-04-25]. Dostupné z: https://www.chmi.cz/podminky-uziti

[78] Holistics Software: Homepage. [online], ©2021, [cit. 2021-05-13]. Dostupné z: https://dbdiagram.io/home

[79] The pgAdmin Development Team: pgAdmin. [online], ©2013-2021, [cit. 2021-04-25]. Dostupné z: https://www.pgadmin.org/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CSV Comma-Separated Values

ČHMÚ Český hydrometeorologický ústav

ČÚZK Český úřad zeměměřický a katastrální

DOM Document Object Model

DP Device independent pixel

GUI Graphical User Ïnterface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ISSaR Informační systém statistiky a reportingu v životním prostředí

JS JavaScript

JSON JavaScript Object Notation

PC Personal Computer

REST Representational State Transfer

SQL Structured Query Language

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

WF Wireframe

WGS84 World Geodetic System 1984

XML Extensible Markup Language

Příloha B

Uživatelská příručka

V této kapitole bude popsáno jak nainstalovat jednotlivé části řešení a jak je ovládat.

B.1 Administrátorská příručka

B.1.1 Instalace

B.1.1.1 Získávání dat

1. Nainstalujte na zařízení Python a Conda.
2. Přesuňte se do složky Data a v příkazovém řádku v této složce spusťte příkaz *conda env update*.

B.1.1.2 Databáze

Následující návod popisuje instalaci databáze pomocí Docker. Při použití lokální instalace Postgres se řiďte návodem dle dokumentace PostgreSQL a pokračujte v návodu od kroku číslo 6.

1. Nainstalujte na své zařízení Docker.
2. Pokud není součástí instalace Docker, nainstalujte Docker Compose.
3. Nainstalujte na své zařízení PGAdmin.
4. Nastavte parametry v souboru .env.example, který se nachází v kmenové složce a přejmenujte jej na .env. Vytvořte .evn soubory i ve složce Backend a Frontend/eco-web.
5. V kmenové složce spusťte příkaz *Docker Compose up -d* v příkazové řádce. Pokud však chcete použít vlastní kontejner, stáhněte si oficiální PostgreSQL image a dle dokumentace vytvořte kontejner.

6. Otevřete PGAdmin a přidejte databázový server.

7. Pokud databázový server neobsahuje databázi s názvem eco-web, vytvořte jej pomocí kliknutí pravím tlačítkem myši na server a zvolením *create new database*.

8. Klikněte pravím tlačítkem na databázi eco-web a zvolte *Restore* pro vytvoření databáze pomocí dumpu, který je přiložen ve složce Data/ Scripts, nebo zvolte *Query Tool* možnost a vložte Create Script, která se také nachází ve složce Data/Scripts, který následně spusťte.

B.1.1.3 Backend

Pro instalaci pomocí Docker se řiďte následujícím návodem.

1. Nainstalujte na své zařízení Docker.

2. Pokud není součástí instalace Docker, nainstalujte Docker Compose.

3. Pokud jste již neprovedli při instalaci databáze, nastavte parametry v souboru .env.example, který se nachází v kmenové složce a přejmenujte jej na .env. Vytvořte .evn soubory i ve složce Backend a Frontend/eco-web.

4. Pokud jste již neprovedli při instalaci databáze, v kmenové složce spusťte příkaz *Docker Compose up -d* v příkazové řádce.

Pro instalaci bez použití Docker se řiďte následujícím návodem.

1. Nainstalujte na své zařízení NodeJS.

2. Přesuňte se do složky Backend.

3. Nastavte parametry v souboru .env.example a přejmenujte jej na .env.

4. V příkazovém řádku v této složce spusťte příkaz *npm install*

B.1.1.4 Frontend

Pro instalaci pomocí Docker se řiďte následujícím návodem.

1. Nainstalujte na své zařízení Docker.

2. Pokud není součástí instalace Docker, nainstalujte Docker Compose.

3. Pokud jste již neprovedli při instalaci databáze, nastavte parametry v souboru .env.example, který se nachází v kmenové složce a přejmenujte jej na .env. Vytvořte .evn soubory i ve složce Backend a Frontend/eco-web.

4. Pokud jste již neprovedli při instalaci databáze, v kmenové složce spusťte příkaz *Docker Compose up -d* v příkazové řádce.

Pro instalaci bez použití Docker se řiďte následujícím návodem.

1. Nainstalujte na své zařízení NodeJS.

2. Přesuňte se do složky Frontend/eco-web.

3. Nastavte parametry v souboru .env.example a přejmenujte jej na .env.

4. V příkazovém řádku v této složce spusťte příkaz *npm install*

B.1.2 Spuštění

B.1.2.1 Získávání dat

Pro spuštění částí pro získání dat se řiďte následujícím návodem.

1. Pro použití Sraperu se přesuňte do složky Data/Scraper, pro použití Downloaderu se přesuťe do složky Data/Downloader a pro použití Data Inserteru se přesuňte do složky Data/DB_Data_Inserter.

2. Použijte ve složce příkaz *conda activate eco-web* v příkazovém řádku.

3. Následně použijte příkaz *scrapy crawl [název_spideru]* pro Scraper, *python CHMI_Downloader [zvolené_přepínače]* pro Downloader a *python Data_Inserter [zvolené_přepínače]* pro Data Inserter.

B.1.2.2 Databáze

Pokud Docker Compose již neběží, použijte příkaz *Docker Compose up -d* v příkazové řádce v kmenovém adresáři. Pro smazání použijte *Docker Compose down* v příkazové řádce v kmenovém adresáři a pro zastavení *Docker Compose stop* v příkazové řádce v kmenovém adresáři.

B.1.2.3 Backend

Pokud Docker Compose již neběží, použijte příkaz *Docker Compose up -d* v příkazové řádce v kmenovém adresáři. Pro smazání použijte *Docker Compose down* v příkazové řádce v kmenovém adresáři a pro zastavení *Docker Compose stop* v příkazové řádce v kmenovém adresáři. Pro spuštění Backendu mimo Docker, spusťte příkaz *npm start* v adresáři Backend.

B.1.2.4 Frontend

Pokud Docker Compose již neběží, použijte příkaz *Docker Compose up -d* v příkazové řádce v kmenovém adresáři. Pro smazání použijte *Docker Compose down* v příkazové řádce v kmenovém adresáři a pro zastavení *Docker Compose stop* v příkazové řádce v kmenovém adresáři. Pro spuštění Frontendu mimo Docker, spusťte příkaz *npm start* v adresáři Frontend/eco-web.

B.2 Uživatelská příručka

B.2.1 Scraper

Nejprve je potřeba provést instalaci potřebných součásti dle návodu v sekci B.1.1.1. Následně lze postupovat dle návodu na spuštění v sekci B.1.2.1. Dostupné přepínače jsou následující:

-h zobrazí nápovědu se všemi dostupnými přepínači

-o slouží k nastavení cesty, kam se mají stáhnout soubory

-r slouží ke zvolení konkrétního kraje, pro který budou stažena data

-t slouží ke zvolení konkrétního datového typu, pro který budou stažena data

-lr slouží k vypsání všech dostupných krajů

-lt slouží k vypsání všech dostupných datových typů

-s slouží k vypnutí veškerých výpisů.

-d slouží k zapnutí debugovacích výpisů.

B.2.2 Downloader

Nejprve je potřeba provést instalaci potřebných součásti dle návodu v sekci B.1.1.1. Následně lze postupovat dle návodu na spuštění v sekci B.1.2.1. Dostupné přepínače jsou následující:

-h zobrazí nápovědu se všemi dostupnými přepínači

-p slouží k nastavení cesty, odkud se mají číst data

-ktf slouží k zachování pomocné struktury při vkládání dat

-im slouží k ignorování již vložených dat při vkládání po částech

-d slouží k zapnutí debugovacích výpisů.

B.2.3 Backend

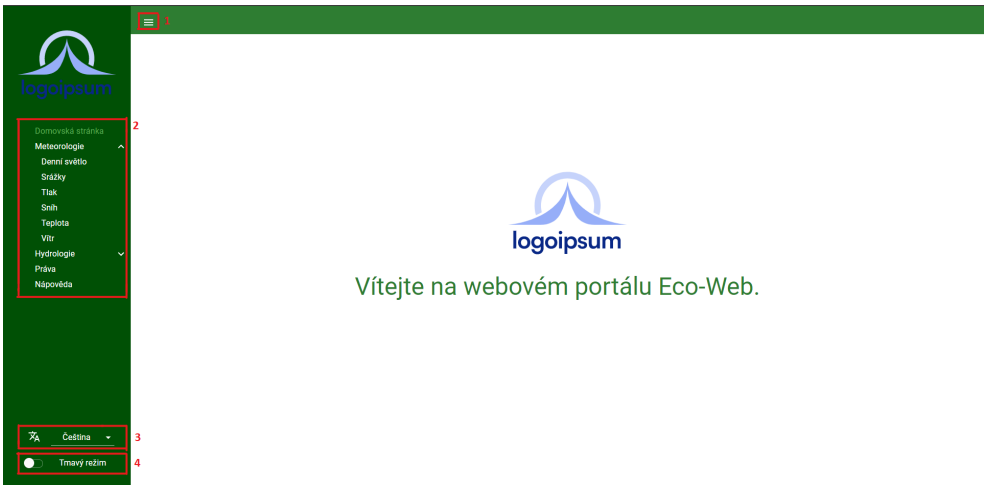
Uživatel nemá důvod pro interakci s backendem. Pokud by však měl zájem, na API jsme nechali zapnutý grafický editor pro prozkoumání API s možností si psát vlastní query dotazů. Toto rozhraní je velmi užitečné, pokud by někdo chtěl psát vlastní aplikaci, která by využívala toto API. Grafický editor je dostuný na *URI/graphql*.

B.2.4 Frontend

Z důvodu větší přehlednosti jsme se rozhodli udělat uživatelskou příručku webového portálu pomocí názorných obrázků implementovaného webu s popisky.

Domovská obrazovka

1. Tlačítko pro skrytí/zobrazení bočního ovládacího panelu.
2. Navigační panel webu.
3. Nastavení jazyka.
4. Přepínač tmavého režimu.

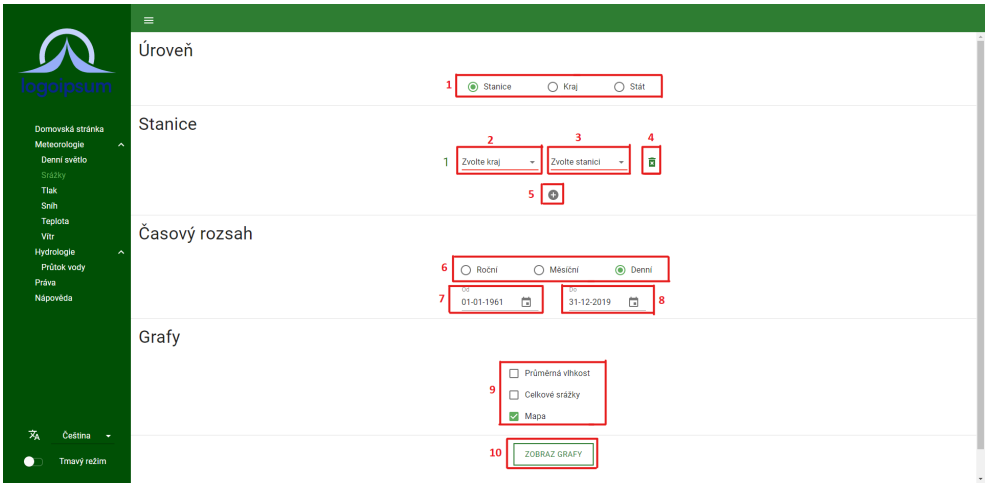


Obrázek B.1: Obrázek zachycující domovskou obrazovku pro návod.

Nabídka parametrů

1. Přepínač úrovně zobrazeni dat.
2. Nabídka krajů.

3. Nabídka stanic.
4. Tlačítko pro odebrání řádku.
5. Tlačítko pro přidání řádku.
6. Přepínač četnosti měření.
7. Nastavení počátečního data měření.
8. Nastavení koncového data měření.
9. Nabídka statistik pro zobrazení.
10. Tlačítko pro zobrazení dat dle nastavených parametrů.

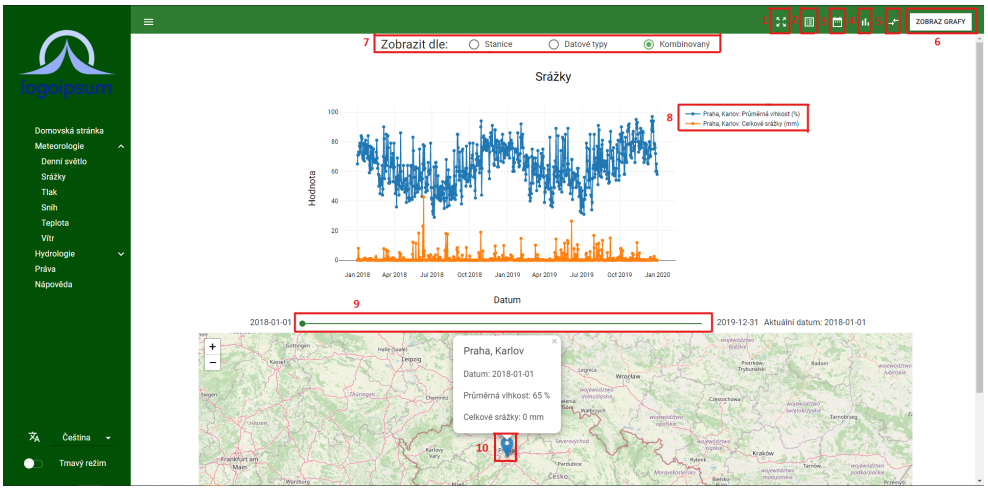


Obrázek B.2: Obrázek zachycující obrazovku s výběrem parametrů pro návod.

Zobrazená data

1. Tlačítko pro změnu úrovně zobrazení dat.
2. Tlačítko pro změnu zdrojů zobrazeních dat.
3. Tlačítko pro změnu datového rozsahu zobrazení dat.
4. Tlačítko pro změnu zobrazovaných statistik.
5. Tlačítko pro přepnutí aplikace do porovnávacího režimu.
6. Tlačítko pro aktualizaci zobrazených dat dle upravených parametrů.
7. Přepínač zobrazení dat v grafech.

8. Legenda grafu s možností vypnutí a zapnutí zobrazení konkrétní datové řady.
9. Posuvník pro posouvání data, pro který se mají ukazovat měření v mapě.
10. Značka na mapě. Po kliknutí dojde k zobrazení statistik.



Obrázek B.3: Obrázek zachycující obrazovku se zobrazenými daty pro návod.

Ovládání je koncipováno tak, že si uživatel nejprve v navigačním panelu zvolí kategorii, která ho zajímá. Následně vyplní požadované parametry zobrazené na obrázku B.2. Poté, co je s nastavením spokojen, zvolí tlačítko zobrazit grafy, které jej přesunou na další obrazovku (B.3). Zde může upravovat zobrazení dat dle preferovaného způsobu, nebo pomocí mapy. Pokud by se uživatel rozhodl upravit parametry zobrazení dat, může toto provést pomocí tlačítek v pravém horním rohu. Nejprve upraví parametry a následným stisknutím tlačítka zobrazit grafy dojde k překreslení zobrazení dat.

Příloha C

Dotazník před testem

- Jaké jsou Vaše znalosti PC:
  - Minimální uživatel
  - Běžný uživatel
  - Pokročilý uživatel
  - Profesionál
- Používal(a) jste někdy nějakou aplikaci na vizualizaci dat?
- Použil/a jste někdy nějakou aplikaci na vizualizaci ekologických dat?
- Měl/a byste zájem použít takovou aplikaci?
- Co byste od takové aplikace očekával/a?

Příloha D

Dotazník po testu

- Nastaly při používání nějaké potíže?
- Jaké jsou Vaše pocity z používání této aplikace?
- Bylo vše přehledné?
- Bylo vše srozumitelné?
- Použil(a) byste v budoucnu takovou aplikaci znovu?
- Můžete specifikovat co se Vám na aplikaci líbilo?
- Můžete specifikovat co se Vám na aplikaci nelíbilo?
- Fungovalo vše podle očekávání?
- Chyběla zde nějaká očekávaná funkcionalita?
- Jakou školní známkou (1 - nejlepší, 5 - nejhorší) byste aplikaci hodnotil(a)?

Příloha E

Odkaz na repozitář implementace na GIT

https://github.com/opendatalabcz/eco-web

Příloha F

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `doc` ..... dokumentace zdrojového kódu implementace
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `obrazky` ..... obrázky grafů a wireframů použitých v práci