Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Tvorba webové aplikace v Node.js

David Homolka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

David Homolka

Informatika

Název práce

Tvorba webové aplikace v Node.js

Název anglicky

Web application development in Node.js

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje a tvorby webové aplikace v Node.js. Hlavním cílem práce je vytvoření reálné aplikace pro prezentaci hydrometeorologických dat s využitím open source řešení. Dílčími cíli práce jsou:
- vypracování přehledu vývoje dynamických technologií na straně webového klienta,
- komparace vytvořené aplikace s podobnými aplikacemi.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část práce spočívá ve vývoji reálné aplikace na základě open source řešení s využitím Node.js. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

30 – 40 stran textu.

Klíčová slova

javascript, Node.js, webová aplikace, vizualizace dat (JSON), meteorologie

---

Doporučené zdroje informací

Cantelon, M, Harter, M., Holowaychuk, T., RajlichForeword, N. Node.js in Action. New York City: Manning Publications, 2013. ISBN 9781617290572

HOLZNER, S. *JavaScript : profesionálně : [kompletní referenční příručka].* Praha: Mobil Media, 2003. ISBN 80-86593-40-1.

KEOGH, J. Java bez předchozích znalostí. Brno: Computer Press, 2005. ISBN 80-251-0839-2.

ODELL, D. Javascript průvodce programováním ajaxových aplikací. Brno: Computer Press, 2010. ISBN 978-80-251-2733-9.

PEHLIVANIAN, A., NGUYEN D., Javascript okamžitě. Brno: Computer Press, 2014. ISBN 978-80-251-4163-2.

ŘEZÁČOVÁ, D. *Fyzika oblaků a srážek.* Praha: Academia, 2007. ISBN 978-80-200-1505-1.

SUEHRING, S. *JavaScript : krok za krokem.* Brno: Computer Press, 2008. ISBN 978-80-251-2241-9.

TEIXEIRA, P. Professional Node.js: Building Javascript Based Scalable Software. United States: John Wiley & Sons Inc., 2012. ISBN 978-1-118-18546-9.

THAU. Velký průvodce JavaScriptem: tvorba interaktivních webových stránek v praxi. Praha: Grada, 2009. ISBN 978-80-247-2211-5.

ZAKAS, Nicholas: JavaScript pro webové vývojáře. Computer Press, 2009. EAN 9788025125090.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 05. 03. 2017

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Tvorba webové aplikace v Node.js" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 05. 03. 2017 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Pavlu Šimkovi, Ph.D. za cenné rady, konzultace a vedení bakalářské práce.

Dále děkuji PhDr. Viktoru Fuglíkovi, Ph.D. i celému kolektivu pracovníků projektu MEDARD z Ústavu informatiky AV ČR, v.v.i. v Praze za pomoc při návrhu tématu, podporu a cenné konzultace, které mi velmi pomohly při realizaci této práce.

Tvorba webové aplikace v Node.js

Souhrn

Bakalářská práce se zabývá problematikou vývoje webové aplikace v Node.js. Webové řešení znázorňuje vizualizace nejrůznějších meteorologických dat a vykresluje je na mapy planety Země. Uživatel si může zobrazit jakákoliv dostupná data o světovém počasí, nebo může vložit vlastní soubor s meteorologickými daty. Pro vývoj je použit open-source projekt, který je dle požadavků přizpůsoben a dále rozšířen. Základ vlastní aplikace tvoří jádro naprogramované v Node.js. Node.js je platforma využívající principy jazyka JavaScript. Stejný programovací jazyk je použit i pro tvorbu dynamického prostředí na straně klienta.

Klíčová slova: Node.js, JavaScript, vizualizace, JSON, webová aplikace, meteorologie

Web application development in Node.js

Summary

Thesis deals with development of web application in Node.js. Web solution represents visualization of many various meteorological data and renders it into map of planet Earth. User can view any available data about world weather or he can upload own file with meteorological data. For development is used open-source project which is customized and expanded by requirements. Base of application is programmed in Node.js platform. The Node.js is a platform that use the principles of JavaScript. The same programming language is used for creating dynamic client-side environment.

Keywords: Node.js, JavaScript, visualization, JSON, web application, meteorology

Obsah

1 Úvod 11

2 Cíl práce a metodika 12

2.1 Cíl práce 12

2.2 Metodika 12

3 Teoretická východiska 13

3.1 HTML5 14

3.1.1 Základní informace 14

3.1.2 Historie 14

3.1.3 Využití HTML5 15

3.2 CSS 16

3.2.1 Základní informace o CSS 16

3.2.2 Využitelnost CSS 17

3.2.3 Historie CSS a počátky CSS3 17

3.3 JavaScript 19

3.3.1 Technologie na straně klienta 19

3.3.2 Základní informace a historický vývoj 20

3.3.3 Postupný vzestup JavaScriptu 21

3.3.4 (NE)Jednoduchý programovací jazyk 21

3.3.5 Praktické využití 22

3.3.6 Objektově orientovaný JavaScript 23

3.3.7 Užitečné spouštění JavaScriptu 23

3.3.8 Moderní JavaScript a současnost 23

3.4 Node.js 24

3.4.1 Použití technologie 24

3.4.2 Vznik nové technologie Node.js 24

3.4.3 Praktické využití 25

3.4.4 Jádro Node.js 25

3.4.5 Node.js není lék úplně na všechno 26

3.5 Data 27

3.5.1 Formát dat JSON 27

3.6 Vizualizace dat 28

3.6.1 Knihovna D3.js 28

3.6.2 Grafické znázornění 28

3.7 Charakteristika významných použitých technologií 29

3.7.1 JQUERY 29

3.7.2 XPath 29

3.7.3 AJAX 30
3.7.3.1 Použití Ajaxu 30
3.7.4 Http metody 31
3.7.4.1 Posílání požadavku pomocí metody GET 31
3.7.4.2 Posílání požadavku pomocí metody POST 31
3.8 Geolokace 31
3.8.1 Vyhledávání pozice 31
4 Vlastní práce 33
4.1 Definice zadání 33
4.2 Analýza dostupných řešení 34
4.2.1 Kritéria hodnocení pro analýzu dostupných řešení 34
4.2.2 Seznam aplikací, které byly podrobeny vícekriteriální analýze variant 35
4.2.3 Provedení komparace 35
4.2.4 Výsledky a ukazatele použité analýzy 37
4.3 Vývoj vlastní aplikace 38
4.3.1 Stavba aplikace založená na open-source 38
4.3.2 Instalace ekosystému Node.js 39
5 Zhodnocení výsledků 43
5.1 Charakteristika dílčích částí aplikace 43
5.1.1 Moderní vizualizace 43
5.1.1.1 Použití klíčové knihovny pro zobrazení 43
5.1.1.2 Projekce 43
5.1.1.3 Interaktivní zobrazení 44
5.1.2 Nahrání souboru 45
5.1.3 Aktualizace dat 46
5.1.4 vstupní formáty GRIB a NetCDF 47
5.1.5 Zvolení časového intervalu 47
5.1.6 Online dostupnost 47
5.1.7 Počet prohlížených veličin 48
5.1.8 Pokročilé znalosti při ovládání aplikace 48
5.1.9 Lokalizace 49
5.1.10 Ovládací prvky 49
5.2 Závěr z hodnocení výsledků 50
6 Závěr 51
7 Seznam použitých zdrojů 52
7.1 Seznam literatury 52
7.2 Seznam obrázků 53

7.3 Seznam tabulek 53

8 Přílohy 54

1 Úvod

Bakalářská práce se zabývá vývojem webové aplikace s použitím Node.js. Node.js je označení pro platformu, která byla vytvořena vývojářem Ryanem Dahlem. V současné době je velice využívaná v oblasti webových technologií. Nejčastěji se používá pro vývoj škálovatelných aplikací. V porovnání s podobnými platformami vystupuje nad ostatní díky rychlému nasazení a snadnému rozšíření.

Podrobnější průzkum a využití technologie Node.js je jednou ze dvou částí, které se opírají o základ jazyka JavaScript. Současně s platformou využitou na realizaci jádra systému jsou prozkoumány i techniky vývoje, které se zabývají problematikou na klientské straně. V této části se práce podrobněji věnuje dynamickým technikám vývoje, kterých lze dosáhnout vhodným použitím JavaScriptu. Samotný jazyk je velmi rozmanitý a jeho využití je široké, jak se i shoduje většina autorů, kteří se o tuto oblast zajímají. Cílem zkoumání jsou i jazyky HTML a CSS, které s vývojem úzce souvisí. Mimo základních informací ohledně těchto jazyků dominujících v oblasti Internetu, je kladem důraz především na charakteristiku funkcí, které jsou součástí nejnovějších verzí.

Práce pojednává také o použití open-source řešení. Stručně vysvětluje výhody a nevýhody použití řešení, které je dílem jiného programátora, ale autor se uvolil dát své know-how veřejně k dispozici. Okrajově práce postihuje i oblasti, které řeší problematiku týkající se práce s datovými soubory. Více se zabývá samotným formátem používaným pro přenos dat typu JSON. V neposlední řadě nechybí ani charakteristika moderní vizualizace v oblasti webových aplikací. V této oblasti stojí za povšimnutí šikovná javascriptová knihovna s názvem D3.js. Její použití zaručuje perfektní vizuální efekt a dynamičnost.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje a tvorby webové aplikace v Node.js. Hlavním cílem práce je vytvoření reálné aplikace pro prezentaci hydrometeorologických dat s využitím open-source řešení. Dílčími cíli práce jsou: Vypracování přehledu vývoje dynamických technologií na straně webového klienta. Komparace vytvořené aplikace s podobnými aplikacemi.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Zpracování literární rešerše zajišťuje prohloubení teoretických znalostí. Názory různých autorů jsou konfrontovány a získané postřehy jsou použity v oblastech vývoje aplikace na straně klienta i serveru.

Práce zahrnuje analýzu dostupných řešení v oblasti aplikací prezentujících počasí. Metoda pro komparaci spočívá ve využití vícekriteriální analýzy variant. Na porovnání jednotlivých řešení je aplikována metoda hodnocení, která se nazývá bodovací. Tato metoda hodnotí všechny analyzované prvky. Hledá nejlepší variantu za použití bodovací škály v rozsahu 1–10 bodů. Cílem analýzy je prověřit, zda již existuje řešení, které je vyhovující zadaným požadavkům na funkcionalitu aplikace.

V rámci práce probíhá průzkum dostupných technologií a ověřených praktik, který se snaží ulehčit samotný vývoj. Praktická část práce spočívá v tvorbě reálné aplikace na základě open-source řešení s využitím Node.js.

Na základě syntézy teoretických poznatků, přehledů a znalostí získaných při zpracování teoretické části je možné aplikovat teorii na praktický vývoj reálné aplikace. Z výsledků a zhodnocení vlastní práce jsou formulovány závěry bakalářské práce.

3 Teoretická východiska

Teoretická část bakalářské práce je věnována teorii jazyka JavaScript na straně klienta i serveru a zkoumá názory a závěry různých autorů o daném tématu. Značná část je věnována javascriptové serverové platformě Node.js. Analytická část práce vysvětluje a do značné míry objasňuje veškeré souvislosti použitých technologií. V aplikační části je teorie aplikována na vývoj reálné aplikace.

Následující práce zkoumá technologie, které jsou nezbytnou součástí při vývoji webových stránek a aplikací. Téměř každá webová stránka je vytvořena za pomoci jazyka HTML, který je základním stavebním kamenem pro každého vývojáře ve webovém prostředí. Vedle značkovacího jazyka HTML stojí za povšimnutí CSS. Kaskádové styly, jak je CSS překládáno do českého jazyka, slouží především pro stylování a grafickou úpravu stránek. Třetím důležitým jazykem při programování webu je JavaScript, který je pro své obrovské využití právem označován za hlavní jazyk v oblasti vývoje webů.

Podrobněji jsou HTML, CSS, Javascript a s nimi související technologie popsány v následujících kapitolách. Jelikož se jedná o velice rozsáhlé téma, jsou vybrány pouze technologie a platformy, které přímo souvisejí s vývojem reálné aplikace.

Obrázek 1- znázornění dominantních technologií používaných v oblasti webových aplikací

Zdroj: https://www.planet-source-code.com/vb/default.asp?lngWId=14

3.1 HTML5

3.1.1 Základní informace

HTML (Hyper Text Markup Language; hypertextový značkovací jazyk) je hlavním jazykem na webu. Kdykoliv si uživatel prohlíží nějakou webovou stránku ve svém prohlížeči, je skoro jisté, že se jedná o dokument jazyka HTML. Tento značkovací jazyk původně vznikl pro formátování a sdílení vědeckých materiálů. Reprezentoval jednotný způsob definice struktury hypertextových dokumentů. Slovo hypertext označuje nějaký kus textu, který odkazuje na jiné texty. Takový text je nelineární a umožňuje logicky odkazovat v rozsáhlých dokumentech. Tím, že je popsána struktura dokumentu, dojde k oddělení obsahu od vzhledu, tzn. od toho, jak jsou texty prezentovány koncovému uživateli. Díky tomu lze dokumenty snadněji sdílet, upravovat a šířit.

V současné době ho používají téměř všechny typy dokumentů v oblasti Internetu. Rovněž slouží k tvorbě vizuálně zajímavých rozhraní pro webové aplikace. Se současnou funkcionalitou lze označit HTML5 spíše za aplikační platformu než za prostý značkovací jazyk (Brown a kol., 2014).

3.1.2 Historie

Kořeny jazyka HTML sahají hluboko do minulosti. Samotný návrh jazyka byl poprvé zveřejněn už v roce 1993. Z návrhu se stala funkční záležitost a postupně byly odtajněny verze 2.0, 3.2. Velký úspěch znamenal rok 1999, kdy vznikla verze HTML 4.0 a pak konečně 4.0.1 (Lubbers a kol., 2011).

Jazyk se stále postupně vylepšoval. Podporoval nové prvky pro tvorbu tabulek a formulářů. Jednotlivé prvky byly standardizovány a vzhled dokumentu byl ovlivněn připojenými styly. Verze 4.01 měla ukončit vývoj HTML a tvůrci webových stránek měli přejít na XHTML – což měl být následník HTML, který využívá univerzální jazyk pro všechny platformy zvaný XML[1]. Jazyk HTML byl v této době označován za slepou uličku vývoje. S tím se ovšem nechtěli smířit všichni, a tak vznikla skupina vývojářů, kteří dál pracovali na rozvoji HTML. Cílem bylo, aby verze HTML5 byla schválena mezinárodním konsorciem W3C[2]. Snažení

[1] XML – rozšiřitelný značkovací jazyk používaný pro serializaci dat.
[2] W3C - The World Wide Web Consortium.

všech lidí, kteří se podíleli na vývoji, nakonec vyústilo v úspěch a po bezmála patnáctileté odmlce vyšla oficiální verze HTML 5.0 (Castro a kol., 2012).

3.1.3 Využití HTML5

S příchodem nové verze HTML5 se z běžného značkovacího jazyka stal nástroj s plnohodnotným rozhraním API (aplikační programové rozhraní) určeným pro vývoj webových aplikací.

Nová verze vylepšuje elementy archaických verzí tohoto jazyka. Pátá verze HTML umožňuje práci se vstupními poli, proto mohou vznikat velmi jednoduše kompletní formuláře, aniž by bylo nutné použít např. JavaSript. Přibyly další typy polí jako například url nebo email, které podporují validaci na straně klienta. Práce s elementy audio a video umožní vkládat různé zvukové soubory a videa do dokumentů. S aplikačním rozhraním lze snadno vytvářet multimediální přehrávače bez použití jakýchkoliv zásuvných modulů ve webovém prohlížeči. Element canvas[3] a podpora formátu SVG (Scalable Vector Graphics; škálovatelná vektorová grafika) dovolí v HTML5 dokonce kreslit. API elementu canvas je určeno pro kreslení bitmapové grafiky a je určené pro tvorbu grafů, 2D a 3D obrázků a například i her. SVG formát naproti tomu podporuje vektorovou grafiku, pomocí které je možné opětovné použití, změna velikosti a úpravy obrázků. Tímto je možné docílit toho, aby se obrázky zobrazovaly korektně napříč širokým spektrem různých zařízení.

Nejvýznamnější změna ve specifikaci HTML5 je doplnění o rozhraní API, která jsou součástí objektového modelu HTML. Níže je vypsáno několik nejdůležitějších a nejzajímavějších:

- Sdílení, nebo odebírání dat.
- Napodobení procesů s více vlákny a úlohy na pozadí.
- Určení pozice dle geolokace.
- Odesílání dat mezi dokumenty a doménami bez zveřejnění modelu DOM.
- Rozhraní WebSocket pro komunikaci klient-server téměř v reálném čase.
- SVG grafika.
- Webové úložiště.

[3] Canvas – plátno, které je součástí HTML5 a umožňuje dynamické vykreslování 2d tvarů a bitmapových obrázků.

- Webové SQL[4] databáze.
- Canvas pro 2D a 3D grafiku.

V seznamu se nachází mnoho skvělých a užitečných funkcí, které napomáhají při tvorbě moderních webových stránek a aplikací (Brown a kol., 2014).

Z hlediska obecného přístupu se HTML5 snaží o následující principy:

- Přístupnost – podpora uživatelů s nějakým postižením. Všechny elementy jsou dostupné i pro handicapované uživatele, kteří mohou využít čtečku obrazovek.
- Nezávislost – tímto se rozumí nezávislost na médiu. HTML5 by mělo být podporováno napříč všemi platformami a zařízeními.
- Podpora všech jazyků – podpora anotace Ruby[5], má za následek zpřístupnění všech světových jazyků včetně jazyků používaných ve východní Asii. (Lubbers a kol., 2011).

3.2 CSS

3.2.1 Základní informace o CSS

Zatímco prostřednictvím jazyka HTML je možné definovat význam obsahu a udělovat webovým stránkách základní strukturu, jazyk CSS specifikuje vzhled. Šablona stylů je standartní textový zápis, který obsahuje jedno nebo více pravidel, která pomocí svých vlastností a hodnot určují, jak se mají jednotlivé elementy zobrazovat. Proto, aby byla využita maximální síla jazyka CSS je nutné stránky při vývoji označovat dobře a konzistentně na základě doporučení jazyka HTML (Castro a kol., 2012).

CSS je zkratka pro Cascading Style Sheets (kaskádové styly) a představuje komplexní kolekci metod pro grafickou úpravu webových stránek. Jde v podstatě o samostatný jazyk, který byl vyvinut jako doplněk k jazyku HTML. CSS umožňuje aplikovat styly na obsah webových stránek. Hlavní smysl vzniku a vývoje kaskádových stylů je umožnit vývojářům webových stránek a aplikací oddělení vzhledu dokumentu od jeho obsahu (Lazaris, 2014).

[4] SQL – dotazovací standardizovaný jazyk používaný v relačních databázích pro práci s daty.
[5] Ruby – skriptovací interpretovaný programovací jazyk kompletně objektově orientovaný.

CSS je možné k HTML připojit jako externí soubor, nebo lze zakomponovat přímo do kódu jazyka HTML. Při stavbě pravidla stylu se vždy vychází z definované struktury. Selektor – určuje na jaký element se následující pravidlo bude vztahovat. Po selektoru přichází deklarační blok, pro který platí, že jeho skladba je neměnná: vlastnost – hodnota. Každá dvojice tvoří deklaraci, jenž určí, co se s označeným obsahem stane.

3.2.2 Využitelnost CSS

V současnosti je k dispozici CCS3, které je plně dostačující pro kompletní tvorbu stylů. V dnešní době se proto již nedoporučuje definovat styly přímo v zápisu HTML. Nesprávná zastaralá praktika nedovoluje využít potenciál, který CSS vývojáři přináší. Hlavní význam použití kaskádových stylů spočívá v tom, že všechny úpravy fungují automaticky na celý dokument a vzhled celého webu lze deklarovat i pouze jedním správně nadefinovaným souborem (Lazaris, 2014).

Možnost využití CSS je velice široké, níže jsou uvedeny některé z možností použití:

- Nastavení libovolného stylu a velikosti písma.
- Automatické formátování stylů a velikosti nadpisů.
- Definice odsazení odstavců, řádkování, odrážek.
- Nastavení pozadí u skupin, nebo jednotlivých elementů (stránky, tabulky, odstavce).
- Zvýraznění odkazů při událostech myši.
- Zneviditelnění, zprůhlednění určitých elementů.
- Nastavení pozic jakýmkoliv objektům.
- Přidání rolovacích lišt, rámečků, okrajů.

3.2.3 Historie CSS a počátky CSS3

Verze jazyka, která předcházela CSS3 byla specifikace CSS2.1. Ta byla revizí verze CSS2, která vznikla roku 1997. Navzdory neustálému vývoji je velice překvapivé, že verze CSS2 se stala oficiálním doporučením konsorcia W3C až v roce 2011. Za zmínku stojí skutečnost, že internetový prohlížeč Internet Explorer ve verzi 8, vydaný v roce 2009, prohlásil, že je jediný prohlížeč s úplnou podporou specifikace CSS2.1, i když oficiálně schválena tato verze kaskádových stylů ještě nebyla. Práce na vývoji nové verze CSS3 sice začaly již někdy kolem roku 1998, ale W3C raději zastavilo vývoj nové větve a soustředilo se na dokončení verze 2.1 (Gaston, 2016).

Není tedy překvapením, že podobně jako tomu bylo u vývoje HTML, následovala dlouhá přestávka, než se CSS ve verzi 3 dostalo do použitelné podoby tak, aby ho podporovala většina webových prohlížečů.

Především webový klient Internet Explorer nevykazoval žádné známky toho, že by se chystal připravit se na podporu CSS3. V průběhu posledního desetiletí však došlo k vytvoření mnoha nových prohlížečů, což mezi nimi rozpoutalo velký konkurenční boj. Předháněly se v tom, kdo nejrychleji zavede nové funkce do své podpory. Hlavní výhoda tohoto měření sil bylo zavedení specifikace CSS3. Skupina výrobců webových prohlížečů chtěla vývojářům a uživatelům dopřát nejnovější funkce a nové webové technologie. Specifikace CSS3 byla z velké části vytvořena, a tak implementace do prohlížečů byla poměrně snadná.

V dnešní době probíhá stále aktivní vývoj CSS3. Drtivá většina prohlížečů tuto specifikaci zahrnuje do svých funkcí a komunita vývojářů na ní staví své webové stránky a aplikace (Gaston, 2016).

Úprava a „zkrášlování“ stránek je stále na lepší úrovni, pomocí CSS3 a různých frameworků[6] je možné docílit skutečně velmi nápaditých výsledků. Navzdory tomuto vývoji však přichází i nutnost počítat s celou řadou zařízení s různými velikostmi displejů. Kancelářské monitory, notebooky, tablety, telefony, ale i herní konzole, nebo chytré hodinky. Různé typy zařízení mají dnes již běžně připojení k internetu a jejich uživatelé si v nich chtějí prohlížet obsah Internetu. V dnešní době je tedy nutné poskytovat různé styly pro rozdílné typy zařízení.

V minulosti se weboví vývojáři potýkali s rozdíly mezi prohlížeči. Problém způsobený nekompatibilitou mezi prohlížeči se do jisté míry podařilo eliminovat, nyní je ovšem třeba věnovat velkou pozornost různorodým koncovým zařízením.

[6] Framework – aplikační rámec obsahující různá API, podprogramy, či vývojové postupy a doporučení.

Obrázek 2 - ukázka rozmanitých typů zařízení s různými typy displejů

Zdroj: http://youdedicated.com/Responsive.aspx

3.3 JavaScript

Skriptovací jazyk, který využívá výhod objektového modelování. Využívá objektů prohlížeče a zabudovaných objektů. Právem je označován za jazyk webu. Klientský JavaScript je závislý na prohlížeči a je vykonáván na straně klienta, tedy prohlížeče. Je multiplatformní, interpretovaný, což znamená, že nemusí být kompilován. JavaScript vychází z rodiny jazyků C, proto lze pozorovat podobnou syntaxi jako například u jazyka C, C++ nebo Java. JavaScript je rovněž „case – sensitive“, tzn. je nutné rozlišovat velikost textu.

3.3.1 Technologie na straně klienta

JavaScript je velmi mocný skriptovací jazyk, který je vložen do webových stránek a jeho úkolem je jejich oživení. Jestliže má web kromě zobrazení prostého textu nějakou funkcionalitu je velmi pravděpodobné, že byl při vývoji použit právě JavaScript. Téměř každá webová stránka nebo aplikace, se kterou je možné se ve světě Internetu setkat, obsahuje JavaScript. Jeho oblíbenost neustále roste, protože se s jeho pomocí dají řešit komplexní situace, které při vývoji webových řešení vznikají.

JavaScript podporuje nepřeberné množství možností, jak zapracovat užitečné funkce do webové stránky nebo aplikace. Detailní představení funkcionality by znamenalo velmi rozsáhlý seznam. Dále je uvedeno několik základních úloh, které lze pomocí JavaScriptu řešit:

- Tvorba dynamického HTML zahrnuje zpracování ovládacích prvků HTML jazyka.
- Otevírání nových oken webového prohlížeče.
- Reakce na události myši a klávesnice, na které je navázaná celá řada funkcí.
- Tvorba interaktivních galerií, formulářů, tabulek apod.

- Používání šablon stylů CSS.
- Nastavení souborů cookies.
- Práce s XML a XSLT soubory a napojením na jiné platformy např. .NET.

3.3.2 Základní informace a historický vývoj

JavaScript je označení pro jedinečný jazyk. Jeho název však může být poněkud matoucí, jelikož JavaScript přímo nesouvisí s programovacích jazykem Java, jak by se mohlo na první pohled zdát. Označení zdánlivě související s Javou však není pouhou náhodou. Během roku 1995, kdy jazyk vznikal, byla Java velmi dominantní, a tak spojení Javy s JavaScriptem nebyl dle jeho autorů špatný nápad.

Následují vývoj poslal jazyky rozdílným směrem. Java se stala celkem složitým a spletitým jazykem, naopak JavaScript v průběhu doby zůstával relativně stejný. Jeho hlavní výhody, kterými jsou především rychlost a modularita, umocňovala přízeň programátorů a v dnešní době je tento jazyk nejpoužívanějším a nejoblíbenějším jazykem ve světě internetových stránek a webových aplikací. Oproti Javě má výhodu ve větší jednoduchosti a v rychlejším nasazení. JavaScript je jazyk skriptovací a před použitím se nemusí kompilovat do bitového kódu, jak je tomu právě u zmiňované Javy, nebo jiných kompilovaných jazyků (Holzner, 2003).

JavaScript byl poprvé představen v roce 1995 na tiskové konferenci společnosti Sun Microsystems, Inc. Účel, který měl původně splňovat, bylo doplnění jazyka Java a HTML v oblasti Internetu. Jeho vývoj ovšem nabral obrátek, unikl ze stínu Javy a postupně ji začal předhánět po všech směrech. U programátorů se stal velice populárním, i přesto, že existovaly problémy s ošetřením chyb a s bezpečností. Jazyk dovoloval dosud nevídané věci, a tak se jeho obliba neustále zvyšovala. Jedno z prvotních využití tohoto skriptovacího jazyka představovala dynamická změna obrázků po najetí kurzoru myši. Dnes zcela běžná a banální funkce byla v době minulé považována za něco neuvěřitelného (Thau, 2009).

Stále větší pokrok pro JavaScript znamenal i neustálý vývoj webových prohlížečů, které podporovaly rychlejší načítání skriptů, a umožňovaly stále více vylepšovat webové stránky. Webové prohlížeče se začaly postupně transformovat a byly přívětivějšími jak pro uživatele, tak pro programátory. Společnosti, které se soustředily na vývoj klientů, byly postupně nuceni

ke sjednocení používané webové technologie. Jak je ale popsáno v následující kapitole, nebylo tomu tak vždy.

3.3.3 Postupný vzestup JavaScriptu

Úspěch společnosti Netscape s vytvořením prvního skriptovacího jazyka logicky upoutával čím dál tím větší pozornost. Společnost Microsoft nechtěla zůstat pozadu a vytvořila jazyk podobný JavaScriptu s názvem Jscript. Jak představuje Holzner v kapitole „Úvod do JavaScriptu“ první verze byla použita ve webovém prohlížeči Internet Exploer 3.0. Touto cestou tedy vznikla druhá forma skriptovacího jazyka. Přinesla velký konkurenční boj mezi oběma společnostmi, ale negativně to odnesli především všichni vývojáři.

V důsledku těchto událostí se Společnosti Sun a Netscape rozhodly vytvořit standardizovaný jazyk ECMAScript, který měl vnést ucelená pravidla pro používání skriptů v rámci vývoje webových stránek. Nový standardizovaný skriptovací jazyk byl dlouhou dobu vyvíjen, a ještě déle schvalován. Po jeho dokončení tedy existovali 3 podobné skriptovací jazyky vedle sebe: JavaScript, Jscript a ECMAScript. Problém spočíval v tom, že skriptovací jazyky používaly odlišné metody v práci s objekty, a tak nebylo možné vytvořit projekt, který by uměl pracovat se všemi webovými prohlížeči stejně. Webové prohlížeče podporovaly jeden nebo druhý formát JavaScriptu, a tak nebylo nic výjimečného, že se pro zcela totožné webové stránky vytvářely dvě a více verzí, aby byla zajištěna kompatibilita a vývojáři uspokojili všechny uživatele (Suehring, 2008).

Praktika vývoje pro různé prohlížeče byla náročná. Vzniklou situaci se snažilo urovnat až W3C (World Wide Web Consortium), které vneslo do objektů prohlížeče určitou shodu. Vznikl DOM – Document objekt model, jak blíže specifikuje ve své knize autor Zakas (2009), a tento standart upravil pravidla využití skriptovacích technologií. Navzdory všem těmto úskalím je JavaScript velmi oblíbený a je to nejpoužívanější jazyk v oblasti vývoje webových stránek a aplikací.

3.3.4 (NE)Jednoduchý programovací jazyk

V mnohé literatuře je JavaScript označován za skriptovací jazyk namísto programovacího. Může to vést k domněnce, že skriptovací jazyky jsou pro méně zkušené a méně zběhlé programátory jednodušší než jazyky programovací. Na první pohled je JavaScript opravdu

poměrně jednoduchý. Někdy bývá také přirovnáván k jazyku BASIC[7]. Toto tvrzení potvrzuje fakt, že některé prvky činí jazyk použitelnějším i pro nezkušené vývojáře. Jazyk je tolerantnější, ale to platí pouze v případě, že je používán jen pro omezené programovací úkoly s jasným předem určeným návodem. David Flanagan ve svém rozsáhlém průvodci přirovnává použití JavaScriptu pro stále se opakující úlohy k použití kuchařské příručky při vaření.

Ve skutečnosti se pod tímto zdánlivě jednoduchým zevnějškem skrývá velice propracovaný a komplexní jazyk. Je stejně složitý jako jiné programovací jazyky, možná i složitější. Především záleží na způsobu použití a na obtížnosti úkolu, který se JavaScriptem má řešit. Programátoři, používající jazyk pro složité úlohy, často narazí na mnoho překážek, které jsou způsobeny jeho špatnou znalostí. Obrat nastává, pokud se JavaScript dostane do rukou zkušeného profesionála, potom se jedná o mocný nástroj, který je značně složitý (Flanagan, 2002).

3.3.5 Praktické využití

V minulosti vznikaly a vznikají statické stránky, které uživateli sdělí potřebné informace, ale nijak ho neohromí. JavaScript umožňuje vytvářet dynamické webové stránky, umožňuje ovládat všechny prvky dokumentu a dokonce i samotnou webovou stránku skripty, které běží přímo v prohlížeči (klientovi).

Jako dynamické HTML (DHTML) je souhrnně označován celý soubor postupů, který umožňuje libovolné umisťování prvků, zpracování obrázků, nebo používání speciálních efektů pro přetváření webových stránek za běhu. Jazyk kombinuje možnost zpracování skriptů interpretem JavaScriptu s objektovým modelem dokumentů (DOM) definovaným webovým prohlížečem. Tyto dvě zcela odlišné technologie se vzájemně doplňují a umožňují distribuci spustitelného obsahu přes web.

Ověřování vstupních dat a další funkce, bez kterých se dnes obejde málokterý web, spočívá v ověřování zadávaných informací například prostřednictvím formulářů. Někdy je potřeba informace automaticky zasílat serveru, což znamená velké časové zdržení. V tomto případě může JS zajistit ověření ještě před odesláním dat na server. Uživatel zadá neplatný formát

[7] Basic – souhrn programovacích jazyků vytvořených pro výuku programování.

emailové adresy a JS ho zkontroluje. Pakliže jsou zadané údaje v pořádku, pak teprve odešle data k dalšímu zpracování.

Některé výpočtové aplikace, jako například kalkulačka bankovních produktů nebo kalkulačka čisté mzdy, fungují pouze na straně klienta. Zajistí rychlou odezvu a pro programátora znamenají i jednoduší práci při tvorbě kódu. Dnes se rychlost vyřízení požadavku odesílaného na server zvýšila. V minulosti se ale právě kódem s JavaScriptem nahrazovalo řešení, který využívají například programovací jazyky: Perl, ASP, nebo C++ pro zpracování dat z klienta na serveru.

Zpracování cookies, což jsou speciální malé soubory, které ukládají informace o zařízení uživatele, který si webové stránky prohlíží, je díky JavaScriptu možné vytvářet, aniž by bylo třeba vyvíjet aplikace běžící na webovém serveru (Holzner, 2003).

3.3.6 Objektově orientovaný JavaScript

JavaScript byl od počátku vytvořen jako objektově orientovaný jazyk. Postupný vývoj znamenal, že programátoři jiných programovacích jazyků (Python, Perl, Ruby, …) vnášeli do jazyka své programátorské názory. Objektově orientovaný kód JavaScriptu vypadá a chová se jinak než v případě ostatních objektových jazyků. Objekty v JS jsou základním stavebním kamenem a prakticky na všechny prvky lze pohlížet jako na objekty a také toho náležitě využít. Na objekty lze použít mnoho různých funkcí a vlastností, které dávají programátorům možnost ovládat všechny prvky prohlížeče i vstupy od uživatele. Objekty JS lze definovat jako skupinu vlastností. Vlastnosti představují atributy a metody (Resig, 2007).

3.3.7 Užitečné spouštění JavaScriptu

JS se spouští ve třech hlavních časových okamžicích. Skript se provede při načtení dokumentu, po jeho načtení, nebo když uživatel provede nějakou akci. Kód JS se nemusí spustit hned, ale je možné počkat až na stažení celého HTML dokumentu. Je to velmi užitečné, protože HTML se nahrává po částech a tímto způsobem umožňuje programátorovi plnou kontrolu nad aplikací (Flanagan, 2002).

3.3.8 Moderní JavaScript a současnost

Jazyk JavaScript se vyvíjel velice dlouho, ale trvale docházelo k jeho vylepšování. Na samém prvopočátku to byl jednoduchý jazyk s velkým potenciálem. Nyní je již velice respektovaným

programovacím jazykem, který je využíván společnostmi a programátory po celém světě k tvorbě neuvěřitelně funkčním aplikacím. Jazyk je považován za solidní, robustní a velice přizpůsobivý nástroj. Za jeho dnešní moderní podobou stojí tisíce programátorů, kteří vytříbili jazyk tak, jak ho známe dnes (Pehlivanian, 2014).

3.4 Node.js

3.4.1 Použití technologie

Navzdory prvotnímu použití JavaScriptu je možné primárně „client – side“ (na straně klienta) jazyk použít rovněž na straně serveru. Běžnými jazyky pro zpracování skriptů na serveru jsou například PHP, Perl, Python nebo ASP. Tyto programovací jazyky provádí chod programu na serveru a do klienta posílají již zpracované informace a data. Stejným způsobem pracuje i serverová verze upraveného JavaScriptu. Oblíbenými a využívanými jsou technologie vytvořené ze základů open – source: Rhino, Apache a Node.js.

3.4.2 Vznik nové technologie Node.js

JS na straně serveru s názvem Node.js byl poprvé představen na evropské konferenci JSConf v roce 2009. Mladý programátor Ryan Dahl představil projekt, na kterém stále pracuje a dále ho vyvíjí. Platforma Node.js v kombinaci s V8 Google JS využívá model událostí a asynchronní I/O operace. Konstrukce je tímto způsobem vytvořena z důvodu maximalizace výkonu a snížení režie procesoru.

Pro nově vznikající platformu byla využita jednoduchost JavaScriptu a zpracování úloh, které probíhá na straně serveru. Projekt získal nevídaně pozitivní ohlasy a od té doby se stal velice žádaným a populárním. Programátory je využíván pro tvorbu škálovatelných webových aplikací.

Celkem rychlého rozvoje se Node.js dočkal kvůli tomu, že JavaScript používalo a používá obrovské množství programátorů. Pro většinu webových vývojářů je JavaScript natolik používaný na straně klienta, že přechod a psaní aplikací na straně serveru jim přišlo velice přirozené. API’s (neboli rozhraní pro programování aplikací) jsou velice elegantní a jednoduché pro použití. Node.js je ovšem velice atraktivní i díky svým modulům. Dostupné moduly třetích stran je velice jednoduché implementovat do vlastního projektu. Modulů je

celá řada a dnes pokrývají většinu potřebné funkcionality. Programátor si jednoduše vybere knihovnu modulů, kterou chce do svého projektu zahrnout a snadno ji stáhne a nainstaluje. Node.js umožňuje rychle a jednoduše vytvářet škálovatelné síťové služby (Teixeira, 2012).

3.4.3 Praktické využití

Hlavní uplatnění této technologie spočívá v tvorbě škálovatelných síťových aplikací. Aplikace mohou běžet na různých zařízeních. Vhodné uplatnění je možné najít v oblasti aplikací běžících v reálném čase, které mohou být datově náročné. Využití Node.js má obrovskou výhodu, protože není příliš náročné na využití CPU[8]. Použití Node.js může značně zvýšit výkon navrženého řešení, a to s minimálními požadavky na hardware serveru. Jinými slovy pro obsluhu více uživatelů postačí méně hardwaru.

Další významnou součástí Node.js je relativně snadné a funkční napojení na databáze. U většiny webových aplikací se neobejdeme bez databáze. Rozhraní pro napojení na databázi je k dispozici pro všechny hlavní relační databáze (MySQL, MariaDB, PostgreSQL, Oracle, SQL Server). Další možností je použití tzv. nerelační databáze (NoSQL), které nabízejí koncepčně jednodušší přístup k ukládání dat. Dnes již existuje velké množství takových databází. Mezi programátory je například velice oblíbená MongoDB (Brown, 2014).

3.4.4 Jádro Node.js

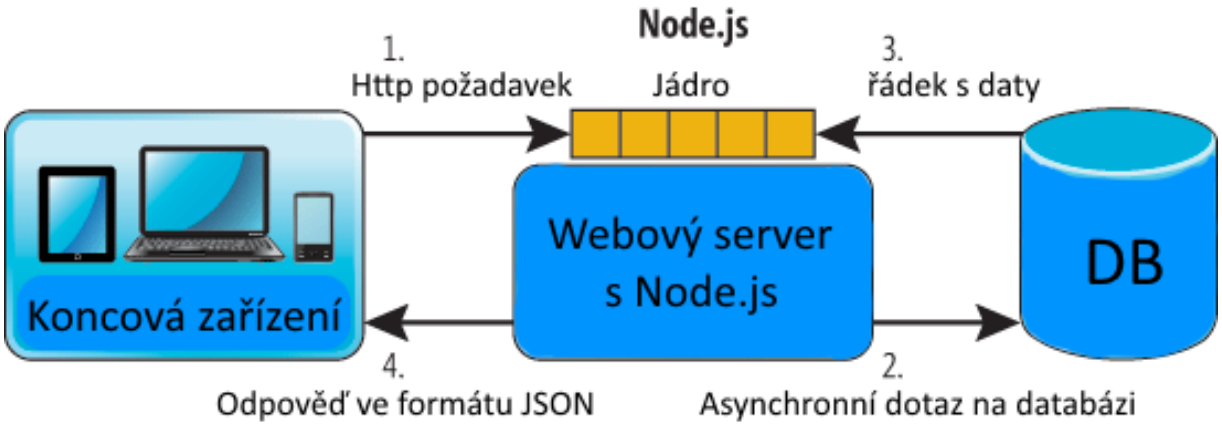
Node.js je platforma JavaScriptu na straně serveru. Je tvořena ze záměrně miniaturní klíčové knihovny a bohatého ekosystému. Je provozována na jádře V8 pro JavaScript, což ji dodává velkou rychlost. Jádro V8 vytvořily vývojáři společnosti Google. Samotná platforma Node.js je postavena na běhovém prostředí JavaScriptu z webového prohlížeče Chrome. Node.js používá neblokující vstupně-výstupní model řízený událostmi. Nad ním je tenká vrstva kódu v C++ poskytující minimální nutné zázemí (event-loop vyhodnocující příchozí události, obsluha I/O bufferů a jiné).

Jestliže autor John Resig ve své knize Moderní JavaScript označil JS jako mocný a moderní nástroj pro web, tak Node.js jeho tvrzení ještě umocňuje. Architektura platformy je postavená

[8] CPU – central processing unit (centrální procesorová jednotka).

na nejlepších technikách. Jádro ve spojení s motorem V8 Google JavaScript je extrémně rychlé. Ve skutečnosti je řešení postavené na Node.js několikanásobně rychlejší, než když je stejný kód napsaný například v jazyku Ruby, nebo Python. Oproti Python3 je přibližně 13x rychlejší. Náskok před rychlostí jazyka Ruby je přibližně osminásobný. Tyto fakta potvrzují neuvěřitelná pozitiva pro dynamický jazyk a jsou důležitá pro optimalizaci a kompilaci do strojového kódu (Ngyuyen, 2016).

Obrázek 3 - schéma znázorňující komunikaci mezi koncovými zařízeními, web serverem a databází



Zdroj: vlastní

3.4.5 Node.js není lék úplně na všechno

Node.js není vhodné použít úplně všude, existuje však určitá řada problémů, v nichž přímo vyniká. Programy lze rozdělit na dvě skupiny podle toho, jestli záleží více na výkonu procesoru nebo na vstupně-výstupních operacích. Problémy, které nastávají při zatěžování procesoru těží z vyššího počtu výpočetních cyklů dostupných pro provedení nějakého výpočtu. Přesně pro tyto účely, řešení problémů spojených s procesorem, není Node.js úplně vhodný. (Ngyuyen, 2016).

Zakladatel Node.js hledal řešení, které by zajistilo, že žádný požadavek nebude moci zatěžovat svým výpočtem déle než 5 milisekund před návratem do cyklu událostí. Pokud by se podmínka porušila, proces by byl ukončen a již by nebyl nikdy spuštěn. Například kdyby programátor měl kód v JavaScriptu, který by nejdříve počítal 10 000 prvočísel, zbytečně by blokoval výpočetní výkon a neměl by být vykonán (Wandschneider, 2013).

Problémy, které se týkají vstupně-výstupních operací těží z vyšší propustnosti vstupů a výstupů. Z této skupiny operací lze uvést například rychlejší diskové operace, paměťové

operace, efektivnější ukládání dat do mezipaměti a propustnosti sítě. V této kategorii můžeme nalézt skutečně mnoho možných problémů. Například problém, který je známý pod názvem C10K. Známý problém představuje to, jak nejlépe zpracovat požadavky na webový server, který v jedné chvíli musí zpracovávat 10 tisíc současných spojení. Právě v tomto příkladu nachází uplatnění fenomén Node.js, protože má neblokující asynchronní architekturu skvěle navrženou pro souběžnost a dokáže tak obsloužit obrovské počty požadavků (Ngyuyen, 2016).

3.5 Data

3.5.1 Formát dat JSON

JavaScript Object Notation. V českém překladu javascriptový objektový zápis. Vedle formátu XML se stal JSON velice oblíbeným. JSON představuje celosvětově využívaný standard pro výměnu dat v oblasti internetu a v implementaci webových API's. Je formátem pro zápis a výměnu dat nejen v JavaScriptu, ale je nezávislý na použité platformě. JSON, zjednodušeně řečeno, je způsob zápisu dat, který je nezávislý mezi platformami. Data v tomto formátu jsou organizována v polích, nebo jsou seskupena v objektech. Vstupem do datového formátu JSON je libovolný datový typ. Výstupním formátem je řetězec. JSON dovede porozumět poli hodnot, ať už je indexované, nebo neindexované, objektům a jednotlivým hodnotám. Datové typy, které můžeme do JSON uložit jsou následující: boolean (logickou hodnotu „true", nebo „false"), string (textový řetězec), číslo (celočíselné, nebo reálné), pole, objekt (Bassett, 2015).

JSON slouží k přenosu dat a je podporován napříč celým spektrem programovacích jazyků. Navíc je dobře čitelný i pro člověka. Data přenášená prostřednictvím této technologie jsou shlukována do souborů. Nedostatky JSONu spočívají v přenosu binárních dat a v tom, že nelze specifikovat použitou jazykovou sadu.

JSON vznikl převážně proto, aby zjednodušil přenos datových souborů, u kterých bylo použití XML zbytečně složité. JSON nekonkuruje formátu XML ve složitém přenosu souborů, jako jsou například celé dokumenty, ale v případě, kdy je nutné přenést strukturovaná data do jiné platformy, nebo webové aplikace je vhodné použít právě JSON (Smith, 2015).

3.6 Vizualizace dat

3.6.1 Knihovna D3.js

Moderní javascriptová knihovna je určena k vizualizaci nejrůznějších dat. Tvůrcem D3.js je vývojář Mike Bostock, jehož cílem bylo vyplnit mezeru mezi statickou a interaktivní animovanou vizualizací. Mike Bostock je zároveň tvůrcem knihovny Protovis, která je oficiálním předchůdcem D3.js. Po přechodu na nástroj D3.js byl její vývoj zastaven. Celý název této knihovny zní „Data – Driven Documents“. Data jsou poskytována uživatelem a slovem „documents“ je myšleno cokoliv, co může být vykresleno webovým prohlížečem (př. HTML, SVG). D3.js slouží k připojení dat k webovému dokumentu. S použitím D3.js lze vytvořit téměř jakoukoliv vizualizaci.

Díky knihovně lze dynamicky manipulovat s atributy a vlastnostmi HTML elementů, stejně tak je možné vytvářet SVG elementy. Další šikovnou vlastností je schopnost napojit DOM nebo SVG element na pole dat a jakákoliv změna v datech se následně překreslí i do vizualizace. Stavebními kameny vizualizační komponenty vytvořené pomocí D3.js jsou jednotlivé dílčí SVG tvary – kruh, čtverec atd. Z toho vyplývá, že i na první pohled složité obrazce jsou v základu pouze jednoduché tvary, které jsou uspořádány dle pravidel v programu (Dewar, 2013).

Knihovna si díky své flexibilitě perfektně rozumí s existujícími webovými technologiemi (HTML, CSS, SVG) a má tak velkou výhodu oproti ostatním nástrojům. Dává tak možnost vzniknout téměř jakékoliv vizualizaci a neomezuje se jen na vybrané oblasti stránky jako jsou nástroje typu Processing.js, Paper.js nebo Raphael.js, či další knihovny založené pouze na využití elementů canvas nebo SVG (Thomas, 2015).

3.6.2 Grafické znázornění

V rámci webových stánek a aplikací je možné setkat se se dvěma typy grafiky. Bitmapová, která zastupuje formáty JPEG, PNG a GIF a někdy je také označována jako grafika rastrová. Technologie zobrazení spočívá v ukazování pixelů, které jsou různě barevné a společně tvoří nějaký obrazec.

Druhý typ, vektorová grafika (formát SVG), definuje body a cesty, jejichž velikost se může libovolně měnit, a přitom nedochází ke ztrátě kvality obrazce. Na webu se používají především pro značky, loga, bannery, obrysové kresby a ilustrace.
Některé výhody použití SVG grafiky:

- Skládá se z textových informací, proto jsou obrazce snáze přístupné a dají se indexovat webovými prohlížeči.
- Snadná úprava grafiky, na kterou lze zároveň aplikovat CSS (kaskádové styly).
- Vektorový obrazec lze umístit nad jiný objekt a definovat mu průsvitnost, aby objekty pod ním byly viditelné.
- Menší datová velikost výsledné grafiky ve srovnání s bitmapovou.
- K dispozici je celá řada knihoven, nástrojů a rozšíření pro tvorbu a úpravu SVG grafiky.

Použití vektorové grafiky má celou řadu výhod, přesto je nutné použití vždy pečlivě zvážit. Například reprodukování realistických fotografií pomocí vektorů je skoro nereálné (Brown a kol., 2014).

3.7 Charakteristika významných použitých technologií

3.7.1 JQUERY

Jquery byl významný přírůstek do světa javascriptových knihoven a přinesl dosti významné změny v psaní kódu pomocí JavaScriptu. Nejprve knihovna vypadala podobně jako cssQuery[9], ale po vydání knihovny Deanem Edardsem se Jquery vydalo svou vlastní cestou. Podporuje CSS 1 až CSS 3 a základní funkcionalitu XPath. Vylepšuje navigaci a manipulaci s objektovým modelem a rovněž podporovaná napříč všemi prohlížeči.

3.7.2 XPath

Výrazy XPath jsou velice výkonnými nástroji v oblasti navigace v dokumentech XML. Tato technologie již nějakou dobu existuje a proto tam, kde lze narazit na implementaci DOM, je pravděpodobné, že bude k dispozici i XPath. XPath jsou mnohem mocnější než cokoliv, co lze zapsat pomocí CSS selektorů, přestože jejich zápis je delší.

[9] CssQuery – první veřejná knihovna poskytující plnou podporu pro CSS od verze 1 až po CCS3.

3.7.3 AJAX

Ajax je termín, který vznikl při vysvětlení pojmu asynchronní komunikace mezi klientem a serverem vývojářem Jesse Jamesem Garrenttem. Zkratka znamená asynchronní JavaScript s XML. Zkratka vystihuje pojmenování technik, které jsou využívány k vytváření dynamických webových stránek a aplikací. Využití Ajaxu spočívá na vysílání požadavku z prohlížeče na server.

3.7.3.1 Použití Ajaxu

Samotná implementace Ajaxu do projektu není složitá a vyžaduje opravdu malé množství napsaného kódu. Přesto je využití velice obsáhlé. Jako příklad lze uvést načítání stránky při odesílání formuláře. Při odeslání formuláře na server není nutné načítat kompletní webovou stránku, Ajax umožní načtení pouze části z celkového obsahu. Data jsou obstarávána asynchronně a načte se pouze malá část výsledků.

Při vzájemné komunikaci serveru a klienta je zpracováván požadavek http (Hypertext Transfer Protocol). Tento protokol byl navržen pro jednoduchý přenos dokumentů HTML. Spojení http se navazuje dynamicky pomocí JavaScriptu a je hojně využíváno ve webových aplikacích díky své pružné reakci na požadavek. Asynchronní komunikace čili posílání dat na server a vracení výsledkových dat je hlavní účel Ajaxu. Vývojář dané aplikace si může libovolně zvolit formát zasílaných dat.

Proto, aby Ajax fungoval správně, je nutné, aby ověřil spojení se serverem. Existuje několik způsobů, jak spojení ověřit. Dle Resiga (2003) je vhodné vést komunikaci dat pomocí objektu XMLHttpRequest, který obsahuje vhodné metody pro navázání komunikace a čtení dat ze serveru. Zásadním krokem před odesláním dat na server je jejich serializace. Data je nutné naformátovat takovým způsobem, aby je server mohl jednoduše přečíst. Pokud proběhne serializace v pořádku, přistoupí se k přenosu dat pomocí požadavku GET nebo POST.

3.7.4 Http metody

3.7.4.1 Posílání požadavku pomocí metody GET

Jedna z tzv. formulářových metod. Předávání dat probíhá prostřednictvím adresního řádku prohlížeče. Data jsou k objektu url připojena za speciálním znakem otazníku. Odeslaná data jsou v adresním řádku viditelná, proto platí, že by se touto metodou neměla posílat žádná citlivá data.

3.7.4.2 Posílání požadavku pomocí metody POST

U této metody se data posílají zcela odlišným způsobem. Požadavek POST umí přenést jakákoliv data (určitý formát dat) jakékoliv délky (velikost souboru). Velikost dat zasílaných touto metodou není nijak omezena.

Pro korektní komunikaci je nutné použít ošetření všech chybových stavů, které mohou nastat a rovněž zajistit kontrolu vypršení časového limitu. Pokud komunikace se serverem proběhla v pořádku, server odpoví a vrátí požadovaná data. Data, která přicházejí ze serveru, mohou být různě naformátovány (JSON, XML, HTML). Záleží na tom, co je pro danou aplikaci výhodnější a s čím je programátor zvyklý pracovat. Z důvodu bezpečnosti není dovoleno posílat požadavky na stránky, které se nenacházejí ve stejné doméně. Toto opatření má zabránit útočníkům v oblasti Internetu zneužívání cizích dat.

Jak zdůrazňuje Resig (2003), využití Ajaxu pro webové aplikace je velmi mocný nástroj. Dynamickým načítáním menších částí obsahu do běžící aplikace vytváří velice rychle reagující rozhraní, které budou rádi uživatelé používat.

3.8 Geolokace

3.8.1 Vyhledávání pozice

Vyhledání pomocí geolokačního rozhraní je velice oblíbenou funkcí, kterou nabízí HTML 5. Pokud uživatel poskytne souhlas s použitím jeho polohy, nic potom nebrání tomu, aby byly tyto údaje zpracovány a použity.

Webové aplikace tedy dokáží prostřednictvím geolokačního rozhraní HTML5 určit polohu. Informace mohou pocházet z různých zdrojů a mohou mít také rozdílnou přesnost. Základní

veličiny, se kterými se pracuje, jsou zeměpisná délka a šířka. Tento údaj bývá většinou zapsán ve formátu desetinného čísla (například Latitude: 45.1887, Longitude: -114.1544). Kromě zápisu v desetinném formátu je také možné setkat se s formátem DMS (Degree, Minute, Second – Stupně Minuty Sekundy) a ten může vypadat například takto (45 ° 18’ 87’). K těmto základním údajům je ještě přidružena informace o přesnosti udávaných souřadnic. V závislosti na typu zařízení, se kterým uživatel pracuje, je také možné obdržet data o nadmořské výšce, směru a rychlosti. Pokud data nejsou k dispozici a vývojář aplikace s nimi počítá, nabývají hodnot null (Lubbers a kol., 2011).

Geolokační rozhraní neuvádí informace o tom, jakým způsobem se má poloha zjistit. Společně se souřadnicemi se sdělí také míra přesnosti poskytnutých údajů. Není však žádná záruka, že poskytnuté údaje jsou pravdivé. Můžeme rozlišit několik druhů zdrojů, ze kterých je možné data získat (Brown a kol., 2014).

Uživatel může informace ručně zadat.

- Adresa IP.
- Zaměření polohy GPS.
- Podle MAC adresy Wi-Fi, nebo Bluetooth.
- Podle identifikátoru mobilních zařízení GSM.

Většina zařízení používá kombinaci výše uvedených zdrojů informací o poloze, aby dosáhla co nejpřesnějšího výsledku.

4 Vlastní práce

V rámci projektu Medard, který je zaměřen na předpovědi počasí a kvality ovzduší, provozuje Ústav informatiky AV ČR numerické modely pro předpověď nejrůznějších meteorologických veličin, převážně z oblasti hydrometeorologie a kvality ovzduší. Výstupem předpovědního numerického modelu jsou data, která jsou vizualizována. Sledován je zejména atmosférický tlak, oblačnost, teplota, rychlost a směr větru. Pro předpověď s třídenním rozsahem je využívána prezentace na internetu, která poskytuje uživatelům přehledné informace o vývoji počasí na území České republiky a střední Evropy. Předpověď je dostupná online a neustále se aktualizuje.

I přes funkční, stabilní a uživateli hojně využívanou aplikaci Medard byl vznesen požadavek na vývoj nové webové aplikace, která by používala pro vizualizaci meteorologických dat moderní prvky a animace, kterých lze dnes v oblasti internetových technologií dosáhnout. Webová aplikace byla vytvořena pro účel interního využití Ústavu AV ČR na nejrůznějších prezentacích, seminářích a konferencích.

4.1 Definice zadání

Primární cíl pro vývoj aplikace je zobrazit meteorologická data v podobě, která upoutá pozornost i běžných uživatelů. Vizualizace samotných dat o počasí se musí co nejvíce přiblížit lidskému vnímání meteorologických vlivů. Zároveň je ovšem nutné dodržet zavedená pravidla a zvyklosti při návrhu grafiky, která musí korespondovat s obecnými základy vizualizace meteorologických veličin. Aplikace je určena jak pro laickou veřejnost, tak i pro samotné odborníky, kteří v oblasti meteorologie testují různá data z numerických modelů.

Speciálním požadavkem při tvorbě aplikace je možnost importu datových zdrojů, které pocházejí z numerických modelů AV ČR. Vzhledem k tomuto požadavku musí být uživatel schopen jednoduchým způsobem nahrát datový soubor a zobrazit si jeho obsah ve webové aplikaci. Vstupním datovým formátem je speciální soubor GRIB[10] nebo NetCDF[11]. Dle vědecké terminologie lze tyto soubory považovat za nezávislý soubor určený pro přenos meteorologických dat. Pro webovou aplikaci je nutné tyto soubory konvertovat do obecnějšího formátu pro přenos dat typu JSON. V souboru jsou uloženy informace o počasí

[10] GRIB – datový formát používaný v meteorologii pro přenos historických dat a předpovědí počasí
[11] NetCDF – soubor softwarových knihoven a nezávislých datových formátů s vědeckými daty

na daném území. Zjednodušeně lze říci, že každý bod na meteorologické mapě má přiřazená data, která obsahují všechny dostupné informace pro tento bod. Konverze datových souborů z formátů GRIB a NetCDF na univerzální formát JSON je velice složitá operace a přímo nesouvisí s daným tématem, proto detaily týkající se konverze formátů nejsou nijak dál specifikovány.

Nahrání souborů je nutné umožnit z uživatelského prostředí. Aplikace umožní vložení souboru prostřednictvím uživatelského rozhraní. Jakmile je soubor vložen, musí proběhnout jeho kontrola a pokud se jedná o očekávaný vstup v požadovaném formátu, provede webová aplikace vizualizaci datových podkladů na mapě Světa. Pokud nese vložený datový soubor informaci pro různá časová období, aplikace umožní postupně prohlédnout dostupné informace pro daný čas. Zároveň je umožněn výběr ze všech dostupných veličin.

4.2 Analýza dostupných řešení

Po analýze požadavků pro vývoj aplikace byl proveden průzkum veřejně dostupných webových aplikací zobrazujících data o počasí. Za účelem dosažení všech výše zmíněných potřeb byla provedena analýza, zda je nutné aplikaci vytvořit, nebo zda nějaké řešení splňující výše uvedené požadavky již existuje a mohlo by být využito.

Byl proveden průzkum a analýza notoricky známých webových portálů zobrazujících meteorologická data. Do průzkumu byla zahrnuta i webová řešení, která nemají tak velký počet uživatelů, ale jejich provedení je nápadité a funkční. Průzkum, který zmapoval oblast aplikací, které ukazují data o počasí, byl zúžen na 7 řešení, které prošly analýzou. Záměrně bylo vybráno několik zástupců webových aplikací i některá desktopová řešení. Kritéria hodnocení byla zvolena následujícím způsobem. Jednotlivé body jsou sestaveny od těch nejdůležitějších.

4.2.1 Kritéria hodnocení pro analýzu dostupných řešení

1. Moderní vizualizace meteorologických dat s použitím animace.
2. Možnost nahrání datového souboru z uživatelského prostředí.
3. Automatická aktualizace dat.
4. Podpora vstupních formátů GRIB, NetCDF.
5. Uživatelsky nastavitelné prohlíženého časového úseku.
6. Dostupnost na internetu online.

7. Větší množství zobrazených meteorologických veličin.
8. Ovládání bez nutnosti pokročilých znalostí.
9. Na základě geolokace určit přibližnou polohu.
10. Intuitivní používání.

4.2.2 Seznam aplikací, které byly podrobeny vícekriteriální analýze variant

Níže jsou uvedeny názvy analyzovaných aplikací. U každého názvu je odkaz do příloh, kde je obrázek daného uživatelského prostředí.

1. Ventusky – viz přílohy obrázek č. 5
2. YR – viz přílohy obrázek č. 6
3. Windguru – viz přílohy obrázek č. 7
4. MeteoEarth – viz přílohy obrázek č. 8
5. Windy TV – viz přílohy obrázek č. 9
6. Panoply – viz přílohy obrázek č. 10
7. ZyGrib– viz přílohy obrázek č. 11

4.2.3 Provedení komparace

Po analýze požadavků na funkcionalitu aplikace bylo provedeno srovnání mnoha řešení, která se zabývají vizualizací počasí. Průzkumu podlehla většina veřejně dostupných aplikací se stejným zaměřením. Každé z těchto přibližně 25 řešení bylo podrobněji analyzováno, aby později mohlo dojít ke zúžení základního výběru a blíže porovnat 7 aplikací. Záměrně byly do srovnání zahrnuty i dvě desktopové aplikace, které samozřejmě nesplňují podmínku online režimu. Namísto toho ale oproti ostatním řešením disponují možností nahrání vlastních datových souborů. Tento fakt slibuje, že v komparaci mohou zaujmout místo plnohodnotných konkurentů jiných online řešení.

Tabulka 1 - vícekriteriální analýza variant na vybraném souboru zkoumaných aplikací

aplikace	moderní vizualizace, použití animace	nahrání souboru z uživatelského prostředí	aktualizace dat
Ventusky	9	1	8
YR	7	1	10
Windguru	5	1	8
MeteoEarth	10	1	1
Windy TV	8	1	8
Panoply	5	5	1
ZyGrib	5	8	1

aplikace	podpora vstupních formátů Grib, NetCDF	zvolení časového úseku	online řešení
Ventusky	1	9	10
YR	1	4	10
Windguru	1	8	10
MeteoEarth	1	2	10
Windy TV	1	5	10
Panoply	10	6	1
ZyGrib	10	7	1

aplikace	počet zobrazených meteo. veličin	použití pokročilých znalostí	geolokace
Ventusky	7	9	9
YR	9	4	2
Windguru	2	6	3
MeteoEarth	2	7	4
Windy TV	5	6	7
Panoply	6	3	1
ZyGrib	8	1	1

aplikace	intuitivní používání aplikace	celkový počet bodů	celkové pořadí aplikací
Ventusky	9	72	1.
YR	6	54	3.
Windguru	5	49	4.
MeteoEarth	8	46	5.
Windy TV	7	58	2.
Panoply	2	40	7.
ZyGrib	1	43	6.

Zdroj: vlastní

4.2.4 Výsledky a ukazatele použité analýzy

V rámci komparace jednotlivých aplikací byla provedena vícekriteriální analýza variant. Výsledkem metody je porovnání jednotlivých řešení, které byly bodově hodnoceny dle zadaných kritérií. Vzhledem k velkému počtu hodnocených oblastí byla využita metoda bodovací. Ve všech zkoumaných kritériích se používá stejná bodová škála, která nabývá hodnot od 1 do 10. Přičemž platí, že 10 bodů znamená nejlepší ohodnocení. Na základě obodování všech dílčích částí byl proveden součet bodů a určeno celkové pořadí.

Zvolená metoda, k provedení komparace, ukázala jako nejlepší volbu webové řešení s názvem Ventusky. Naopak nejhůře se umístila desktopová aplikace s názvem Panoply. Pouze o 3 body více získala aplikace ZyGrib. Aplikace na 6. a 7. místě nejvíce zaostaly u požadavku na online řešení. I přes nejhorší umístění v celkovém srovnání jsou jedinými zástupci, které podporují nahrání souborů GRIB a NetCDF. S podobným počtem bodů se uspořádaly aplikace na druhém až pátém místě. Dle výsledků lze pozorovat, že nejlépe hodnocená aplikace Vetusky s celkovým počtem 72 bodů získala prvenství před druhým nejlepším řešením o 14 bodů. I přes zvolení nejlepší z variant není možné tuto aplikaci využít, protože nesplňuje některé kritické body zadání.

Na základě analýzy bylo zjištěno, že žádné z veřejně dostupných řešení nesplňuje požadavky v takovém rozsahu, aby ho bylo možné použít k účelům prezentace vlastních datových zdrojů a současně bylo dostupné online bez nutnosti instalace a nutnosti použití odborných znalostí.

Webové aplikace splňují požadavky z hlediska vizualizace, grafického zpracování a zároveň intuitivního použití. Hlavním nedostatkem u všech porovnávaných internetových aplikací je absence použití jiných datových zdrojů než těch, které jsou součástí. Nelze uživatelsky nahrát jakýkoliv jiný soubor, ze kterého by se zobrazila uložená data o počasí.

Některé aplikace umožňují přepnutí mezi předpověďmi od různých poskytovatelů. Vylepšená funkcionalita je kladně hodnocena a umožňuje komparaci odlišných datových zdrojů. I přes mírné vylepšení je nutné konstatovat, že možnost vložení dat z klientského prostředí chybí.

Desktopové aplikace jsou rovněž nevhodné pro využití. Umožnují prohlídku vlastních meteorologických dat, ovšem nad touto výhodou převažuje spoustu nevýhod, které brání použití tohoto řešení. Nutnost instalace a vyšší odborná znalost potřebná pro obsluhu těchto aplikací jsou hlavní důvody, které vylučují použít tuto možnost. Všeobecně lze říci, že desktopové aplikace jsou vhodné pro pokročilejší uživatele v této problematice. Po instalaci

aplikace nejsou k dispozici žádná online data nebo jakékoliv připravené testovací soubory. Software je určen výhradně k prohlídce uživatelsky vložených dat. Grafické prostředí je vytvořeno tak, aby korespondovalo se všemi zažitými standardy a vytvářelo pracovní prostředí pro profesionální uživatele. V tomto ohledu nelze desktopovým aplikacím nic vytknout, ovšem samotná vizualizace nepoužívá žádných animací, je kompletně statická, a proto běžného uživatele ničím nezaujme.

4.3 Vývoj vlastní aplikace

Analýza prokázala nutnost vývoje vlastní aplikace, která splní stanovené cíle ve všech směrech. Pro vývoj byla použita platforma Node.js, která zvládne skvěle obsluhovat požadavky z webových prohlížečů. Je navržena pro tvorbu robustních, škálovatelných webových aplikací, který by si měly poradit s velkým provozem. Samotná platforma je modernější než alternativy v podobě php, python, nebo ASP.NET. K její volbě došlo i z důvodu, že vychází z klientské verze jazyka JavaScript. JS je hojně používán k vývoji webových řešení na klientské straně, Node.js umožňuje jeho výhody přenést i do prostředí backendu a postavit na něm kompletní a stabilní webovou aplikaci.

Vzhledem k tomu, že se k vývoji používá mnoho javascriptových knihoven, je použití Nodej.js vhodnou volbou i s ohledem na čerpání znalostí z obecného použití JavaScriptu. Studium dalšího programovacího jazyka a prostředí pro vývoj backendu by prodloužil samotný vývoj aplikace.

4.3.1 Stavba aplikace založená na open-source

V rámci časové úspory byla aplikace vytvořena na open-source základě a následně upravena dle zadání. Open-source je počítačový software, který disponuje otevřeným zdrojovým kódem. Autor daného projektu tak dává možnost ostatním vývojářům použít jeho základ na stavbu vlastního řešení. Někdy open-source neboli otevřený software podléhá určitému omezení. Kód může být například zveřejněn pouze pro prohlížení. Jiné podmínky umožňují úpravu zdrojových kódů. Někdy je nutné striktně dodržet pravidla, které jsou určeny pro použití open-source, aby tak nedošlo k zneužití autorských práv vývojáře, který své zdrojové kódy poskytnul.

Pro vývoj webové aplikace v Node.js byl použit open-source projekt, který zahrnuje stavbu a nastavení technologie Node.js. Základ také navádí na vhodné použití javascriptových

knihoven, které umožňují vytvářet působivé vizualizace. Open-source projekt obsahoval pouze základní stavební prvky pro vývoj aplikace. Napojení na servery s daty o počasí bylo nutné nastavit tak, aby se automaticky data stahovala po určitých časových intervalech a umožnila mít vždy aktuální podobu dat online. Protože jsou tyto části aplikační logiky velmi složité a přímo nesouvisí s vybraným tématem práce, není jejich charakteristika podrobněji rozpracována.

4.3.2 Instalace ekosystému Node.js

Instalaci Node.js je možné provést několika způsoby. Nepoužívanějšími z nich jsou instalace prostřednictvím příkazové řádky nebo instalace za pomoci správce balíčků. Pro vytvoření jádra Node.js byla vybrána varianta pomocí správce balíčků. Je jednodušší a podobá se instalaci balíků na operačních systémech Linux. Na oficiálních stránkách platformy Node.js je k dispozici podrobný návod k instalaci[12]. Dobře funguje i fórum, které spravuje komunita vývojářů vytvářejících aplikace na této platformě a v případě nějakých problémů lze dohledat velké množství odpovědí na různá témata.

Node.js je ve své základní podobě vybaven vestavěným http serverem. Nad ním je postavena mezivrstva s názvem Connect, která podporuje například soubory cookies. Nad mezivrstvou se nachází vrstva Express, což je framework, který například podporuje šablonovací systém Jade a umožňuje směrování. Aplikační rámec nazývaný Express je základním balíčkem, bez kterého by se aplikace navržená následujícím způsobem neobešla.

Pro správnou instalaci a následné použití Node.js je nutné mít na stroji, který bude hostovat jádro systému, nainstalovanou dvojici JRE a JDK. JRE neboli Java Runtime Environment je důležitý prvek pro velké množství webových aplikací a Node.js se bez tohoto prostředí neobejde. JDK (Java Development Kit) je také nedílnou součástí a zajišťuje privátní běhové prostředí pro aplikaci.

Po vytvoření všech nutných programů je možné přejít k instalaci samotného ekosystému. Jako první je nutné nainstalovat správce balíčkovacího systému NPM (Node package manager). Balíčky lze přidávat jednotlivě, dle potřeby daného projektu, nebo je možné vytvořit soubor s názvem „package.json“, ve kterém jsou nadefinovány přesně konkrétní názvy balíčků, které

[12] Odkaz na návod k instalaci: https://nodejs.org/en/

je potřeba získat a nainstalovat. Při vývoji webové aplikace byl použit takový soubor a vše potřebné se naistalovalo pomocí několika jednoduchých příkazů.

Níže je package.json, který byl použit pro vytvoření a instalaci balíčků. Oproti open-source verzi jsou přidány mimo jiné i balíčky Accepts a Multiparty, které jsou primárně využívány pro možnost nastavení uživatelského uploadu souborů.

Výhoda použití souboru k instalaci je na první pohled patrná. Dojde ke značné úspoře času. Jednoduší způsob obnáší i jistá rizika. Při definici balíčků je nutné ověřit, zda jsou mezi sebou jednotlivé verze (u ukázky se jedná o číslo ve tvaru „x.x.x“) balíčků kompatibilní a nebudou při realizaci aplikace způsobovat žádné problémy. Postupné přidávání balíčků je sice časově náročnější, ale vývojář si může ušetřit spoustu problémů, pokud by nastala nekompatibilita.

Ukázka souboru package.json použitého pro instalaci balíčků:

```
{
    "engine": "node >= 0.10.21",
    "dependencies": {
        "when": "2.6.0",
        "swig": "1.2.2",
        "mkdirp": "0.3.5",
        "express": "3.4.4",
        "accepts": "3.2.2",
        "body-parser": "1.2.3",
        "compression": "1.0.1",
        "cookie-parse": "1.2.2",
        "debug": "1.2.2",
        "morgan": "3.5.2",
        "multiparty": "1.0.1",
        "jade": "1.2.2"
    }
}
```

Některé z použitých balíčků stojí za povšimnutí, protože tvoří základ a významnou funkcionalitu, bez které by webová aplikace nebyla funkční. Výše bylo uvedeno, že Express je velmi důležitou součástí celého ekosystému Node.js. Umožňuje nastavit mezivrstvu, aby reagovala na požadavky protokolu http. Lze nadefinovat směrovací tabulku, které se používá k provádění akcí založených na metodě url a protokolu http. Šablonovací systém umožňuje dynamické vykreslování HTML stránek prostřednictvím předávaných parametrů.

Systém šablon „Jade“ je použit proto, aby urychloval zápis HTML. Toto rozšíření Node.js umožňuje využít předávání parametrů jako proměnných, povoluje použití podmínek i cyklů a s jeho pomocí lze kód zapsat v alternativní formě.

```
div.priklad
 p Zapsany text jako odstavec
   a(href="/priklad") odkaz
   | na vytvoreny priklad.
```

```
<div class="priklad">
 <p>Zapsany text jako odstavec <a href="/clanek">odkaz</a> na
vytvoreny priklad.</p>
</div>
```

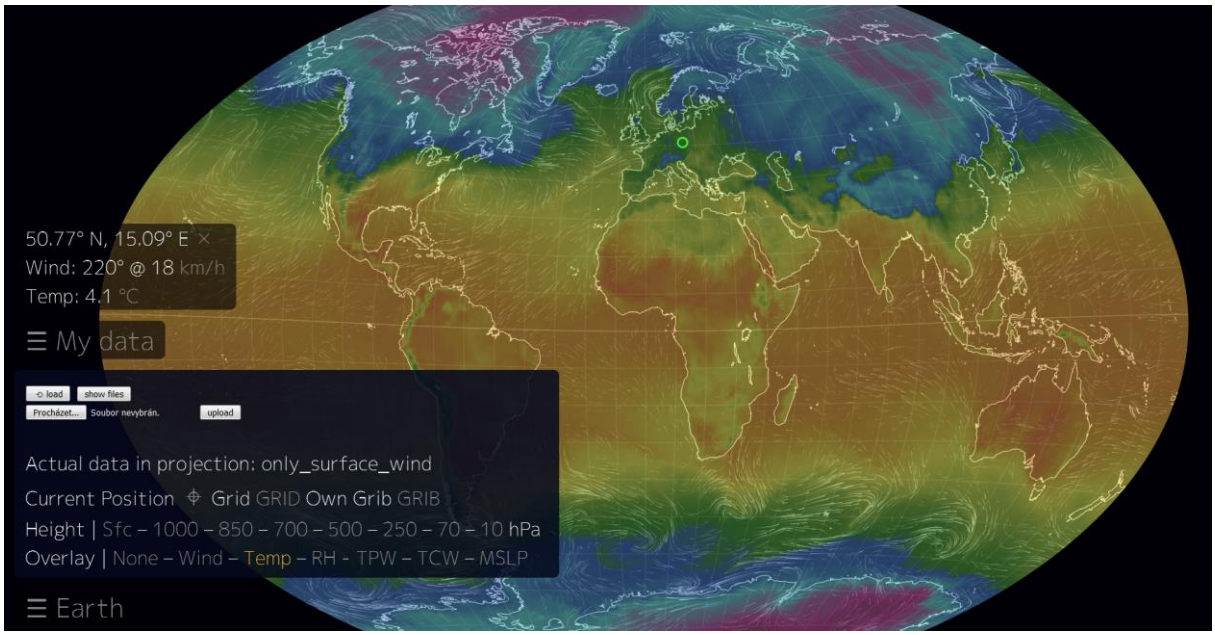
Příklad uvedený výše ukazuje, jak lze zápis zjednodušit. Při vykonávání úlohy se zapsaný kód převede na standartní HTML a daná stránka je zobrazena.

Aplikační rámec multiparty je primárně použit na upload souborů. Modul umožňuje naprogramovat ukládání souborů z uživatelského prostředí. Nechybí možnost kontroly velikosti souboru, který chce uživatel nahrát. Během tvorby aplikace bylo testováno nahrání souboru, který měl přes 1 GB a vše proběhlo bez problémů. Ve finální verzi aplikace je maximální velikost nastavena na 300 MB. Toto omezení není kvůli platformě samotné, ale kvůli samotným datovým formátům s meteorologickými daty, které se do aplikace nahrávají. Není potřeba nahrávat data větší velikosti. Toto opatření slouží jako prvotní kontrola toho, že se opravdu jedná o data, která je možné vizualizovat. Další stupeň kontroly je nastaven tak, aby nedocházelo k uploadu souborů, které jsou jiného formátu než podporované soubory typu GRIB(grib2) a NetCDF(mpas).

Pokud systém dostane informaci o tom, že má k dispozici soubor pro uložení, provede se otevření soboru a přečtení prvního řádku, kde je definován jeho typ. Pakliže typ odpovídá správnému formátu je soubor uložen, v opačném případě operace skončí chybou.

V předchozí části byly podrobněji popsány nejdůležitější použité moduly. Další balíčky ekosystému Node.js, které jsou součástí aplikace řeší záležitosti týkající se předání parametrů mezi klientem a serverem, mají na starosti správnou funkčnost a kontrolu použitých šablon, generují a kontrolují cookies a řeší další dílčí úlohy.

Obrázek 4 - ukázka vlastní aplikace, aktivní panel pro nahrání souboru a zobrazená data dle aktuální polohy



Zdroj: vlastní aplikace

5 Zhodnocení výsledků

5.1 Charakteristika dílčích částí aplikace

5.1.1 Moderní vizualizace

Jedním z požadavků na aplikaci byl záměr ukázat meteorologická data způsobem, který by zaujal každého návštěvníka při příchodu na webovou stránku s aplikací. Za tímto účelem byla vybrána k prvotnímu zobrazení veličina větru. Vítr je simulován na mapových podkladech pomocí bílých čar a barevného podkladu. Čáry vykreslují směr větru a barvený podklad charakterizuje jeho intenzitu. Vzhledem k tomu, že je vizualizace větru zajímavá, je použita jako základní vrstva pro všechny ostatní veličiny. Pohybující se čáry jsou zobrazeny neustále a změnou prohlížené veličiny se mění podbarvení, které se používá na základě barevných škál vybraných veličin.

5.1.1.1 Použití klíčové knihovny pro zobrazení

Pro zobrazení je použita javascriptová knihovna s názvem D3.js. Její nasazení umožňuje využít její univerzálnost pro zobrazení různorodých dat. Tento framework se nesnaží pokrýt mnoho problémů, ale soustředí se pouze na vizualizaci, proto se ním dobře pracuje. Použití rozšíření umožňuje dynamickou manipulaci s objekty prohlížeče a vytvoření SVG elementů. Mapový podklad vizualizace je rozdělen na pravidelnou síťovou strukturu. Každému bodu v síti je na základě datového souboru přidělena hodnota. Pro každou veličinu se hodnota porovná s tabulkou, kde je přesně definovaná barevná škála. Dle roztřídění hodnot pro jednotlivé intervaly jsou jim přiřazeny odpovídající barvy a vzniká vizualizace.

K obnovení vizualizace dochází nejen při změně zobrazované veličiny, ale i pokud se mapový podklad posouvá, přibližuje nebo oddaluje. Při každé změně se musí provést funkce pro aktualizaci, aby zobrazení odpovídalo skutečnosti a bralo v úvahu měnící se polohu a měřítko. Aktualizování vždy trvá různou dobu, proto je do uživatelského prostředí implementován progres-bar, který ukazuje stav načteného obsahu.

5.1.1.2 Projekce

Z uživatelského prostředí je možné zvolit typ projekce meteorologických dat. Primárně je vybrané zobrazení zeměkoule. Toto zobrazení je známé například z aplikace Google Earth. Virtuální globus zobrazující data o počasí je možné libovolně otáčet všemi směry

a přibližovat si konkrétní oblasti. Mimo základní zobrazení, aplikace disponuje ještě 7 dalšími typy projekce. Veškeré druhy projekce mají stejné výhody. Mapu lze libovolně přibližovat, otáčet s ní a stále je zachované zobrazení meteorologických veličin na mapovém podkladu. Zároveň všechny typy vizualizace podporují kliknutí na kterýkoliv bod v mapě. Po kliknutí se zobrazí okno s aktuální hodnotou prohlížené veličiny pro daný bod. Místo v mapě je zároveň přesně definováno prostřednictvím souřadnicové soustavy.

Pro možnost porovnání vybraných oblastí je podporována funkce, která rozdělí mapu na mřížku, kde jsou stejně velká pole. Tato možnost lze zapnout nebo vypnout kliknutím na tlačítko „GRID“.

Ukázka části kódu, který definuje rozmístění dat:

```
[{
     "header": {
         "refTime": "2017-01-30T18:00:00.000Z",
//čas pro zobrazená dat
         "parameterCategory": 1,
//paremetr definující veličinu
         "parameterNumber": 2,
//číslo definující veličinu
         "surface1Type": 0,
//typ zobrazení
         "surface1Value": 10.0,
//vzdálenost od povrchu
         "forecastTime": 1,
//definice předpovědi
         "nx": 181,
         "ny": 360, "lo1": 0,
         "la1": 90,
         "lo2": 360,
         "la2": -90,
         "dx": 1,
         "dy": 1
//souřadnicový systém
     },
     "data": [2.42, 2.31, 2.19, 2.08, 3.96, 3.84, 3.72, 3.6,
3.47, ...]
//uložená data pro každý bod na Zemi, pole obsahuje tisíce
hodnot
 }]
```

5.1.1.3 Interaktivní zobrazení

Jádro aplikace je vytvořeno v Node.js, což je ekosystém, který vznikl z klientského JavaScriptu. Zatímco na serveru se s ním řeší komunikace mezi klientem a serverem, úkolem

klientského JavaScriptu je co nejvíce komunikaci omezit. Na tuto úlohu je využívaná technologie Ajax, která dokáže modifikovat obsah, aniž by komunikovala se serverem.

Ajax je jedna z nejoblíbenějších a nejvyužívanějších technologií vycházejících z JavaScriptu. Jeho samotná implementace není složitá, vyžaduje opravdu malé množství napsaného kódu. Přesto je jeho použití velmi výhodné a značně zjednoduší práci při vývoji. Ve webové aplikaci je jeho úkolem načítat dílčí části z celkového obsahu. Probíhá časté načítání obsahu po malých částech, takže uživatel ani nepostřehne, že se stránka aktualizuje. Postupná aktualizace má takovou výhodu, že prvky, které jsou stálé a načtou se při prvním spuštění, není třeba obnovovat. Dynamicky se načítá pouze obsah, který se neustále mění.

Tento proces lze pozorovat například při zoomovaní meteorologické mapy. Mění se prohlížená oblast, proto se musí aktualizovat data. Jejich aktualizace probíhá postupně. Dle zobrazené části se natáhnou pouze data pro prohlíženou oblast. Kdyby se na začátku načetl kompletní obsah pro všechny možnosti zobrazení, trvalo by to velice dlouho a aplikace by byla nepoužitelná. Použití Ajaxu a asynchronního načítání dat má tedy v této aplikaci velký význam.

5.1.2 Nahrání souboru

Za největší přidanou hodnotu webové aplikace lze považovat možnost nahrání vlastních meteorologických dat prostřednictvím uživatelského prostředí a jejich následnou vizualizaci. Nahrání souborů do aplikace je vyřešeno standardním způsobem, který je znám buď z jiných webových aplikací, nebo z prostředí operačních systémů. Po kliknutí na tlačítko je uživateli umožněn výběr souboru z jeho zařízení. Pokud soubor odpovídá svým typem a velikostí je nahrán na server, kde proběhne jeho konverze do formátu JSON. Soubor již nese všechny potřebné informace o datech, které se mají zobrazit. Zobrazení uživatelsky vložených dat má stejné možnosti jako prohlídka automaticky stahovaných datových souborů. Uživatel si tak může zobrazit jakákoliv data z právě nahraného souboru.

Webová aplikace umožňuje správu nahraných meteorologických dat. Je tedy možné soubory nejen nahrávat, ale i odstranit. Zároveň je umožněno vybírat ze seznamu nahraných souborů. Seznam lze zobrazit po kliku na tlačítko, pokud je vybrán soubor, provede se jeho konverze na podporovaný formát a v aplikaci se zobrazí vybraná data. Po vybrání souboru se automaticky spustí konverze do formátu JSON. Operace probíhá na pozadí a její doba trvání

je různá v závislosti na velikosti souboru. Aby byl uživatel informován o tom, že se na pozadí něco děje, objeví se informace, která upozorňuje, že systém připravuje soubor pro vizualizaci.

Ukázka funkce, která zajišťuje korektní generování hlavičky načítaného souboru:

```
function gfs1p0degPath(attr, type, surface, level) {
    var dir = attr.date,
        var stamp = (dir === "current") ? "current" : dir ===
"owngrib" ? "owngrib" : attr.hour;
    var file = [stamp, type, surface, level, "gfs",
"1.0"].filter(µ.isValue).join("-") + ".json";
    return [WEATHER_PATH, dir, file].join("/");
}
```

5.1.3 Aktualizace dat

K aktualizaci datových souborů dochází celkem 4x denně. Data o počasí jsou automaticky stahovány ze serveru NOAA (National oceanic and atmospheric administration)[13] Jedná se o skutečné meteorologické údaje, které zahrnují data pro celou planetu. Data jsou historická, jsou tedy k dispozici vždy s tříhodinovým zpožděním. Pro předpověď počasí jsou používány soubory s označení GFS (global forecast system)[14]. U vyvíjené webové aplikace je kladen důraz na korektní zobrazení historických dat, proto je primárně využíván server s uloženými numerickými modely NOAA.

Aplikace využívá ke komunikaci klienta se serverem jednu z http metod. Pro vyčítání a aktualizaci seznamu dostupných souborů k použití je využita metoda GET. Data jsou k objektu url připojena za speciálním znakem „?“. Aplikační framework a webový server dokáže data vložená za speciální znak interpretovat jako serializovaný sled párů: klíč + hodnota. Tímto opatřením je zajištěná neustálá aktualizace dat na stránce, aniž by musela být znovu kompletně načtena. V podstatě se obsah načítá po malých částech neustále a vytváří velice rychlé a dynamické rozhraní, které uživatelům vyhovuje.

Metoda Get je použita i pro načítání veškerých dat do vizualizace. Jednotlivé interaktivní komponenty v uživatelském prostředí vyvolají akci, která na základě informací o daném prvku vytvoří request url (žádost url) ve správném tvaru. Na zaslaný požadavek již reaguje JavaScript, ve kterém se provede funkce pro identifikaci požadavku a po vyhodnocení jsou ze

[13] http://www.noaa.gov/
[14] https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs

serveru odeslána příslušná data zpět do klienta. Pokud požadavek neproběhne a skončí chybou, uživatel je o tomto stavu informován a je vyzván k tomu, aby opakoval akci. Další chybový stav může nastat ve chvíli, kdy je odeslán požadavek, který je předán bezchybně, ale na serveru chybí daný datový soubor. V takovém případě komunikace proběhne v pořádku, ale vizualizace se nenačte z důvodu chybějících dat a uživateli je opět chybový stav oznámen prostřednictvím uživatelského rozhraní.

5.1.4 vstupní formáty GRIB a NetCDF

Podpora datových formátů nesoucí označení GRIB a NetCDF je významnou funkcionalitou aplikace. Bližší informace o daných formátech byly představeny v rámci kapitoly, která definovala zadání. Oba tyto formáty jsou v aplikaci podporovány. Tím, že je nahrání souborů umožněno, řadí se vzniklá aplikace ke složitějším řešením, která jsou k dispozici pouze v offline podobě a vyžadují instalaci. Aplikace umožňuje načítání výše zmíněných formátů a zároveň uživateli poskytuje data o aktuálním počasí. Žádné řešení, které bylo součástí komparace, tyto dvě funkcionality nekombinuje. Zároveň ani v celkovém průzkumu dostupných webových aplikací nemá podobné řešení obdoby.

5.1.5 Zvolení časového intervalu

Meteorologická data jsou stahována v pravidelných intervalech po šesti hodinách. K aktualizaci dochází vždy se stejnou dobu: 06:00, 12:00, 18:00, 24:00.[15] Pro zobrazování počasí v různých časových intervalech slouží kontrolní panel, kterým se lze posouvat po šesti-hodinových nebo po denních intervalech. Vždy se lze vrátit na právě aktuální data. Z důvodu omezené kapacity serveru, historická data, starší než 30 dní, jsou automaticky odebrána.

5.1.6 Online dostupnost

Aplikace je online dostupná z jakéhokoliv zařízení, které má přístup k internetu. Její funkčnost je ověřena ve webových prohlížečích Google Chrome, Mozilla Firefox, Internet Explorer (aktuální verze), Microsoft Edge, Opera, Safari. Webová aplikace je zároveň optimalizovaná pro zobrazení na mobilních zařízeních. Její obsah se přizpůsobí displejům na tabletech a chytrých mobilních telefonech. Vzhledem k tomu, že jádro aplikace je postaveno na platformě Node.js, umožňuje přístup mnoha uživatelů najednou.

[15] Doba je uvedena v UTC – koordinovaný světový čas.

Po otevření aplikace se uživateli zobrazí mapa světa, která ukazuje pohyby větru na planetě Zemi v posledním dostupném souboru meteorologických dat. Uživatel je informován o přesném datu a čase jakému zobrazení odpovídá.

5.1.7 Počet prohlížených veličin

Webová aplikace umožňuje prohlídku různých meteorologických veličin. Konktrétně slouží pro zobrazení rychlosti a směru větru, teploty, relativní vlhkosti, a to pro různou nadmořskou výšku. Díky aplikaci je možné sledovat veličiny na povrchu planety, nebo v celkem deseti různých výškách, respektive v různých hladinách atmosférického tlaku. Data pro oblačnost, srážky a tlak vzduchu je možné pozorovat pouze na povrchu. V porovnání s ostatními aplikacemi je podpora různých veličin mírně omezena, ale v dalším možném rozšíření je reálné doplnit aplikaci například o vizualizaci nejrůznějších plynů v ovzduší, nebo znázornění mořského proudění.

Ukázka přiřazení barevné škály pro veličinu teploty:

```
scale: {
    bounds: [193, 328],
    gradient: µ.segmentedColorScale([
        [193, [37, 4, 42]],
        [206, [41, 10, 130]],
        [219, [81, 40, 40]],
        [233.15, [192, 37, 149]],
   // - 40 ° C/F
        [255.372, [70, 215, 215]],
   // 0 ° F
        [273.15, [21, 84, 187]],
   // 0 ° C
        [275.15, [24, 132, 14]],
   // pouze nad 0 ° C
        [291, [247, 251, 59]],
        [298, [235, 167, 21]],
        [311, [230, 71, 39]],
        [328, [88, 27, 67]]
    ])
}
```

5.1.8 Pokročilé znalosti při ovládání aplikace

K ovládání aplikace není zapotřebí žádných speciálních dovedností. V porovnání s desktopovými aplikacemi, které rovněž podporují nahrávání datových souborů, je webová aplikace použitelná bez jakékoliv instalace. Při instalaci programů a aplikací na počítač

mohou být a často jsou vyžadovány pokročilejší znalosti pro práci se specializovanými softwary. Navíc programy, které byly součástí komparace, není možné instalovat na mobilní telefony nebo tablety. Webová aplikace běží na kterémkoliv zařízení, přizpůsobí se velikosti displeje a její ovládání je jednoduché a intuitivní.

5.1.9 Lokalizace

Pro lokalizaci aktuální polohy je k dispozici tlačítko, které označí bod na mapě, kde se uživatel právě nachází. Společně s označením v podobě zeleného kruhu je také zobrazena informace o hodnotě prohlížené veličiny pro daný bod. U vyhledávání polohy pomocí geolokačního rozhraní je zapotřebí, aby tuto akci uživatel povolil. Musí být informován o tom, že se někdo chystá zjistit jeho polohu a dál údaj zpracovávat. Při takovém požadavku se aktivuje vyskakovací okno, na které uživatel může reagovat. Prozradí svou polohu nebo ji neprozradí. Dále má možnost tyto informace ve webové aplikaci nastavovat automaticky. Je to podobné jako opětovné přihlášení pomocí jména a hesla na jiný server, kde má uživatel také možnost, aby si prohlížeč zapamatoval jeho volbu.

Použití geolokačního rozhraní striktně neuvádí to, jakým způsobem se má aktuální poloha zjistit. Pokud návštěvník přistupuje k aplikaci například prostřednictvím mobilního telefonu a telefon má k dispozici zaměření polohy přes GPS, je velmi pravděpodobné, že výsledná poloha bude přesná. Problém nastane v případě, že se lokalizace místa získává na základě IP. U této varianty mohou být výsledky velice zkreslené. Lokalizace funguje na základě určení adresy IP a následném dohledání fyzické adresy registrátory adresy. Popisovaná metoda je ovšem velice nespolehlivá. Pokud je návštěvník webu připojen k internetu u lokálního poskytovatele, tak ten registruje svou vlastní fyzickou adresu a přesnost takového výsledku se může lišit v desítkách kilometrů. Další způsoby pro určení polohy jsou například lokalizace podle MAC adresy Wi-Fi nebo Bluetooth či dle identifikátoru mobilních zařízení GSM. Webový klient si v základním nastavení automaticky zvolí metodu, kterou použije pro zjištění polohy. Pro dosažení co nejvyšší přesnosti bývá použita kombinace několika metod.

5.1.10 Ovládací prvky

Webová aplikace obsahuje dvě základní menu. Při otevření aplikace obě menu zůstávají schovaná a až po kliknutí se rozbalí. Uživatel může mít obě menu zavřená, otevřená nebo

může používat jen jedno z nich. Menu s názvem „Earth“ umožňuje nastavení typu projekce, nastavení vrstvy prohlížených meteorologických veličin, nastavení vzdálenosti od povrchu Země a další možnosti, které byly představeny v předchozí části. Oproti druhému menu s názvem „My data“ obsahuje i pole, které zobrazuje název zdroje, původ dat, barevnou škálu a časovou značku pro právě zobrazené počasí. Ovládací panel „My data“ umožňuje uživateli práci s nahraným souborem.

5.2 Závěr z hodnocení výsledků

Častý výskyt spojení “nahrání vlastních souborů ” by mohl navodit pocit, že je aplikace schopná přijímat data například z vlastní domácí meteorologické stanice. Tuto možnou domněnku je nutné uvést na pravou míru. Nahráním vlastních dat je míněno to, že lze uživatelsky nahrát datové soubory stažené z některého ze serverů, který poskytuje data v podporovaném formátu. Nahrání dat z domácích meteostanic by mohlo být zajímavé, ale aplikace je vytvořena za účelem pozorování počasí na rozsáhlejším území, takže by ani data z jednoho místa nemělo smysl vizualizovat.

Nahrávání historických dat o počasí může mít význam například pro pojišťovny. Instituce poskytující pojištění proti přírodním živlům mohou aplikaci využít pro likvidování nahlášených pojistných událostí. Z historických dat je možné určit, jaké bylo počasí v době vzniku řešené události. Primární využití uploadu souborů je pro účely Ústavu informatiky AV ČR. Instituce, zabývající se počasím, vlastní numerický model, ze kterého generuje meteorologická data a ty je možné zobrazovat v aplikaci. Vizualizace dat prostřednictvím webové aplikace slouží k ukázkám na interních poradách nebo jako prezentace na konferencích.

6 Závěr

Výsledkem bakalářské práce je funkční online řešení, které umožňuje prezentaci meteorologických dat a uživatelské nahrání souborů.

Na základě teoretických poznatků, získaných studiem odborných informačních zdrojů a zpracováním literární rešerše, bylo možné získat nezbytně nutný teoretický základ pro přístup do problematiky vývoje webových aplikací.

V teoretické části práce jsou podrobněji analyzované techniky vývoje, které tvoří základ aplikace. Postupné studium HTML, CSS a JavaScriptu zaručovalo potřebné informace pro pozdější realizaci prostředí na klientské straně. Hlubší průzkum JavaScriptu usnadnil vypracování přehledu vývoje dynamických technologií na straně klienta, které přispělo k realizaci moderního webového řešení.

Nutnost vývoje vlastní aplikace potvrdil i výsledek analýzy, která byla provedena. Vzhledem k tomu, že pro parametry, které byly stanoveny zadavatelem pro vývoj aplikace, neexistuje veřejně dostupné řešení, přistoupilo se k samotné realizaci. Vývoj reálné aplikace je založen na základu open-source řešení, které je rozšířené o mnohou funkcionalitu. Jádro webové aplikace tvoří Node.js.

Výsledná aplikace podlehla zhodnocení a svou stavbou i funkcionalitou plně odpovídá stanoveným požadavkům, které byly zvoleny před samotnou realizací. Jako jednoznačný přínos práce může být vnímána možnost zobrazení libovolných historických záznamů o počasí v online webové aplikaci. Řešení slouží zadavateli a v budoucnu může být využito pro potřeby jiných institucí s podobným zamšřením.

7 Seznam použitých zdrojů

7.1 Seznam literatury

BASSET, L., Introduction to JavaScript Object Notation, A To-the-Point Guide to JSON. USA: O'Reilly Media, 2015. ISBN 978-1-4919-2942-1.

BROWN, E. Web development with Node and Express. Boston: O'Reilly Media, 2014. ISBN 978-1-4919-4930-6.

BROWN TIFFANY, B., BUTTERS, K., PANDA, S. HTML5 Okamžitě. Brno: Computer Press, 2014. ISBN 978-80-251-4296-7.

CASTRO, E., HYSLOP, B. HTML5 a CSS3 Názorný průvodce tvorbou WWW stránek. Brno: Computer Press, 2012. ISBN 978-80-251-3576-1.

DEWAR, M., Getting Started with D3. USA: O'Reilly Media, 2012. ISBN 978-1-449-32879-5.

FLANAGAN, D. JavaScript Kompletní průvodce (2. aktualizované vydání). Brno: Computer Press, 2002. ISBN 80-7226-626-8.

GASSTON, P. Průvodce vývojáře moderního webdesignu. Brno: Computer Press, 2016. ISBN 978-80-251-4641-5.

HOLZNER, S. JavaScript: profesionálně: [kompletní referenční příručka]. Praha: Mobil Media, 2003. ISBN 80-86593-40-1.

LAZARIS, L. CSS Okamžitě. Brno: Computer Press, 2014. ISBN 978-80-251-4176-2.

LUBBERS, P. a kol. Programujeme moderní webové aplikace. Brno: Computer Press, 2011. ISBN: 978-80-251-3539-6.

NGYUYEN, D., Node JS okamžitě. Brno: Computer Press, 2016. ISBN 978-80-251-4820-4.

ODELL, D. JavaScript průvodce programováním ajaxových aplikací. Brno: Computer Press, 2010. ISBN 978-80-251-2733-9.

PEHLIVANIAN, A., NGUYEN D., JavaScript okamžitě. Brno: Computer Press, 2014. ISBN 978-80-251-4163-2.

RESIG, J. JavaScript a Ajax Moderní programování webových aplikací. Brno: Computer Press, 2007. ISBN 978-80-251-1824-5.

SMITH, B., Beginning JSON. USA: Apsress, 2015. ISBN 978-1-4842-0202-9.

SUEHRING, S. JavaScript : krok za krokem. Brno: Computer Press, 2008. ISBN 978-80-251-2241-9.

TEIXEIRA, P. Professional Node.js: Building JavaScript Based Scalable Software. United States: John Wiley & Sons Inc., 2012. ISBN 978-1-118-18546-9.

THAU, D. Velký průvodce JavaScriptem: tvorba interaktivních webových stránek v praxi. Praha: Grada, 2009. ISBN 978-80-247-2211-5.

THOMAS, S.A., Data Visualization with JavaScript. San Francisco: No Starch Press, Inc., 2015. ISBN 978-1-59327-605-8.

WANDSCHNEIDER, M. Learning Node.js: A Hands-On Guide to Building Web Applications in JavaScript. Boston: Addison Wesley, 2013. ISBN 978-0-321-91057-8.

ZAKAS, N. JavaScript pro webové vývojáře. Computer Press, 2009. ISBN 978-80-251-2509-0.

7.2 Seznam obrázků

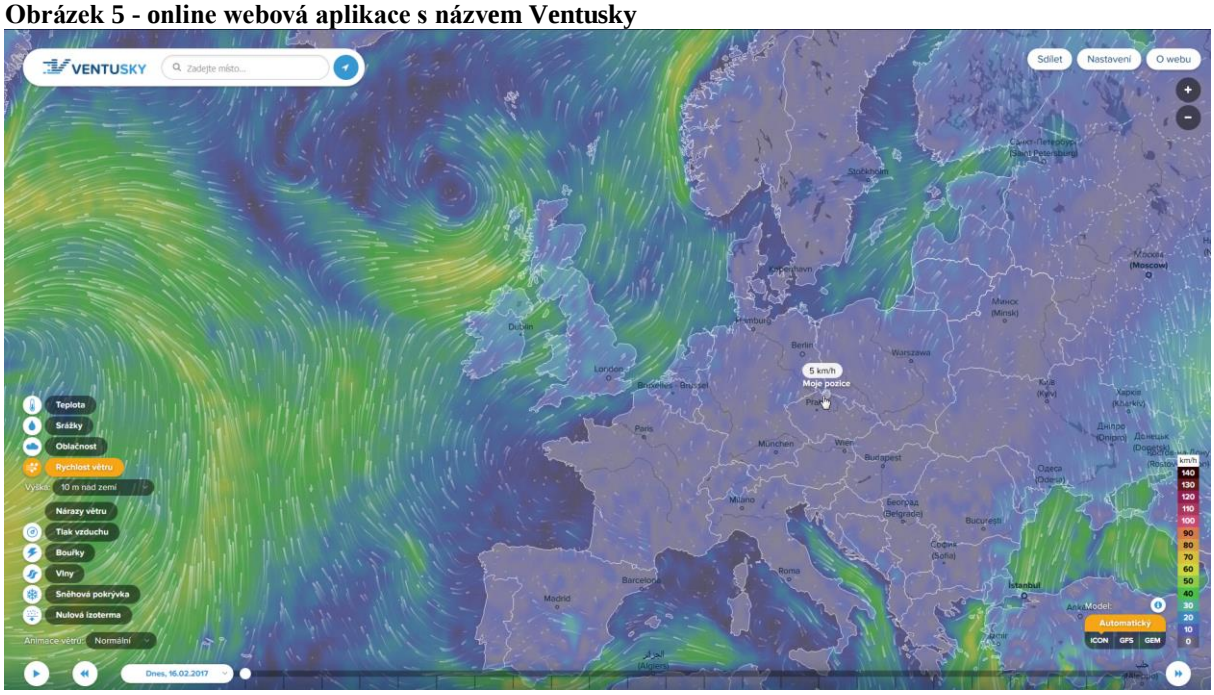
Obrázek 1- znázornění dominantních technologií používaných v oblasti webových aplikací. 13
Obrázek 2 - ukázka rozmanitých typů zařízení s různými typy displejů 19
Obrázek 3 - schéma znázorňující komunikaci mezi koncovými zařízeními, web serverem a databází 26
Obrázek 4 - ukázka vlastní aplikace, aktivní panel pro nahrání souboru a zobrazená data dle aktuální polohy 42
Obrázek 5 - online webová aplikace s názvem Ventusky 54
Obrázek 6 - online webová aplikace s názvem YR 54
Obrázek 7 - online webová aplikace s názvem Windguru 55
Obrázek 8 - online webová aplikace s názvem Meteo Earth 55
Obrázek 9 - online webová aplikace s názvem Windy TV 56
Obrázek 10 - desktopová aplikace pro OS Windows s názvem Panoply 56
Obrázek 11 - desktopová aplikace pro OS Windows s názvem ZyGrib 57

7.3 Seznam tabulek

Tabulka 1 - vícekriteriální analýza variant na vybraném souboru zkoumaných aplikací 36

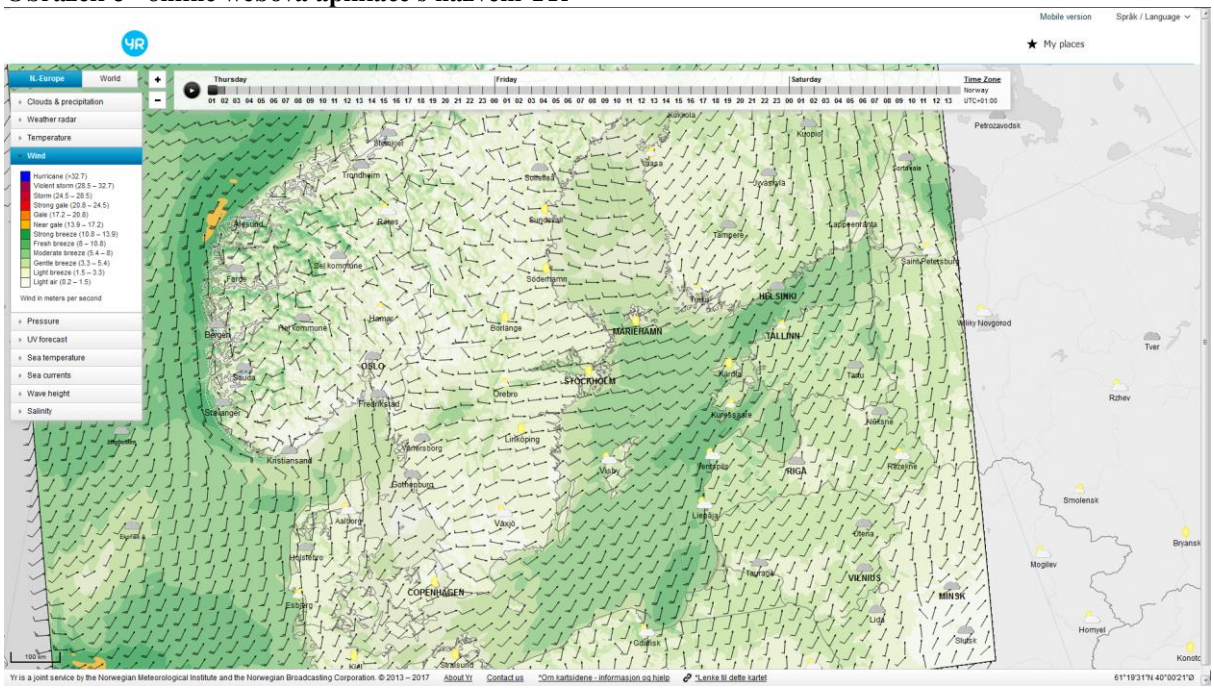
8 Přílohy

Obrázek 5 - online webová aplikace s názvem Ventusky



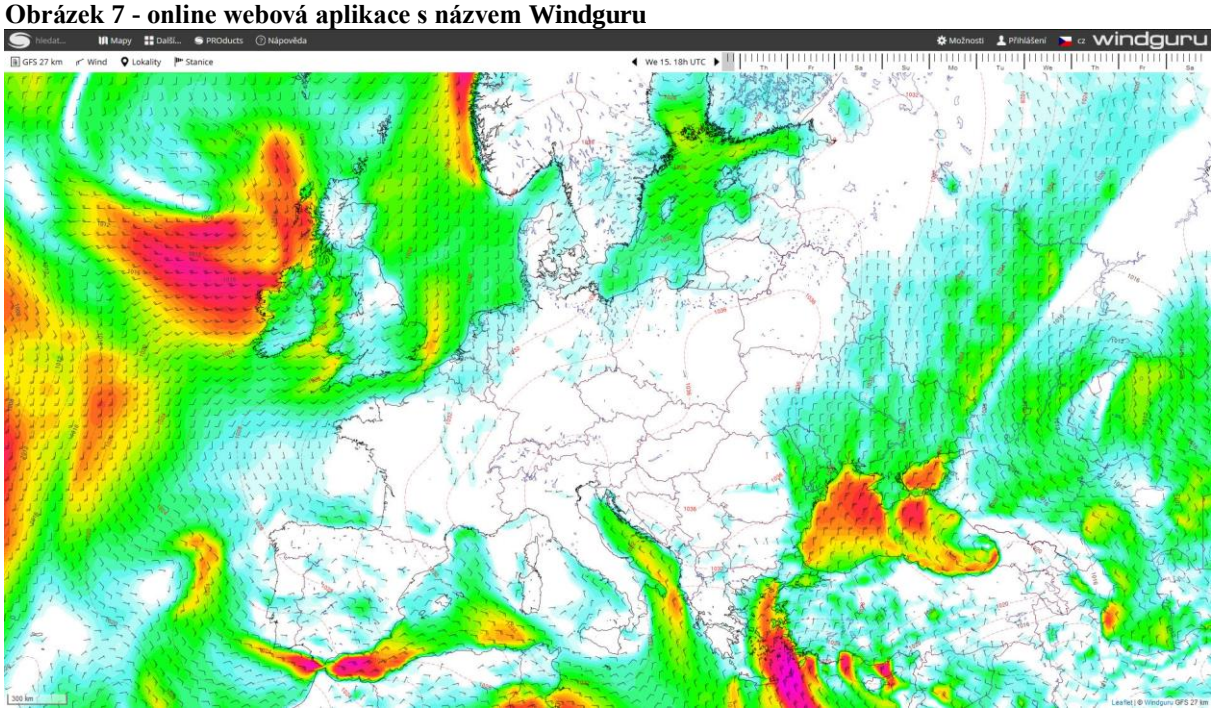
Zdroj: https://www.ventusky.com

Obrázek 6 - online webová aplikace s názvem YR



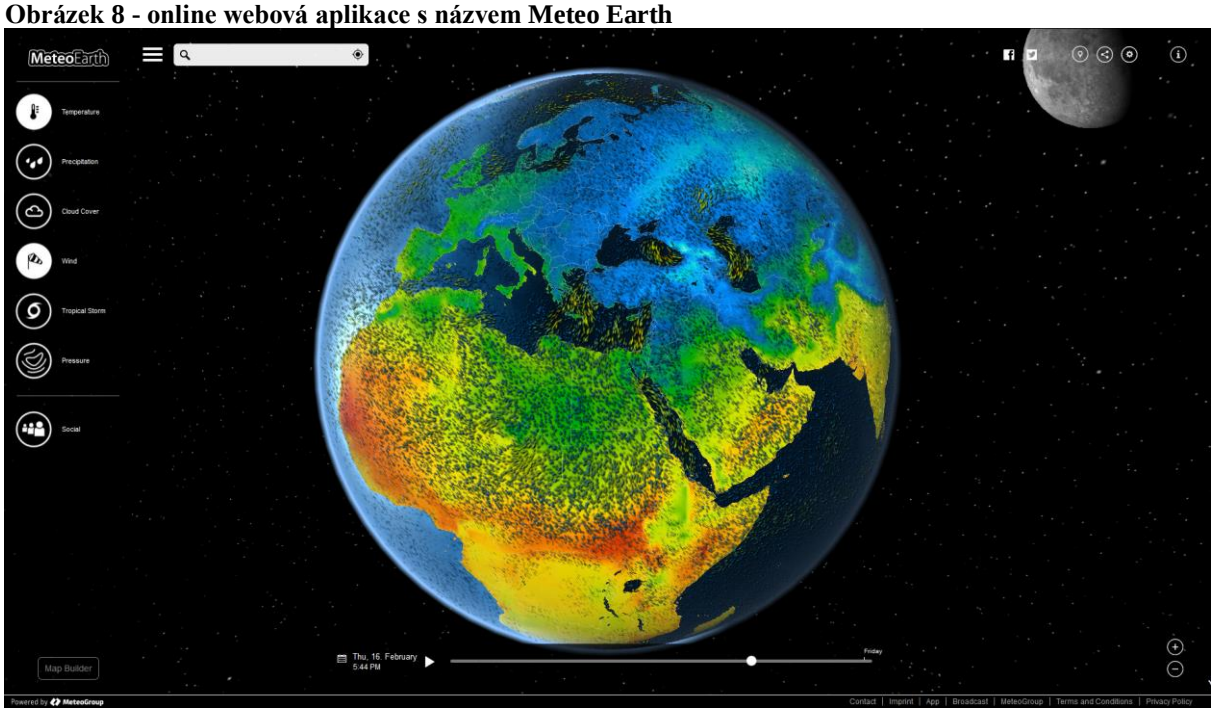
Zdroj: https://www.yr.no/

Obrázek 7 - online webová aplikace s názvem Windguru



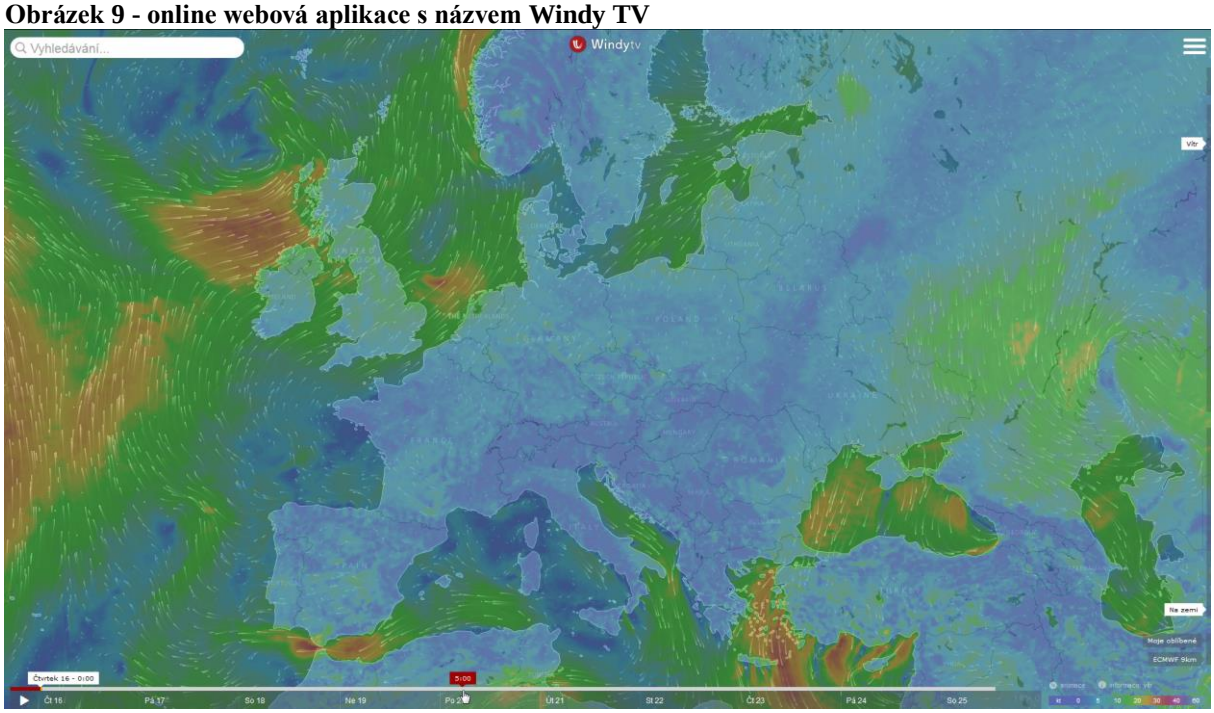
Zdroj: https://www.windguru.cz/

Obrázek 8 - online webová aplikace s názvem Meteo Earth



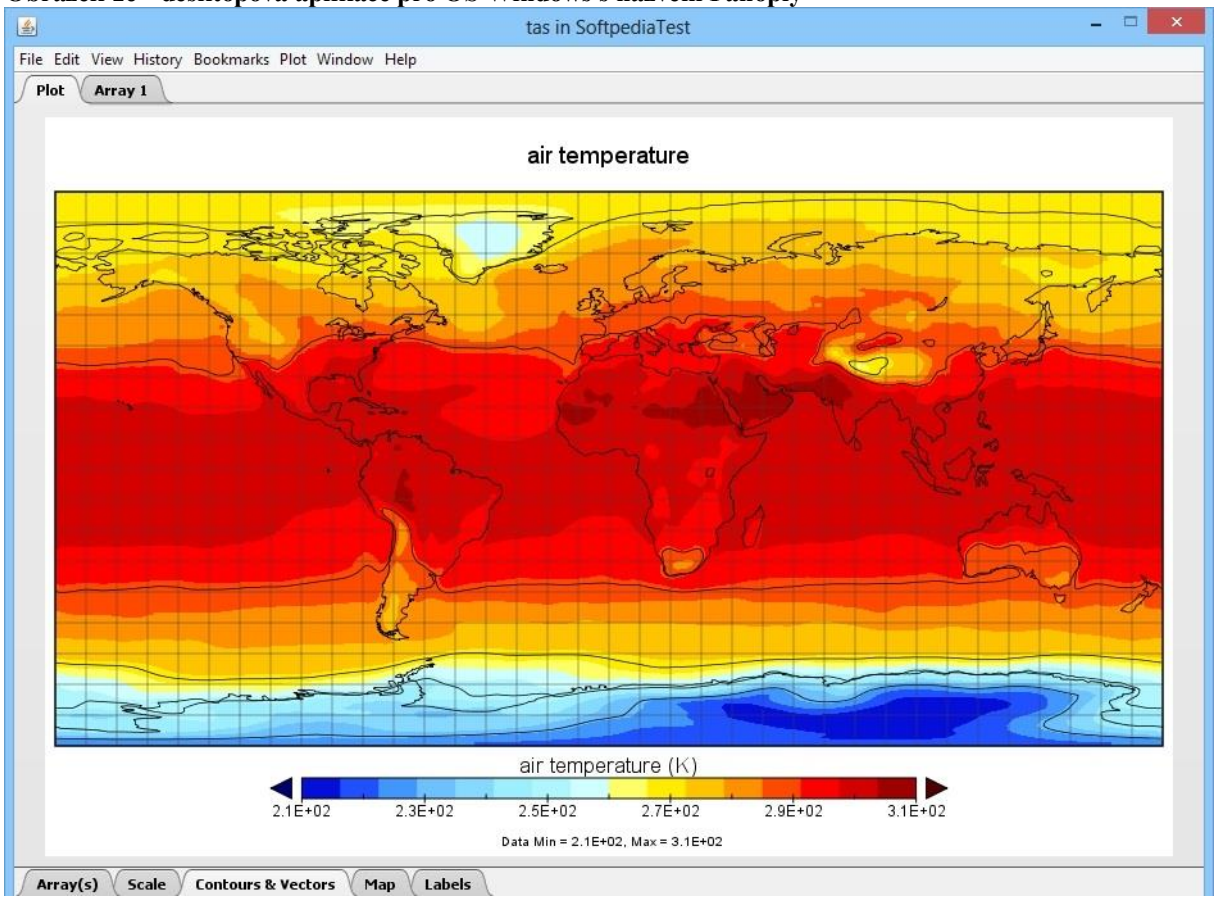
Zdroj: http://www.meteoearth.com/

Obrázek 9 - online webová aplikace s názvem Windy TV



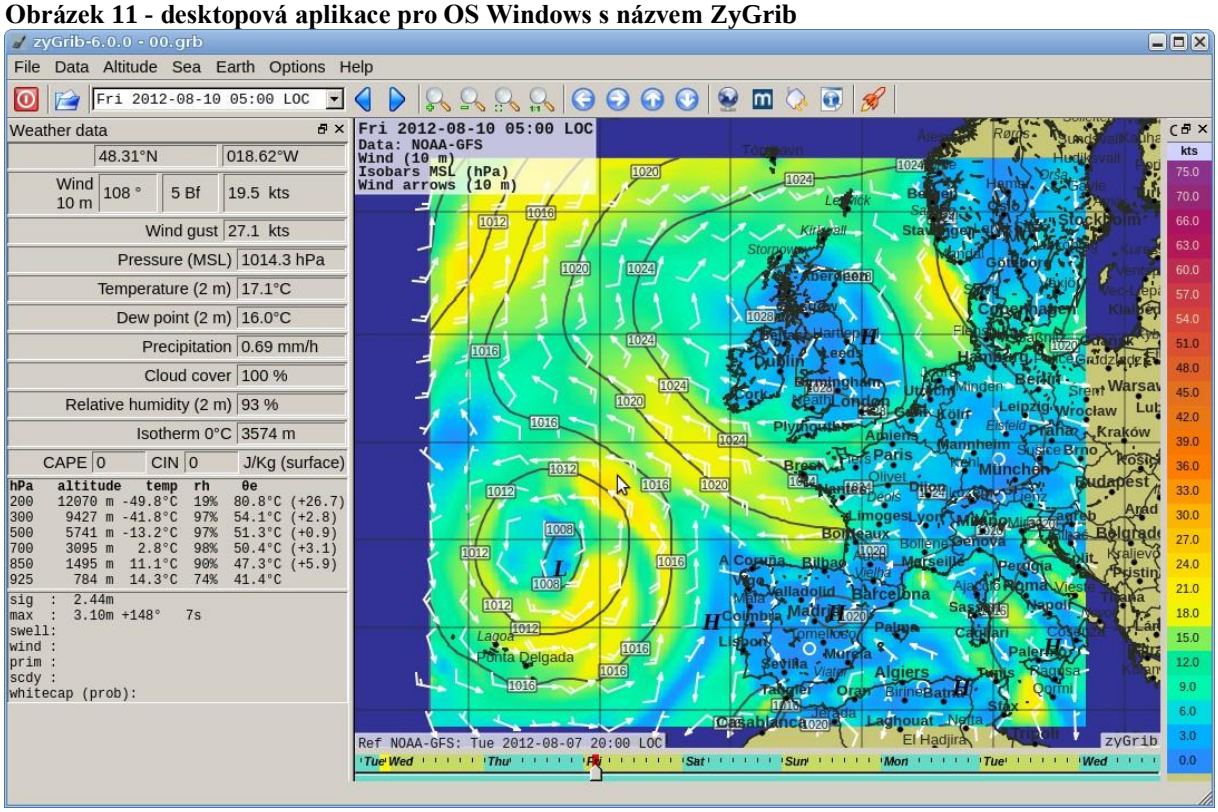
Zdroj: https://www.windytv.com/

Obrázek 10 - desktopová aplikace pro OS Windows s názvem Panoply



Zdroj: Náhled na aplikaci Panoply v lokální instalaci

Obrázek 11 - desktopová aplikace pro OS Windows s názvem ZyGrib



Zdroj: Náhled na aplikaci ZyGrib v lokální instalaci