VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ NÁSTROJ PRO ANOTACI OBRAZOVÝCH DAT

WEB-BASED IMAGE ANNOTATION TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DAVID DVOŘÁČEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Dvořáček David**
Program: Informační technologie
Název: **Webový nástroj pro anotaci obrazových dat**
Web-Based Image Annotation Tool
Kategorie: Zpracování obrazu
Zadání:

1. Seznamte se s principy vývoje webového nástroje za použití knihovny Flask (python).
2. Navrhněte webový nástroj pro anotaci obrazových dat, zejména pro vkládání anotací typu - bod, přímka, tah, obdélník, polygon
3. Zaměřte se na modularitu a lehkou rozšířitelnost tohoto nástroje pro různé typy anotací.
4. Implementujte funkce pro manipulaci s obrazem (zoom, posunutí) a funkce pro transformaci obrazu (jas, kontrast, atd.)
5. Iterativně vyvíjejte a testujte vytvořený nástroj na jeho uživatelích.
6. Vytvořte plakát a video prezentující vaší práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Špaňhel Jakub, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Tato práce se zabývá tvorbou webového nástroje pro anotaci obrazových dat. Teoretická část specifikuje aplikaci, její design a funkcionalitu. Praktická část se zabývá implementací webového nástroje pro anotaci obrazových dat jako bod, přímka, obdelník, polygon se zaměřením na modularitu a lehkou rozšířitelnost nástroje pro ruzné typy anotací a dále implementací funkcí pro manipulaci s obrazem a jeho transformaci. Pro praktickou část této práce byla použita knihovna Flask s využitím jazyků Python, HTML, Javascript. Nástroj byl vytvářen a vyvíjen iterativně.

Abstract

This work deals with the creation of web tool for image data annotation. The theoretical part specifies the application, its design and functionality. The practical part deals with the implementation of the web tool for image data annotation such as point, line, rectangle, polygon with focus on modularity and easy extensibility of the tool for various types of annotations and implementation of image manipulation and transformation functions. For practical part of this work was used library Flask using Python, HTML, Javascript. The tool was created and developed iteratively.

Klíčová slova

Webový nástroj, anotace, anotace obrazových dat, manipulace s obrazem, transformace obrazu, uživatelské testování, Flask, Python, HTML, Leaflet, Leaflet, Javascript

Keywords

Web tool, Annotation, Image data annotation, Image manipulation, Image transformation, User testing, Flask, Python, HTML, Leaflet, Javascript

Citace

DVOŘÁČEK, David. *Webový nástroj pro anotaci obrazových dat.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Webový nástroj pro anotaci obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jakuba Špaňhela.

Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
David Dvořáček
27. května 2020

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce panu Ing. Jakubu Špaňhelovi za kvalifikované vedení, vstřícný přístup, cenné rady a zpětnou vazbu.

Obsah

Kapitola 1

Úvod

V současnosti mnoho lidí, firem i státních organizací využívá prostředí internetu pro svou osobní i pracovní potřebu. Potřebují vytvářet, zpracovávat, kontrolovat, ukládat svá data, komunikovat se svým okolím, seznámit se s informacemi, co se děje ve světě, nakupovat s pomocí internetu a webových aplikací. Uživatelská příznivost, design webových stránek a aplikací patří k jednomu z nejdůležitějších motivátorů k jejich návštěvnosti a četnosti využití, čímž se zvyšuje jejich hodnota. Jednodušší intuitivní aplikace lidi nezatěžují a mohou je využívat všichni bez věkového omezení. V opačných případech složitost aplikace lidi odradí od práce s ní. Aplikace jsou různé a odlišují se podle toho, k jakému účelu slouží.

V průběhu celé své práce, zejména v první části, chci seznámit s informacemi souvisejícími s touto prací jako jsou webový nástroj, anotace, grafická data, WEB, webová aplikace, dále s použitými metodami a postupy (programování – frontend a backend, vizualizace a serializace), jakými nástroji byla aplikace vytvořena a pomocí jakých jazyků (Python, Javascript, HTML).

Při spouštění každé aplikace nelze opominout aplikační bezpečnost, protože každá aktivita uživatele může pro něj znamenat určitou hodnotu. Narušení bezpečnosti webové aplikace by mohlo uživateli způsobit poškození či trvalou škodu. Z těchto důvodů se zabývám v práci i bezpečností aplikace a možnými útoky.

Cílem této práce je vytvořit webový nástroj jako aplikaci, která by umožnila anotovat obrázky podle uživatelské konfigurace, například vozidlo pomocí tahu myší, město pomocí bodu, vzdušnou vzdálenost bodů na mapě pomocí přímky. Aplikace umožní uživateli obrázky či segmenty nahrát do aplikace, anotovat je a následně uložit s možností jejich podrobnějšího popisu. Z detailu obrázku člověk může zjistit bližší informace o daném objektu jako registrační značku vozidla, typ vozidla, popis člověka, popis okolí, místo výskytu objektu na obrázku. Tyto informace pak může využít pro trénování neuronových sítí. Po vytvoření takovou aplikaci otestovat na její funkčnost a intuitivnost.

Kapitola 2

Rešerše

2.1 Webový nástroj

Webové nástroje umožňují sdílet analýzu s ostatními na portálu. Data jsou ukládána a jejich zpracování probíhá na serveru. Aplikace tak mohou provádět analýzu. Webový nástroj se může skládat z jednoho či více nástrojů s využitím dat v klientské aplikaci. Nástroje data zpracovávají a zpět vrací funkční výstupy formou map, sestav nebo souborů jako vlastní model nebo skriptovací nástroje sdílené na portálu. Protože webový nástroj může odkazovat na jakýkoli geoprocesní nástroj, možnosti pro něj jsou nekonečné. Může provádět např. tyto úkony:

- výpočet pravděpodobné oblasti evakuace pro únik nebezpečné chemikálie
- výpočet předpokládané dráhy a síly hurikánu
- vytvořit zprávu o krajinném pokryvu a půdách v určeném povodí
- vytvořit pozemkovou mapu s historickými podrobnostmi o vlastnictví
- geokódovat adresu a vstoupit do povolovací aplikace pro systém renovace domů

Hlavním rozdílem mezi webovými nástroji a nástroji ve stolním počítači je to, že když spustíte webový nástroj, spustí se na serverovém počítači pomocí prostředků serverového počítače, na rozdíl od stolního počítače.[6]

2.2 Anotace

Anotace znamená poznámku, či její připojení, vysvětlivku a stručnou charakteristiku díla.[4] Ruční anotace vzniká před metodami založenými na strojovém učení, přičemž s vyšším množstvím ručních anotací lze aplikovat s vyšší úspěšností metody strojového učení. Problémem ruční anotace je selhání lidského faktoru, jako chybovost nebo nesprávné pochopení úlohy, čímž vznikají nesoulady, rozdílnosti.

Anotace, co nejpřesněji označí dané údaje, jako obrázek, video atd. pro další referenční účely. Provádí to tak, že přiřadí nějaké klíčové slovo pro konkrétní oblast textu, obrázku. Tato myšlenka se obvykle používá v oblasti pro účely výcviku strojů. Umělá inteligence představuje pole velkého rozsahu zaměřující se na vývoj inteligentních strojů pro různé úkoly.

Anotace obrázků

Technologie anotace obrázků nám pomáhá v dnešní časově náročné době zefektivnit práci s obrázky (např. fotkami), kterých je velké množství, a proto je potřeba práci s nimi uspořádat, zorganizovat, obnovovat a jinak s nimi pracovat. K vyhledání jedinečné sady obrázků lze využít technologii anotace obrázků, která využívá metody přidávání metadat, jako jsou titulky, klíčová slova nebo popisy k obrázkům.[7] Ty pak lze použít k vyhledávání díky anotačním slovům, např. na mapě: vodní toky, hory, města atd. Vzniká tak klasifikace s více štítky, která umí přiřadit více štítků k jednotlivým datům.

V dnešní době tato technologie využívá aplikaci technik počítačového vidění pro systémy, pro vyhledávání obrázků k uspořádání a lokalizaci obrazů zájmu z databáze. Objekt, který se může vyskytnout v různých situacích, lze zatřídit do více kategorií, avšak nelze odhlédnout od složitých korelací, které existují mezi různými štítky. Automatická anotace obrazu je schopna klasifikovat více štítky a spojit sadu textu s obrázkem popisujícím jeho sémantiku. Díky tomu je využívána k získávání a zobrazování relevantních obrázků z celosvětového webu uživateli na základě klíčových slov. Získané obrázky jsou využívány v téměř každém pracovním prostředí, jak při rozvoji, zpracovávání, tak i při ochraně.

Anotace obrázku přiřazuje metadata do obrázku. Metadata jsou různá data obrázku specifikována klíčovým slovem (datem). Například na obrázku má být označen strom. Strom je boxován s určitým obrysem a označen jako strom. Tato metadata jsou přiváděna do algoritmů. Na základě algoritmů existují různé typy anotací obrázků jako:

- Ohraničení anotace rámečku
- Anotace mnohoúhelníku
- Bodová anotace
- Řádková anotace
- Polygonální anotace
- Sémantická anotace
- Anotace klíčového bodu
- Video anotace

Pro pochopení obrázků je důležité jejich označení a prezentace v jazyce, v němž je schopen je počítač přečíst, čímž vznikne počítačové vidění, anotace obrazu. Anotace obrázku je tedy:

- jeho označení lidmi tak, aby byl čitelný pro stroje
- proces označování objektů v celém obrazu lidmi

Návrh anotovaných obrázků

Anotace k obrázkům popisují obsah obrázku. Jsou užitečné při jednoduchém vyhledávání obrázků. Například Google vyhledává obrázky používá pro vyhledávání obrázků klíčová slova – text na stránce, kde se obrázek nachází. Chybovost pak spočívá v tom, že text mnohdy nesouvisí s obrázkem. Snadno se uživatel vyhledá obrázek, který nesouvisí s danou

tématikou, nebo nenajde obrázek vůbec. Proto jsou přesné anotace k obrázkům tak užitečné a důležité.

Tato označená data jsou široce používána k trénování autonomních modelů vnímání jízdy pro chodce, dopravních značek, překážek v pruhu atd.

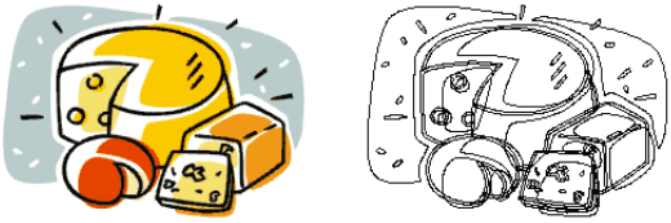
2.3 Grafická data

Grafická data se dělí na data vektorová a rastrová, mezi nimiž je rozdíl ve způsobu popisu objektu.

Vektorová využívají k popisu geometrických parametrů a rastrová data popisují shluk bodů, např. úsečka je popsána jako množina bodů o různých souřadnicích, které dohromady tvoří obraz úsečky. Ve vektorovém zápisu bude obsahovat pouze počáteční a koncový bod a data o barvě a šířce.[2]

2.3.1 Vektorová data

Vektorový grafický soubor obsahuje informace o objektech složených z křivek, jednoduchých těles umožňující geometrickou konstrukci. Ohledně kružnice soubor neobsahuje informace o všech bodech, co na ní leží, avšak uvádí, že se jedná o kružnici se souřadnicemi jejího středu a jednoho bodu, který na ní leží. Poslední bod určuje rovinu její konstrukce. Další informace jsou o barvě, tloušťce čáry k sestrojení. Program musí obsahovat algoritmus pro sestrojení. Typickými vektorovými daty jsou technické výkresy.[3]



Obrázek 2.1: Obrázek je popsán pomocí vektorových dat jako soustava křivek[2]

Výhody:

- změnou měřítka nedochází ke ztrátě informace
- výstupní soubory jsou poměrně malé
- vhodné na tvorbu logických map, tady obrázků využívaných v různých rozměrech
- jednoduché kreslení
- výstupní soubory: .eps, .pdf, .pict (Mac).

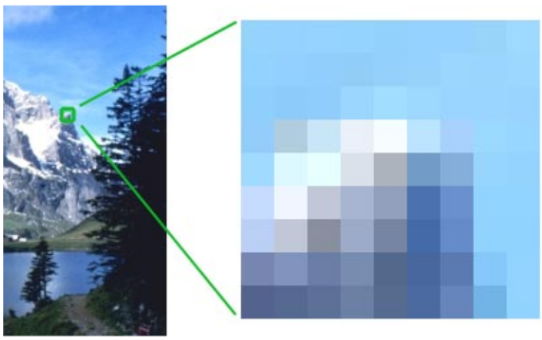
Nevýhody:

- při velkém zmenšení mohou zmizet některé čáry
- při velkém zvětšení jsou vidět chyby v kresbě

2.3.2 Rastrová data

Rastrová data dohromady tvoří obrázek. Skládající se z matice jednotlivých teček (pixelů), které mají svou vlastní barvu. Mnohdy bývají k dispozici data o kompresi či kódování barev.

Rastrově jsou ukládána data, která již nebudou upravována systémem, kterým byla vytvořena (např. žánrový pohled na součást stroje), nebo obrazy, které nebyly pořízeny počítačem (např. fotografie). Na rastrovém principu funguje většina zobrazovacích zařízení (tiskárny, televize, monitory). Na obrázku 2.2 je rastrový obrázek, kde je v detailu vidět složení ze čtverců s barvou. Lidské oko nedokáže rozlišit jednotlivé čtverce, proto působí výsledek spojitě. Počet pixelů je závislý na tom, v jaké velikosti chceme obrázek tisknout či prezentovat. Pro porovnání rozlišení existuje jednotka DPI-Dots Per Inch (počet bodů na palec).



Obrázek 2.2: Rastrový obrázek[2]

Rastrové obrázky mohou obsahovat libovolný počet barev. Rozlišujeme 4 základní kategorie:

- line art – obsahují dvě barvy – černou a bílou, každý pixel je definován pouze jedním bitem (on=černá, off=bílá)
- odstíny šedi – obsahují paletu stínů šedé, také čistou černou či bílou
- tónované – obsahují odstíny dvou či více barev, nejoblíbenější z černé a druhé barvy
- plně barevné – informace popsána užitím množství barevných prostorů RGB, CMYK apod.

Výhody:

- jednoduché pro výstup, pokud má tiskárna dostatečnou paměť
- výstupní soubory: .eps, .gif, .jpg, .tiff, .pict (Mac)

Nevýhody:

- velká paměťová náročnost (A4 formát středí kvalitně bez komprese má 40 MB)
- nelze libovolně zvětšovat, dochází ke čtvercování
- při zmenšení ztrácí ostrost

Obrázek 2.3: 1. Line-art, 2. Odstíny šedi, 3. Tónované, 4. Plně barevné[2]

2.4 Web

Web, www (World Wide Web), je celosvětovou soustavou dokumentů dostupných na internetu na témž webovém serveru nebo internetové doméně nejnižšího stupně.

Koncem 80. let minulého století se v CERNu vyvíjel Tim Berners Lee hypertextový značkovací jazyk HTML a tím spustil velký pokrok v informatizaci světa. První webová stránka vznikla počátkem 90. let.

Web tvoří hlavní službu poskytovanou v rámci světové sítě internet a jeho hlavní princip spočívá v propojování webových stránek pomocí takzvaných linků (hyperlinků), které jsou reprezentovány adresou URL (uniform resource locator).

2.4.1 Povrchový web

Povrchový web tvoří běžně využívané webové stránky (noviny, eshopy, profily firem, wikipedie apod.). Sbírá indexované informace pomocí speciálních robotů (sběračů) jako např. vyhledávače Seznam, Google, Bing a další. Jejich cílem je prohledat obsah vzniklých webových stránek a zobrazit jej přes vyhledávání uživatelům. Pouze uživatel může posoudit kvalitu informací, která je mnohdy prezentována stránkami se spamy, reklamou aj. Jedná se proto o velmi složité prostředí. Problémem jsou například sociální média, která generují obrovské množství informací a mnohdy právě dezinformací, které se poté lavinově šíří napříč společností. Jedním z hlavních šiřitelů je Hoax.cz, který je vlastně největší českou databází nesmyslů, poplašných, řetězových a lživých zpráv, které jsou lidmi často sdíleny dále.[29]

2.4.2 Hluboký web

Vzhledem k tomu, že roboti povrchového webu se nedostanou do hloubky, existuje web s těžko dostupným nebo nedostupným obsahem (např. vědecké články, studie, patentové dokumenty, normy, obchodní informace), a tím je hluboký web. Hluboký web tvoří neindexované a těžko dostupné informace. Je pravdou, že např. Google Scholar rovněž zpřístupňuje vědecké články nebo patenty, nicméně knihovny, či univerzity je za prvé zpřístupňují oněm robotům a za druhé, jedná se pouze o zlomek skutečného množství těchto dokumentů. Velikost povrchového webu přesáhla v roce 2014 jednu miliardu webových stránek, což je malým zlomkem hlubokého webu a jeho velikosti. Kvalitní, cenné informace lze najít v oblasti hlu-

bokého webu, kam se lze dostat pouze za pomocí znalosti příslušných informačních zdrojů a specifických vyhledávacích postupů.

2.4.3 Temný web

Temný web (též temný internet) bývá pojmenován jako darkweb nebo darkinternet. Je to ta část webu, která je sice veřejnosti přístupná, ale již za pomocí sofistikovaných nástrojů (prohlížečů) s primární funkcí anonymního pohybu. To samozřejmě není samoúčelné, protože temný web dnes slouží především k nelegální činnosti a obchodu.

2.5 Webová aplikace

Webová aplikace je aplikací poskytovanou uživatelům z webového serveru přes počítačovou síť Internet nebo vnitropodnikovou síť Intranet prostřednictvím webového prohlížeče jako tenkého klienta, který nezná logiku aplikace. Jsou oblíbeny a využívány k implementaci informačních systémů jako freemailů, internetových obchodů, online aukcí, webblogů. Jsou charakteristické způsobem komunikace s uživatelem přes webový prohlížeč napříč celým světem přes internet. Rozšiřováním internetu jsou přístupné velkému množství uživatelů. Sestávají z modelu klient x server, přičemž v současnosti jsou třívrstvé, a to:

- Vrstva databázová – relační databáze
- Vrstva aplikační
- Vrstva prezentační – webové technologie

Výhody:

- široká dostupnost webové aplikace
- možnost pružně reagovat na potřeby zákazníka
- centralizace dat na jednom serveru
- jednoduchá správa aplikace
- lze spustit na mobilech, tabletech i na stolních počítačích

Nevýhody:

- bezpečnost – hackerské útoky, únik a ztráta dat aj.
- závislost na výpadcích a kvalitě sítě
- závislost na rychlosti připojení k internetu
- její běh ve webovém prohlížeči je méně efektivní než běh desktopové aplikace

Historie

U dřívějších typů klient – server obsahovala aplikace klientský program poskytující uživatelské rozhraní na osobním počítači. Aktualizace klientských programů probíhala na každé pracovní stanici, což bylo nákladné a neefektivní. Nové webové aplikace vytváří webové stránky ve formátu HTML/XHTML podporován běžnými prohlížeči (univerzální klient). Webová stránka funguje jako statický dokument, ale jejich sled může díky reakci na vstup uživatele vyvolat interaktivitu. K tvorbě dynamických prvků je využíván skriptovací jazyk JavaScript. Nová verze jazyka HTML5 je více zaměřená na podporu vývoje webových aplikací.

Rozhraní

Webové rozhraní mnohdy omezuje možnosti klienta. Pro přidání funkčnosti a pocitu interaktivity se často využívá skriptování na straně klienta s obnovou stránky, což způsobuje rušení . Některé technologie umožňují spolupráci skriptů na klientské straně se serverovou částí aplikace bez nutnosti obnovovat stránku, např. AJAX: technika vývoje webu využívající kombinaci HTML, JavaScriptu a rozhraní XMLHttpRequest, které načítají klientským skriptům informace ze serveru bez obnovy celé stránky.

Technické aspekty

Výhodou vývoje je schopnost pracovat podle určení, bez ohledu na operační systém či verzi na klientském počítači. Implementace HTML, CSS, DOM je problematická z důvodu možnosti uživatelů nastavit způsob zobrazení ve svém prohlížeči (např. jiný řez či velikost písma, barvy či vypnout podporu skriptování) znehodnocující vzhled aplikace. Většina prohlížečů podporuje použití Adobe Flash či javových appletů pro část nebo celé uživatelské rozhraní formou zásuvných modulů. Poskytují větší kontrolu vývojářům nad uživatelským rozhraním, ale neřeší problémy s nastavením prohlížečů. Komplikace dělá i rozdílnosti mezi implementacemi Flashe či Javy. Nevýhodou je vysoká závislost na poskytovateli a nutnost stabilního silného připojení k serveru poskytovatele. Při ukončení služby poskytovatelem ji nelze nadále použít, na rozdíl od lokálně provozovaného software. Nedostupnost služby vznikne i přerušením spojení se serverem poskytovatele. Výhodou je prakticky nulová údržba a minimální náklady (pevné měsíční či roční poplatky).

Struktura

Webové aplikace jsou obvykle strukturovány jako třívrstvé (three-tier architecture).[16] Jednotlivé vrstvy jsou:

- První (prezentační) vrstva zobrazuje data (vizualizuje) prováděná klientem (tlustší nebo tenčí klient) uživateli formou grafického uživatelského rozhraní, může kontrolovat zadávané vstupy, neobsahuje však zpracování dat (webový prohlížeč).
- Střední aplikační (logickou) vrstva je jádrem aplikace, jedná se o nástroje pro dynamické generování stránek (např. logika a funkce, výpočty a zpracováním dat). Využívá aplikační skripty PHP, Java, .NET, Python, Ruby aj. a dále různá rámcová řešení (frameworky).

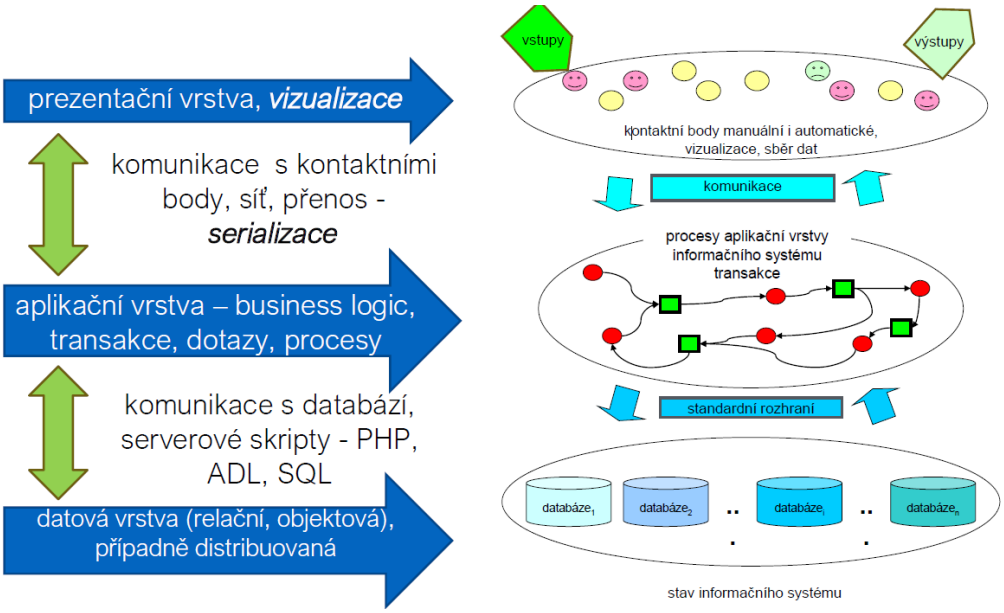
- Třetí (datovou) vrstvou je zpravidla databáze či síťový souborový systém, webové služby nebo jiná aplikace. Datovou vrstvu tvoří buď relační, nebo objektový databázový model.

Webový prohlížeč posílá požadavky střední aplikační vrstvě (HTTP, serializace), která je vyřizuje přes dotazy do datové vrstvy (databáze) generováním uživatelského rozhraní (standardizované databázové rozhraní SQL).

Pro bližší vysvětlení Tier je fyzickou vrstvou, tzv. jednotkou nasazení (deploymentem). Jedná se o fyzické členění systému na klient, aplikační server a databázový server, čemuž odpovídá volba technologií pro realizaci jednotlivých částí.

Logická vrstva Layer je jednotkou organizace kódu, která je řešena převážně v aplikační vrstvě a dělí se na tyto části:

- Data layer – komunikace s databází
- Business layer – implementace logiku aplikace
- Presentation layer – komunikace s klientem



Obrázek 2.4: Struktura webové aplikace[16]

Obchodní využití

Za používání vytvořených webových aplikací účtují jejich poskytovatelé (application service provider, ASP) měsíční či roční poplatek. Výhodou je, že tyto aplikace si nemusí uživatelé instalovat na svůj pevný disk.

2.6 Programování webových aplikací

Webové aplikace jsou psány v čistých programovacích jazycích jako PHP či Perl nebo jsou vytvořeny pomocí frameworků, které mohou díky jednoduchosti a přehlednosti snížit počet chyb.

Programovací prostředí

Tvorba webové aplikace probíhá ve dvou fázích, a to frontendu a backendu. Frontend je spuštěn u uživatele na internetovém prohlížeči a backend (server) uchovává získané anotace a synchronizuje komunikaci mezi více uživateli.

2.6.1 Frontend

Internetové prohlížeče podporují omezené množství technologií, proto je nutné zvolit takové, které jsou kompatibilní s co největším počtem zařízení a prohlížečů. Já jsem se rozhodl použít: Javasript, HTML5.

HTML 5

HTML (Hypertext Markup Language) je základním stavebním kamenem (značkovacím jazykem) tvorby hypertextových dokumentů.[12] Je označován příponami .html nebo .htm.

Obsahuje:

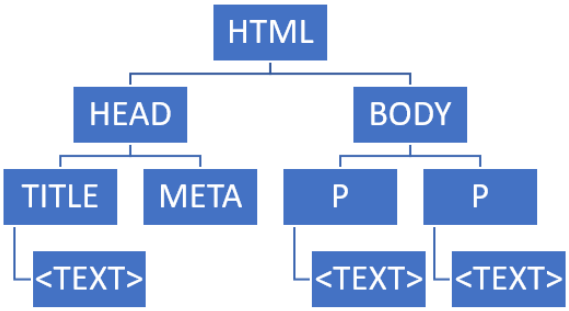
- Neformátovaný text s kódováním znaků UTF-8, v němž se větší počet mezer redukuje na jednu, konec řádku se převádí na mezeru a prázdné řádky se ignorují.
- Vložené značky, elementy a znakové entity; HTML element je úsekem dokumentu vymezeným značkami, má své jméno, atributy, obsah.
- Poznámky zůstávají v dokumentu a dají se využít k vložení příkazů pro server či poznámek autora.
- Znakové entity slouží pro vkládání speciálních znaků. Viz. obrázek 2.5

&lt;	<
&gt;	>
&amp;	&
&reg;	®
&copy;	©
&euro;	€
	nedělitelná mezera

Obrázek 2.5: Vkládání speciálních znaků[12]

Společně s CSS upravuje strukturu. Struktura se skládá se z hlavičky (head), těla (body) dokumentu a často ze zápatí. Následně se do ní přidávají prvky (elementy) HTML jazyka. Strukturu lze popsat jako strom elementů, v němž otcovský element činí HTML a synovské uzly tvoří elementy vnořené do otcovského.

Tělo definuje obsah (text, obrázky, multimédia, interaktivní prvky) a strukturu (sekce, podsekce, víceúrovňové nadpisy, odrážkované a číslované seznamky, tabulky, formuláře atd.)



Obrázek 2.6: HTML diagram

dokumentu. Vstupy dat jsou formuláře (texty, zaškrtávací pole, výběrové prvky) a výstupy dat jsou prezentace (texty, seznamy, tabulky). [25]

HTML určuje, jak má stránka vypadat, kolik objektů má na ní být umístěno a kde. Prohlížeče přečtou kód v jazyce HTML a zobrazí podle něj stránku. Nejnovější verzí je HTML 5, která již umí vkládat videa, zvuk do stránky. Dříve se musela použít technologie Flash. Užitečnou utilitou je element canvas, který slouží jako plátno k vykreslení různých obrazců (přímek, obrázků, kružnic apod.). Základní struktura HTML kódu je uvedena na obrázku 2.7.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Obrázek 2.7: Kostra HTML kódu

CSS 3

CSS, alias kaskádové styly, je jazyk, který slouží k uspořádávání elementů na webové stránce v souboru s HTML jazykem. Autorem CSS je Hawkom Lium Vie a navrženy byly organizací W3C. Kaskádové styly jsou propojeny s HTML jazykem pomocí tříd. Hlavním smyslem je tedy upravovat vzhled webové stránky bez změny struktury a obsahu.

Jeho úkolem je rozdělit prezentaci stránky do jiného dokumentu než do struktury. Jde o jednodušší zpracování dokumentů, zjednodušení tvorby stránek, protože CSS dokument

je většinou v jednom souboru a stanovuje vzhled pro všechny ostatní HTML dokumenty. Odpadá tak opakování stejných značek pro vzhled stránky ve všech HTML dokumentech. Nynější verze pracují s moderním designem a s tvorbou animovaných prvků, které byly dříve tvořeny pouze pomocí Javascriptu nebo Flashe.[23]

Obrázek 2.8: CSS 3 logo

JavaScript

Javascript je objektově orientovaný interpretovaný programovací jazyk pro zápis programů k webovým stránkách.[13] Jeho jádro je implementováno ve většině webových prohlížečů a určeno pro programování webu. Přikládá objekty určující obsah okna prohlížeče. Díky Javascriptu nejsou webové stránky pouze statické, ale umožňuje do nich vkládat spustitelný obsah.[19]

Vlastnosti:

- vychází z jazyků C a Java
- je interpretovaný
- je objektově orientovaný
- je case sensitive (citlivý na velká a malá písmena)

JavaScript nemá nic společného s Javou, je jí pouze syntakticky podobný.

Použití:

- dynamický obsah, tj. generování obsahu dokumentu podle různých okolností (vložení aktuálního času apod.)
- interaktivní práce s dokumentem, tj. reakce na události (pohyb myši, klik, změna hodnoty atd.) a změna vzhledu prvků
- ovládání prohlížeče, tj. pohyb v historii, otevírání oken, stavový řádek atd.

JavaScript může být vložený či externí skript. Vložený v hlavičce či těle dokumentu se spustí v době, kdy na něj prohlížeč narazí, a pak se zpracuje zbytek dokumentu. Výstup nejčastěji generuje HTML kód.

2.6.2 Backend

HTTP protokol

Zkratka HTTP znamená HyperText Transfer Protocol. Jedná se v podstatě o protokol přenášení hypertextu, přičemž hypertext je text s odkazy.[22] HTTP protokol popisuje

způsob komunikace prohlížeče se serverem, při němž se stahují stránky. Využívá se zejména u složitějších věcí a pokročilejších optimalizací pro vyhledávače, kde je důležité vědět o dění mezi severem a klientem. Informace o přesměrování, kešování, o cookies, o komprimaci či o referreru poskytnou HTTP hlavičky. Pro tvorbu serverových skriptů nebo složitějších webů je důležité znát, kdy, jak a kterou HTTP odpověď zaslat.

HTTP protokol funguje tak, že klient komunikuje se serverem. Klient zadává, co chce, a server na jeho požadavek reaguje. Nejvíce využívané internetové prohlížeče jako Explorer, Mozilla, Opera mohou být klienty, ale stejně tak i vyhledávací robot či jiný program. HTTP server označuje program fungující v počítači v serverovně, mezi nejvíce používaný patří Apache. HTTP protokol je jazykem dvou programů, které spolu komunikují nejvíce po internetu. Požadavkem klienta bývá zejména vyhledání určité stránky, proto se připojí k serveru a zadá URL dané stránky. Požadavek se naformuluje v HTTP protokolu, server jej přijme, zpracuje odpověď v HTTP protokolu, kterou zašle klientovi. Může jít například o HTTP hlavičky s textem stránky v HTML. Když klient obdrží odpověď, přečte hlavičky a zobrazí stránku.

Mezi jednu z metod HTTP patří metoda GET. Tato metoda vyslovuje určité požadavky na soubory s konkrétním umístěním na serveru. Vzhledem k tomu, že na jednom serveru může být více domén, informace HOST stanoví doménu, ze které soubor získá. Takové volání příjme server, který hostí doménu a podle informace HOST určí adresář, kde bude soubor hledat. Z kořeně dokumentů získá server soubor, připojí k němu HTTP hlavičky odpovědi a zašle ji zpět klientovi. Klient (prohlížeč) projde jednotlivé HTTP hlavičky, zapamatuje si je a zpracuje požadovaný HTML dokument, který následně zobrazí uživateli. HTTP hlavičky nejsou zobrazeny, avšak lze je vidět pomocí rozšířením prohlížeče, ve vlastnostech stránky v prohlížeči (klávesa F12), vlastním programem nebo telnetem.

Po zadání požadavku GET serveru, server může reagovat následovně:

- kódy 1xx – informační zprávy definované konkrétní aplikací
- kódy 2xx – úspěšné zpracování požadavku
- kódy 3xx – oznamují například, že byl dokument přesměrován
- kódy 4xx – oznamují chybu na straně klienta
- kódy 5xx – oznamují chybu na straně serveru

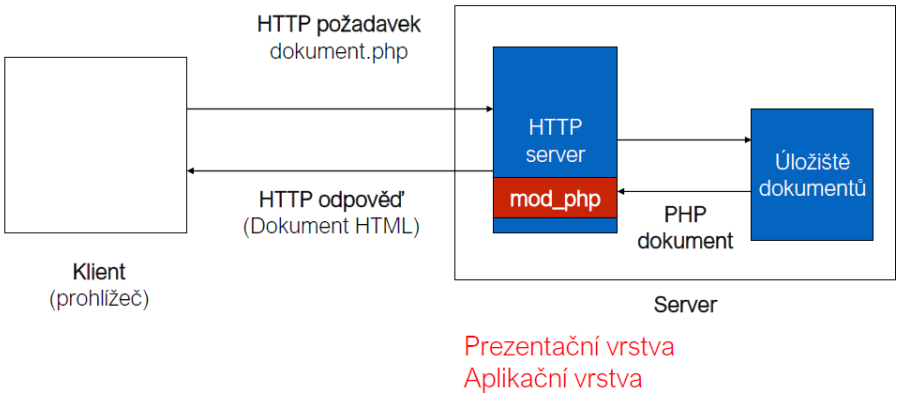
Kódy uvozují odpovědi serveru, proto je důležité znát nejčastější kódy. Za trojmístnými kódy bývá obvykle uveden anglický komentář.

Pro programování serverové části jsem volil z několika možných technologií a jazyků, které jsou objektově orientované, dynamicky typované a interpretované, tj.:

- PHP
- Python
- Ruby[26]

PHP

Personal Home Page (Hypertext Preprocessor) je nejvíce využívaný jazyk pro vývoj webu. Jednoduchý nástroj pro tvorbu dynamických webových stránek dostupný na všech klíčových platformách zdarma. Jeho cílem je vývoj jednoduchých webových aplikací, u nichž se nevyplatí drahé aplikační prostředí.[15]



Obrázek 2.9: Architektura serveru s PHP[15]

Jeho nevýhodou je slabý typový systém, nepřesnost a více možností zpracování úkolů, které narušují čistotu kódu. Výhodou je však přímé vkládání zdrojového kódu do HTML souborů, zejména u tvorby jednodušších stránek či aplikací. Programátorům, kteří mají zkušenosti s jazykem C, pomáhá syntaxe obou jazyků.

Ruby

Ruby je dynamický jazyk s objektovými datovými typy. V jeho běhu lze střídat metody všech objektů. Při různých přístupech může dojít k tomu, že primitivní objekty jsou přetypovány kdekoliv v programu a při přesunu cizích knihoven se mohou chovat neočekávaně.

Python

Python neslouží přímo k vývoji webových aplikací, avšak pomocí WSGI (Web Server Gateway Interface) lze tento jazyk využít i v tomto směru. Nabízí velkou standardní knihovnu a kvalitní dokumentaci. Může využít soubor kvalitních externích modulů. Zaměřuje se na čitelnost a znovupoužitelnost kódu (PEP-8) pomocí syntaxe. Přehlednost kódu spočívá v odsazených blocích textu. Využívá silný typový systém. Nízký počet klíčových slov jej činí méně složitým.

Srovnání jazyků pro backend

Jazyky pro backend byly porovnány podle následujících kritérií:

- přehlednost
- bezpečnost
- popularita
- výkon
- dynamičnost

Nejvíce přehledným jazykem z důvodu odsazených bloků textu byl vybrán Python. Z pohledu bezpečnosti se umístily na stejném místě Ruby a Python, neboť PHP může chybovat v čistotě kódu. Právě naopak je tomu v popularitě, kde obliba PHP je vyšší než u Pythonu, který je umístil na druhém místě, a Ruby, který se umístil na místě třetím. Výkonnost všech třech jazyků je téměř na stejné úrovni s ohledem na provedení kódu se stejnou funkcí v cca obdobném čase. Nejvíce dynamickým je jazyk Ruby, který mění metody za běhu programu. Pro svou práci jsem si vybral jazyk Python s ohledem na jeho přehlednost, bezpečnost, střední popularitu i výkonnost.

Python Framework

Mezi multifunkční frameworky patří Flask, Twisted, Django. Vytvářejí základní funkce všech webových aplikací jako např. napojení k databázi a práce s ní, vznik uživatelů, autentizace, funkcionality se session a cookies, základní bezpečnostní ochranu. Django a Flask jsou založeny na stejném účelu. Twisted lze kombinovat, protože se zabývá komunikací po síti.

Twisted

Framework napsaný v Pythonu pro programování síťových služeb a aplikací, jehož hlavním cílem je vznik neblokujícího asynchronního serveru. Podporuje klientskou i serverovou část a využívá široký výběr operačních systémů a platforem. [28]

Django

Tento jednoduchý, flexibilní, vysokoúrovňový open-source framework vytváří webové aplikace s relativně nízkým počtem řádků zdrojového kódu. Na objektově orientovaném jazyku Python vytváří aplikace k řešení různých druhů problémů a možností znovupoužitelnosti všech komponentů na základě principu „DRY“ (Don't repeat yourself). [20]

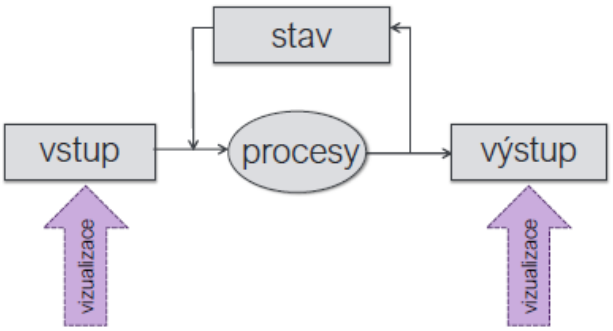
Flask

Jedná se o jednoduchý microframework pro Python. Jeho zásadou je k osekanému jádru využít funkcionalitu pluginů. Tento program jsem si vybral pro svou práci pro jeho jednoduchost.[24]

2.7 Vizualizace a serializace dat

Data v informačních systémech jako jednoduché hodnoty, struktury a kolekce mohou být reprezentována pomocí vhodných modelů a těmi jsou:

- Vizualizace (2D reprezentace dat), která využívá zejm. HTML, CSS, pokročilí JavaScript, XSLT a grafické prezentace.
- Serializace (1D reprezentace sériovou formou pomocí řetězce), která využívání zejm. JSON, XML, další specializované formáty.[17]



Obrázek 2.10: Vizualizace v informačním systému[17]

Vizuální prezentace

Vizuální prezentace může představovat seznamy, formuláře, tabulky. Musí být zvlášť určená, nezávisle spravována. Generovaný HTML kód vyjadřuje obsah výsledného dokumentu a jeho strukturu. Má více možností vzhledu pro desktop, mobil apod. Cílem je udržení nejjednoduššího serverového kódu.

CSS

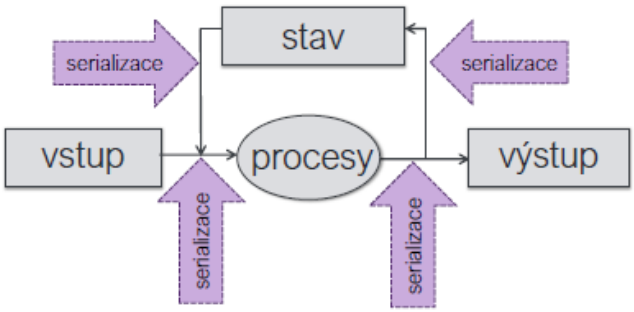
Cacsading Style Sheets je samostatnou definicí vzhledu. Style sheet znamená stylový předpis. CSS pracuje se sadou pravidel, kde každé pravidlo stanovuje vlastnosti nějakého okruhu HTML elementů. Okruh HTML elementů volí tzv. selektor pravidla. Definice určité vizuální vlastnosti je označena jako její deklarace. Větší množství elementů může spojovat jedno pravidlo a naopak aplikace většího množství pravidel může způsobit vzhled elementu.

CSS Frameworky

CSS Frameworky představují hotová CSS řešení, která jsou nejčastěji využívanými prvky uživatelského rozhraní pro rychlý návrh aplikace. Využívají přidání různých tříd HTML elementů (standardní atribut class), přičemž nechtěným efektem je, že se větší část specifikace vzhledu se přesouvá do HTML. Řešením je minimalizace kombinací s vlastními CSS předpisy, např. Bootstrap.

Serializace

Serializace (marshalling) je proces konvertování datových struktur nebo stavů objektů do formátu, který může být buď uložen jako např. textový soubor nebo přenášen síťovým přenosem.



Obrázek 2.11: Serializace v informačním systému[17]

Deserializace

Deserializace (unmarshalling) je rekonstrukce hodnoty na stejný nebo jiný použitelný i netextový formát (vznik sémanticky ekvivalentního klonu původní datové struktury). Použitím referujících hodnot (vztahů) vzniká složitý proces u objektů i u metod.

Syntaxe serializačních formátů

Pokud chápeme pojem serializace vyjádřením hodnoty struktury ve formálním jazyce, musí mít hodnoty struktury a kolekce stanovený tvar. U hodnoty typu struktura jde o uspořádanou n-tici (a1, a2, ... an) a u hodnoty typu kolekce jde o uspořádanou množinu b1, b2, ... bm, přičemž na úrovni hodnoty (výskytu) jsou stejné. Vzhledem k tomu, že formální jazyk musí být bezkontextový, tak se jedná o závorkované zanořené věty tvaru an... bn. Jakýkoliv bezkontextový jazyk by měl mít možnost zápisu literálů strukturovaných hodnot (např. JavaScript –JSON) nebo lze použít speciální bezkontextový jazyk (např. značkovací XML).

Serializační formáty

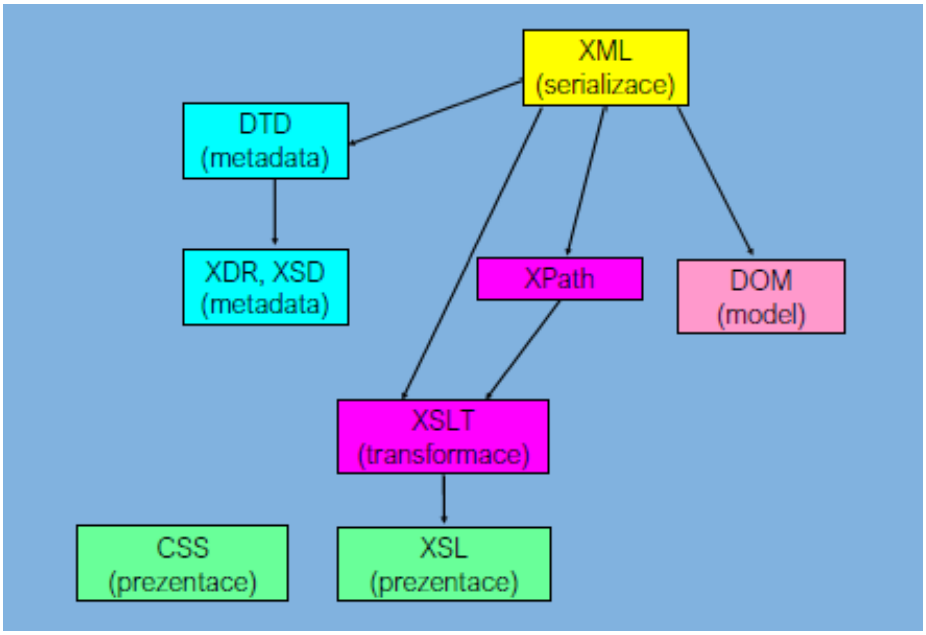
Serializační formáty jsou převážně proprietární (privátní, uzavřené) formáty jednotlivých aplikací závisející na architektuře (např. dumppaměti). Nejsou schopny sdílet existující nástroje jako např. parser. Z historie:

- V roce 1987 zavedla firma Sun Microsystems External Data Representation (XDR)
- V roce 1990 vyvinut firmou XML Working Group jazyk XML založený na SGML a rozšířil možnosti HTML
- V roce 2001 vznikl JSON jako konkurence XML
- V roce 2001 vznikl YAML jako nadmnožina JSON

2.7.1 XML

Extensible Markup Language byl primárně určen jako značkovací jazyk pro tvorbu dokumentů. Vznikl ze standardu SGML. Je snadno rozšiřitelný pro další aplikace, protože umí

formálně definovat vlastní schéma, validaci, transformaci, objektovou reprezentaci, je jedním z nejvhodnějších kandidátů pro reprezentaci dat. Nevýhodou je jeho těžkopádnost.



Obrázek 2.12: Komponenty XML technologie[14]

XML dokumenty

XML dokumenty jsou syntakticky obdobné jako v HTML.[18] Vznikají na základě hierarchického zanořování XML prvků (elementů), které mají své jméno, příp. atributy. Jeden z elementů je kořenovým. XML dokumenty mají syntaktické rozdíly proti HTML. XML neurčuje sám jména ani význam elementů či atributů, protože vše stanovuje konkrétní aplikace. Všechny značky jsou párové (parser nezná význam) a hodnoty atributů se uvádějí v uvozovkách. Je-li element prázdný (bez obsahu), ale může mít atributy, bude mít tvar: <jménopřípadnéatributy/> nebo ekvivalentně: <jménopřípadnéatributy></jméno>.

Typová kontrola

Pro konkrétní aplikaci můžeme stanovit určitá jména značek a atributů. Zavedením možnosti typového určení a kontroly (text, čísla, enumerace apod.) obsahu značek lze definovat, kontrolovat, přenášet obecné datové struktury, čímž vzniká ze značkovacích jazyků obecný prostředek pro serializaci strukturovaných dat.

Definice typu dokumentu

Definice typu dokumentu (DOCTYPE) určuje strukturu zvolené aplikace (metadata). Jedná se o instrukci, která XML dokumentu stanoví definici typu dokumentu (DTD – Document Type Definition). V serializovaném tvaru vzniká krátký řetězec vyhovující syntaxi. Samotné definice nejsou vyjádřeny v jazyce XML. XSD (XML SchemaDefinition) definuje základní kameny dokumentu XML jako elementy a atributy, podelementy, jejich pořadí. U elementů

určuje počet, obsah, datový typ (i atributů) a výchozí a fixní body (i atributů). Pracuje s XML syntaxí. Další možností je např. RelaxNG.

Entity

Entita je taková jednotka (prvek) dokumentu, která se v dokumentu často objevuje, má své jméno a lze ji fyzicky izolovat a samostatně ukládat. Tím, že se jednotka vyskytuje v několika místech dokumentu, může její duplikací docházet k chybám. Nekompatibilní systémy ji mohou odlišně reprezentovat. Z těchto důvodů se doporučuje rozsáhlý dokument rozčlenit. Entita existuje v jiném datovém formátu, nežli text XML (multimédia).

Klasifikace entit:

- Interní zastupují nějaký text určený přímo v XML dokumentu (kratší, složitější texty, které se v dokumentu několikrát opakují).
- Externí zastupují nějaký jiný soubor, buď textový dokument nebo binární soubor (např. obrázek).

Externí entity se dělí na:

- Textové
- Binární

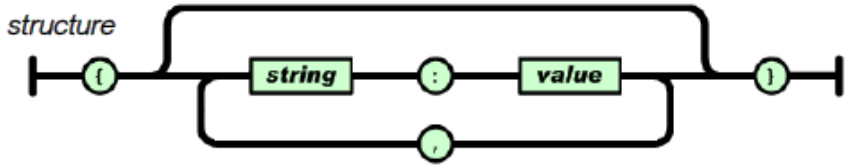
V souhrnu mohou existovat entity v kombinacích: interní textová, externí textová a externí binární.

Deklarace

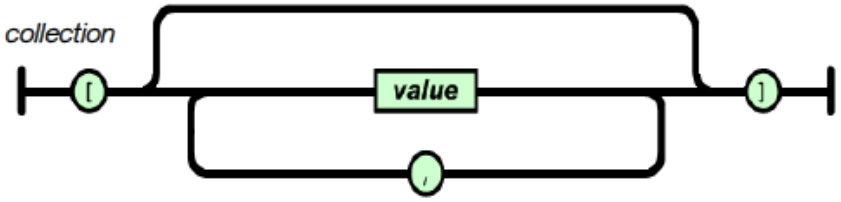
Entita musí být v textu definována před prvním odkazem. Při vícenásobné definici se bere první a ostatní se ignorují. U deklarace interní textové entity je textový obsah uzavřen v uvozovkách (nebo apostrofech). Vestavěné entity není třeba deklarovat, protože nahrazují metaznaky jazyka XML. Deklarace typu dokumentu se musí objevit před prvním elementem dokumentu a lze ji využít jen pro pojmenování a další část za závorkami se vynechá.

2.7.2 JSON

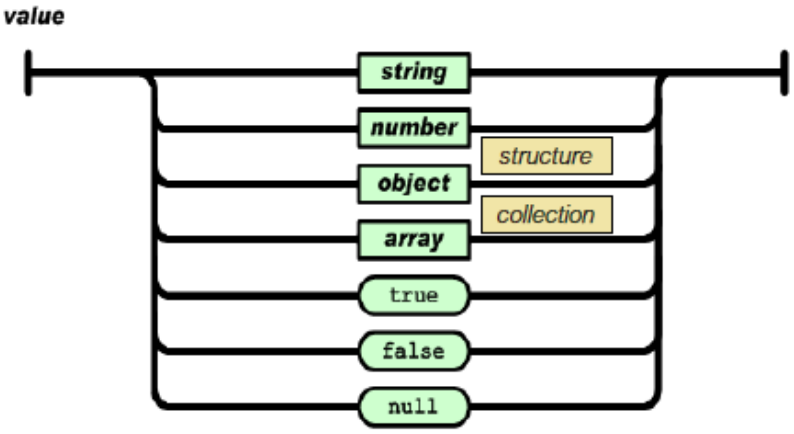
Java Script Object Notation byl navržen na podmnožině programovacího jazyka JavaScript, Standard ECMA-262 3rd Edition-December1999 jako jazyk pro přenos dat mezi serverem a prohlížečem. Pro zápis využívá syntaxi literálů v jazyce JavaScript, v jehož aplikacích dochází k snadnějšímu zpracovávání. Není na této platformě závislý, protože má podporu na všech hlavních serverových i klientských platformách.



Obrázek 2.13: Struktura v JSON – serializace neuspořádané struktury[17]

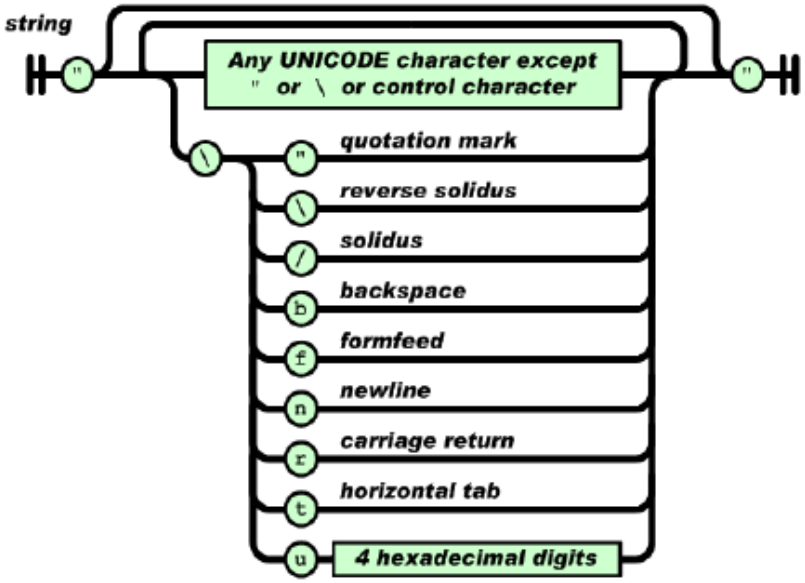


Obrázek 2.14: Kolekce v JSON – serializace uspořádané kolekce – seznamu[17]

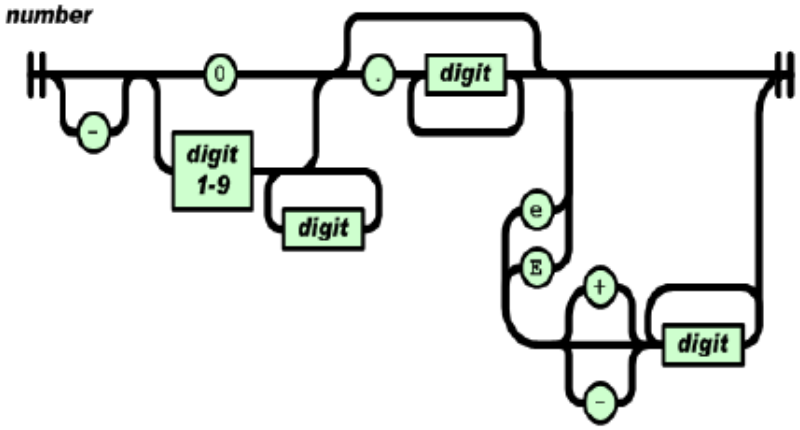


Obrázek 2.15: Základní typy v JSON[17]

Hlavní výhodou JSON oproti XML je menší velikost přenášených dat. Uvádí se, že obsah XML je až ze 40% tvořen značkami a atributy. Malé projekty neomezí, velké však ano. Další výhodou je úspora finančních prostředků. Za nevýhodu, která sama o sobě nemusí nevýhodou být, je někdy chápána nemožnost definovat znakovou sadu přenášeného obsahu. Výchozí kódování je moderní UTF-8.



Obrázek 2.16: Textové řetězce v JSON[17]



Obrázek 2.17: Čísla v JSON[17]

2.7.3 YAML

Ain't Markup Language obsahuje rysy pro jednodušší, uživatelsky příznivější a kompaktnější serializaci bez složitých konstrukcí s možností se vyjádřit v čistém textu. Má podporu nehierarchických struktur a strukturuje data indentací, tedy zanořením (bez tabelátorů).

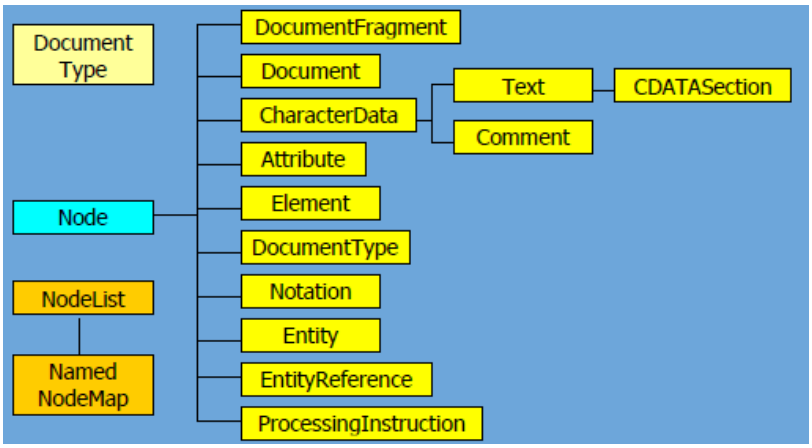
Mezi ostatní jazyky pro serializaci lze zařadit XDR od firmy SUN, dále Cocoa (propertylist v systémech NeXSTEP, GNUstepa Mac OS X). Pro rozsáhlé soubory dat jako jsou satelitní a meteorologická data byl vyvinut binární serializační formát HDF a jeho varianty.

Omezené serializační formáty

Omezené serializační formáty jsou určeny pouze pro speciální použití, např. pro předávání obsahu formulářů v HTML protokolem HTTP. Omezují hloubku zanoření a typy položek strukturovaných hodnot.

2.7.4 DOM

Objektový model (Document Objekt Model) stanovuje hierarchii objektů, které podporuje internetový prohlížeč, jejich metod a vlastností. Jinak řečeno jde o objektovou reprezentaci XML či HTML dokumentu pro práci s ním jako např. pro čtení či změnu. Jde o prezentaci ve formě stromu objektů. Rozhraní objektů nezávisle na platformě definuje W3C standard. Jeho znalost je nezbytná pro programování v JavaScript, Java, PHP, C++, C# apod.



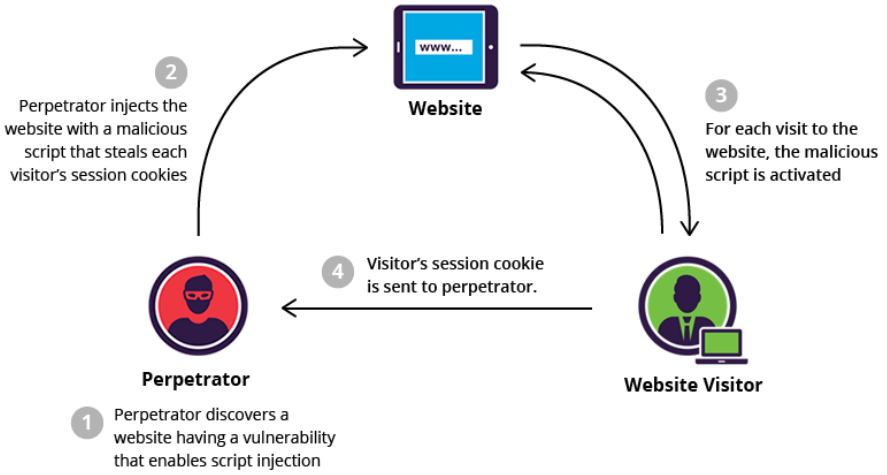
Obrázek 2.18: Třídy DOM[17]

2.8 Bezpečnost webových aplikací

Zabezpečení webových aplikací je jednou z nejdůležitějších věcí, protože webové aplikace jsou aktivy uživatele, tedy znamenají pro něj určitou hodnotu. Narušení bezpečnosti webové aplikace by způsobilo uživateli poškození či trvalou škodu. Může jít například o získání osobních údajů (jméno, heslo, adresu) a jejich zneužití. Pokud aplikace ztratila důvěryhodnost, nikdo by neměl zájem s ní pracovat. Z těchto důvodů se zabývám i bezpečností aplikace a možnými útoky.

XSS (Cross-site scripting)

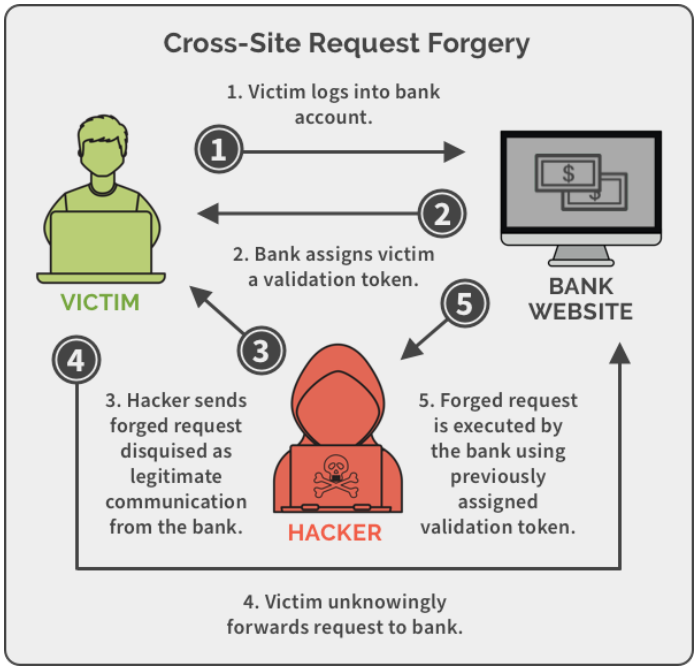
XSS přestavuje jeden z možných ataků na webovém rozhraní tím, že vkládá vlastní Javascript kód, čímž provozuje tzv. phishing. Může proběhnout tak, že svému dítěti chcete předvést fotku zvířátka, kytky či auta a místo toho se v odkazu otevře video nebo obrázek s něčím zcela nechutným, např. s roztrhaným zvířetem. Vyvolá to hněv a odmítavou reakci k webu. Tento moment útočník využije například pro získání důvěryhodných dat, finančních prostředků nebo dokonce ke spadnutí celého webu.



Obrázek 2.19: Cross-site scripting[5]

Cross Site Request Forgery

Cross Site Request Forgery je jednou z 20 nejvíce zneužívaných bezpečnostních chyb. Útočí tak, že jeho stránka navádí prohlížeč uživatele k zaslání HTTP request na důvěrnou stránku jako běžnou interakci uživatele s touto stránkou. Připojení uživatele a stav cookies v prohlížeči uživatele mu umožní připojit se jako uživatel a provádět veškeré operace s jeho oprávněním s tím, že to zneužívá ve svůj prospěch. Na obrázku 2.20 je schéma útoku.[9]



Obrázek 2.20: Cross Site Request Forgery[1]

Custom HTTP Headers

Jako ochranu proti Cross Site Request Forgery lze využít vlastní http hlavičky, které prohlížeče brání zasílání vlastních hlaviček z jedné webového adresy na další, ale nebrání tvorbě vlastní http hlavičky za použití XMLHttpRequest objektu. Stránka musí tedy kontrolovat hlavičky HTTP requestů a vyloučit ty nesprávné.

Secret Validation Token

Další obranou je dodatečné zaslání dat ke každému HTTP požadavku, které garantují autorizovaný zdroj. „Validační token“ by měl být pro útočníka neznámý a těžce zjistitelný. V případě, že by HTTP request neobsahoval validační token či neměl očekávanou hodnotu, server by jej měl odmítnout. Validační tokeny lze generovat pomocí různých technik, například náhodně, a následně jej uložit ve formě cookies.

Kapitola 3

Implementace

3.1 Použité technologie

K tvorbě aplikace jsem využil virtual environment. Do environmentu jsem stáhl a nainstaloval Python a Flask, které byly v zadání práce.

3.1.1 Virtuální prostředí

Stahování, ukládání a řešení modulů v programovacím jazyku Python je jako u mnoha dalších moderních programovacích jazyků jedinečné. Jedinečnost stahování má své výhody, avšak některá rozhodnutí o skladování a řešení balíčků přinesla určité problémy, zejména s tím, jak a kde jsou uloženy. Balíčky mohou být nainstalovány do systému na více míst, např. v podřízeném adresáři cesty uložené v sys.prefix. Balíčky třetích stran nainstalované pomocí easy_install nebo pip obvykle umístěny do jednoho z adresářů, na které odkazuje server site.getsitepackages. Každý projekt v systému tak používá ve výchozím nastavení stejné adresáře k ukládání a načítání balíčků webů (knihovny třetí stran). Problém nastane, pokud dva různé projekty vyžadují jinou verzi programovacího jazyka Python, proto tímto způsobem nelze rozlišovat jednotlivé verze pro daný projekt. Projekty jsou ukládány podle názvů a neobsahují informaci o verzi jazyka, nelze tak rozlišit jednotlivé verze. Dané projekty by musely používat stejnou verzi, což by nefungovalo. Vzniklý problém řeší virtuální prostředí a nástroje virtualenv/venv.[27]

Pro zpracování projektů v Pythonu je vytvoření izolovaného virtuálního prostředí velmi důležité. Jde zejména o to, že každý projekt je jedinečný v tom, že používá různé verze programovacích jazyků a knihoven. Každé samostatné virtuální prostředí pro projekt by mohlo být závislé na jiné verzi jazyka. Není omezen počet takto vytvořených virtuálních prostředí, které lze snadno vytvořit pomocí příkazového řádku.

Nástroj venv jsem získal instalací programovacího jazyka Python 3, který tento nástroj obsahuje. Nový adresář ke zpracování projektu jsem vytvořil na základě příkazu: $ mkdir bakalarskaprace && cd bakalarskaprace. Poté, co vznikl nový adresář, jsem v něm vytvořil nové virtuální prostředí: $ python3 -m venv env.

Při instalaci izolovaného virtuálního prostředí nástrojem venv musíte určit verzi interpretu Python 3, která instalaci provede. Předejdeme tím problémům s instalací nového prostředí v Pythonu. Vzhledem k tomu, že jsem pracoval s Pythonem 3.8.2, tak jsem nevyužil nástroje pyvenv určeného pro nižší verze, ale příkaz: python3 -m venv.

Každá složka tohoto prostředí obsahuje následující položky:

- bin: soubory, které ovlivňují virtuální prostředí

- include: záhlaví C, která sestavují balíčky Pythonu
- lib: kopie verze Pythonu spolu se složkou site-packages, kde je nainstalována každá závislost

Izolace od globálního prostředí lze dosáhnout pomocí několika různých nástrojů Pythonu a také pomocí samotných spustitelných souborů Pythonu. Všechny kódy a příkazy tak jsou prováděny ve vazbě na aktuální prostředí.

Aktivační skripty v adresáři Scripts jsem použil k nastavení shellu, aby se použila daná verze Pythonu spolu s balíčky webu nainstalovanými ve virtuálním prostředí. K aktivaci izolovaného prostředí jsem spustil následující skript: $ env/Scripts/activate. Po aktivaci prostředí se zobrazí indikátor s názvem aktuálního prostředí. Program v jazyce Python tudíž bude pracovat pouze s balíčky a nastavením aktuálního prostředí. Při spuštění Pythonu a existenci dvou spustitelných souborů neexistuje mezi nimi žádný rozdíl, liší se pouze v adresářích.

3.1.2 Porovnání textových editorů

S ohledem na skutečnost, že v současné době existuje mnoho textových editorů, musel jsem se pro aplikaci rozhodnout mezi jedním z nich. Prostudoval jsem tak porovnání kladů a záporů čtyř nejoblíbenějších Atom, Visual Studio Code, Vim a Sublime, abych zjistil, který mi bude nejvíce vyhovovat. [8]

Atom

Jeho adresou URL je https://atom.io/, dá se získat zdarma (licence MIT), vyvinul jej GitHub a funguje na platformách OSX, Windows a Linux. Je pojmenováván Hackabilním textovým editorem pro 21. století. Jeho první vydání proběhlo v roce 2014 a je stále dynamičtější. Nabízí možnost přidání dalších funkcí k úpravám přes správce balíčků nainstalovaném ve výchozím nastavení. Tyto balíčky a témata (6 tisíc a více) jsou pak hostovány na severu vývojáře GitHub. I základní funkce Tree View a Settings View jsou předinstalovanými balíčky. Zásadním problémem Atomu je vědět, které balíčky je potřeba nainstalovat do základní verze (tzv. od nuly).

Atom potěší uživatele funkcemi minimap, která pomáhá vizuálně přeskočit na části souboru, automatické doplňování pomocí automatického doplňování +, rozděleného podokna při práci na několika souborech současně, podpora drag / drop souboru / složky ve stromovém zobrazení (chybí v Sublime). I úpravy stylů jako oříznutí mezer na uložení a uložení na ztracené zaměření souboru lze v Atomu snadno nastavit a přepsat, např. i různé odsazení pro JS vs CSS vs HTML.

Výkon Atomu má při otevření souboru nebo přepínání mezi kartami zpoždění. Lze říci že je stále frustrující, ale od doby, co byl vydán se výrazně zlepšil. Přesto je Atom skvělým nástrojem k editaci pro nové uživatele. Největší nevýhodou zůstávají stále problémy s výkonem.

Kód Visual Studio

Jeho adresou URL je https://code.visualstudio.com/, dá se získat zdarma, vyvinul jej Microsoft a funguje na platformách OSX, Windows a Linux. Je relativním nováčkem ve světě textových editorů, protože byl zveřejněn v dubnu 2015. Jedná se o vytvoření výkonného a flexibilního editoru napříč platformami. VSCode nabízí ekosystém pluginů (rozšíření),

a to přes několik tisíc. Některé pluginy i jejich správa je nainstalována ve výchozím nastavení. Určitou dobu zabere výběr vhodných pluginů pro konkrétní projekt. Mezi oblíbené patří Debugger pro Chrome, který disponuje možností nastavení tzv. break pointů a ladit JavaScriptový kód. Pro úpravy a pracovní postup funguje obdobně jako Atom a používá Elektron, Node a HTML s CSS.

VSCode je náročnější na konfiguraci, ale po čase začne ukazovat, co všechno umí. Velkým bonusem je integrace Git, který je výhodou pro běžné operace, mezi něž patří například odevzdání a rozdíly, a ušetří čas. Stejně jako u jiných editorů obsahuje funkce přizpůsobení jako například obtékání, odsazení, vytváření témat, vylepšení jazyka atd. VSCode díky používání Node.js, Elektronu, HTML a CSS je rychlejší než Atom. Při práci se soubory se nezpožďuje a vyhledává plynule.

Lze říci, že hlavním rozdílem mezi Atomem a VSCode je rozdíl ve výkonu. Výkon VSCode je srovnatelný se Sublime. VSCode nabízí zajímavou konfiguraci k tomu, aby splňoval stejné standardy jako Sublime. Integrace Git a ladící program jsou jeho bonusem.

Vim

Jeho adresou URL je http://www.vim.org/, dá se získat bezplatná licence kompatibilní s GPL, vyvinul jej Bram Moolenaar a funguje na platformách OSX, Windows a Linux. Nabízí možnost úpravy a prohlížení souboru na vzdáleném serveru pomocí terminálu. Vim nabízí širou funkcionalitu, avšak začátky jsou těžké. Více než 14 000 balíčků jsou poměrně velkým výběrem. Mezi nejzajímavější patří průzkumníci stromů, zvýrazňovače syntaxe, tématické úpravy, integrace Git. Patří mezi flexibilní a výkonné editory. Opět je u něj důležité znát nejvíce prospěšné pluginy k instalaci, které můžeme získat vlastními zkušenostmi, získanými prací ve Vimu, nebo na základě doporučení.

Vim funguje jako textový editor příkazového řádku, což znamená, že se neovládá na základě dvojkliků či pohybů myši, ale své funkce jako otevírání, zavírání, úpravy, ukládání ovládá na základě klávesových zkratek. Pokud se naučíte společné příkazy klávesnice, tak se ve Vimu pracuje celkem dobře. Hlavním problémem Vimu je tedy znalost klávesových zkratek, díky níž se pak snadno upraví soubory, prohledává a nahrazuje. Pokud je neznáte, rychle se v něm ztratíte. Je přizpůsobitelný, přes Google lze vyhledat konfigurační soubory.

Vim je výkonným editorem s rychlým a efektivním vývojovým prostředím. Jeho výkonnost však blokuje znalost příkazů a rychlost jejich psaní uživatelem. K získání znalosti příkazů je potřebná trpělivost.

Sublime

Jeho adresou URL je https://www.sublimetext.com/, platí se licenční poplatek s možností využití časově neomezené zkušební bezplatné verze. Vyvinul jej Jon Skinner bývalý inženýr a funguje na platformách OSX, Windows a Linux. Sublime byl spuštěn v roce 2007. Jde o jedno nejoblíbenějších vývojových prostředí podle průzkumu společnosti Stackoverflow 2016. Prvním instalačním balíčkem je plugin Sublime Package Control, pomocí kterého se instalují další balíčky. Sublime obdobně jako Atom nabízí spoustu balíčků. Některé z nich jsem využil a mám je popsány v následující kapitole. Editor vytváří prostředí, které podporuje soustředění na důležité soubory a jeho vyhledávání je jednoduché a rychlé i ve větším množství souborů.

Po rozšíření o klíčové pluginy, jejichž instalace nějakou dobu trvá, u Sublime stejně jako u Atomu získáte správné zvýraznění syntaxe, formátovacích jednotek JSON atd. Sublime umí ovládat aspekty obdobně jako Atom jako např. oříznutí mezery. Má jednoduché

konfigurační soubory ve formátu JSON. Ve výkonnosti (otevírání, zavírání, vyhledávání) Sublime překonává Atom, pracuje plynule a rychle. Obecně řečeno Sublime je flexibilním a rychlým editorem a jedno z nejlepších vývojových prostředí.

Všechny výše uvedené editory mají své klady a zápory. Vzhledem k tomu, že Sublime již nějakou dobu využívám, dále pro jeho výkonnost, jednoduchost ovládání, flexibilitu, intuitivnost a k faktu, že se jedná o jedno z nejlepších vývojových prostředí, jsem si pro svou práci vybral právě tento editor.

3.1.3 Sublime Text 3

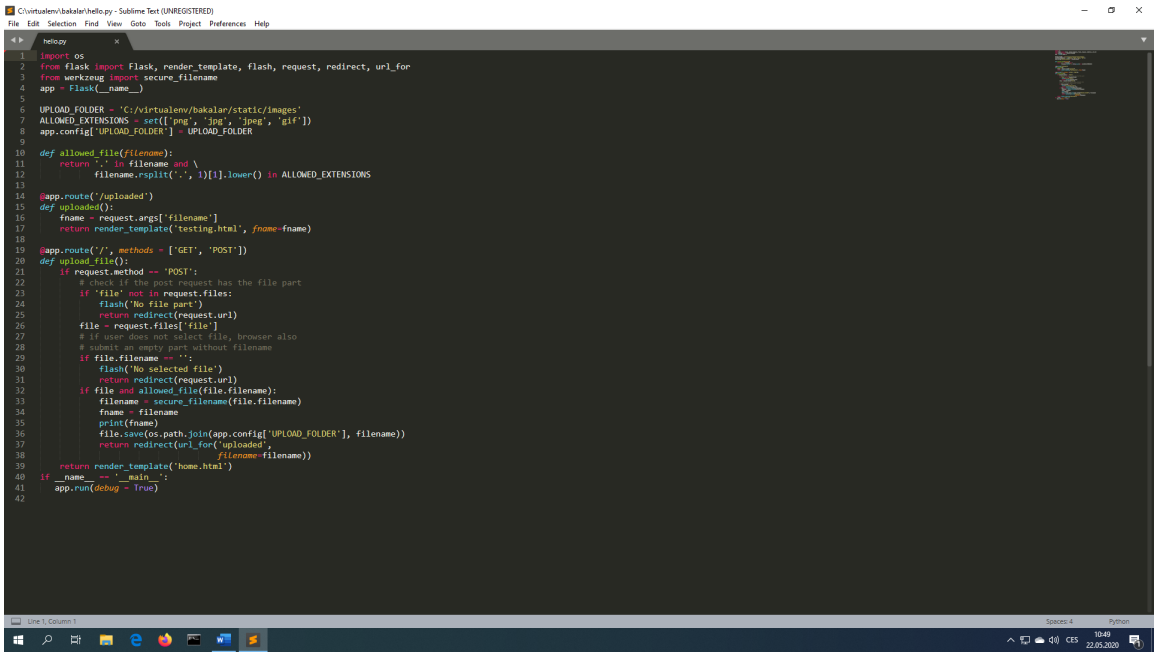
Samotné implementování programu probíhalo za pomocí programu Sublime Text 3, který zvýrazňuje syntaxi jazyků HTML, JavaScript, Python, které jsem k tvorbě aplikace použil.

Jedná se o komerční editor napsaný v jazyku C++. Sublime Text je multiplatformní, čistý, výkonný a rychlý code editor s přehledným uživatelským prostředím (inspirace Vimem).[10] Jak jsem již výše uvedl, podporuje hodně programovacích jazyků a umí zvýraznit jejich syntaxi. Jeho komunita vytváří různá rozšíření jako pluginy, snippety atp. Je rychlý, co se týče vypnutí i zapnutí oproti jiným editorům jako např. Eclipse. Sublime Text 3 to zvládne do 2 sekund, přičemž jiné do 2 minut.

Editor dokáže pracovat v různých módech jako fullscreen, který jsem nejvíce využíval. Dále pracuje v distraction free mode, kde se zobrazí pouze místo na kód a menu. Tento jsem nepoužíval. Kromě této nabídky umí editor rozdělit pracovní plochu horizontálně až na 4 plochy vedle sebe, vertikálně 3 plochy pod sebou nebo na 2x2. Rozdělení ploch je praktické při prezentaci kódu, abychom nemuseli přepínat mezi jednotlivými soubory.

Sublime Text 3 nabízí stylovou a rychle návykovou minimapu kódu, kterou lze měnit podle toho, zda ji chcete využít či ne. Python konzole je bonusem navíc, jenž lze využít minimálně jako kalkulačku. V Sublime Text 3 probíhá plná konfigurace ve speciálních JSON souborech.

S editorem se dá pracovat jako s klikacím editorem, nebo můžete využít svých znalostí práce s klávesovými zkratkami a použít klávesnici. Další jeho super výhodou je možnost ovládat více kurzorů najednou. Mezi další funkčnosti, které usnadňují práci v editoru je klávesová zkratka Ctrl + D. Vybereme s ní část kódu a postupně se označí další výskyty, které můžeme pak upravovat. Na obrázku 3.1 je znázorněno zobrazení editoru.



Obrázek 3.1: Sublime Text 3

Klávesové zkratky

Pro rychlejší práci v editoru je výhodné pracovat s klávesovými zkratkami, kterých je hodně, ale mezi nejvíce používané patří tyto:

- Ctrl + P nabídne výběr souboru, pokud zadáte @ získáme seznam funkcí a pokud zadáte : a číslo řádku, tak přesune kurzor na zadaný řádek
- Ctrl + Shift + P nabídne seznam příkazů
- Ctrl + D přidá kurzor na další výskyt vybrané položky, jak jsem již popsal výše
- Ctrl + Click přidá na vybrané místo další kurzor
- Ctrl + F vyhledá zadaný řetězec kódu v aktuálním souboru
- Ctrl + Shift + F vyhledá zadaný řetězec kódu ve více souborech, enter vám zobrazí výpis s nalezením
- Ctrl + Shift + nahoru/dolů posune řádek nahoru nebo dolů
- Ctrl + Shift + D duplikuje řádek
- Ctrl + Shift + K nebo Shift + Delete smaže řádek
- Ctrl + (Shift) + Tab přesune na další otevřený soubor

Pluginy

Abychom mohli pracovat s pluginy, je nutno nainstalovat Package Control, a to otevřením konzole pomocí Ctrl + ; a vložením kódu Install Package Control, potvrdíme klávesou Enter.

Po úspěšné instalaci se při použití klávesové zkratky Ctrl + Shift +P objeví nový příkaz. Jedná se o příkaz Package Control: Install Package, který nám po vybrání poskytne textové okno, kam napíšeme název požadovaného pluginu. Plugin si můžete vybrat podle vlastního uvážení na webu https://packagecontrol.io/. Po instalaci nového pluginu je doporučeno celý editor restartovat. Mezi nejzajímavější pluginy, které mohou usnadnit práci v editoru patří:

- Alignment přehledně zarovná vybraný kód
- Colorpicker nabídne možnost využít klasický color picker
- Emmet (Zen Coding) jako doplněk pro HTML vytváří HTML zápis, obdobně jako zápis CSS selektrorů, nabízí násobení, znak $ k inkrementaci, text vyplní obsah textem, závorky () pro vytvoření výrazů atp. Spouští se Ctrl + Alt + Enter
- SidebarEnhancements slouží k navýšení možností v přehledu souborů
- SublimeCodeIntel je velmi užitečný, napovídá dostupné funkce a metody, po zadání Alt + Click na jméno funkce Vás přesune na její zápis, který se může vyskytovat i v jiném souboru
- Theme Soda obarví text do tmava a použije upravený zvýrazňovat kódu Monokai z Bonus options
- Theme Spacegray je velmi oblíbený a nastavuje vzhled editoru

Snippety

SublimeText nabízí několik snippetů, tj. útržků kódů, které se dají vložit ze seznamu příkazů. Jde zejména o HTML kostru, lorem ipsum atp. V případě, že Vám nabídka nepostačuje, je k dispozici možnost vytvořit si snippet vlastní. K tvorbě nového snippetu se využívá cesta v záložce: tools -> developer -> new snippet.

Sublime Text jsem si vybral právě proto, že se jedná o jednoduchý editor se spoustou bonusových kvalitních doplňků. Jeho velkými konkurenty se stávají editory vytvářené webovými technologiemi, např. Atom či VSCode.

3.1.4 HTML 5 Canvas

Jedním z možných nástrojů pro vytváření anotací v obraze je skrze HTML5 Canvas, který jsem nejdříve vyzkoušel.

HTML5 je novou verzí HTML a standardem ke vzniku dalších aplikací. Obsahuje nových tagů a technologií jako např. tag <canvas>. Canvas znamená plátno, které používáme ke kreslení.[21] Firma Apple jej ve svém operačním sytému uvedla jako první a zabudovala jej i do svého prohlížeče Safari. V dnešní době je podporován všemi moderními prohlížeči. Internet Explorer jej podporuje od verze 9.0.

Syntaxe je rozdílná od tagu <img /> tím, že canvas je párový tag a jeho zápis vypadá následovně:

<canvas id="htmlcanvas"width="500"height="300» Canvas nebyl vykreslen. </canvas>

Zadaný kód zobrazí plátno o velikosti 500x300 pixelů. Následně se nastavuje atribut ID pro abychom se na něj mohli odkazovat např. v JavaScriptu. Výška obsahu plátna je vyjádřena atributem height a šířka obsahu plátna je vyjádřena atributem width. V CSS bychom nastavili pouze rozměry plátna, avšak ne obsah. V případě, že nenastavíme atributy

plátna, automaticky se nastaví rozměr 300x150 pixelů. Pokud Váš prohlížeč nepodporuje HTML 5 canvas, tak se zobrazí namísto vykresleného plátna pouze textový obsah tohoto tagu.

V aplikaci jsem zkoušel použít 2D vykreslovací kontext Canvasu. Existuje již také WebGL, který umí vykreslovat i 3D objekty, avšak tento je zatím experimentální. Systém souřadnic je obdobný jak v CSS, tudíž souřadnice [0;0] stanovují levý horní roh plátna, ne stránky. Souřadnice vychází z bodu nula, protože jde o absolutní pozicování, které je uváděno v pixelech.

Kontext neboli obsah plátna jsem získal z elementu canvas, kde je pro každé plátno právě jeden kontext. Kontext plátna s vykreslovacím režimem jsem nastavil funkcí getContext. V JavaScriptu jsem zadal následující příkazy:

var canvas = document.getElementById("htmlcanvas");

var context = canvas.getContext("2d");

Jakmile jsem získal kontext Canvas, tak jsem mohl začít kreslit. Základním objektem je obdélník, který lze nakreslit třemi různými metodami, a to: fillRect, strokeRect a clearRect. Argumenty jako výška x a šířka y jsou pro všechny tři metody stejné. Rozdíl je ve vykreslení, kdy funkce fillRect obdélník vyplní, strokeRect znázorní pouze jeho obrys a clearRect oblast obdélníku smaže.

Čáry jsem kreslil na plátno pomocí tzv. cest, které je nutné začít a uzavřít. Funkcí beginPath() jsem začal cestu tvořit a uzavřel jsem ji pak funkcí closePath(). Čáru jsem vykreslil díky metodě lineTo(x, y). Pozici jsem určil funkcí moveTo(x, y) od poslední pozice kurzoru. Nejdříve dojde k přesunu kurzoru na pozici a pak funkce fill() vyplní cestu barvou, nebo funkce stroke() vykreslí čáry z cesty. Cesta se musí uzavřít, protože se takto dají kreslit vlastní tvary.

Další, co jsem na plátno kreslil jsou kruhy, kružnice a jejich výseče díky funkci context.arc(). Absolutní pozici stanovují argumenty x a y a rádius udává poloměr kružnice. Počáteční úhel určuje úhel v radiánech, od kterého se má kružnice, resp. výseč, vykreslit. Pro nastavení je důležité vědět, že obvod kružnice je $2*\pi$ a stupně převedeme na radiány následovně: $(\pi/180)$*stupně. Směr je logickou hodnotou k vykreslení po či proti směru hodinových ručiček. Hodnota true vykresluje po směru hodinových ručiček a false proti směru hodinových ručiček. Kruh tedy nakreslíme takto: context.arc(100, 100, 80, 0, Math.PI*2). Na níže uvedeném obrázku 3.2 je znázorněn výsledek mnou zadaného příkazu.

Obrázek 3.2: HTML Canvas kruh

Vzhled kresby upravují styly pro vyplnění (fill) a vykreslení obrysu (stroke). Můžeme je použít na objekty, které jsme již zmínili výše. Metody vzhledu pracují se dvěma základními proměnnými, a to jsou fillStyle a strokeStyle, jejichž hodnoty jsou zápisy barev. Když chceme změnit vzhled jiných věcí, můžeme využít dalších dostupných proměnných, např. ke změně velikosti vykreslených čar proměnné lineWidth, ke změně stylu zakončení čar lineCap s hodnotami "butt", "round", "square".

Krom barvy vykreslení můžeme stylovat i jiné věci. Například pro styly čar jsou dostupné proměnné lineWidth - velikost vykreslených čar, lineCap - zakončení čar. Hodnoty jsou butt, round, square, kde hodnota butt znamená zakončení bez přesahu, round zakulacené zakončení a square zakončení rovné s přesahem.

Plátno lze využít nejen ke kreslení na něj, ale také k vykreslení externího obrázku, který načteme z objektu Image či z tagu img. Načtení obrázku zjistíme v DOM klasicky funkcí onload a obrázek připevníme na plátno metodou drawImage(obrazek, x, y). Metoda drawImage nabízí variabilní práci s obrázkem, jako například zobrazení opakování obrázku na jednom plátně, změnu velikosti obrázku, tvorba výřezu obrázku.

K popiskům ke grafům či obrázkům lze vykreslit na plátno kromě tvaru i text pomocí základní funkce fillText(text, x, y). Absolutními pozicemi jsou x a y a text vyjadřuje textový řetězec. Je však bez stylů a malý, proto pomocí proměnné context.font jej můžeme nastavit stejně jako u CSS zápisu při zadávání font a velikosti textu.

Vzhledem k tomu, že práce s plátcem HTML5 Canvas nesplňovala veškeré požadavky na mnou vypracovávanou aplikaci, tak jsem od této možnosti upustil a využil jsem místo ní knihovnu Leaflet. Práci s touto knihovnou popisuji podrobněji v dalším textu.

3.1.5 Flask

Z webových frameworků jsem použil Flask, dle zadání projektu a protože se řadí mezi nejjednodušší frameworky na pochopení. Flask jsem naistaloval do virtualenvu, jak je uvedeno v obrázku 3.3.

```
C:\Users\Uživatel>cd C:\virtualenv\bakalar\Scripts

C:\virtualenv\bakalar\Scripts>activate

(bakalar) C:\virtualenv\bakalar\Scripts>python -m pip install Flask
```

Obrázek 3.3: Instalace Flasku

Po dokončení implementace programu je potřeba pro start webové aplikace nejdříve nastavit proměnnou FLASK_APP, která říká Flasku, kde aplikaci najít. On pak hledá v souboru proměnnou app. Ladicí režim pak nastavuje proměnná FLASK_DEBUG, který se nevyužívá při běžné provozu. Jelikož jsem vyvíjel aplikaci v operačním systému WINDOWS musel jsem využít příkaz SET namísto EXPORT, jak je uvedeno na obrázku 3.4.

```
(bakalar) C:\virtualenv\bakalar\Scripts>set FLASK_APP=helllo.py

(bakalar) C:\virtualenv\bakalar\Scripts>set FLASK_DEBUG=1

(bakalar) C:\virtualenv\bakalar\Scripts>flask  run
 * Serving Flask app "helllo.py" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 262-224-592
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Obrázek 3.4: Nastavení Flasku

V programu jsem do proměnné app vytvořil flaskovou aplikaci. Flask potřebuje vyhledávat soubory jako jsou šablony (templates) nebo sdílené soubory ve složce static, k čemuž slouží jméno modulu získané argumentem ___name___.

Z důvodu bezpečnosti a správného chodu aplikace kontroluje formáty vkládaných obrázků pomocí funkce allowed_file. Mezi povolené formáty patří .png, .jpg, .jpeg. Architektura stránky je na obrázku 3.5 viděna pomocí dekorátoru @app.route. Tato funkce pro danou URL vrací cestu k souboru, který je k této adrese určený.

Ke změně URL adresy jsem využil funkce redirect(url_for()), kde funkce redirect vyvolá přesměrování na adresu získanou z funkce url_for. U funkce url_for můžeme přidat argument, jak je uveden na obrázku 3.5, kde se přenáší jméno obrázku, které je zapotřebí pro jeho načtení na cílové stránce.

Pro přehlednost se nekombinuje psaní HTML do programu Flask, ale používají se tzv. templates šablony, které obsahují HTML popis stránky doplněný o případné další argumenty. Funkce render_template pošle Flasku danou HTML šablonu s možnými argumenty.

```
app = Flask(__name__)

UPLOAD_FOLDER = './static'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/uploaded')
def uploaded():
    fname = request.args['filename']
    return render_template('pokus2.html', fname=fname)

@app.route('/')
def loadhome():
    return redirect(url_for('home'))

@app.route('/home', methods = ['GET', 'POST'])
def home():
    if request.method == 'POST':
        file = request.files['file']
        if file.filename == '':
            err=1;
            return render_template('home.html', err=err)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            fname = filename
            print(fname)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            return redirect(url_for('uploaded',
                                    filename=filename))
        if file and not allowed_file(file.filename):
            err2=2;
            return render_template('home.html', err=err2)
    return render_template('home.html')
if __name__ == '__main__':
   app.run(debug = True)
```

Obrázek 3.5: Flask kód

Jedním z argumentů, který je uveden na obrázku 3.6 a který jsem využil ke své práci byl název obrázku, který potřebuji vykreslit, abych ho mohl následně anotovat.

```
img.src = './static/{{ fname }}';
```

Obrázek 3.6: využití argumentu funkce render_template

3.1.6 Leaflet

Po nadměrné složitosti implementace skrze HTML5 Canvas při jakékoliv editaci anotovaných objektů pro vytváření objektů, jsem zvolil Leaflet, který nabízí ve svém rozšíření nabízí knihovnu Leaflet.editable[11]. Tato knihovna umí vytvářet geometrické obrazce a přímo je editovat.

Leaflet je oblíbená knihovna s open source kódem JavaScript k tvorbě webových aplikací. Již v roce 2011 podporoval většinu mobilních a stolních platforem s dále HTML5 a CSS3. Knihovna byla vytvořena Vladimirem Agafonkinem a je stále zdokonalována.

Obdobně jako Leaflet funguje knihovna od OpenLavers, avšak práce s touto knihovnou je náročnější. Obě jsou open source, avšak Leaflet je mnohem menší cca 7000 řádků kódu, oproti OpenLavers, která existuje od roku 2015 a obsahuje cca 230 000 řádků. Za zmínku stojí také srovnatelná Google Maps API s uzavřeným zdrojem, která je rychlá a jednoduchá, ale není flexibilní.

Základním zobrazovacím modelem v knihovně Leaflet je mapa, v mém případě obrázek, která může obsahovat žádnou či více vrstev, přičemž na povrchu je zobrazen žádný nebo více vektorových objektů. Ke své práci jsem využil několik z hlavních prvků Leafletu, a to z rastrových typů ImageOverlay, z vektorový typů (bod, křivku, polygon, kruh a obdelník), a z ovládacích prvků Zoom a Layers.

Inicializace

Vzhledem k tomu, že jsem chtěl použít knihovnu Leaflet, tak jsem nejdříve na ni odkázal pomocí příkazů, které jsou obsahem obrázku 3.7. Dále jsem vytvořil element div s indikátorem map, do kterého se obrázek vykreslí.

```
<link rel="stylesheet" href="../static/leaflet/leaflet.css" />
<script src="../static/leaflet/leaflet.js"></script>
<script src="../static/leaflet/Leaflet.Editable/src/Leaflet.Editable.js"></script>
```

Obrázek 3.7: Odkazy na knihovnu Leaflet

Na obrázku 3.8 je zobrazeno vytvoření mapového objektu, který je vložen do připraveného elementu map. Proměnná L znázorňuje přístup k samostatné knihovně Leaflet. Abych nepoužíval mapy, které byly v Leafletu vytvořeny, existuje možnost využití funkce CRS.Simple, která umožňuje za specifického nastavení použít vlastní podklad zobrazení. Následuje nahrání obrázku, jehož jméno jsem získal argumentem flaskové funkce render_template ze sdílené složky static. Po načtení obrázku dojde k inicializaci mapového elementu tak, aby využil celého prostoru prohlížeče. Obrázek se vykreslí v jeho původní velikosti uprostřed mapového elementu.

```
var map = L.map('map', {
    crs: L.CRS.Simple,
    minZoom: -5,
    editable: true
});

var img = new Image();
img.src = './static/{{ fname }}';
img.onload = function() {
    var x = this.height;
    var y = this.width;

    var bounds = [[0,0], [x,y]];
    var image = L.imageOverlay(img, bounds).addTo(map);

    map.fitBounds(bounds);
```

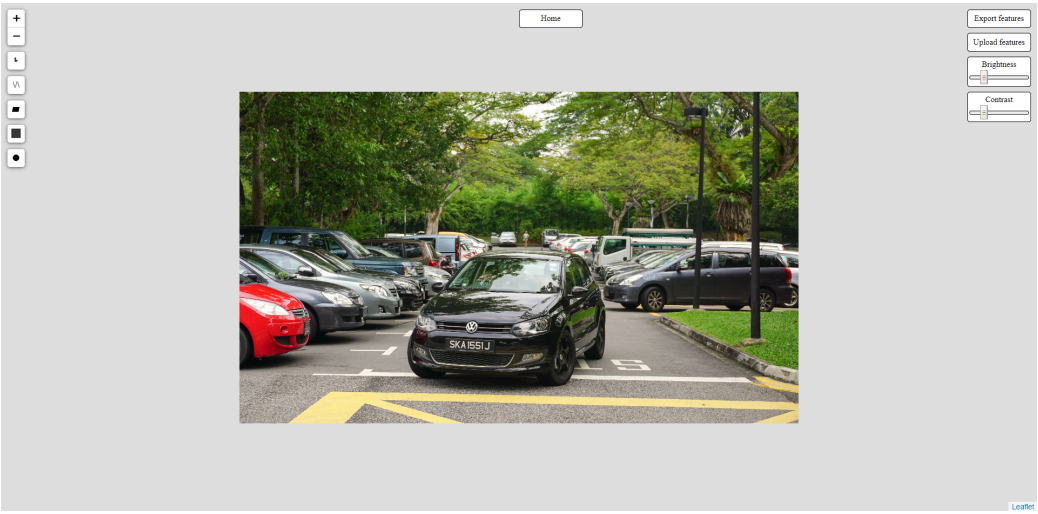
Obrázek 3.8: Vytvoření mapového objektu

K pohybu a zoomování obrázků je vhodná knihovna Leaflet, která se používá pro různé mapy a vytváření značek, objektů na mapě, avšak k samotné editaci slouží knihovna Leaflet.Editable, která umožňuje práci s objekty na obrázku, zejm. s nimi pohybovat, editovat je. Z tohoto důvodu jsem knihovnu Leaflet.Editable využil ke své práci.

Architektura aplikace

Hlavní stránka aplikace umožňuje nahrání obrázku sloužícího k anotaci ve formátu: .jpg, .jpeg, .png. Po nahrání obrázku se načte Leaflet image overaly s daným obrázkem. Po pravé straně obrazovky se nachází Leaflet control panel pro zoomování a vytváření nových editovatelných objektů. Pokud by se jednalo o již anotovaný obrázek, příslušné anotace k tomuto obrázku uložené ve formátu JSON lze nahrát zmačknutím tlačítka Upload features, které se nachází na levé straně obrazovky. Po levé straně máme dále tlačítko Export features pro uložení objektů s anotacemi. Je zde také možné nastavit css filter pomocí brightness a contrast sliderů. Nahoře uprostřed obrazovky se je umístěno tlačítko Home pro zpětné navrácení na úvodní stránku, za účelem načtení nového obrázku.

Obrázek 3.9: Domovská stránka



Obrázek 3.10: Pracovní stránka

Vytváření Leaflet objektů

Pro vytváření objektů lze v knihovně Leaflet použít jednoduché příkazy pro vytvoření objektů na určitých souřadnicích. Jelikož jsem potřeboval, aby se daly tyto objekty upravovat a případně posouvat na obrázku podle uživatelských nároků, využil jsem možnost vytvořené knihovny Leaflet.Editable, která všechny zmíněné požadavky umožňuje. Vytvoření objektů v knihovně Leaflet.Editable lze provést pomocí kontrolního panelu na levém horním rohu obrazovky. Uživatel si vybere tlačítko dle znaku, který symbolizuje objekt. Dostane se do funkce kreslení, kdy stačí pomocí kliknutí či tahu myši jednotlivé geometrické obrazce vytvořit. Při tvoření křivky či polygonu se ukončí kreslení dalšího bodu kliknutím na nějaký již vytvořený bod tohoto obrazce. Na obrázku 3.11 je vidět, jak se přidá tlačítko do kontrolního panelu pro vytvoření křivek.

```
L.NewLineControl = L.EditControl.extend({

    options: {
        position: 'topleft',
        callback: map.editTools.startPolyline,
        kind: 'line',
        html: '\\/\\'
    }
});
```

Obrázek 3.11: Vytváření křivky

Po kliknutí na tlačítko nové křivky se spustí metoda knihovny startPolyline, která je vidět na obrazku 2. Ostatní obrazce fungují na stejném principu jako výše popsané vytvoření křivky.

```
// method startPolyline(latlng: L.LatLng, options: hash): L.Polyline
// Start drawing a Polyline. If `latlng` is given, a first point will be added. In any case, continuing on user click.
// If `options` is given, it will be passed to the Polyline class constructor.
startPolyline: function (latlng, options) {
    var line = this.createPolyline([], options);
    line.enableEdit(this.map).newShape(latlng);
    return line;
},
```

Obrázek 3.12: Leaflet.Editable funkce startPolyline

Při vytváření kruhů v knihově Leaflet.Editable docházelo k problému s tím, že poloměr neodpovídal vzdálenosti vyznačené myší. Kruh se vykreslil přes celé pozadí plochy. Jelikož jsem využil funkci CRS.Simple, tak docházelo k chybnému vykreslení kruhu na pozadí plochy. Funkce na vykreslení obrázků v knihově Leaflet defaultně předpokládá, že pracuje s funkcí L.CRS.Earth a nastaví poloměr jako výsledek funkce L.LatLng#distanceTo. Souřadnicový systém tudíž předpokládá zkreslení mapy, protože obrázek je plochý a nemá zkreslení jako mapa, musíme tuto funkci upravit. Úpravu jsem provedl zadáním kódu:

L.LatLng.prototype.distanceTo = function (other) {
var dx = other.lng - this.lng;
var dy = other.lat - this.lat;
return Math.sqrt(dx*dx + dy*dy);
}

Anotování a smazání objektů

Anotace jsem řešil pomocí tzv. Popupů, které je možné připnout k jednotlivým geometrickým útvarům. Pro přidání Popup informace slouží funkce bindPopup(). Anotace jednotlivých geometrických obrazců lze přidat pomocí kombinace klávesy Shift a kliknutí myši na daný obrazec. Smazání objektu docílíme kombinací stisknutí klávesy Ctrl a kliknutím myši na daný objekt.

```
var editShape = function (e) {
    if (e.originalEvent.ctrlKey && this.editEnabled()) map.removeLayer(this);
    if (e.originalEvent.shiftKey && this.editEnabled()) {
        var anotinfo = prompt("Please enter your info:", "Harry Potter");
        if ((anotinfo != null) && (anotinfo != "")) {
            this.bindPopup(anotinfo);
        }
    }
};

var editMarker = function (e) {
    if (e.originalEvent.ctrlKey && this.editEnabled()) map.removeLayer(this);
    if (e.originalEvent.shiftKey && this.editEnabled()) {
        var anotinfo = prompt("Please enter your info:", "Harry Potter");
        if ((anotinfo != null) && (anotinfo != "")) {
            this.bindPopup(anotinfo);
        }
    }
};

map.on('layeradd', function (e) {
    if (e.layer instanceof L.Path) {e.layer.on('click', L.DomEvent.stop).on('click', editShape, e.layer)}
    else {e.layer.on('click', L.DomEvent.stop).on('click', editMarker, e.layer)};
    if (e.layer instanceof L.Path) e.layer.on('dblclick', L.DomEvent.stop).on('dblclick', e.layer.toggleEdit);
});
```

Obrázek 3.13: Anotování objektů

Ukládání projektů

Leaflet obsahuje strukturu, která nabízí informace o veškerých objektech vytvořených na obrázku, tudíž pomocí přenosu do JSON struktury následné jeho zřetězení nebo přeměně struktury na řetězec můžu tyto informace uložit do souboru, který můžeme načíst a pokračovat v práci. Přesnější popis funkcionality jsem napsal níže.

Zapsání všech objektů a jejich anotací do formátu JSON a následné uložení

Pro uložení všech nakreslených objektů a jejich anotací slouží tlačítko Export features po pravé straně obrazovky. Knihovna Leaflet obsahuje třídu Layer, která je nadřízená všem objektům. Proto třída map obsahuje funkci eachLayer, pomocí které se prochází skrz všechny nakreslené objekty. Zjistí se, o jaký geometrický útvar se jedná, a následně se vytvoří JavaScriptový objekt se souřadnicemi jednotlivých bodů(vrcholů) tohoto útvaru. Uložení souřadnic je nezbytné pro případné znovu nakreslení geometrických útvarů. K těmto souřadnicím se do JavaScript Objektu ukládají taky anotace. K získávání anotace slouží funkce getPopup(), která získá přístup k Popup části nakresleného útvaru, následně funkce getContent() získá námi požadovaný řetězec. Následně se vkládají jednotlivé JavaScript objekty podle obrazce, o který se jedná, do polí k nim vytvořeným. Proměnná skip slouží pro přeskočení určitého počtu bodů(vrcholů), které jsou součástí daného útvaru. Je to proto, že funkce iteruje i přes Markery, které jsou součástí třídy, například Rectangle. Pokud je na obrázku nakreslený jen jeden obdélník, tak funkce eachLayer vrátí jako první třídu Rectangle a následně čtyři vrcholy tohoto obdélníku jako třídu Marker. Tento příklad můžete vidět na obrázku 3.14.

```
map.eachLayer(function(layer) {
    if (layer instanceof L.Rectangle) {
        pom = layer.getLatLngs();
        popup = layer.getPopup();
        if (popup == undefined) {
            content = "";
        }
        else {
            content = popup.getContent();
        }
        irectangle = {latlngs: pom.toString(), text: content};
        rectangles.push(irectangle);
        skip = 4;
    }
```

Obrázek 3.14: Ukládání obdelníků

Nakonec se vytvoří objekt se všemi danými poli, který se pomocí funkce JSON.stringify() převede do řetězce ve formátu JSON a následně je poskytnut ke stažení jako soubor data.json. Pro lepší porozumění je vše vidět na obrázku 3.15.

```
var annotations = {markers: markers, rectangles: rectangles, circles: circles, polygons: polygons, polylines: polylines};
var convertedData = 'text/json;charset=utf-8,' + encodeURIComponent(JSON.stringify(annotations));
document.getElementById('export').setAttribute('href', 'data:' + convertedData);
document.getElementById('export').setAttribute('download','data.json');
```

Obrázek 3.15: Ukládání do JSON formátu

Načtení JSON zápisu ze souboru a následné nakreslení objektů a přidání jejich anotací

Načtení souboru s anotacemi spustíme tlačítkem Upload features. Pro přečtení souboru jsem využil JavaScriptovou funkci Filereader(). Po parsování JSONu na JavaScript objekt se zkontroluje typ objektu, a podle něho získat souřadnice a anotace k jednotlivým geometrickým útvarům. Tahle funkcialita je vidět na obrázku 3.16.

```
var fileTypes = ['txt','json'];
function readSingleFile(e) {
    var file = e.target.files[0];
    if (!file) {
        alert("You didn't select file.")
        return;
    }
    var extension = e.target.files[0].name.split('.').pop().toLowerCase();
    if (fileTypes.indexOf(extension) <= -1) {
        alert("Supported file types are: .txt, .json")
        return;
    }
    var reader = new FileReader();
    reader.onload = function(e) {
        var contents = e.target.result;
        drawUploadedFeatures(contents);
    };
    reader.readAsText(file);
}
```

Obrázek 3.16: Načtení JSON pomocí funkce FileReader

Na obrázku 3.17 je zobrazen příklad zpracování všech kruhů z nahraného JSON souboru. Nejprve se pomocí regexu získají z řetězce souřadnice středu. Na rozdíl od ostatních objektů je u kruhů potřeba radius, neboli poloměr. Poslední informací, která se získává, je případná anotace jednotlivých objektů, jenž je uložená v proměnné item.text.

```
if (obj.circles.length > 0) {
    obj.circles.forEach(drawCircles);
    function drawCircles(item, index) {
        var pommatches = item.latlng.match(/\d+\.*\d*/g);
        var pomlat = pommatches[0];
        var pomlng = pommatches[1];
        var circ = L.circle([pomlat, pomlng], {radius: item.radius}).addTo(map);
        circ.enableEdit();
        if (item.text != "") {
            circ.bindPopup(item.text);
        }
    }
}
```

Obrázek 3.17: Nakreslení kruhů ze souboru

Slidery na jas a kontrast

Slidery na jas a kontrast jsou zhotoveny přes HTML input type range, kde lze nastavit minimální a maximální hodnotu a taky o kolik se mají hodnoty změnit při jednom kroku. Jedná se o nastavování css filtru na jas a kontrast.

```
<div id="cntdiv">
    <label for="contrast-slider" id="contrast-label">Contrast</label>
    <input type="range" id="contrast-slider" min="0" max="500" step="0.5" value="100">
</div>
<script>

    var sliderc = document.getElementById('contrast-slider');
    sliderc.addEventListener('input', function(e) {
        var value = e.target.value;
        var imgoverlay = document.getElementsByClassName('leaflet-image-layer');
        var pomsliderb = document.getElementById('brightness-slider');
        imgoverlay[0].style.filter = 'contrast(' + value + '%) brightness(' + pomsliderb.value + '%)';
    });
```

Obrázek 3.18: Kontrast slider

Kapitola 4

Testování

Testování nově navržené aplikace včetně všech jejich funkcionalit je důležité pro ochranu uživatele i vývojáře. Zjistí se jím nedostatky, které by mohly způsobit případné chyby. Takové chyby by bez jejich odstranění mohly stát provozovatele nemalé finanční prostředky.

Projekt testujeme automaticky v průběhu jeho spouštění, např. zatížením serveru. Tester při splnění zadaných požadavků zahájí testování jednotlivých částí aplikace. Takto realizované testy uloží a vyznačí v nich chyby, po jejich opravě se provede kontrolní test.

Testování funkčností aplikace

Testování funkčnosti aplikace je důležité pro získání zpětné vazby a informací vedoucím k vylepšení aplikace. Ohledně testování jsem oslovil respondenty se zkušenostmi s anotováním obrázků. K vyzkoušení celé aplikace jsem vytvořil seznam níže uvedených aktivit:

1. Výběr obrázku určeného k anotaci
2. Zoom obrázku
3. Posuny obrázku
4. Zahájení anotace
5. Výběr typu anotace
6. Výběr nástroje polygon, označení vybraného segmentu a ukončení kresby polygonu
7. Úprava tvaru polygonu
8. Vložení popisu do polygonu
9. Uložení vytvořené anotace 1
10. Smazání polygonu
11. Načtení uložené anotace 1
12. Výběr typu anotace 2
13. Výběr nástroje obdélník, označení vybraného segmentu

14. Úprava tvaru obdélníku
15. Vložení popisu do obdélníku
16. Uložení vytvořené anotace 2
17. Výběr typu anotace 3
18. Výběr nástroje kruh, označení vybraného segmentu
19. Úprava tvaru kruh (nastavení poloměru, posun)
20. Vložení popisu do kruhu
21. Uložení vytvořené anotace 3
22. Návrat na úvodní obrazovku aplikace

Průběh testování s respondenty

Testování se zúčastnili dva respondenti v počítačové místnosti s minimálním rušením. Probíhalo na zařízeních respondentů s operačním systémem Windows 10. Zadané úkoly jsem předal respondentům ke splnění. Poznámky v průběhu testování zapisovali na papír. Na závěr jsem zjišťoval jejich dojmy a pocity z práce s aplikací jako celkem.

Číslo	Otázka	Odpověď
1.	Je podle Vás aplikace uživatelsky příznivá?	Ano, pracovalo se mi s ní dobře, je intuitivní.
2.	Vyhovovalo Vám rozložení ovladačů na obrazovce?	Ano, jsou umístěny přehledně po okrajích obrazovky.
3.	Co Vás z nabídky aplikace nejvíce potěšilo?	Nejvíce se mi líbil, vykreslený kruh, který se dal ještě zvětšit a umístit do něj popis.
4.	Co se Vám na aplikaci nelíbilo a proč?	Chyběla mi možnost zvýšení jasu tmavého obrázku.
5.	Máte nějaký nápad, jak aplikaci vylepšit?	Navrhl bych rozšíření nabídky aplikace o zvýšení jasu.

Tabulka 4.1: Odpovědi na otázky 1. respondenta

Číslo	Otázka	Odpověď
1.	Je podle Vás aplikace uživatelsky příznivá?	S aplikací se pracovalo pohodlně.
2.	Vyhovovalo Vám rozložení ovladačů na obrazovce?	Ano.
3.	Co Vás z nabídky aplikace nejvíce potěšilo?	Úprava jasu, zoom obrázku a popis.
4.	Co se Vám na aplikaci nelíbilo a proč?	Aplikace neobsahuje úpravu kontrastu.
5.	Máte nějaký nápad, jak aplikaci vylepšit?	Doplnit úpravu kontrastu.

Tabulka 4.2: Odpovědi na otázky 2. respondenta

Úprava aplikace

Na základě poznámek respondentů v průběhu testování aplikace a na základě odpovědí na otázky respondentů jsem do aplikace zapracoval možnost úpravy jasu a kontrastu. Tuto úpravu rozbírám v aplikační části práce s názvem Slidery na jas a kontrast. Nastavil jsem, o kolik se mají hodnoty měnit v jednotlivých krocích. Tím jsem splnil obě doporučení, která mi respondenti dali během testování.

Vlastní testování

Po dokončení aplikace a po zapracování doporučení respondentů jsem provedl ještě její vlastní testování. Zadané úkoly, včetně doporučení, aplikace optimálně splnila, načetla výběr obrázku, jednotlivé tvary, které jsou v její nabídce (bod, křivka, polygon, obdélník, kruh) vykreslila, doplnila je o požadovaný popis a vytvořené anotace uložila. Aplikace fungovala pružně a plynule.

Kapitola 5

Závěr

Cílem této bakalářské práce je zpracování projektu webové aplikace, která umožňuje vytváření anotací obrazových dat podle uživatelské konfigurace, například vozidlo pomocí tahu myší, město pomocí bodu, vzdušnou vzdálenost bodů na mapě pomocí přímky. Aplikace umožňuje uživateli obrázky či segmenty nahrát do aplikace, anotovat je a následně uložit s možností jejich podrobnějšího popisu. S takto nahraným obrázkem může uživatel manipulovat po plátně, lze pomocí aplikace zvětšit či zmenšit, upravit jeho jas a kontrast, čímž zlepší jeho čitelnost. Po úpravě obrázku si uživatel může vybrat z kreslících segmentů bod, křivku, polygon, obdélník či kruh, kterým vyznačí část obrázku. Z detailu obrázku člověk může zjistit bližší informace o daném objektu jako registrační značku vozidla, typ vozidla, popis člověka, popis okolí, místo výskytu objektu na obrázku. Po vyznačení části obrázku lze tuto část upravovat a vložit k ní popis.

Celou prací se prolíná popis základních elementů aplikace, metod, postupů, jazyků, které jsem přímo využil či přímo souvisí se webovou aplikací pro anotaci obrazových dat. Na základě výběru jsem pro zhotovení daného problému vybral jednu z nich a postup programování aplikace jsem v implementační části práce rozvedl po jednotlivých krocích. Důležitou součástí bylo přiblížit téma čtenáři práce, aby se dokázal v dané problematice pohybovat. Z důvodu vypsání více metod, jak problematiku řešit, si může čtenář vybrat dle svých preferencí. Po přečtení zhotovené rešerše by měl čtenář získat základní přehled o tom v jakém prostředí, s jakým typem dat a s jakými programátorskými jazyky lze pracovat.

K tvorbě aplikace jsem využil virtual environment. Do environmentu jsem stáhl a nainstaloval Python a Flask, které byly v zadání práce. Samotné implementování programu probíhalo za pomocí programu Sublime Text 3, který zvýrazňuje syntaxi jazyků HTML, JavaScript, Python, které jsem k tvorbě aplikace použil. Jedním z možných nástrojů pro vytváření anotací v obraze je skrze HTML5 Canvas, který jsem nejdříve vyzkoušel. Vzhledem k tomu, že práce s plátcem HTML5 Canvas nesplňovala veškeré požadavky na mnou vypracovávanou aplikaci, tak jsem od této možnosti upustil a využil jsem místo ní knihovnu Leaflet, který nabízí ve svém rozšíření nabízí knihovnu Leaflet.editable. Tato knihovna umí vytvářet geometrické obrazce a přímo je editovat. Prostřednictvím této knihovny jsem postupně aplikaci inicializoval, vytvořil jsem architekturu, vytvořil Leaflet objekty, vytvořil anotování a smazání objektů, uložení projektů přes zapsání do JSON a uložení a následné načtení JSON zápisu.

Celou aplikaci jsem testoval na funkčnost a intuitivnost dvěma nezávislými respondenty a jejich doporučení (jas a kontrast) byla do aplikace zapracována. Po zapracování doporučení jsem rozšířenou funkčnost aplikace opakovaně otestoval. S ohledem na funkčnost aplikace,

její hodnocení v testech, rozšíření její funkčnosti a možnost dalšího využití aplikace jsem přesvědčen, že cíl práce byl splněn.

Literatura

[1] *Cross Site Request Forgery* [online]. [cit. 2020-05-27]. Dostupné z: https://www.alten.com/wp-content/uploads/2019/09/cross-site-request-forgery-example-1.png.

[2] *Grafická data* [online]. [cit. 2020-05-27]. Dostupné z: http://mathonline.fme.vutbr.cz/pg/flash/TeorieGrafika/pocGrafika1.pdf.

[3] *Grafická data* [online]. [cit. 2020-05-27]. Dostupné z: https://mathonline.fme.vutbr.cz/Graficka-data/sc-1194-sr-1-a-182/default.aspx.

[4] *Slovník cizích slov* [online]. [cit. 2020-05-27]. Dostupné z: https://slovnik-cizich-slov.abz.cz/.

[5] *Sorted XSS* [online]. [cit. 2020-05-27]. Dostupné z: https://www.imperva.com/learn/wp-content/uploads/sites/13/2019/01/sorted-XSS.png.

[6] *What is a web tool?* [online]. [cit. 2020-05-27]. Dostupné z: https://pro.arcgis.com/en/pro-app/help/analysis/geoprocessing/share-analysis/what-is-a-web-tool.htm.

[7] *What is image anotation?* [online]. [cit. 2020-05-27]. Dostupné z: https://www.quora.com/What-is-image-annotation.

[8] *Best text editor* [online]. 2019 [cit. 2020-05-27]. Dostupné z: https://cs.bccrwp.org/compare/best-text-editor-atom-vs-sublime-vs-visual-studio-code-vs-vim-341dc7/.

[9] Barth, A., Jackson, C. a Mitchell, J. C. *Robust Defenses for Cross-Site Request Forgery* [online]. 2008 [cit. 2020-05-27]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/1455770.1455782?fbclid=IwAR2SXjWoC1PIN7oKmqXodWGnMi3WmhzAzQEr8cmIT72vORvCI2Xxv-5Vx9k.

[10] Bittner, J. *Sublime Text* [online]. 2016 [cit. 2020-05-27]. Dostupné z: https://www.itnetwork.cz/software/sublime-text.

[11] Boniface, Y. *Leaflet.Editable* [online]. 2019 [cit. 2020-05-27]. Dostupné z: https://github.com/Leaflet/Leaflet.Editable.

[12] Burget, R. *HTML* [online]. 2018 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp03b_HTML.pdf&cid=12157.

[13] Burget, R. *JavaScript* [online]. 2018 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp08_JavaScript.pdf.

[14] Burget, R. *Pokročilá vizualizace* [online]. 2018 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp07_Pokrocila_vizualizace.pdf&cid=12157.

[15] Burget, R. *Dynamické stránky v PHP* [online]. 2019 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp03_PHP.pdf.

[16] Burget, R. *Informační systémy - Architektury IS* [online]. 2019 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp02_Architektury.pdf.

[17] Burget, R. *Vizualizace a serializace dat* [online]. 2019 [cit. 2020-05-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp06_Vizualizace_serializace.pdf.

[18] Burget, R. *Přednášky předmětu Tvorba webových stránek ITW* [online]. 2020 [cit. 2020-05-27]. Dostupné z: https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/.

[19] Flanagan, D. *JavaScript: The Definitive Guide* [online]. 2011 [cit. 2020-05-27]. Dostupné z: https://books.google.cz/books?id=4RChxt67lvwC&printsec=frontcover&dq=javascript%20the%20definitive%20guide&fbclid=IwAR21J8xWCs_lbaHKwod9YQ8IlNj9o2P1snBzX-Q3IRfPXRAzxKqguyOKzmA#v=onepage&q=javascript%20the%20definitive%20guide&f=false.

[20] Forcier, J., Bissex, P. a Chun, W. J. *Python Web Development with Django* [online]. 2008 [cit. 2020-05-27]. Dostupné z: https://books.google.cz/books?id=M2D5nnYlmZoC&printsec=frontcover&dq=Python%20web%20development%20with%20Django&hl=cs&sa=X&ved=0ahUKEwjmka-ZwdTpAhVR4KYKHYTaBZcQ6AEINDAB&fbclid=IwAR1YDjC_yH0iFLG_3awGmxJMpFz3JIpMJURLES23ZLQ4hGlduGYmbFeU7gQ#v=onepage&q=Python%20web%20development%20with%20Django&f=false.

[21] Hanák, D. *Canvas aneb grafika JavaScriptem* [online]. 2013 [cit. 2020-05-27]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/zdrojove-kody/canvas-aneb-grafika-javascriptem.

[22] Janovský, D. *HTTP protokol* [online]. [cit. 2020-05-27]. Dostupné z: https://www.jakpsatweb.cz/server/http-protokol.html.

[23] Meyer, E. A. *Cascading Style Sheets: The Definitive Guide* [online]. 2004 [cit. 2020-05-27]. Dostupné z: https://books.google.cz/books?id=nU4EiJ9ZPf8C&printsec=frontcover&dq=Cascading%20style%20sheets&hl=cs&sa=X&ved=0ahUKEwj8l8KZvNTpAhVn1qYKHXG9AKMQ6AEIKDAA&fbclid=IwAR0omS3miSWp53fare_TLJSjmUawM7ya_N7tPyBUN9rO-xujXsXPSmuCkP0#v=onepage&q=Cascading%20style%20sheets&f=false.

[24] Pallets. *Flask documentation* [online]. 2010 [cit. 2020-05-27]. Dostupné z: https://flask.palletsprojects.com/en/1.1.x/?fbclid=IwAR1YDjC_yH0iFLG_3awGmxJMpFz3JIpMJURLES23ZLQ4hGlduGYmbFeU7gQ.

[25] Pilgrim, M. *HTML5: Up and Running: Dive into the Future of Web Development* [online]. 2010 [cit. 2020-05-27]. Dostupné z: https://books.google.cz/books?id=Mk3sW0on7OAC&printsec=frontcover&dq=Pilgrim%20Dive%20into%20HTML5&hl=cs&sa=X&ved=0ahUKEwjHoP7bvNTpAhUHJZoKHVoMA9MQ6AEIKzAA&fbclid=IwAR10NAsonLauzMrivw-0CPahlAstGwbj48v_U5rvoKslflKiFgSJqV9BV4U#v=onepage&q=Pilgrim%20Dive%20into%20HTML5&f=false.

[26] Purer, K. *PHP vs. Python vs. Ruby–The web scripting language shootout* [online]. 2009 [cit. 2020-05-27]. Dostupné z: https://www.researchgate.net/publication/228940751_PHP_vs_Python_vs_Ruby-The_web_scripting_language_shootout.

[27] Santos, A., Bader, D., Jablonski, J. a Herman, M. *Python Virtual Environments: A Primer* [online]. 2018 [cit. 2020-05-27]. Dostupné z: https://realpython.com/python-virtual-environments-a-primer/.

[28] Zadka, M. a Lefkowitz, G. *The Twisted Network Framework* [online]. [cit. 2020-05-27]. Dostupné z: https://twistedmatrix.com/users/glyph/ipc10/paper.html?fbclid=IwAR2OgQgbAqhSaCnSHPXPSOWp6sn9WUbjEZ5yGhPmXm35J9fbPqB-qmnNqoc.

[29] Černý, J. *Tři druhy webu: povrchový, hluboký a temný* [online]. 2017 [cit. 2020-05-27]. Dostupné z: https://www.informacnigramotnost.cz/tri-druhy-webu/.

Příloha A

Obsah přiloženého paměťového média

- xdvora2e.pdf - Technická zpráva ve formátu PDF
- xdvora2e-txt.zip - Zdrojové soubory technické zprávy
- xdvora2e-app.zip - Zdrojové soubory aplikace
  - README.txt - Manuál pro instalaci a spuštění
  - hello.py - zdrojový soubor
  - /templates/home.html - Domovská stránka aplikace
  - /templates/pokus2.html - Kreslící stránka
  - /static/leaflet - Adresář s knihovnou leaflet a Leaflet.Editable