

Gymnázium Josefa Jungmanna



Závěrečná práce

Webová aplikace

Vojtěch Netrh

© 2021 GJJ v Litoměřicích

Čestné prohlášení

Prohlašuji, že svou závěrečnou práci „Webová aplikace“ jsem vypracoval samostatně pod vedením vedoucího závěrečné práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené závěrečné práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Litoměřicích dne 29. dubna 2021

Poděkování

Rád bych touto cestou poděkoval Mgr. Pavlu Beránkovi za ochotu při vedení závěrečné práce a jeho cenné rady, paní Mgr. Evě Hrdličkové za korekturu po stránce jazykové i typografické.

Webová aplikace

Abstrakt

Závěrečná práce na téma „Webová aplikace“ spočívá ve vytvoření aplikace pro výpočet FINA bodů. Spojil jsem dohromady své dvě oblíbené aktivity (programování a sport), aby vznikla užitečná aplikace. V teoretické části jsem se věnoval především tématu otevřených dat a použitých technologií, v praktické část jsem navrhl řešení, a poté ho převedl v realitu. Pro realizaci byl použit Python (s frameworkem Flask) a typické moderní webové technologie, kterými jsou JavaScript a Bootstrap.

Klíčová slova

Python, Flask, XML, otevřená data, plavání.

Web application

Abstract

This thesis on the topic „Web application“ consist of creating an app for calculating FINA points. I combined my two favourite hobbies (programming and swimming) to create this useful application. In the theoretical part I focused on the topic of open data and used programming technologies, in the practical part I designed a solution and after that I transformed it into reality. For realization was used Python (with Flask framework) and modern web technologies like JavaScript and Bootstrap.

Keywords

Python, Flask, XML, open data, swimming.

Obsah

1.	Úvod.....	1
2.	Cíle.....	3
3.	Teoretická východiska	5
3.1.	Otevřená data	5
3.1.1.	Otevřená data obecně.....	5
3.1.2.	Přínos otevřených dat.....	5
3.1.3.	Zajímavé aplikace pro práci s otevřenými daty.....	6
3.1.4.	Otevřená data ve sportu	7
3.2.	Porovnání výkonů různých disciplín v plavání.....	7
3.2.1.	Fina body.....	7
3.3.	Webové technologie	8
3.3.1.	Co se děje, když otevíráme webové stránky?	8
3.3.2.	World Wide Web.....	9
3.3.3.	HTTP a HTTPS	10
3.3.4.	REST.....	10
3.3.5.	Moderní web	10
3.4.	Správa front-endu.....	11
3.4.1.	HTML	11
3.4.2.	CSS	12
3.4.3.	Bootstrap	12
3.4.4.	UX a UI design.....	12
3.5.	Správa back-endu.....	13
3.5.1.	Python	13
3.5.2.	Flask.....	13
3.5.3.	JavaScript	13
3.5.4.	Databáze.....	14
3.5.5.	NoSQL databáze.....	14
3.5.6.	XML.....	15
4.	Vlastní práce.....	17

4.1.	Shrnutí požadavků.....	17
4.2.	Implementace back-end.....	18
4.2.1.	Struktura souborů aplikace.....	19
4.2.2.	Python	19
4.2.3.	Flask.....	19
4.2.4.	JavaScript	21
4.3.	Práce s daty	24
4.3.1.	XML.....	24
4.3.2.	Propojení XML s Pythonem.....	25
4.4.	Implementace front-end.....	27
5.	Diskuse.....	29
5.1.	Problémy otevřených dat.....	29
5.1.1.	Je zveřejnění vždy přínosem?	29
5.1.2.	Sportovní odvětví	29
5.2.	Přínosy aplikace	29
5.3.	Budoucnost aplikace	30
5.4.	Komplikace při programování	30
6.	Závěr	31
7.	Seznam použitých zdrojů	33
7.1.	Textové zdroje	33
7.2.	Obrázky	36

Seznam obrázků

Obr. 1 – Jak funguje web	9
Obr. 2 – Responzivní vzhled.....	11
Obr. 3 – Návrh vzhledu.....	17
Obr. 4 – Schéma fungování aplikace.....	18
Obr. 5 – Rozdělení funkcí v Pythonu	20
Obr. 6 – Úprava časů a výpočet bodů.....	21
Obr. 7 – Výstup z Pythonu.....	21
Obr. 8 – AJAX druhá část.....	22
Obr. 9 – AJAX první část	22
Obr. 10 – Výměna vstupního pole.....	23
Obr. 11 – Vyčištění formuláře	23
Obr. 12 – Odpočítávací tabulka dnů.....	24
Obr. 13 – Import XML do Pythonu.....	26
Obr. 14 – Určení vhodného rekordu.....	26
Obr. 15 – Prohledávání XML.....	26
Obr. 16 – Vzhled aplikace	27

1.Úvod

V každém sportovním odvětví jsou v dnešní době důležité statistiky. To je zapříčiněno především téměř stejnou úrovní široké špičky sportovců, kde o vítězství rozhodují detaily. Velmi často se také různé pozice či disciplíny (ve stejném sportu) porovnávají. Nejpřesnější srovnání nám zpětně nabídne videozáznam, avšak ten je zdoluhavý a nemá žádný stanovený systém, podle kterého by byl hodnocen, proto se častěji používají statistiky psané (forma bodů nebo procent). Další pádný důvod ke zveřejnění těchto dat může být vzájemná kontrola dodržení pravidel, jelikož podle výsledků jsou sportovci odměňováni a oceňováni, a je tedy důležité, aby vše probíhalo podle pravidel. Kromě detailních statistik, které slouží především pro trenéry a jiné specialisty, je neméně podstatné zavést jednoduchou, avšak přesnou metodu porovnání i pro diváky. Přece především díky nim se daný sport stává zajímavý a může se posouvat kupředu. V plavání tuto funkci splňují tzv. FINA body. Body zavedená organizací FINA (*Fédération Internationale de Natation*, česky: *Mezinárodní plavecká federace*). Jednotka je hojně užívána mezi diváky (jednoduchá pro pochopení), ale i v profesionálním prostředí (klíč pro výběr plavců do reprezentace ČR). Jejich výpočet si chceme co nejvíce zjednodušit, k čemuž nám výborně poslouží webová aplikace. Žádná pravidelně aktualizovaná zatím neexistuje.

2. Cíle

Hlavní cíle této maturitní práce jsou celkem tři, dále je podrobně popíši. Nejdůležitějším z nich je naprogramování aplikace pro výpočet FINA bodů. Aplikace bude zpřístupněna na veřejné webové stránce pro všechny návštěvníky a měla by být schopna se na základě reklamy sama zaplatit (proplacení domény a webhostingu). Druhým cílem je naučit se pracovat s webovými frameworky a umět zakomponovat databáze do různých programů. Především vidím velkou budoucnost ve webových aplikacích díky jejich nenáročnému používání, pravidelné automatické aktualizaci, nízkými požadavky na hardware a jednoduchému používání i na mobilních zařízeních. Databáze také nachází uplatnění ve všech typech aplikací, navíc nejen v oboru IT (informační technologie), nýbrž i v jiných oborech. Nakonec bych rád ověřil dostupnost veřejných dat od sportovních organizací (v tomto případě FINA), jakými jsou různé statistiky, informace o světových rekordech, včetně jejich držitelů.

3. Teoretická východiska

3.1. Otevřená data

3.1.1. Otevřená data obecně

Jsou to informace, které jsou zveřejněné na internetu. Musí být úplná, snadno dostupná, obsahovat jméno autora, používat standardy s volně dostupnou specifikací a obsahovat minimum omezení používání. Pocházejí ze všech oblastí života – univerzit, nevládních organizací, vědy, státní správy či soukromých podniků (např.: jízdní řády, státní rozpočet, kalendáře státních představitelů, měření čistoty pitné vody či katastr nemovitostí). Největší zájem lidé projevují o data ze státní správy, jelikož se týkají všech a setkávají se s nimi každodenně. Celkově máme 5 tříd otevřených dat [3]. Čím je třída vyšší, tím jsou data považována za více otevřená. Na druhou stranu běžný uživatel si s prohlížením dat stupně 5 bude muset dát obrovskou práci, na rozdíl od stupně 1, kde jsou data například ve formě PDF souboru. Nejrozšířenějším typem jsou data textová (např.: uspořádaná do tabulek), dále mapy, audio či video záznamy atd. Jejich masivní rozšíření a zájem veřejnosti o ně přišel s rozmachem práva na duševní vlastnictví a digitalizací (příchod internetu a WWW), díky které se mohly spustit první weby (data.gov, data.gov.cz atd.). [1], [2]

3.1.2. Přínos otevřených dat

Podle mnoha institucí a lidí zabývajících se touto problematikou by publikace dat a jejich volná dostupnost měla umožnit posun ve více směrech [4]. Smysl této publikace a jejich přínos bych přirovnal ke konkurenci mezi komerčními společnostmi. Otevře se totiž prostor, kdy běžní občané či instituce mohou na data nahlédnout a vyvodit z nich různé závěry, určit predikci do budoucna, případně upozornit na podezřelé nebo nevhodné praktiky (např.: platové výměry, zakázky s nejasným dodavatelem atd.) a v neposlední řadě pomohou lidem v demokratických státech rozhodovat se ve volbách (jestliže má občan nesprávné údaje o chování vlády, nemůže správně vyhodnotit svou volbu, jako kdyby těmito daty disponoval).

Z hlediska státní správy se každý občan může více a přímo zapojit do chodu státu, což posiluje nezávislost a demokracii. Výzvu v tomto ohledu přináší publikace i pro soukromý sektor (konkrétně IT firmy), které budou vyvíjet aplikace pro práci s otevřenými daty, jež bude intuitivní pro každého občana (dat je veliké množství a orientace v nich nemusí být jednoduchá, zvláště pro nezainteresované lidi).

Tento přínos jsme si mohli potvrdit především v posledních měsících, kdy celý svět zužuje pandemii covidu-19. Otevřená data hrála a doteď hrají naprosto zásadní roli. Hned ze začátku se z Číny šířily klíčové informace se zpožděním a zkreslené, což se ukázalo jako jeden z problémů nedostatečné přípravy zbytku světa na pandemii [5], [6]. V České republice na alarmující čísla upozornil člověk, který se epidemiologickými scénáři nezaobírá, avšak jeho excelová tabulka spustila bleskovou reakci vlády [41]. V průběhu pandemie může být pozitivní vliv těchto informací v podobě zodpovědného chování občanů a dodržování opatření (např.: jestliže vláda nastaví určitá omezení a občan na statistikách uvidí jejich účinnost, psychicky ho to bude motivovat k jejich důslednému dodržování, případně k větší obezřetnosti i mimo povinné restriktce). Jako další změna, kterou tato data mohou odstranit, je zdravotní péče (očkování, léčba), okolo které se šíří mnoho dezinformací [7]. Je tedy důležité, aby si každý mohl přečíst klinické studie a nezávislé testy a sám se rozhodnout a vyvrátit případně dezinformace. A v neposlední řadě se dá z dat vycházet při zpětném hodnocení pandemie a postupů použitých v ní – zda byla opatření účinná, postupy dostatečně rychlé, co stálo za ohnisky nákazy, jestli byl stát připraven, jak probíhal nákup potřebného zdravotnického materiálu [8] (toto je položka, která může přinést velké problémy, jelikož většina této výroby se odehrává v Číně, což je země mimo EU, kde pro tyto prostředky platí jiné certifikace než u nás [9]).

3.1.3. Zajímavé aplikace pro práci s otevřenými daty

Pro českého občana bude jednou z nejzásadnějších webových stránek <https://data.gov.cz/>. Zmíněná webová stránka poskytuje otevřená data ze státní správy (ministerstva, kraje, obce a jiné orgány). Nedílným prvním krokem je digitalizace dat, která se v České republice pohybuje spíše pod evropským průměrem [10]. Dále pak už jen zbývá posoudit, zda jsou data vhodná ke zveřejnění (určité informace nemusí být, např.: z hlediska bezpečnosti), a pokud ano, musí stát mít zájem o jejich publikace, nebo k tomu musí být „donucen“ obyvateli, novináři nebo opozičními poslanci. Příkladem z poslední doby může být kauza v oblasti platů Hradní kanceláře [11]. Dále jen weby zmíním a nebudu více rozepisovat jejich použití, každý si je může sám vyzkoušet a posoudit jejich přínos nebo zajímavost.

- Hlídač státu (dostupné z: <https://www.hlidacstatu.cz/>)
- Katastr nemovitostí (dostupné z: <https://nahlizenidokn.cuzk.cz/>)
- Hodnocení veřejných zakázek (dostupné z: <https://www.zindex.cz/>)

3.1.4. Otevřená data ve sportu

Přestože se sportovní odvětví na první pohled nemusí zdát důležité pro zveřejňování a práci s otevřenými daty, je tomu právě naopak. Sport jakožto kompetitivní odvětví závisí nejen na nás samotných (z pohledu sportovce), ale také na soupeřích (jejich chování sice nemůžeme ovlivnit, zato potřebujeme mít důvěru v to, že i oni soutěží podle pravidel a nedopouští se žádných „přešlapů“). Tento problém mohou v jistých specifických případech vyřešit právě otevřená data, např.: detailní záznamy závodů/zápasů dostupné na stránkách federací/svazů či podrobné statistiky. V momentální době nejpodrobnější statistiky z plavání nabízí aplikace třetí strany swimrankings [12], která eviduje nejen plavecké mítinky, ale i profily většiny plavců z desítek zemí. Bohužel například záznamy světových rekordů (v celé délce) jsou především v poslední době zveřejňovány pouze za poplatek na Fina TV.

Momentálně stanovená pravidla plavání [13] neumožňují pro jejich kontrolu používat kamerové záznamy, jak je běžné v jiných sportech, nejčastěji pod pojmem „jestřábí oko“ (např.: ve fotbale, hokeji nebo nejznámější v tenise). Dopad této změny byl značný ve všech sportech, kterých se dotkla [14]. Sporné momenty se přestaly stávat spornými a celá hra byla více transparentní. V dnešní době sportu jsou maličkosti, jež rozhodují o konečném výsledku a kamerové záznamy bezpochyby patří mezi technologie, které změnily sportovní svět. V plavání všechna kontrola správné techniky způsobu zůstává na očích rozhodčího, což není vůbec jednoduché, především v dnešní rychlosti a práci pod vodou (okolo plavce se vytvoří bublinky vody přes které není vidět). Právě tento problém by mohly vyřešit kamery, jež se už vyskytují běžně i pod vodou, jelikož televize se snaží nabídnout co nejlepší záběry divákům. Výhoda videozáznamu spočívá především v několika bodech – můžeme si zvolit pomalejší rychlost přehrávání, kamery pokrývají více úhlů, jednotlivé úseky se dají přezkoumat více lidmi. Nejznámější případy, kdy nepovolené záběry pod vodou pomohly i ke světovému rekordu jsou shrnuty na tomto [videu](#) [42].

3.2. Porovnání výkonů různých disciplín v plavání

3.2.1. Fina body

Fina body (dále jen FP – Fina points) jsou parametr umožňující porovnání plaveckých výkonů napříč disciplínami. Nejvyšší možná hodnota je 1000 bodů (čím více tím lépe), i když to není úplně stoprocentní pravda, jelikož oněch 1000 bodů odpovídá světovému rekordu, tudíž plavec při jeho překonání tuto hranici překročí [např.: Světový rekord na 100 m prsa měl od roku 2012 hodnotu 58,46 (= 1000 FP), při jeho překonání v roce 2015 byl zaplavan čas 57,92,

který nabyl hodnoty 1028 FP]. Nové tabulky jsou vydávány vždy k novému roku, tzn. když opět použijeme poslední příklad zjistíme, že až v roce 2016 nabyl čas 57,92 hodnotu 1000 FP, tudíž kdyby ještě v roce 2015 někdo zaplavával čas pod 58,46 dosáhl by také více než 1000 FP.

Zjištění FP pro daný čas má dva možné způsoby. Hodnoty času nebo bodů se dá samozřejmě vyčíst v tabulek vydávaných FINA, což je ovšem zdlouhavé, proto je jednodušší použít tento vzorec $X = 1000 * (WR/ST)^3$ [15].¹

Jakožto každá mince má dvě strany, ani FP tomu nejsou výjimkou. Pro úplnost by bylo vhodné zmínit i pár nevýhod. Většina disciplín má světovou špičku velmi blízko u sebe, přesto se v dnešní době najdou výjimky, v plavání je to například mužská disciplína 100 m prsa na dlouhém bazénu. Jelikož se řada limitů odvíjí právě od FP, může zde dojít k problému, jelikož jeden plavec (který drží světový rekord) může mít vysoký náskok oproti druhému zaplavanému času, tzn. dosažení stejného počtu bodů v různých disciplínách může být různě náročné (např.: standardem u 200 m prsa může být mezi 10 nejlepšími na světě plavat 970 FP, zatímco u 100 m prsa je standard 930 FP). Jako řešení se nabízí sestavování limitů podle průměru prvních x časů nebo využití určitého umístění z několika posledních mezinárodních akcí (typu ME, MS a OH) a průměrování těchto časů. Další stránka nedostatků je z hlediska technologického. Neexistuje jednoduchá aplikace, kde by se daly FP počítat a běžela by multi-platformě. Tento účel skvěle splní aplikace webová.

3.3. Webové technologie

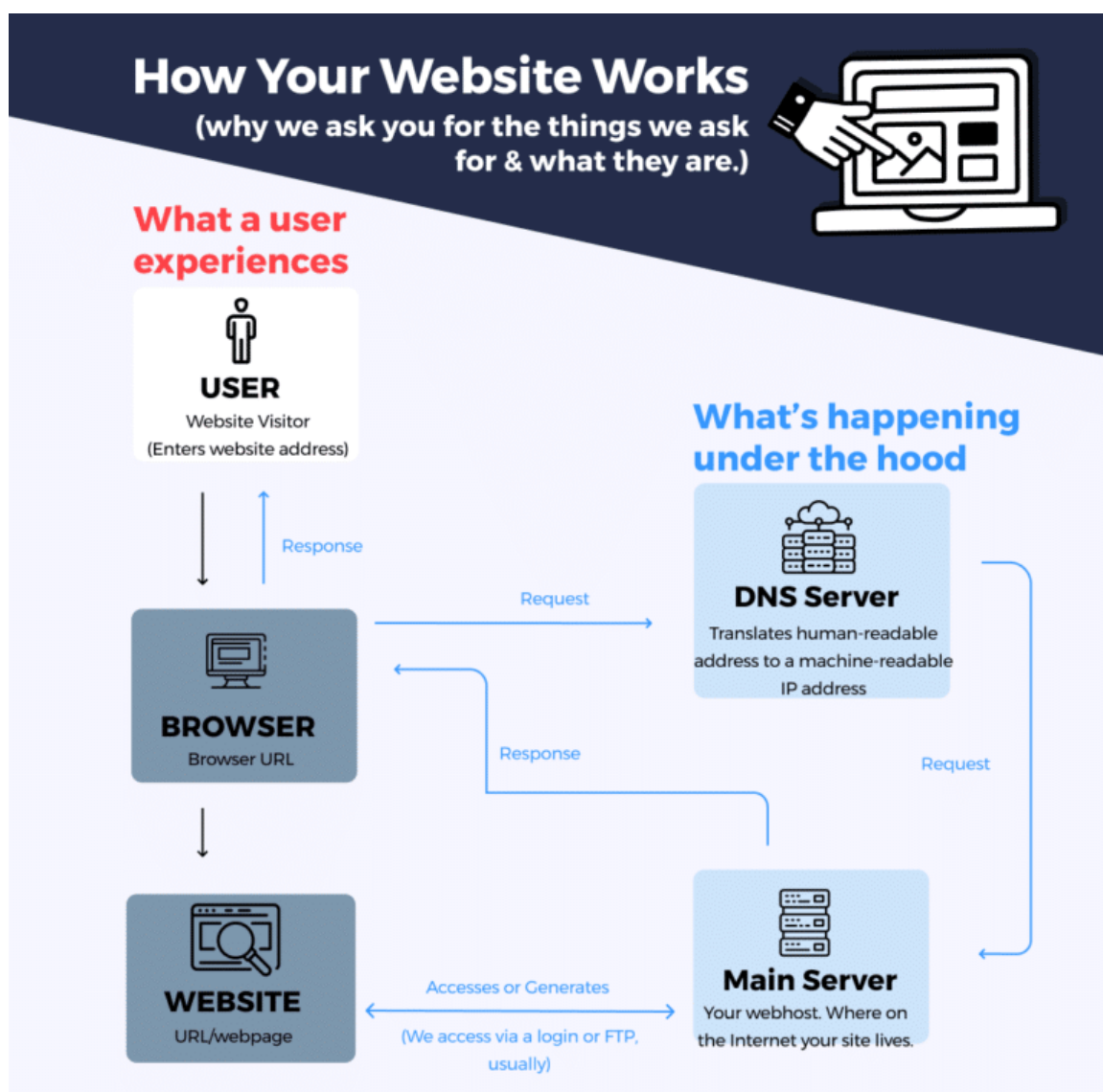
3.3.1. Co se děje, když otevíráme webové stránky?

Veškerá funkcionalita je založena na komunikaci uživatele se serverem. Naprosto zásadní je zde internet a také WWW, bez těchto služeb by žádná webová aplikace nemohla fungovat.

Ve zjednodušené podobě jde o to, že uživatel zadá do webového prohlížeče webovou adresu stránky (pod tou se nachází IP adresa, webová adresa má pro nás tu výhodou, že je jednodušejší zapamatovatelná). Pokud se pod touto IP adresou nachází nějaký funkční web, dostaneme odpověď od serveru v podobě dat potřebných pro prohlížeč k zobrazení webové stránky. Tyto operace probíhají přes protokol HTTPS. Celý popsany postup výborně znázorňuje obrázek (obr. 1). [16]

¹ WR = světový rekord (world record)
ST = zaplavaný čas (swam time)

K vývoji jakékoliv aplikace, která má obsahovat i uživatelské rozhraní je nutné programovat obě stránky věci – back-end i front-end. Back-endem se rozumí kód „v pozadí“, který vykonává službu (např.: u kalkulačky obstarává sečtení čísel). Naopak front-end se zaobírá stránkou vzhledu (např.: u kalkulačky určuje jakým stylem se nám výsledek zobrazí). Technologie, jež jsou potřebné ke komunikaci či samotnému vývoji aplikaci (ty, které jsem využíval já) budou popsány v následujících odstavcích. [17]



Obr. 1 – Jak funguje web

3.3.2. World Wide Web

Jde o aplikaci (případně službu), která je poskytována v rámci internetu. V dnešní době je používána k záležitostem všech druhů (sociální sítě, čtení zpráv, obchodování, sledování videí atd.). Zjednodušeně řečeno, vše co děláme ve webovém prohlížeči, je nám umožněno díky této službě. [18]

Důležité je nezaměňovat s pojmem internet, který znamená něco jiného. Internet je pojem užívaný pro propojení počítačových sítí (sít' sítí) tak, aby mohli vzájemně komunikovat. Zatímco WWW je služba, která v rámci internetu poskytuje uživateli funkci prohlížení webových stránek.

3.3.3. HTTP a HTTPS

Protokol, který je určen ke komunikaci se servery v rámci WWW. Funguje na principu *dotaz – odpověď*. Jelikož samotný HTTP protokol umí provádět pouze nešifrovanou komunikaci, je nutné v dnešní době použít HTTPS protokol, který tuto funkcionalitu umožňuje (doporučené pro všechny stránky, i tam kde nepracujeme s citlivými daty). Použití HTTPS je nutné především pro přenos dat, jež by neměla být viděna třetí osobou (např.: internetové bankovníctví nebo tréninkový deník profesionálních sportovců).

Protokol neumí uchovávat žádné informace (neví, zda různé dotazy spolu souvisejí), proto byl rozšířen o HTTP cookies, které umožňují uchovat informace o spojení uživatele se serverem. [19], [20]

3.3.4. REST

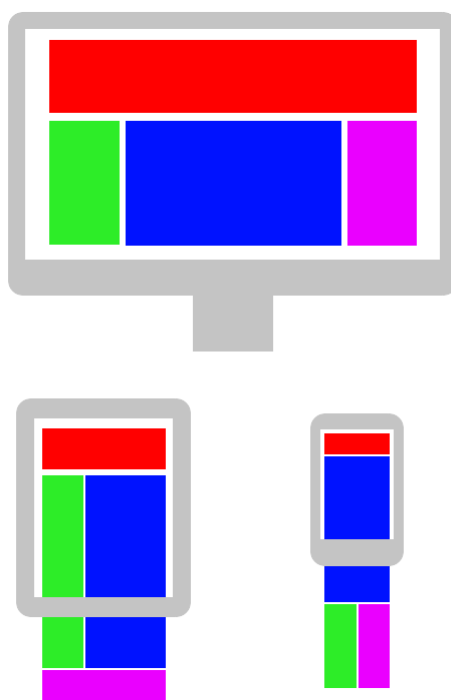
Zkratka pro *Representational state transfer* (neexistuje český ekvivalent). Jde o architekturu rozhraní, která umožňuje pomocí jednoduchých HTTP volání číst, editovat, vytvářet či mazat informace ze serveru. Tyto funkce jsou implementovány pod označením GET, PUT, POST a DELETE. K výměně dat používá dva jednoduché standardizované formáty ATOM/RSS a JSON (který používám i já ve své aplikaci). [21]

JSON je zkratkou pro *JavaScript Object Notation* (*JavaScriptový objektový zápis*). Jde tedy o datový formát určený pro přenos dat. Vstupem může být libovolná datová struktura (číslo, řetězec, boolean apod.), ale výstupem je vždy řetězec. V Pythonu se používá pro komunikaci s JavaScriptem. [22]

3.3.5. Moderní web

Dnešních vysoká konkurence mezi společnostmi klade důraz na co nejlepší funkci jakýchkoliv prvků, které se prezentují před zákazníky. Mezi tento prvek určitě patří webové stránky, v poslední době ještě umocněné pandemií, na které zákazník pohlíží většinou jako první a často na nich i finální produkt nakupuje.

Jedním z klíčových faktorů posledních let se stala responzivnost (termín, který byl do webdesignu přinesen až v roce 2010). Tento termín značí způsob stylizace (front-end) takový, že se bude bezproblémově a optimalizovaně zobrazovat na všech typech zařízení (počítač, tablet, chytrý telefon, televize – viz obr. 2). Tato funkcionalita je vyžadována především s rozšiřujícím se portfoliem zařízení, která každý uživatel používá. Stránky tak nevypadají všude úplně stejně, ale jsou optimalizované pro daný poměr stran, velikost atd. [23]



Obr. 2 – Responzivní vzhled

3.4. Správa front-endu

3.4.1. HTML

Jedná se o značkovací jazyk, který se používá pro tvorbu webových stránek. Jeho vývoj je ovlivněn především vývojem webových prohlížečů, podle čehož vznikají nové verze (standards). Je charakteristický skupinou značek (tagů) a jejich vlastnostmi (atributy). Názvy značek se uzavírají mezi úhlové závorky a obvykle jsou párové. Dokumenty formátu HTML mají přesně danou strukturu *deklarace typu - kořenový element - hlavička – tělo*.

Poslední je verze HTML5 (a její další nástavby 5.1, 5.2 apod.). Tato verze přišla patnáct let po verzi předchozí a přinesla podstatné změny – možnost fungování aplikací i v offline módu (při splnění specifických podmínek). Prohlížeče mohou fungovat i v režimu zpětné kompatibility, kdy jsou schopné zobrazit stránky psané podle aktuálních standardů, ale i ty, které je nesplňují. [24]

3.4.2. CSS

Celým názvem „*Kaskádové styly*“ je další z jazyků používaný při tvorbě webových stránek. Jeho úloha spočívá v popisu způsobu zobrazení (vzhledu) HTML dokumentu.

Hlavním důvodem vzniku bylo programátorům umožnit oddělit vzhled od struktury dokumentu (jednodušší pro orientaci, provádění změn).

CSS má stejně jako HTML danou strukturu, která je velmi primitivní. Je rozdělen do *pravidel* a každý selektor obsahuje *deklarace (blok deklarací)*, které definují vzhled daného selektoru. Např.: pokud jako *pravidlo* uvedu h1 a jako *deklaraci* color: blue, znamená to, že nadpisy h1 budou mít modrou barvu. [25]

3.4.3. Bootstrap

Jedná se o svobodnou a otevřenou sadu nástrojů kaskádových stylů, sloužící pro tvorbu webů. Pomocí Bootstrapu jsme schopni upravovat veškerý vzhled prvků pro uživatele (typografii, tlačítka, vstupy, formuláře apod.). Pro jeho použití je nutná předchozí základní znalost HTML a CSS.

Největší výhody spočívají v jednoduchém nastavení, responzivnosti a nápaditému stylu. Tímto způsobem dokáže ušetřit spoustu času a práce a zároveň není nutné se za jeho použití stydět. Obvykle zaručuje dobře fungující web po stránce front-endu. [26]

3.4.4. UX a UI design

Obě profese se zaměřují na spokojenost uživatele, každá z trochu jiného směru. UI designér navazuje na UX designéra a bez vzájemné interakce ztrácí profese velkou část svého přínosu.

UX (user experience) designér začíná komunikací s uživatelem a snaží se pro něj navrhnout co nejvhodnější prostředí (pohodlná práce, touha chtít zůstat atd.). Snaží se prostředí co nejlépe seřadit, zařadit použití interaktivních prvků a později pracovat s daty o používání a snažit se o neustále vylepšení. Jde v podstatě o smysl dotažení věcí k perfekcionismu.

UI (user interface) designér všechny tyto požadavky realizuje do grafické podoby. Jedná se z velké části o práci grafickou (vhodný font, barvy, styl designu). V dnešní době často uživatel zvolí spíše aplikace s hezčím prostředím než funkcemi navíc. [27]

3.5. Správa back-endu

3.5.1. Python

Python je vysokoúrovňový programovací jazyk, který byl navržen v roce 1991. Podporuje dynamickou kontrolu datových typů (není potřeba je definovat). Od roku 2018 roste jeho popularita. Momentálně patří mezi nejoblíbenější programovací jazyky. Disponuje výbornou vyjadřovací schopností, neboť je dobře čitelný a zápisy jsou krátké. To také velmi přispívá k jeho častému doporučení při výuce programování, na druhou stranu při použití v praxi Python spadá spíše k pomalejším jazykům (v dnešní době s výkonnými počítači to většinou nebývá problém, kromě užití např.: v Raspberry Pi, při vědeckých simulacích a jiných výpočtově náročných operacích). [28]

3.5.2. Flask

Jde o webový framework pro Python. Framework je pracovní rámec, která pomáhá programátorům při vývoji projektů. Obsahuje knihovny, návrhové vzory a jiné podpůrné programy pro zjednodušení vývoje a dovoluje programátorovi se soustředit pouze na zadání (např.: při vývoji webu pro elektronickou žákovskou knížku se může zaměřit na funkci přidávání známek pro žáky, zato nemusí řešit navigaci mezi jednotlivými okny webu).

Je to jednoduchý framework, to ovšem neznamená, že by neposkytoval řadu důležitých a pokročilejších funkcí. Pokud by přesto někomu cokoliv chybělo, není problém použít rozšíření vytvořená komunitou.

Flask podobně jako Python získává velikou oblibu v posledních letech (druhé místo na GitHub v rámci vývojových frameworků v Pythonu). Mezi nejznámější velké uživatele patří LinkedIn a Pinterest. [29], [30]

3.5.3. JavaScript

JavaScript je také programovací jazyk, vhodnější je však název skriptovací jazyk (slouží především k automatizaci úloh a manipulaci s prostředky stávajícího systému). Jeho syntaxe patří do skupiny C/C++/Java. Slovo java má v názvu především z marketingových důvodů, kromě názvu má s Javou jen velmi málo společného. Nejčastější využití má na webových stránkách jako client-side jazyk.

V případě webových stránek běží jazyk na zařízení uživatele (client-side), což přináší výhody, např.: systém uživatele upozorní na špatné zadání formy emailu ještě před odesláním, a nemusí se tedy zbytečně čekat. Tento fakt sebou nese i negativní stránku věci, např.: z hlediska

soukromí a bezpečnosti (nemůže ověřovat údaje, jelikož by uživatel mohl systém snadno obejít, podobného principu využil student při „prověřování“ webu na očkování proti covidu-19 [43]). Nejčastěji ovládá animace, efekty obrázků či interaktivní prvky GUI (grafické uživatelské rozhraní), mohou jimi být např.: textová políčka, tlačítka apod. Jeho standardizaci i implementaci provázeli problémy a masivní rozšíření se připisuje Googlu, který poprvé v Gmailu použil AJAX. [31], [32]

Technologii AJAX jsem využil i já, a proto je dobré si ji představit. Jedná se o mírně pokročilou technologii, která umožňuje měnit obsah na stránce bez jejího opětovného načtení, což zpříjemňuje uživatelský zážitek. [33]

3.5.4. Databáze

Definice databáze je systém s pevnou strukturou záznamů, které jsou mezi sebou propojeny pomocí klíčů. Jelikož tato definice je pro laickou veřejnost hůře pochopitelná, je lepší si databázi představit jako kartotéku (např.: kartotéku pacientů u lékaře). S kartotékou je podobná i jejich správa.

Slouží jako zdroj dat pro aplikace, které v nich dokáží vyhledávat, zapisovat do nich a obecně s nimi pracovat. Pro koncového uživatele není podstatné znát, jak tento proces probíhá (systém řízení). Typ databáze (např.: relační, xml, dokumentová atd.) ho však přímo ovlivňuje, jelikož různé databáze umožňují více či méně pokročilé funkce. [34]

3.5.5. NoSQL databáze

Jedná se o databázový koncept, který využívá jiné prostředky než tabulková schémata jako běžné relační databáze, a proto se jim také říká i „nerelační“. Výhoda spočívá především v jeho jednoduchosti a případné algoritmické složitosti. Přestože v názvu máme slovo „no“ (česky „ne“), tak i tento koncept dokáže akceptovat dotazy v jazyce SQL (nebo jemu podobném).

V praxi je využití tohoto typu momentálně jen malé, nejvíce mu brání použití nízko úrovněvých dotazovacích jazyků, absence podpory transakčních modelů a také vysoké investice společností do SQL (v minulosti).

Na druhou stranu dnešní doba poskytuje velké množství dat (v různých datových typech) a zdá se, že v tomto ohledu by jejich využití mohlo stoupat. Především jde o flexibilnější a rychlejší změnu schématu i dotazů (podle požadavků konkrétních dat). Uplatnění se týká aplikací, které používá běžný občan (např.: mobilní). [35]

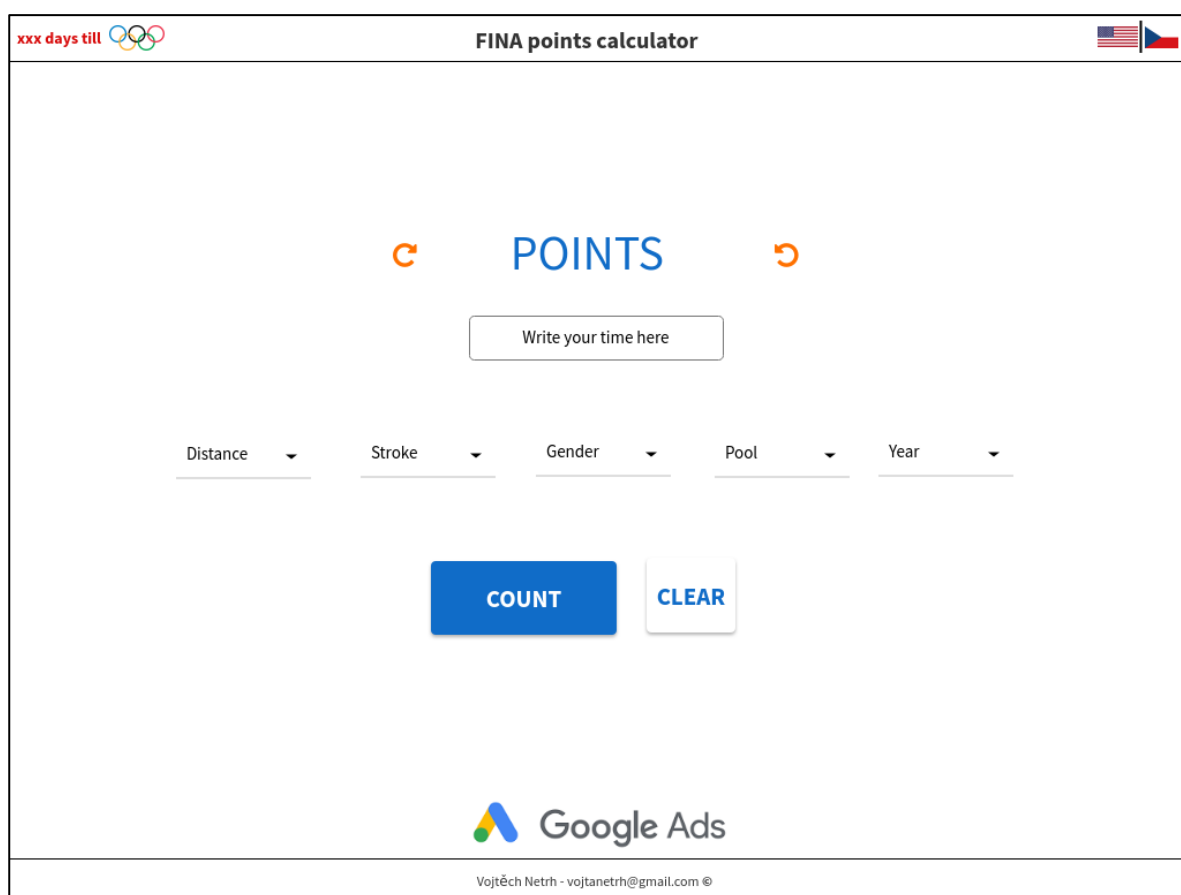
3.5.6. XML

Obeční značkovací jazyk, jenž byl vyvinut a poté i standardizován konsorciem W3C. Jeho předchůdcem byl jazyk SGML. Jeho použití spočívá ve vytváření konkrétních značkovacích jazyků (neboli aplikací). Pomocí něj si tedy můžeme naprogramovat aplikaci, ve které budeme mít uložena data (přesně na míru nám), jež potom budeme vyměňovat s hlavní aplikací, která data nějakým způsobem využívá (např.: srovnávač cen potřebuje získat data z různých e-shopů, což může udělat za pomoci XML a u daných produktů si autoři zvolí, jaké všechny parametry chtějí porovnávat – počet kusů skladem, cenu, dopravu atd.). Je podporovaný valnou většinou jazyků, což je nesporná výhoda. [36]

4. Vlastní práce

4.1. Shrnutí požadavků

Po provedení neúspěšné rešerše v této oblasti se pro mě tvorba webové aplikace stala výzvou, a to hned z několika důvodů. Služba, kterou aplikace poskytuje není běžná (narozdíl od např.: e-shopu, diskusního fóra), takže podrobnější tutoriály k fungování alespoň částí aplikace jsou nedostupné. Byla to má první zkušenost s Flaskem, ale především jsem poprvé musel použít JavaScript, s kterým jsem se do té doby nesetkal a od Pythonu se velmi liší. V neposlední řadě mě čekalo propojení těchto různorodých technologií dohromady. Z hlediska front-endu jsem hledal větší inspiraci jen marně, jako jediný hrubý návrh (jaké prvky musí aplikace obsahovat) mi poskytla mobilní aplikace [37] na platformě Android, která plní stejný účel. Pro vizualizaci svých představ a hrubého návrhu jsem použil službu (taktéž webovou aplikaci) Mockflow [38]. Výsledek mé vizualizace je vidět na obr. 3.



Obr. 3 – Návrh vzhledu

Požadavky, které jsem si stanovil z hlediska funkcí (v podstatě jsou definovány smyslem aplikace) byly vcelku jednoduché na pochopení. Uživatel při zadání svého času a výběru disciplíny (délka trati, plavecký způsob, pohlaví, délka bazénu) dostane vypočítané FP.

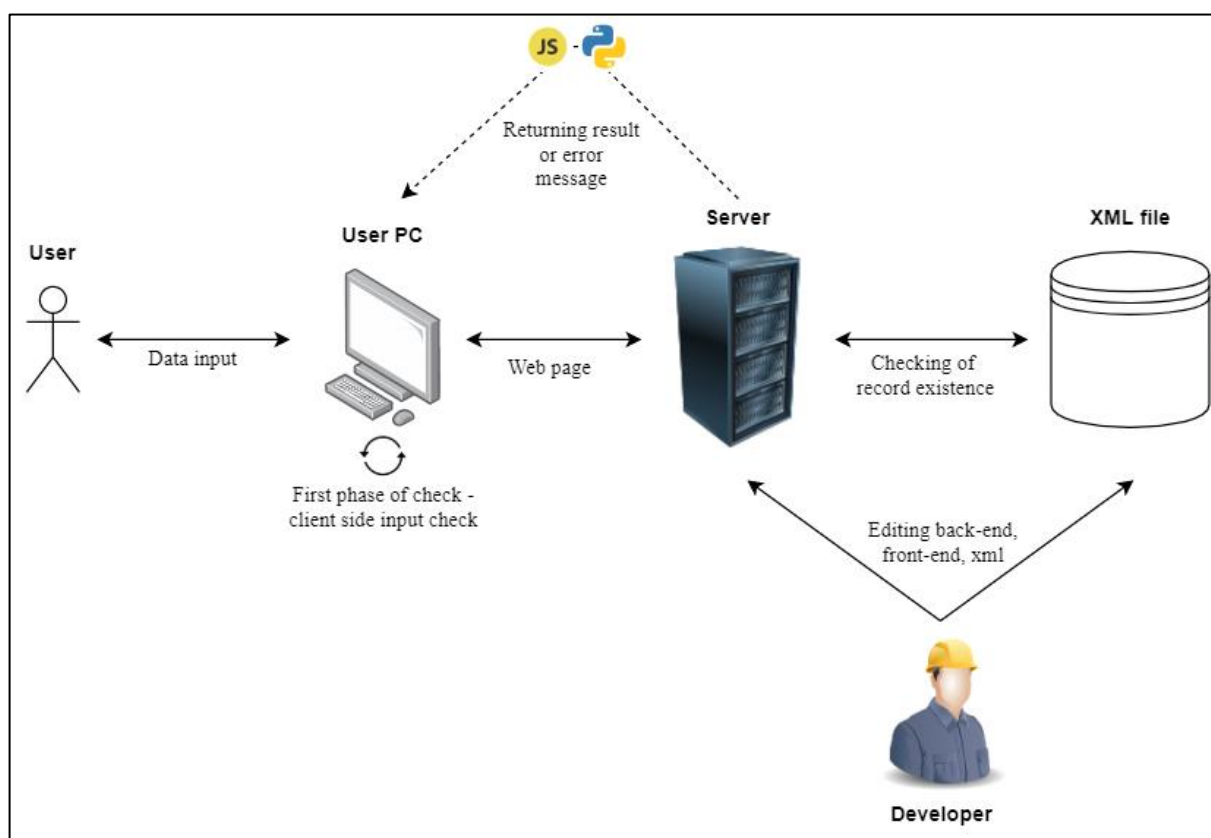
V průběhu programování jsem odstranil kolonku *year* (rok), jelikož by bylo nutné vytvářet více datových sad pro dané roky a vzhledem k malému vytižení tohoto parametru, by implementace nebyla výhodná. Jako funkce navíc by mohl být obrácený postup, kdy uživatel místo času zadá body a opět zvolí disciplínu a aplikace mu spočítá čas, jenž je nutný zaplavat.

4.2. Implementace back-end

Jelikož už dávno webové stránky neslouží jen k prohlížení statického obsahu, ale čím dál více jsou do nich implementovány různé funkce (případně celé aplikace), jež uživatel ovládá a poskytují mu službu, používají pro generování stránek programovací jazyky, které tuto službu umožňují.

Jako jazyk psaní kódu jsem zvolil angličtinu, a to hned z několika důvodů. Neobsahuje diakritiku, která je pro programování nevhodná, názvy jsou obvykle kratší, kódu bude snadno rozumět kdokoliv ve světě. Stejný důvody jsem měl i k použití angličtiny u uživatelského rozhraní. Mým vývojovým prostředím (IDE) byl Visual Studio Code.

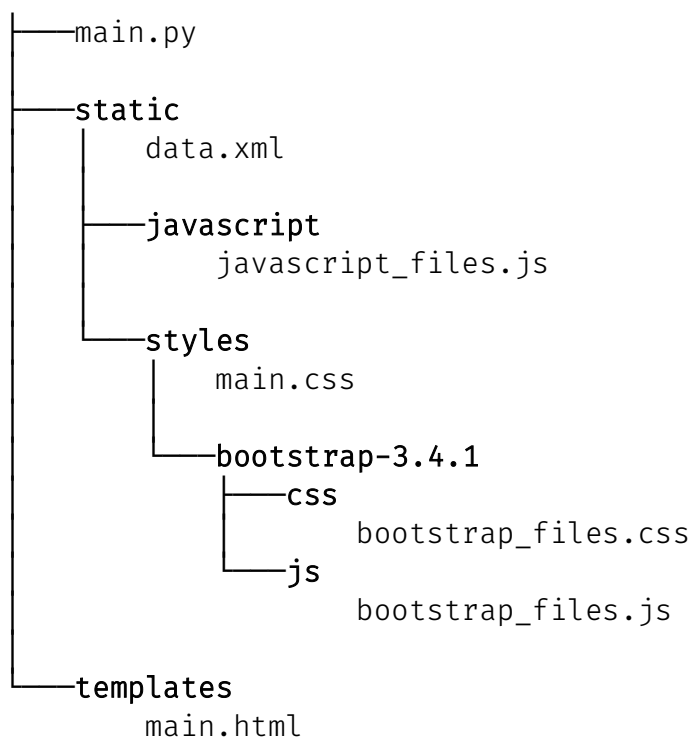
Jednoduché schéma fungování systému je zobrazeno na obr. 4.



Obr. 4 – Schéma fungování aplikace

4.2.1. Struktura souborů aplikace

Strukturování aplikace do souborů a složek umožňuje zlepšit orientaci v celé aplikaci a u některých frameworků je dokonce nezbytné. Strukturu jsem zvolil pro Flask typickou a vcelku jednoduchou. Ovládání celé aplikace se provádí přes jediný soubor Pythonu, který vše řídí a z dalších složek si „volá“ jednotlivé soubory (HTML, CSS, JS atd.). Struktura je pro lepší přehlednost zobrazena, složky jsou zapsány tučně.



4.2.2. Python

Pro back-end jsem zvolil jazyk Python, jež jsem používal v průběhu celého studia programování, a byla to jednoznačná volba. Navíc obsahuje řadu frameworků, které jsou pro začátek vhodné stejně jako samotný Python a zároveň jsou moderní.

4.2.3. Flask

Jako konkrétní „správce“ webu jsem zvolil framework Flask. Jeho systém je totiž velmi jednoduchý, výborně čitelný a jednoduše rozšiřitelný pro další funkce aplikace (např.: nové záložky pro přihlášení uživatelů, procházení databáze rekordmanů atd.)

Program je rozdělen do 2 částí. Každá obsahuje jednu definovanou funkci. Funkce *home* slouží pouze pro zobrazení správného HTML souboru při načítání stránky (stránka se při vykonávání služby neobnovuje). Druhá funkce *process* se volá v případě, že uživatele odešle formulář ke zpracování (v mém případě pomocí metody *POST*) a provádí všechny úkony nutné

k výpočtu a zobrazení výsledku. Můžeme si všimnout, že je uložena v jiné části než funkce první, což podrobněji vysvětlím v další podkapitole JavaScript.

Prvním nezbytným krokem bylo importovat údaje vybrané uživatelem v *select tag* a čas zadaný v poli *input*. Zde také přichází první změna, když používáme AJAX ke zobrazení výsledků. Základní postup je stejný, při použití pouze Flasku do `[“”]` uvádíme atribut *name* požadovaného vstupu ve formuláři (viz obrázek – v tomto případě bychom uvedli `[“time”]`, ale při použití AJAXu uvedeme název proměnné, který jsme si stanovili v JavaScriptu). Část o práci s daty (ze souboru XML) je podrobně vysvětlena v kapitole „[Práce s daty](#)“.

Po úspěšném importování všech dat, která potřebujeme jsem se přesunul na závěrečnou část – samotný výpočet FP. Nejdříve ze všeho jsem oba časy (čas zadaný a čas WR) převedl do formátu sekund pomocí primitivního algoritmu. Prvním dílčím krokem bylo rozdělit čas (rozdělovacím znakem byla dvojtečka) do seznamu pomocí `split` (např.: čas 2:15:42 => `time_split = [2, 15, 42]`). Následně jsem jednotlivé položky ze seznamu vynásobil jejich „měřítkem“ vzhledem k sekundám (např.: 2 min = 120 s => 2×60) a jednotlivé hodnoty sečetl do jedné proměnné (např.: z 2:15:42 dostanu proměnou `edited_time = 135.42`). Důležité je také nezapomenout, že všechny hodnoty musíme uvést jako *integer* (celočíslný datový typ).

```
9  @app.route("/")
10  def home():
11      return render_template("main.html")
12
13  @app.route("/process", methods=["POST"])
14  def process():
15
16      #importování údajů ze select tagů a input tagu
17      time = request.form["time"]
18      distance = request.form["distance"]
19      style = request.form["style"]
20      course = request.form["course"]
21      gender = request.form["gender"]
```

Obr. 5 – Rozdělení funkcí v Pythonu

Posledním krokem je využití vzorce pro výpočet FP, který jsem uvedl již v rešerši. Výsledek vzorce (spočítané FP) jsem uložil do proměnné *points*. Pro označení mocnění se používá specifický zápis (např.: běžně - 2^3 , Python - $2^{**}3$). Výsledek opět nezapomeneme uvést jako *integer*, jelikož z logiky věci vyplývá, že počet bodů musí být celočíselný.

Výsledkem funkce *process* mohou být 2 výstupy – úspěšný a neúspěšný. S rozhodováním, který z nich je pro daný případ ten vhodný, slouží operátor *if*. V obou případech jsou výstupem data ve formátu JSON, který se u Flasku používá pro zpracování v JavaScriptu.

```
45 #úprava formátu času WR
46 split_wr_time = wr_time.split(":")
47 edited_wr_time = int(split_wr_time[0]) * 60 + int(split_wr_time[1]) + int(split_wr_time[2]) * 0.01
48
49 #úprava formátu času plavce
50 time_split = time.split(":")
51 edited_time = int(time_split[0]) * 60 + int(time_split[1]) + int(time_split[2]) * 0.01
52
53 v points = int(1000 * ((edited_wr_time/edited_time)**3))
```

Obr. 6 – Úprava časů a výpočet bodů

```
57 #jestliže neexistuj proměná points příkaz se nevykoná (nebyly zvoleny existující disciplíny)
58 if "points" in locals():
59     return jsonify({"name_of_variable" : value})
60 else:
61     return jsonify({"error" : "Data are not valid!"})
```

Obr. 7 – Výstup z Pythonu

4.2.4. JavaScript

Primární úloha JavaScriptu spočívá ve způsobu zobrazení požadovaných výsledných údajů na obrazovce. Mé konečné rozhodnutí padlo na technologii AJAX. Způsob, kterým zobrazuje data se mi zdál ideální z hlediska uživatelského komfortu. Inspirací pro ni mi byla již zmíněná mobilní aplikace, ve které se výsledek zobrazoval tímto způsobem. Sice je vyžadováno použití moderních webových prohlížečů, avšak s jinými je v dnešní době takřka nemožné se setkat. Pro lepší přehlednost a lepší rozšiřitelnost v budoucnu jsem i JavaScript vložil do samotných souborů.

Nejdříve ze všeho bylo nutné definovat, za jakých podmínek se má HTML soubor odvolávat na *ajax.js*. Touto podmínkou je odeslání (potvrzení) formuláře, které se provádí pomocí kliknutí na tlačítko. Následně jsem musel i do JavaScriptu importovat údaje zadané do formuláře. Tento krok, narozdíl od Flasku neprovádím pomocí atributů *name*, nýbrž atributu *id*. Z těchto dat následně čerpá i Flask, jak jsem již zmiňoval. Tyto údaje pomocí metody POST se odešlou zpět do Flasku a přesměrují na adresu *"/process"*, kde se provádí další kroky, než bude opět nutné vrátit se do JavaScriptu.


```

1  $(document).ready(function() {
2
3      $("form").on("submit", function(event) {
4
5          $.ajax({
6              data : {
7                  time : $("#timeInput").val(),
8                  point_input : $("#pointInput").val(),
9                  distance : $("#distanceInput").val(),
10                 style : $("#styleInput").val(),
11                 course : $("#courseInput").val(),
12                 gender : $("#genderInput").val()
13             },
14             type : "POST",
15             url : "/process"
16         })

```

Obr. 9 – AJAX první část

Do JavaScriptu se opět vrátím poté, co ve Flasku získám všechna data potřebná k výpočtu a výpočet dokončím, nebo při opačné situaci, kdy jsou údaje špatně zadane a výpočet nemůžu dokončit, takže dostanu chybovou hlášku. Zde též aplikace rozhoduje, zda je schopna zobrazit výsledky, nebo dostala nevhodné zadání, a musí zobrazit chybovou hlášku. Toto rozhodnutí proběhne jednoduše pomocí operátoru *if*, který reaguje na existenci proměnné, jež reprezentuje chybovou hlášku (*error*).

```

18     .done(function(data) {
19
20         if (data.error) {
21             $("#errorAlert").text(data.error).show(),
22             $("#div_points").hide(),
23             $("#div_time").hide(),
24             $("#div_distance").hide(),
25             $("#div_style").hide(),
26             $("#div_course").hide(),
27             $("#div_gender").hide()
28         }
29         else {
30             $("#errorAlert").hide(),
31             $("#div_points").text(data.point).show(),
32             $("#div_time").text(data.tim).show(),
33             $("#div_distance").text(data.dis).show(),
34             $("#div_style").text(data.sty).show(),
35             $("#div_course").text(data.cour).show(),
36             $("#div_gender").text(data.gen).show()
37         }
38     });

```

Obr. 8 – AJAX druhá část

JavaScript ještě mimo svůj hlavní úkol plní tři funkce. První z nich je „vyčištění“ formuláře (obr. 10), kdy bez znovu načítání stránky smaže výsledky (data ve formuláři není potřeb mazat přes JavaScript, jelikož v HTML existuje funkce pro tento účel). Druhou funkcí je změna vstupního pole z času na body (i obráceně). Opět vše probíhá bez znovu načítání stránky. Ovšem tato funkce není hotová, tudíž se opačný výpočet nedá použít, prostředí je pouze připraveno na implementaci (obr. 11). Obě funkce se vyvolají stisknutím příslušného tlačítka.

```
1 function clearData() {
2     $("#div_distance").empty();
3     $("#div_style").empty();
4     $("#div_points").empty();
5     $("#div_course").empty();
6     $("#div_gender").empty();
7     $("#errorAlert").empty();
8 }
9
```

Obr. 11 – Vyčištění formuláře

```
1 function changeInput() {
2     var a = document.getElementById("timeInput");
3     var b = document.getElementById("pointInput");
4
5     if (a.style.display === "none") {
6         a.style.display = "inline";
7         b.style.display = "none";
8     }
9     else {
10        a.style.display = "none";
11        b.style.display = "inline";
12    }
13 }
```

Obr. 10 – Výměna vstupního pole

Poslední funkcionalitou, kterou zajišťuje JavaScript je zobrazení odpočtu. Opět se jedná o jednoduchý algoritmus, který odpočítává dny do určitého data (v mém případě Olympijské hry). Nejdřív je nutné nastavit data, první do kterého bude odpočet trvat (v mém případě 24. července 2021), druhé datum bude datum dnešní (každý den se aktualizuje), tak aby se i odpočet dynamicky měnil s tím, jak se budou Olympijské hry blížit. Nakonec jen stačí data odečíst a dostaneme počet zbývajících dnů. Pro přehlednost je zde nastavena i hláška, která poté co Olympijské hry začnou vypíše místo odpočtu daný řetězec.

```

1  var final_date = new Date("Jul 24, 2021");
2
3  var x = setInterval(function() {
4
5      // Dnešní datum
6      var now = new Date();
7      // Vzdálenost do finálního data
8      var distance = final_date - now;
9
10     // Převod na dny
11     var days = Math.floor(distance / (1000 * 60 * 60 * 24));
12
13     document.getElementById("countdown").innerHTML = days + " days left until the Olympics";
14
15     // Vypsání chybové hlášky
16     if (distance < 0) {
17         clearInterval(x);
18         document.getElementById("countdown").innerHTML = "Olympics have already begun";
19     }
20 }, 100);

```

Obr. 12 – Odpočítávací dny

4.3. Práce s daty

4.3.1. XML

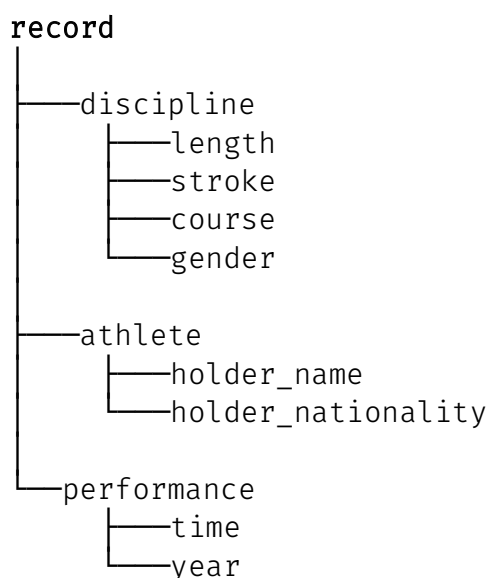
Při prvotní části řešení jsem si do části pro práci s daty zařadil použití relačních databází. Při bližším zkoumání a informování se o dalších možnostech se mi však tento koncept jevil jako zbytečně složitý a nevhodný pro soubor mých dat (data nebudou přibývat v průběhu času, avšak je v budoucnu možné rozšíření jednotlivých úseků dat o podrobnější informace) a zvláště, pokud se chci zabývat i problematikou otevřených dat, bylo by vhodné tento systém uskutečnit i u mé práce, abych šel příkladem. Z těchto výše zmíněných důvodů jsem využil soubory typu XML, které umožňují vytvořit si vlastní (tím pádem ideální strukturu) pro data, a přesto, že data pravděpodobně nebudou přibývat, mohu jednotlivé informace rozšířit i data přidat. Navíc XML spadá mezi 3. a 4. třídu otevřených dat.

Jelikož se jedná o obecný jazyk, je nutné si pro vlastní projekt vytvořit vlastní strukturu (včetně jednotlivých tagů, i jejich názvů). Návrh, z něhož jsem vycházel, spočíval v tom, že jeden celý blok bude tvořit jeden světový rekord. Ten dále rozšířím o tři podtřídy, které uchovávají jednotlivé informace ve skupinách, jenž spolu souvisí (informace o plavci, o rekordu a disciplíně).

První větev *discipline* sdružuje informace, pomocí kterých se daný světový rekord vyhledává (informace, které zadává uživatel). Tato větev má informace vždy jedinečné a v průběhu času se žádné tagy přidávat nebudou.

Následující větev *athlete* poskytuje informace o držitele daného světového rekordu. Informace jsou pouze stručné, avšak dostatečné, jelikož v plaveckém prostředí o nic jiného zájem není. U této větve se v budoucnu počítá s rozšířením (při použití souboru např.: pro databázi plavců). Není jedinečná a běžně se může opakovat. Nejlepší světoví plavci obvykle nedominují pouze v jedné disciplíně, ale ovládají soubor více disciplín (např.: 50 m a 100 m prsa, 200 m a 400 m volný způsob).

Poslední větev *performance* nabízí pohled do souvislosti přímo s rekordem. Nejdůležitější je zde parametr *time*, bez kterého bychom nebyli schopni zprovoznit celou aplikaci, jelikož je nutný k vypočítání FP. Tato větev se také může v budoucnu dočkat rozšíření např.: o místo zaplávání rekordu či mezičasy.



4.3.2. Propojení XML s Pythonem

Jelikož s daty je nutné pracovat, musíme s XML souborem manipulovat přes Python jakožto jazyk, který nám obsluhuje back-end. Pro propojení nebylo nutné importovat knihovnu, ale stačilo použít *xml.dom.minidom*, což je implementace DOM (objektový model dokumentu) ve své minimalistické verzi. Tato implementace je standardně zahrnuta v Pythonu.

Jako první jsem dokument s daty analyzoval do Pythonu pomocí funkce *parse*. Používám relativní cestu k XML souboru, aby aplikace fungoval na libovolném zařízení (za předpokladu, že mám soubor uložený ve složce s aplikací).

```
6 #přiřazení xml dokumentu s data o WR
7 document = xml.dom.minidom.parse("static\\data.xml")
```

Obr. 13 – Import XML do Pythonu

K vyhledávání byl použit *for* cyklus, aby se mohly projít všechny možnosti a poté co se narazím na hledanou možnost, cyklus se ukončí a pokračovat se bude výpočtem a vrácením výsledků. *For* cyklus prochází všechny záznamy označené jako *record*. Začínám od první větve, která mi určí, zda se jedná o správný rekord. Hledám tedy pomocí názvu tagu *discipline*. Z něho jsem poté vybral všechny potomky a jejich hodnoty nahrál do proměnných. Vždy jsem vybíral hodnotu první (označená jako *[0]*), jelikož více než jedna neexistuje. Ty jsem v dalším kroku porovnal s hodnotami ze vstupu od uživatele a pakliže se rovnaly, přistoupil jsem k další části.

```
23 records = document.getElementsByTagName("record")
24
25 for record in records:
26     discipline = record.getElementsByTagName("discipline")[0]
27
28     distance_xml = discipline.getElementsByTagName("length")[0].childNodes[0].nodeValue
29     stroke_xml = discipline.getElementsByTagName("stroke")[0].childNodes[0].nodeValue
30     course_xml = discipline.getElementsByTagName("course")[0].childNodes[0].nodeValue
31     gender_xml = discipline.getElementsByTagName("gender")[0].childNodes[0].nodeValue
```

Obr. 15 – Prohledávání XML

Ve chvíli, kdy program nalezne světový rekord v XML souboru, začíná sbírat další informace o něm. Postup byl stejný jako o tagu *discipline*. Našel jsem vždy daný tag podle názvu (*athlete* a *performance*) a hodnoty jejich potomků nahrál do proměnných. *For* cyklus bylo možné tedy ukončit pomocí *break* (není potřeba procházet další záznamy, jestliže existuje pouze jeden vhodný, který jsem již našel). Tímto krokem byl proces práce s XML ukončen.

```
33 if distance_xml == distance and stroke_xml == style and course_xml == course and gender_xml == gender:
34
35     #informace o plavci
36     athlete = record.getElementsByTagName("athlete")[0]
37     athlete_name = athlete.getElementsByTagName("holder_name")[0].childNodes[0].nodeValue
38     athlete_nationality = athlete.getElementsByTagName("holder_nationality")[0].childNodes[0].nodeValue
39
40     #informace o rekordu
41     performance = record.getElementsByTagName("performance")[0]
42     wr_time = performance.getElementsByTagName("time")[0].childNodes[0].nodeValue
43     wr_year = performance.getElementsByTagName("year")[0].childNodes[0].nodeValue
```

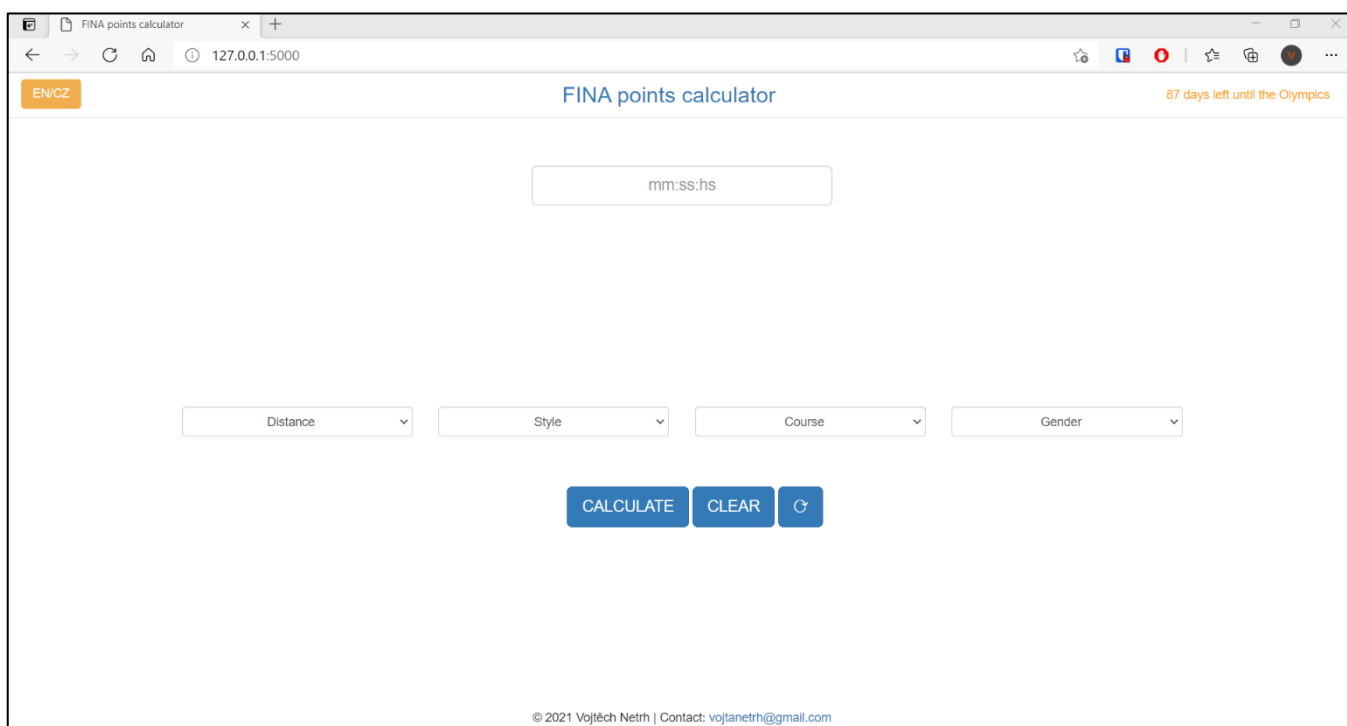
Obr. 14 – Určení vhodného rekordu

4.4. Implementace front-end

Pro tvorbu statického obsahu na stránce byl použit jazyk HTML (ve verzi HTML 5). Pro lepší přehlednost nemá v sobě napřímo implementován JavaScript ani stylování prvků, obojí je řešeno v externích souborech.

Hrubý vzhled aplikaci dává technologie Bootstrap (ve verzi 3.4.1), podle které jsem jednotlivé prvky rozdělil do různých tříd. Tím pádem jsou automaticky responzivní a jejich styl hrubě nastavený a stačí provést pouze malé úpravy. V samostatném CSS souboru jsou poté provedeny úpravy jednotlivých prvků na míru aplikaci.

Výsledný vzhled je možno vidět na následujícím obrázku (obr. 16), případně si můžete celou aplikaci stáhnout na mém GitHubu (<https://1url.cz/CKJQF>).



Obr. 16 – Vzhled aplikace

5. Diskuse

5.1. Problémy otevřených dat

5.1.1. Je zveřejnění vždy přínosem?

Většina případů zveřejňování dat přináší zlepšení dané oblasti a větší důvěru veřejnosti v dané téma. V poslední době se stalo zveřejňování všemožných informací velmi populární a firmy se snaží navzájem předbíhat. V poslední době je velmi populární Bitcoin (BTC), a proto bych jako příklad zmínil zveřejnění platebního procesu BTC od Tesly [39]. Tyto kroky umožňují běžným občanům přiblížit jakým způsobem „pracují“ občané bohatší, a dokázat tak, že dělají především věci přínosné pro všechny a nesnaží se záměrně škodit jiným ve svůj prospěch.

Přestože spousta lidí by protestovala, vždy budou informace, které musí zůstat tajné, např.: z hlediska bezpečnosti, udržení výrobního či obchodního tajemství. Kromě toho, že si tyto informace přečte nezávislá část populace, jistě si je také přečte konkurence (v některých případech spíše nepřítel), což má negativní vliv. V poslední době je ideálním případem odhalení důkazů o podílu ruských tajných služeb na výbuchu muničních skladů ve Vrběticích v roce 2014 [40].

5.1.2. Sportovní odvětví

Ve sportovním odvětví je celá situace více jednoznačná. V poslední době se ukazuje, že uveřejňování má jen své plusy, to i v oblasti tréninkových deníků (a jejich know-how). Přestože dříve si každý trenér své metody střežil jako oko v hlavě, dnes se ukazuje, že možnost nechat nahlédnout do údajů i jiné lidi, může přinést další zlepšení a odhalit možné nevhodné metody.

5.2. Přínosy aplikace

Výhody oproti současnému řešení (ať už v podobě tabulek či mobilní aplikace) jsou především v ušetření času a multi-platformního používání. Pokud by se aplikace uvedla do reálného provozu, bude kód uveřejněn na GitHub, tudíž každý si ho bude moci prohlédnout a navrhnout vhodné úpravy.

Kromě přímého vlivu by aplikace mohla působit i hlediska zvýšení poptávky a požadavků po otevřených datech a tím pádem urychlit posun ve sportovním odvětví. Ideální vliv by byl samozřejmě mezinárodní, ale i vnitrostátní zlepšení bude jistě obrovským příslibem do budoucna.

5.3. Budoucnost aplikace

Do budoucna je plán jednoduchý. Snažit se z velmi jednoduchého nástroje udělat nástroj komplexní a co nejvíce automatizovaný. Konkrétním návrhem může být propojení s již zmiňovaným swimrankings, což by přineslo automatické aktualizace světových rekordů, a případně možnost počítání bodů v předešlých letech. Vylepšení stávajícího odpočtu, kterých by aktualizoval příští mezinárodního plavecký mítink (typu ME, MS, OH) automaticky a byl schopný zobrazit základní informace (např.: při najetí na políčko myši). Možnost funkce v režimu offline, podobně jako Google dokumenty. Z hlediska vzhledu pak aplikovat lákavější design i s animovaným prostředím. Posledním krokem může být vícejazyčná aplikace, kde by se jazyk stránky měnil pomocí tlačítka.

5.4. Komplikace při programování

Jelikož práce programátora je především o opravě chyb a hledáním možností, které mohou aplikaci přivést na neočekávaný scénář, kde nastane chyba, i já jsem musel několik zádrhelů řešit.

Jedním z problémů, které přetrvávaly delší dobu, ale měly triviální řešení byla implementace CSS souboru. Když jsem styly uvedl přímo do HTML souboru vše fungovalo bez problémů, ovšem když jsem je vložil do externího souboru, přesto že jsme ho měl správně napojený, moje stylizace zmizela. Nakonec jsem objevil možnost tzv. hard refresh okna v prohlížeči (zkratka: *Ctrl + Shift + R*), která konečně můj problém vyřešila.

Mým nevyřešeným problémem zůstalo výpočet z FP na čas. Problém spočívá v importování dat mezi uživateli zadanými údaji, JavaScriptem a Flaskem. Flask totiž vypíše *error*, pokud se snažíte z formuláře importovat prázdný řetězec. Z tohoto důvodu je tedy nutné nějakým způsobem vyřešit, za jakých okolností se bude jaký *input* importovat. Pravděpodobným řešením může být vytvoření nové části ve Flasku, která bude z valné většiny totožná s tou již existující, rozdíl bude jen v importu z jiného pole formuláře. V tom případě poté bude JavaScript rozhodovat, kterou z těchto dvou funkcí zavolá. Druhým kompromisem by mohlo být psaní do jednoho pole *input*, kde bychom však museli slevit z nároku na jeden formát a nemohlo by proběhnout již první částečné client-side ověření.

6. Závěr

Na začátku své závěrečné práce jsem si stanovil tři hlavní cíle. Práci bych zhodnotil pozitivně, jelikož se mi zadané cíle povedlo splnit. Naprogramování aplikace bylo úspěšné, a kromě rozšiřujících funkcí, je chod bezproblémový. Tento krok jsem z velké části splnil i u vzhledu, kde se výsledek blíží mému návrhu. Dalším bodem bylo osvojit si práci s webovými frameworky. Spíše než práce s webovými frameworky se mi v průběhu programování otevřela možnost k moderním technologiím webů, jelikož jsem jich použil více, na rozdíl od jednoho frameworku. Ve výsledku jsem však přidal nové dovednosti a zkušenosti do svého portfolia, což bych považoval jako splnění obecnějšího účelu tohoto cíle. Především jsem si oblíbil JavaScript a práci s front-endem, na tuto oblast bych se rád více zaměřil v následující době. Nakonec jsem si chtěl ověřit dostupnost otevřených dat od sportovní organizace FINA. Tento bod byl nevyhnutelný ke splnění prvního cíle, avšak výsledky mého zjištění nejsou moc pozitivní. Určitá data zveřejněna jsou, ale dle terminologie dat otevřených nejsou zvoleny ideální formáty. Zároveň jednoduchá dostupnost na webových stránkách také není naplněna (orientace byla složitější) a v neposlední řadě o detailních statistikách už vůbec nemůže být řeč. XML soubor, který používám pro správu rekordů aplikace jsem si musel vytvořit sám, pomocí přepisování údajů z jejich webových stránek. Ideální variantou by byla nabídka několika formátů souborů, společně s nabídkou počtu informací (zda budu chtít jen časy cílové nebo mezičasy i startovní reakci).

7. Seznam použitých zdrojů

7.1. Textové zdroje

[1] Open data. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-17]. Dostupné z: https://en.wikipedia.org/wiki/Open_data

[2] Otevřená data. Jak na Internet - Otevřená data [online]. Praha: CZ.NIC, 2012 - 2014 [cit. 2021-03-17]. Dostupné z: <https://www.jaknainternet.cz/page/1701/otevrena-data/>

[3] Stupně otevřenosti otevřených dat. Stupně otevřenosti otevřených dat a česká legislativa [online]. Praha 7: Ministerstvo vnitra ČR, 2020 [cit. 2021-04-01]. Dostupné z: <https://opendata.gov.cz/informace:stupn%C4%9B-otev%C5%99enosti-datov%C3%BDch-sad>

[4] Proč otevřená data a jaký mají přínos pro veřejnou správu? Svaz měst a obcí České republiky [online]. Praha 4: Ing. Luděk Galbavý, 2014 [cit. 2021-04-15]. Dostupné z: <https://www.smocr.cz/cs/cinnost/informatika/a/proc-otevrena-data-a-jaky-maji-prinos-pro-verejnou-spravu>

[5] China Covid-19: How state media and censorship took on coronavirus. BBC News [online]. Londýn: BBC, 2020 [cit. 2021-04-15]. Dostupné z: <https://www.bbc.com/news/world-asia-china-55355401>

[6] Covid-19 pandemic: China 'refused to give data' to WHO team. BBC News [online]. Londýn: BBC, 2021 [cit. 2021-04-15]. Dostupné z: <https://www.bbc.com/news/world-asia-china-56054468>

[7] Vakcíny (nejen) proti covid-19 pod obrovským tlakem dezinformací. Asociace inovativního farmaceutického průmyslu [online]. Praha: AIFP, 2020 [cit. 2021-04-15]. Dostupné z: <https://aifp.cz/cs/vakciny-nejen-proti-covid-19-pod-obrovskym-tlakem-/>

[8] Stát podcenil přípravu na pandemii. NKÚ [online]. Praha 7: NKÚ, 2021 [cit. 2021-04-15]. Dostupné z: <https://1url.cz/PKuYw>

[9] Jaký je rozdíl mezi respirátorem FFP2 a KN95? NanoSPACE [online]. Domažlice: nanoSPACE, 2020 [cit. 2021-04-15]. Dostupné z: <https://www.nanospace.cz/blog/jaky-je-rozdil-mezi-respiratorem-ffp2-a-kn95/>

[10] Vysvědčení pro digitální Česko. Jen osmnácté místo z osmadvaceti zemí Evropské unie. Světchytře.cz [online]. Praha 6: Tomáš Doseděl, 2019 [cit. 2021-04-15]. Dostupné z: <https://1url.cz/5KuYl>

- [11] Hrad zveřejnil platy vedoucích úředníků, Mynář měl ročně 1,6 milionu korun. ČT24 [online]. Praha 4: Česká televize, 2020 [cit. 2021-04-15]. Dostupné z: <https://ct24.ceskatelevize.cz/domaci/3184167-hrad-zverejnil-plty-vedoucich-uredniku-mynar-mel-rocne-16-milionu-korun>
- [12] Swimrankings. Swimrankings [online]. Switzerland: Splash Software, 2014 - 2021 [cit. 2021-04-15]. Dostupné z: <https://www.swimrankings.net/index.php?page=home>
- [13] FINA SWIMMING RULES. In: FINA [online]. Švýcarsko: FINA, 2017 [cit. 2021-04-15]. Dostupné z: <https://1url.cz/DKunT>
- [14] Sports Evolving Through The Hawk-Eye Technology. Online Goa [online]. India: Jesslee Crasto, 2021 [cit. 2021-04-20]. Dostupné z: <https://opspl.com/blog/sports-evolving-through-the-hawk-eye-technology/>
- [15] SWIMMING POINTS. Official FINA Website [online]. Switzerland: FINA, 2018 - 2021 [cit. 2021-04-21]. Dostupné z: <https://www.fina.org/swimming/points>
- [16] Web - How it Works? Tutorialspoint [online]. India: Tutorials Point, 2021 [cit. 2021-04-21]. Dostupné z: <https://1url.cz/AK1cT>
- [17] Frontend vs Backend. Geeks for Geeks [online]. Uttar Pradesh: GeeksforGeeks, 2021 [cit. 2021-04-21]. Dostupné z: <https://www.geeksforgeeks.org/frontend-vs-backend/>
- [18] World Wide Web. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/World_Wide_Web
- [19] Hypertext Transfer Protocol. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- [20] HTTPS. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/HTTPS>
- [21] Representational State Transfer. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Representational_State_Transfer
- [22] JavaScript Object Notation. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript_Object_Notation

[23] Responzivní web design. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Responzivn%C3%AD_web_design

[24] Hypertext Markup Language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language

[25] Kaskádové styly. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styls

[26] Bootstrap. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Bootstrap>

[27] Vlastní zdroj – práce na téma „Role ve vývoji softwaru“

[28] Python. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Python>

[29] Flask. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Flask>

[30] Framework. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Framework>

[31] JavaScript. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/JavaScript>

[32] Lekce 1 - Úvod do JavaScriptu. Itnetwork.cz [online]. Praha: David Čápka, 2021 [cit. 2021-04-21]. Dostupné z: <https://1url.cz/GK1Ym>

[33] AJAX. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/AJAX>

[34] Databáze. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Datab%C3%A1ze>

[35] NoSQL. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2021-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/NoSQL>

[36] Extensible Markup Language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Extensible_Markup_Language

[37] <https://1url.cz/JK16V>

[38] <https://www.mockflow.com/>

[39] Tesla Just Helped Patch a Bug in This Open-Source Bitcoin Payment Processor. CoinDesk Bitcoin News [online]. New York: Colin Harper, 2021 [cit. 2021-04-21]. Dostupné z: <https://www.coindesk.com/tesla-helped-patch-bug-open-source-bitcoin-payment-processor>

[40] Za výbuchem ve Vrběticích je šest ruských agentů včetně velitele komanda, píše Bellingcat a Respekt. ČT24 [online]. Kavčí Hory: ČT24, 2021 [cit. 2021-04-21]. Dostupné z: <https://1url.cz/vKeKk>

[41] Tajemný muž, který na jaře „zachránil Česko“? Exředitel České pojišťovny. Seznam zprávy [online]. Praha 5: Seznam.cz, 2021 [cit. 2021-4-21]. Dostupné z: <https://1url.cz/6KeKs>

[42] 3 swimmers who did illegal breaststroke pullout. YouTube [online]. USA: SwimStrength, 2018 [cit. 2021-4-26]. Dostupné z: https://www.youtube.com/watch?v=5vy-ak0l_w

[43] Twitter [online]. Filip Šedivý, 2021 [cit. 2021-4-26]. Dostupné z: <https://twitter.com/filipsedivy/status/1350046619429908480>

7.2. Obrázky

Obr. 1 How A Website Works: A Simple Guide. In: SuperWebPros [online]. East Lansing: Duke Kimball, 2021 [cit. 2021-04-21]. Dostupné z: <https://www.superwebpros.com/blog/how-does-a-website-work/>

Obr. 2 Responzivní web design. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Responzivn%C3%AD_web_design

Obr. 3 Vlastní práce za pomoci softwaru <https://www.mockflow.com/>

Obr. 4 – 16 Vlastní zdroj