ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačov

Diplomová práca

Využití současných trendů při tvorbě webových stránek a text miningu k automatickému vytvoření struktury webové stránky

Bc. Lukáš Figura
Študijný program: Otevřená informatika, Odbor: Softwarové inženýrství

Január 2019
Vedúci práce: Ing. Jiří Šebek

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Figura**	Jméno: **Lukáš**	Osobní číslo: **420087**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Studijní obor:	**Softwarové inženýrství**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Využití současných trendů při tvorbě webových stránek a text miningu k automatickému vytvoření struktury webové stránky

Název diplomové práce anglicky:

Use current website trends and text mining to automatically create web page structure

Pokyny pro vypracování:

Při tvorbě webové stránky v redakčních systémech neodborným uživatelem se daný uživatel potýká s problémy správného strukturálního návrhu jeho webové stránky.
Cílem práce bude těmto uživatelům automaticky nabídnout co nejlepší strukturu jeho nové webové stránky podle vstupu, který zadá a současných trendů, aby výslední stránka byla co nejkvalitnější.[1]
1. Seznamte se s technologiemi a postupy pro vývoj webových stránek a ukládání dat v databázi [2].
2. Nastudujte a zpracujte rešerši o aktuálních trendech struktur webových stránek nasledujících kategorií:
- firemní prezentační stránky,
- blogy.
3. Navrhněte a implementujte systém, který uživatelům navrhne strukturu webové stránky na základě současných trendů v designech webových stránek dané kategorie, vstupů od uživatele a na základě pravděpodobnostního modelu rozložení elementů na webové stránce.
4. Využijte agilní způsob vývoje prototypu.
5. Implementovaný systém otestujte na zkušebním vzorku uživatelů, kterým bude automaticky navržena struktura stránky. Následně na základě počtu následných úprav od uživatele empiricky vyhodnoťte do jaké míry vyhovovala navržená struktura danému uživateli.
6. Vyhodnoťte testy, výhody, možnosti dalších vylepšení a možná omezení řešení.
Vstup od uživatele:
- volba kategorie nové webové stránky (firemní prezentace, blog),
- URL exemplárních stránek, které mají být inspirací.

Seznam doporučené literatury:

1. Bevan, Nigel. "Usability issues in web site design." HCI (2). 1997.
2. Clutch.co. (2018). Small Business Websites in 2017: Survey | Clutch.co. [online]
Available at: https://clutch.co/web-designers/resources/small-business-2017-websitesurvey [Accessed 19 Aug. 2018].
3. ian sommerville: Software engineering, Global edition, Pearson Higher Ed, 2016, isbn 1292096144

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jiří Šebek, laboratoř inteligentního testování softwaru FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.09.2018** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **19.02.2020**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání	Podpis studenta

Poďakovanie / Prehlásenie

V prvom rade by som sa chcel poďakovať vedúcemu práce pánovi Ing. Jiřímu Šebkovi za všetky rady, ktoré pomohli pri vypracovaní tejto práce. Ďalej patrí moje poďakovanie rodine a priateľke, práve oni stáli pri mne v tých najťažších chvíľach a prebdených nociach nielen počas tvorby tejto práce, ale aj počas celého štúdia. V neposlednom rade by som sa chcel poďakovať spoločnosti Web Klikem, ktorá umožnila testovaciu prevádzku tohto systému a pomohla tak k zdarnému dokončeniu tejto práce. Nakoniec by som rád poďakoval všetkým kamarátom za ich konzultácie a cenné rady, ktoré pomohli pri vypracovaní tejto práce.

Prehlasujem, že som predloženú prácu vypracoval samostatne, a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiaváni etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 5. 1. 2019

.....

Abstrakt / Abstract

Táto práca obsahuje podrobný popis systému, ktorý umožňuje automatické generovanie štruktúr webových stránok v kategóriách webová prezentácia a blog. Tento systém je určený ako nástroj, pomocou ktorého majú vedieť rôzne redakčné systémy generovať štruktúry webových stránok.

Úvod je venovaný motivácií a podrobnému popisu cieľa tejto práce. Rešerš, ktorá je súčasťou tejto práce, opisuje aktuálne postupy pri tvorbe štruktúr webových stránok v kategóriách blog a webová prezentácia. Ďalšie kapitoly sa postupne venujú analýze, návrhu a implementácii riešenia. Súčasťou práce je aj testovanie, ktoré bolo formou testovacej prevádzky systému. V závere práce sa autor venuje hodnoteniu výsledkov a možnosti ďalšieho vývoja.

Kľúčové slová: element, štruktúra, webová stránka, mriežka, blog, webová prezentácia, generovať

This thesis contains a detailed description of a system that allows automatically generate web page structures in categories web presentation and blog. This system is designed as a tool by which different content management systems can create web page structures.

Introduction is devoted to motivation and detailed description of the purpose of this work. Research, which is part of this work, describes current procedures for creating web page structures in categories blog and web presentations. Next chapters deal with the analysis, design and implementation of the solution. Part of this work is testing, which was in the form of a test system operation. Final part evaluates a results and possibilities for further development.

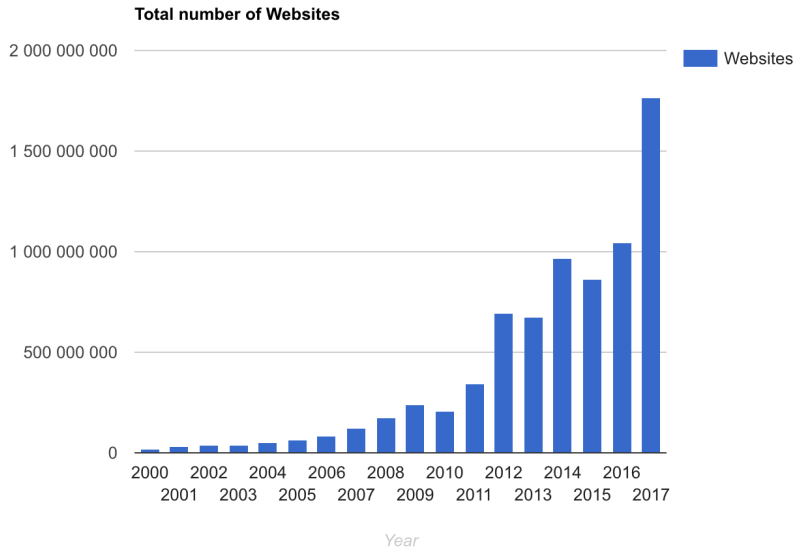
Keywords: element, structure, web page, grid, blog, web presentation, generate

Obsah /

Tabuľky / Obrázky

Kapitola 1
Úvod

V roku 2017 bol počet webových stránok v rámci celého internetu 1 766 926 408.[1] Pre tento prípad rozumieme webovou stránkou unikátne doménové meno, ktoré má pridelenú svoju IP adresu.[1] Je nutné poznamenať, že okolo 75% z týchto webových stránok sú neaktívne.[1] Avšak, ako je možné vidieť na obrázku nižšie, počet webových stránok má tendenciu sa zvyšovať. Preto je veľmi dôležité, aby mal potencionálny vlastník novej webovej stránky dostatočné množstvo možností, ako svoju webovú stránku vytvoriť.

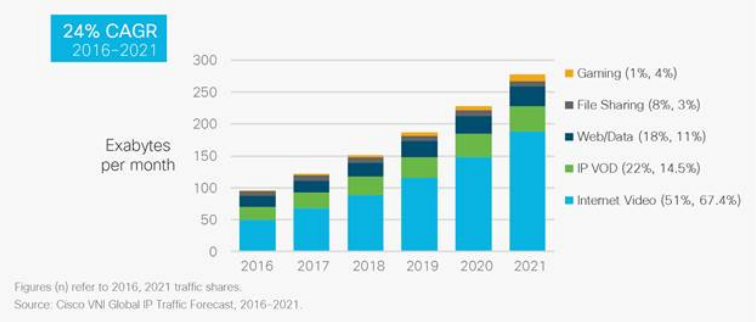


Obrázok 1.1. Celkový počet webových stránok[1]

1.1 Motivácia

V súčasnej dobe sa na internete nachádza obrovské množstvo informácií. Ľudia častokrát hľadajú informácie rôzneho druhu a to práve na internete. V roku 2016 bolo celosvetovo priemerne mesačne prenesených skoro 100 EB dát z toho 18% v rámci webových aplikácií.[2] Obrázok nižšie je predpoveďou od firmy Cisco, ktorá hovorí, ako to asi bude vyzerať v prenosoch dát v najbližších rokoch.

[1] zdroj: `http://www.internetlivestats.com/total-number-of-websites/`



Obrázok 1.2. Predpoveď celkovo prenesených dát[1]

Ako je môžné všimnúť si vyššie, samotný web tvorí skoro jednu pätinu celkového prenosu dát. Tento podiel má zároveň tendenciu sa zvyšovať, a preto je mimoriadne dôležité práve na internete publikovať informácie každého druhu formou webových stránok, príspevkov na sociálnych sieťach a mnohých ďalších. V prílohe B je možné vidieť vývojový diagram, ktorý daný scenár znázorňuje.

Pre novovzniknuté firmy je dôležité, aby čo najskôr existenciu svojej firmy zdigitalizovali, a teda umožnili potencionálnym klientom vyhľadať firmu na internete.[3] Bohužiaľ, nové firmy častokrát nemajú veľký vstupný kapitál a snažia sa svoje financie starostlivo rozdeliť. Vývoj webovej stránky profesionálom, ba dokonca agentúrou, sa môže pohybovať v rozmedzí niekoľkých desaťtisícov CZK.[4] Firmy preto častokrát siahajú po možnosti vytvoriť si svoju webovú stránku svojpomocne pomocou webových nástrojov ako Wix[2], Weebly[3], Squarespace[4] alebo Klikem[5] a pod. Cena takto vytvorenej stránky sa pohybuje na úrovni niekoľkých CZK mesačne. Náklady pre novovzniknutú firmu sú tak neporovnateľne nižšie.

Pri použití takéhoto nástroja sa však firma dostáva pred nový druh problému, a tým je správne štrukturálne vytvorenie stránky. Častokrát človek, ktorý je firmou poverený, aby danú situáciu vyriešil nemá predstavu o tom, ako by mala ich nová webová stránka vyzerať, a aké designové pravidlá by mala spĺňať. Aby boli informácie na takejto webovej stránke ľahko vyhľadateľné a zároveň, aby stránka nepôsobila dojmom zastaralosti je potrebné, aby stránka bola kvalitná aj po designovej stránke[5]. Takáto webová stránka musí spĺňať veľké množstvo pravidiel.[6] Tieto pravidlá budú bližšie predstavené v ďalších častiach tejto práce.

Nástroje na tvorbu webových stránok samozrejme ponúkajú veľké množstvo šablón, medzi ktorými si užívateľ môže vybrať, no práve takýto druh užívateľa si stránku svojej firmy častokrát nevie vôbec predstaviť. Preto je potrebné mu ponúknuť, na základe čo najmenšieho množstva od neho získaných informácií, čo najlepšiu možnú štruktúru jeho novej stránky, ktorá bude spĺňať prísne pravidlá súčasných trendov. Informácie, ktoré chce daná firma zverejniť a zviditeľniť na internete musia byť prehľadné a zároveň prezentované formou, ktorá užívateľa zaujme. Ak mu systém dokáže ponúknuť v modernej šablóne prehľadnú štruktúru jeho novej stránky, tak užívateľ nebude mať pocit bezmocnosti, ale naopak dostane impulz k tomu, v akom duchu by mal svoju stránku vytvoriť.

[1] zdroj: `https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.docx/_jcr_content/renditions/vni-hyperconnectivity-wp_12.jpg`
[2] `http://wix.com`
[3] `http://weebly.com`
[4] `http://squarespace.com`
[5] `http://klikem.cz`

1.2 Cieľ práce

Cieľom tejto práce je analyzovať aktuálne trendy v štruktúrach webových stránok v kategóriách:

- prezentačná stránka,
- blog

a vytvoriť systém, ktorý umožní moderným redakčným systémom, akým je napríklad systém Klikem, automaticky generovať štruktúry webových stránok na základe vstupu užívateľa, ktorým sú URL stránky, ktoré majú byť inšpiráciou.

Po získaní možnosti automatického generovania štruktúr stránok, systém už nebude ponúkať iba preddefinované štruktúry v šablónach. Tým sa stránky generované takýmito systémami stanú rozmanitejšie a budú viac odpovedať predstavám užívateľov. Systém Klikem sa zároveň stal partnerským systémom tejto práce a umožnil testovaciu prevádzku tohto systému.

Kapitola 2
Rešerš

Súčasťou tejto práce je aj rešerš v oblasti štruktúr webových stránok. Nižšie v tejto kapitole sú popísané aktuálne postupy pri tvorbe štruktúr v kategóriách webová prezentácia a blog. Dané kategórie stránok sú vysvetlené v podkapitolách, aby čitateľ korektne porozumel akým typom stránok sa táto práca venuje.

2.1 Webová prezentácia

Webová stránka je zbierka verejne prístupných prepojených webových stránok, ktoré zdieľajú jeden doménový názov.[7] Webové stránky môžu byť vytvorené a udržiavané jednotlivcom, skupinou alebo organizáciou. Taktiež môžu slúžiť rôznym účelom. Spoločne všetky verejne prístupné webové stránky tvoria celosvetový web.[7] Webová stránka je tiež známa ako webová prezentácia.

2.1.1 Všeobecne

Dobrá webová stránka musí byť viac ako len pekná tvár firmy alebo pútavý blog. Webová stránka musí "pracovať". [8] Taká webová stránka neprináša len štýl, ale aj obsah.[8] Každý návštevník webovej stránky ju navštívi s úmyslom získania určitých informácií, a teda je dôležité, aby tieto informácie dokázal efektne na danej webovej stránke nájsť.

2.1.2 Domovská stránka

Domovská stránka každej webovej stránky nenecháva návštevníkom žiadne pochybnosti o tom, o čom je webová stránka. [8] Je najdôležitejšou stránkou na väčšine webových stránok a má viac prezretí než každá iná stránka. [9] Samozrejme, že užívatelia neprídu stále na webovú stránku cez jej domovskú stránku.[9] Webová stránka je ako dom, v ktorom každé okno je zároveň aj dverami.[9] Ľudia môžu prísť na webovú stránku z rôznych vyhľadávačov, alebo iných stránok, ktoré odkazujú hlbšie do webovej stránky.[9] Avšak, jednou z prvých vecí, ktorú títo užívatelia urobia je, že vstúpia na domovskú stránku.[9] Účinná domovská stránka uľahčuje návštevníkom navigáciu po webovej stránke.[8]

Má dve časti: nad a pod záhybom. Pri navrhovaní domovskej stránky je veľmi dôležite, čo bude v daných častiach.

- Nad záhybom: obsah umiestnený v tejto časti by mal byť najdôležitejší, pretože návštevník sa nemusí posúvať nadol, aby ho mohol vidieť.[8] Výborným príkladom pre obsah v tejto časti je jedna veta, ktorá sumarizuje hodnotu, ktorú webová stránka prináša[9].
- Pod záhybom: podrobnejší obsah, ktorý stručne podporuje zámer webovej stránky. Napríklad `zoznam výhod`. Ak webová stránka prezentuje firmu, tento zoznam vysvetľuje, prečo by si návštevník mal vybrať práve ju. Ďalším sekundárnym obsahom môžu byť ukazovatele dôvery, ako napríklad rýchly náhľad recenzií.[8]

Menej je viac. Hlavne domovská stránka nesmie byť preplnená, alebo príliš komplikovaná.[8]

2.1.3 Logo

Ak je k dispozícii logo, máme niekoľko možností pri výbere, kam ho umiestniť.[8] Ak sa webová stránka zameriava na produkty alebo služby, nie je potrebné venovať logu veľa miesta. Ak sa webová stránka zameriava viac na značku, je vhodné umiestniť logo do stredu pod lištu ponuky, teda nad záhyb webovej stránky.[8]

2.1.4 Navigácia

Navigačný panel je menu, ktoré návštevníci používajú na navigáciu na webovej stránke.[8] Existuje preto, aby pomohlo návštevníkovi nájsť obsah, ktorý hľadá. [10] Dôvod, prečo existujú konvencie je, že sú založené na myšlienkach, ktoré fungujú.[10] Jediný čas, kedy by ste mali prelomiť konvencie, je, ak máte lepší spôsob, ako niečo urobiť. [10]

Vedieť, ako navrhnúť navigáciu webových stránok zahŕňa:

- pridanie iba potrebných stránok,
- budovanie kategórií súvisiacich odkazov,
- názvy stránok sú krátke a jasné,
- navigačná lišta je ľahko viditeľná.[8]

Každá časť navigácie musí mať pre užívateľa jasný koncový cieľ.[8] Názvy stránok v navigácii musia byť špecifické napríklad: `Preprava tovaru` namiesto `Naše služby`. Je dôležité návštevníkov nepreťažovať veľkým množstvom možností.[8] Povedie to k nerozhodnosti.[8] Menej položiek v navigácii je pre užívateľov lepším riešením.[10] Krátkodobá pamäť udrží približne iba 7 položiek. [11] Väčšina webových stránok obsahuje navigačný panel v hornej časti stránky. Návštevníci webu očakávajú, kde navigáciu nájdu. [10] Najdôležitejšími položkami navigácie sú stránky `Kontakt` a `O nás`.[10] Tieto stránky totiž odpovedajú na individuálne otázky väčšiny užívateľov [10].

2.1.5 Obsah

Webový obsah je textový, vizuálny alebo zvukový obsah, ktorý sa zobrazuje užívateľovi na webových stránkach. Môže to zahŕňať okrem iného text, obrázky, zvuky, videá a animácie. Animácie ako také nepredkladajú užívateľovi žiadne informácie, ktoré môže nadobudnúť. Slúžia k tomu, aby užívateľovi bolo prezeranie webovej stránky prívetivejšie. Animované obrázky sú v tejto práci chápané ako obrázky.

2.1.5.1 Text

Textový obsah webovej stránky by mal byť viac než len skrášlenie návštevníkového webového prehliadača.[8] Musí slúžiť svojmu účelu. Zároveň musí byť elegantný a pútavý.[8] Pre vytvorenie častí vo veľkých blokoch textu sa odporúča ich obklopiť bielym priestorom.

Je potrebné mať tiež na pamäti SEO. SEO znamená optimalizáciu pre vyhľadávače. Je to možnosť pre získavanie návštevnosti z vyhľadávacích nástrojov, ako je Google. Veľkou časťou SEO je písanie zacieleného obsahu.[8] Ak teda webová stránka slúži na prezentáciu firmy, ktorá poskytuje IT služby, tak potom obsah stránky by mal byť zacielený práve na tieto služby. Taktiež je potrebné, aby sa kľúčové slová,ktoré súvisia s účelom webovej stránky, vyskytovali v nadpisoch vyššej úrovne. [8]

2.1.5.2 Obrázky

Obrázky predstavujú možnosť, ako zveľadiť skromný web.[8] Vizuálny obsah zvyšuje počet kliknutí a interakciu.[8] Navyše, obrázky sa v pamäti udržia dlhšie. Ak človek

počuje nejakú informáciu, pamätá si len 10%, ale ak je k nej pridaný obrázok, tak si pamätá neuveriteľných 65%. [12] Je však potrebné myslieť na to, aby webová stránka nebola zahltená veľkým množstvom obrázkov. Takto zahltená webová stránka vykazuje hneď niekoľko problémov:

- spomalenie načítania obsahu,
- veľké množstvo informácií, ktoré upútajú na prvý pohľad.[8]

Spomalené načítanie obsahu má za následok netrpezlivých návštevníkov, ktorí následne môžu na základe tohto problému z webu odísť.[8]

2.1.5.3 Videá

Webový design určite ťažil z vytvorenia HTML5, najmä preto, že zjednodušil proces aplikácie videa, ako pozadia webovej stránky.[13]

Ak sa to robí správne, webové stránky s videom môžu vyzerať umelecky a štýlovo.[13] V skutočnosti zaujímavé videá vytvárajú „vau“ efekt na mnohých webových stránkach, aj keď nie sú hlavným zdrojom informácií danej webovej stránky (produkty a služby sú často vyjadrené tradične obrázkami a textom).[13]

Inými slovami, videá môžu byť oveľa zaujímavejším médiom ako obrázky alebo text, bez ohľadu na to, čo majú povedať.[13] Je však potrebné znova podotknúť, že v dnešnej dobe, ako už bolo spomenuté sú webové stránky zväčša navštevované hlavne z mobilných zariadení. Na základe tejto skutočnosti je potrebné pri umiestňovaní videa na webovej stránke myslieť na vyššiu dátovú záťaž, ktorá môže ovplyvniť načítanie samotnej stránky.

2.1.5.4 Zvuky

Zvuky, ako súčasť webovej stránky sú na obrovskom ústupe. Predstavme si túto situáciu: pri surfovaní po webe, hľadaním inšpirácie, kliknutím na niektoré odkazy sa stránky otvárajú na pozadí pre neskoršiu kontrolu.[14] Náhle začnú zo zariadenia vychádzať zvuky.[14] Ľudia považújú hudbu na pozadí webovej stránky za nepríjemnú.[14] Webová stránka jednoducho nie je vec, ktorá by očakávane vydávala nejaký zvuk.[14] Pri prehliadaní webu ľudia často počúvajú hudbu, a aj preto nie je priestor pre ďalšiu vrstvu zvuku.[14]

Je však rozdiel v tom či je zvuk použitý ako pozadie, alebo slúži na konkrétnu prezentáciu určitého zamerania webovej stránky.[14] Správne používanie správnych zvukov môže skutočne zlepšiť používateľskú skúsenosť s webovou stránkou.[14] Jedná sa prevažne o umelecké webové stránky, ktoré napríklad prezentujú hudobnú tvorbu.

2.2 Blog

Jedným z najvýraznejších prvkov éry Web 2.0 je vzostup blogovania. Najzákladnejší blog pozostáva z osobnej stránky vo formáte denníka. [15] Blog je online časopis, alebo informačná webová stránka zobrazujúca informácie v opačnom chronologickom poradí, pričom posledné príspevky sa zobrazujú ako prvé.[16] Je to platforma, v ktorej spisovateľ, alebo dokonca skupina spisovateľov zdieľajú svoje názory na jednotlivé témy. [16] Kľúčové prvky, ktoré odlišujú príspevok na blogu od príspevku na statickej stránke, zahŕňajú dátum zverejnenia, referenciu autora, kategóriu a značky. Hoci nie všetky blogové príspevky obsahujú všetky vyššie spomenuté prvky, statické webové stránky nemajú žiadnu z týchto položiek.[16] Z pohľadu návštevníkov sa obsah na statickej stránke z jednej návštevy na druhú nezmení. Obsah blogu má napriek tomu potenciál

ponúknuť niečo nové každý deň, týždeň alebo mesiac v závislosti od plánu publikovania vlastníka blogu, alebo spisovateľov, ktorí na daný blog prispievajú.

Dnešné blogy obsahujú rôzne funkcie, avšak väčšina blogov obsahuje určité štandardné funkcie a zachovávajú si aj svoju štruktúru. Medzi štandardné funkcie, ktoré bude obsahovať typický blog patrí:

- záhlavie s navigačnou lištou,
- hlavná oblasť obsahu so zvýraznenými, alebo najnovšími blogovými príspevkami,
- pätička s relevantnými odkazmi, akými sú napríklad zásady ochrany osobných údajov, kontaktná stránka.

Obrázok 2.1. Ukázka štruktúry dnešných blogov

2.2.1 Všeobecne

Keďže sa za posledné obdobie výrazne zvýšila výkonnosť mobilných zariadení, zaznamenali sme migráciu od hustých, textovo náročných blogov až po ďalšie vizuálne stránky s veľkými farebnými obrázkami. Jedným z najvýznamnejších trendov v oblasti tvorby blogov je rastúca popularita obrovských snímok, ktorý pravdepodobne bude pokračovať aspoň v predvídateľnej budúcnosti.[17] Sú zaujímavé, vizuálne pôsobiace a môžu slúžiť ako základ pre celé návrhy, a to predovšetkým pre jednostránkové webové stránky s dlhou hĺbkou posúvania stránky, ktoré sa spoliehajú na nádherné obrázky s vysokým rozlíšením, ako potravinové blogy.[17] Samozrejme, veľké nádherné obrázky nie sú bez nevýhod. Stránky, ktoré sa rozhodnú použiť tento prvok dizajnu musia zabezpečiť, aby boli ich obrázky dôkladne optimalizované z technických dôvodov, ako je napríklad čas načítania stránky.

Aj keď mnoho webových stránok využíva silu tohto dizajnu, ďalšie stránky presadzujú iný, výrazne odlišný estetický dizajn - skutočne minimalistickým prístupom, ktorý úplne vylučuje obrazy. [17] Minimalistické návrhy blogov môžu byť pre určité typy obsahu veľmi účinné. Táto estetika sa ukázala ako populárna medzi spisovateľmi. [17] Pri

niektorých typoch blogových príspevkov, ako sú napríklad články založené na názore, môže tento formát fungovať veľmi dobre. [17]

2.2.2 Navigácia

Navigačné panely v prípade blogu sa držia pravidiel, ktoré sú bližšie popísané v podkapitole 2.1.4. Ako položky navigačného menu sa v blogu používajú kategórie tém, na ktoré sa v danom blogu publikujú príspevky.

2.2.3 Obsah

Bočné panely preplnené tlačidlami, odznakmi, reklamami a inými položkami už dávno nie sú súčasťou mnohých blogov. [17] Existuje pre to niekoľko nasledujúcich dôvodov. Po prvé, keďže metódy UX dizajnu sa zamerali do iných oblastí dizajnu, dôraz sa posunul od vypĺňania veľkého množstva obsahu do bočných panelov smerom k čistejším a efektívnejším rozloženiam blogov.[17] Tieto princípy sa uplatňujú aj na iné aspekty webového dizajnu, ako je napríklad štruktúra stránky a navigácia, ktoré tiež pomáhajú v SEO oblasti, ako aj pri znižovaní technických nákladov. [17] Po druhé, zvyšujúca sa miera používania mobilných zariadení si vyžiadala nové prístupy v oblasti vytvárania blogov, ktoré zvýhodňujú rýchlosť a výkon - obidva môžu byť negatívne ovplyvnené ďalšími prvkami, ako sú husto vyplnené bočné panely a designy, ktoré vyzerajú skvele aj na menších obrazovkách.[17]

Kapitola 3
Analýza

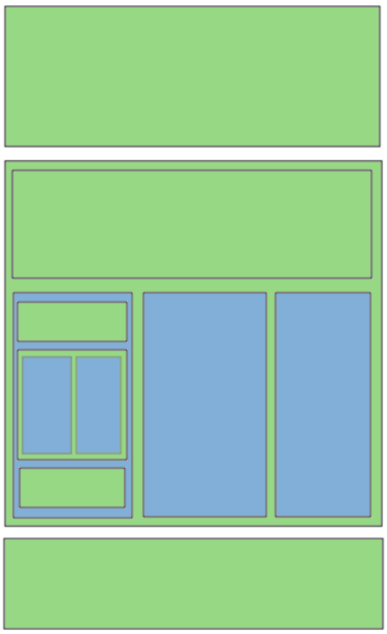
Analýza je dôležitou súčasťou vývoja softvéru. V tejto etape vývoja softvéru sa prehlbujú vedomosti o probléme, ktorý daný softvér rieši.[18] Dochádza k analýze dát, procesov a požiadaviek, ktoré sú súčasťou daného problému.[18] Táto kapitola sa teda venuje analýze dát, SWOT, existujúcich riešení a požiadavkou na nástroj, ktorý má byť produktom tejto práce.

3.1 Základné pojmy

Keďže táto práca nemá byť návodom, prípadne referenčnou príručkou k HTML, CSS a javascripte, autor tejto práce predpokladá aspoň základnú znalosť týchto technológii a ich pôsobenie v oblasti webových stránok. Pre korektné pochopenie významu niektorých pojmov použitých v tejto práci budú nižšie predstavené. Tieto pojmy sa budú v práci vyskytovať často, preto je dôležité, aby čitateľ korektne porozumel ich významu v tejto práci.

3.1.1 Štrukturálny element

Pod týmto pojmom rozumieme časť webovej stránky, ktorá jednoznačne štrukturálne vymedzuje priestor pre vkladanie dátových elementov. Rozlišujeme dva štrukturálne elementy - riadok a stĺpec. Štrukturálne elementy môžu byť rekurzívne vkladané do iných štrukturálnych elementov tak, ako je to znázornené na obrázku nižšie.



Obrázok 3.1. Rekurzívne vkladanie štrukturálnych elementov

Na obrázku vyššie sú riadky vyplnené zelenou a stĺpce modrou farbou. Hĺbku rekurzívneho vkladania jednotlivých štrukturálnych elementov obmedzuje iba technické riešenie konkrétneho redakčného systému.

3.1.2 Dátový element

Pod dátovým elementom rozumieme HTML element, ktorý koncovému užívateľovi prezentuje dáta, ktoré so sebou nesie. Tieto dáta môžu byť v obrazovej, textovej alebo zvukovej podobe.

3.1.2.1 Obrázok

Tento dátový element je v HTML zastúpený nepárovým tagom img[1]. Tento tag sa používa práve pre vloženie obrázkového dátového elementu na webovú stránku. Samozrejme existujú aj iné možnosti ako vložiť obrázok na webovú stránku. Pri definovaní kaskádových štýlov webovej stránky (CSS) sa môže obrázok definovať ako pozadie niektorého zo štrukturálnych, alebo dátových elementov. Avšak pri analýze HTML nevieme takéto použitie obrázku zistiť a preto budeme pracovať iba s tagmi img.

3.1.2.2 Video

V HTML je viacero možností, ako takýto dátový element zahrnúť do webovej stránky. Prioritne je k tomu určený párový tag video, avšak v praxi sa používajú aj ďalšie tagy ako napríklad:

- iframe[2] (párový),
- embed[3] (nepárový),
- object[4] (párový).

Použitie tagov embed a object je označené ako zastarané, preto nebudeme uvažovať v tejto práci nad používaním týchto elementov ako dátových. Element iframe je do značnej miery univerzálny a môže slúžiť aj na vkladanie samotných webových stránok. Ak by sme chceli jednoznačne určovať tento element ako nositeľa videa, bolo by nutné vytvoriť funkciu, ktorá by dokázala zo zdrojového súboru pre element iframe vytiahnuť informáciu, či sa jedná o video. Takáto funkcia môže byť možnosťou pre ďalší vývoj. Budeme teda pracovať iba s tagom video.

3.1.2.3 Nadpis

K definícii nadpisu sa v HTML používajú párové tagy h1, h2, h3, h4, h5 a h6[5]. Jednotlivé číslice tagov definujú nadpisovú úroveň. Čím je číslo vyššie tým je úroveň nadpisu nižšia.

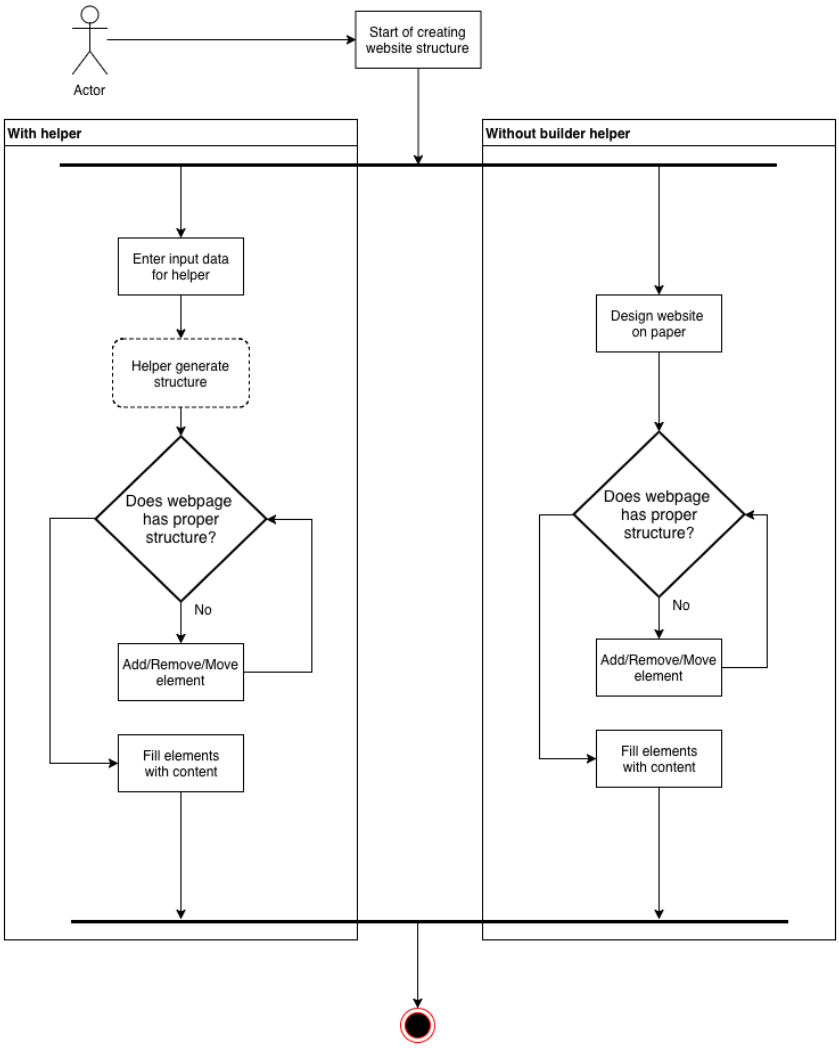
3.1.2.4 Text

Textový dátový element v HTML bude predstavovať pri analýze webovej stránky najväčší problém. Text sa totiž v HTML môže vyskytovať voľne. Ako textový element budeme teda brať dátový element, ktorý nie je obrázkom, videom ani nadpisom. Šírka a výška musí byť väčšia ako 0px a zároveň musí element obsahovať text.

[1] https://www.w3schools.com/tags/tag_img.asp
[2] https://www.w3schools.com/tags/tag_iframe.asp
[3] https://www.w3schools.com/tags/tag_embed.asp
[4] https://www.w3schools.com/tags/tag_object.asp
[5] https://www.w3schools.com/html/html_headings.asp

3.2 Existujúce riešenie

Existuje mnoho rôznych redakčných systémov pre tvorbu a správu webových stránok. Tieto systémy sa vyznačujú rôznymi prístupmi v tvorbe štruktúry webovej stránky. Jedným z týchto prístupov je vytvorenie prvotného rozloženia stránky pomocou riadkov a stĺpcov. Následne sa do týchto štrukturálnych prvkov vkladajú už konkrétne dátové elementy. Druhým rozšíreným prístupom je priame vkladanie dátových elementov do webovej stránky. Pri tomto prístupe je tvorba riadkov a stĺpcov obmedzená iba technickým riešením zo strany redakčného systému.



Obrázok 3.2. Tvorba stránky bez a s asistentom pre tvorbu štruktúr

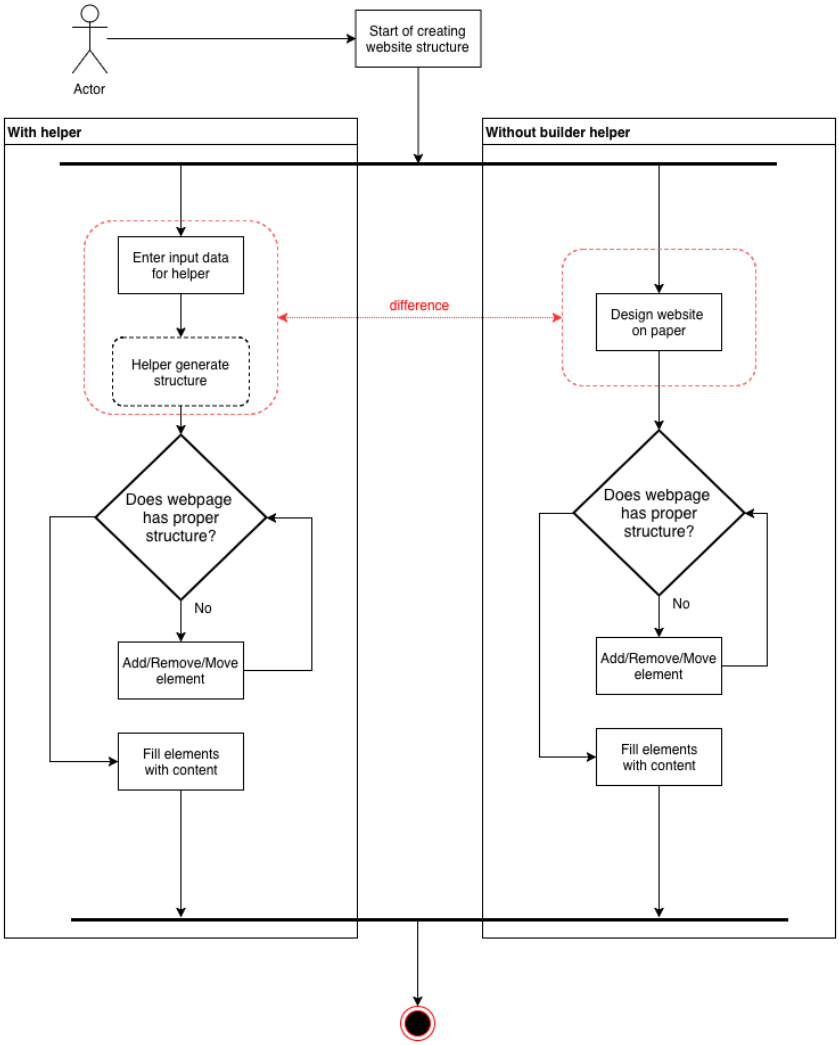
Jediným redakčným systémom, ktorý umožňuje užívateľovi vytvorenie automatickej štruktúry webovej stránky je systém Wix. Funkcia, ktorá sa stará o vytvorenie štruktúry sa nazýva Wix ADI (z anglického Wix Artificial Design Intelligence). Spoločnosť Wix prezentuje túto funkciu ako jedinú na svete, ktorá využíva ku generovaniu webových stránok umelú inteligenciu.[19] Túto funkciu si do značnej miery strážia a do momentu vypracovania tejto práce sa autorovi nepodarilo zistiť žiadne bližšie špecifiká-

cie. Užívateľovi je predložená séria jednoduchých otázok, na základe ktorých Wix ADI vygeneruje užívateľovi unikátnu štruktúru webovej stránky.[19]

Bohužiaľ ostatné redakčné systémy takúto funkciu nemajú a preto sú užívatelia odkázaní na svoju predstavivosť. Niektoré, redakčné systémy ponúkajú ešte po voľbe designovej šablóny prednastavenú štruktúru webovej stránky. Tieto štruktúry, ale nie sú často aktualizované a často nesúhlasia s predstavami používateľa.

3.3 Navrhované riešenie

Predmetom tejto práce je vytvorenie systému, ktorý dokáže koncovému užívateľovi redakčného systému vygenerovať štruktúru webovej stránky na základe vstupu zo strany užívateľa v podobe kategórie webovej stránky a URL adries webových stránok, ktoré majú byť inšpiráciou. Zároveň majú byť pri generovaní zohľadnené súčasné trendy v oblasti štruktúr webových stránok, ktoré boli bližšie popísané v kapitole 2.



Obrázok 3.3. Rozdiel medzi existujúcimi riešeniami a navrhovaným

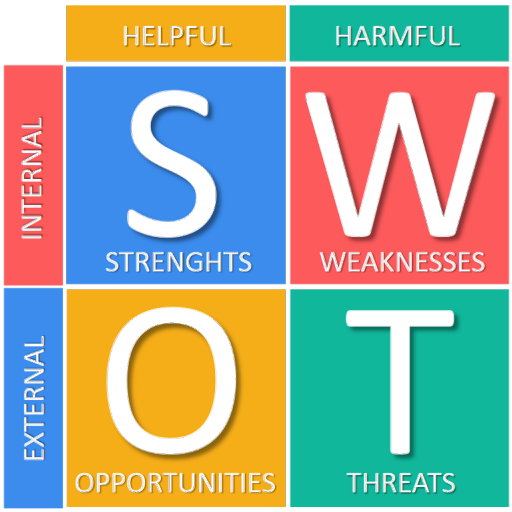
Existujúce riešenie, ktoré ako tvrdí Wix, je tvorené umelou inteligenciou. Takéto riešenie môže byť do značnej miery prestrelené. Wix ADI potrebuje k svojej funkcii

veľké množstvo dát, ktoré získava nielen zo samotných webových stránok, ale aj zo sociálnych sietí.[19] Vývoj takéhoto systému, ktorý je postavený na umelej inteligencii, môže byť do značnej miery náročný nielen po finančnej stránke ale aj časovej.

Navrhované riešenie preto pristupuje k problému jednoduchšou cestou, cestou analýzy webových stránok formou text miningu a následnému vytvoreniu elementových mriežok, ktoré abstrahujú štruktúry analyzovaných webových stránok. Tieto elementové mriežky sú následne kombinované pre vytvorenie výslednej elementovej mriežky, z ktorej redakčný systém dokáže vytvoriť webovú stránku.

3.3.1 SWOT analýza

Analýza SWOT slúži na hodnotenie vonkajších a vnútorných vplyvov. Následne z nich vyplývajú silné a slabé stránky strategického cieľa. Cieľom tejto práce je rozšírenie tohto nástroja medzi redakčnými systémami, aby mohli ponúknuť generovanie štruktúry webovej stránky.



Obrázok 3.4. SWOT diagram[1]

3.3.1.1 Silné stránky

Nástroj bude pre redakčné systémy poskytovať možnosť jednoduchého využitia pre získanie možnosti generovania štruktúr webových stránok. Realizácia bude formou softvéru ako služby, ktorý bude použiteľný prostredníctvom REST API.

3.3.1.2 Slabé stránky

Slabou stránkou nástroja môže byť forma, akou budú štruktúry stránok generované. Či je spôsob generovania štruktúr slabou stránkou nástroja ukáže až testovacia prevádzka. Ďalšou slabou stránkou môže byť aj to, že významný vplyv na konečnú podobu stránky majú samotné redakčné systémy, ktoré musia vygenerovanú štruktúru "preložiť" do konečnej podoby webovej stránky.

3.3.1.3 Príležitosti

Kontaktovanie množstva redakčných systémov a ich následné používanie tohto nástroja, môže zapríčiniť samovoľné používanie tohto nástroja aj ostatnými redakčnými systémami.

[1] zdroj: `https://www.business-to-you.com/wp-content/uploads/2017/06/SWOT-Analysis.png`

3.3.1.4 Ohrozenia

Nezáujem redakčných systémov o využívanie externého nástroja pre generovanie štruktúr webových stránok. Ohrozenie môže spočívať aj v napadnutí systému cez REST API a následné poškodenie funkčnosti nástroja. Autor tejto práce sa pokúsi toto ohrozenie eliminovať zabezpečením REST API. Tomuto zabezpečeniu sa venuje práca nižšie.

3.3.2 Funkčné požiadavky

3.3.2.1 Autentifikácia a autorizácia

Systém je potenciálnemu užívateľovi prezentovaný jednoduchou prezentačnou stránkou. Pre použitie nástroja je však nutná registrácia a prihlásenie. Následne má užívateľ možnosť registrovania tokenu, ktorý bude používať pri práci s REST API. Užívateľ sa bude prihlasovať emailom a heslom, ktoré zadal pri registrácii.

3.3.2.2 Uloženie hesla

Diskrétnosť hesla je zabezpečená uložením jeho hashovanej formy. Hash hesla bude vypočítaný pomalou hashovacou funkciou.[20] V databáze budú heslá uložené s prefixom - soľou.

3.3.2.3 Reset hesla

Ak nastane situácia, že užívateľ svoje heslo zabudne, má možnosť svoje heslo resetovať. Samotné obnovenie hesla bude pozostávať z dvoch krokov. V prvom kroku užívateľ zadá svoj registračný email, na ktorý mu bude zaslaný odkaz s unikátnou URL. Následne po prejdení na danú URL bude užívateľovi ponúknuté zadanie nového hesla.

3.3.2.4 Kategória webovej stránky

Webové stránky môžu byť rôzneho typu. Nástroj, ktorý je predmetom tejto práce bude poskytovať možnosť práce s webovými stránkami v kategóriách webová prezentácia a blog.

3.3.2.5 Analýza inšpiračnej stránky

Nástroj bude poskytovať možnosť štrukturálnej analýzy webovej stránky, ktorú bude definovať predložené URL. URL budú spĺňať normu RFC 18081[1]. Stránky, ktoré majú byť analyzované musia byť dostupné pomocou protokolu HTTP alebo HTTPS.

3.3.2.6 Náhľad analyzovanej stránky

Systém bude poskytovať náhľad analyzovanej stránky. Náhľad bude zabezpečený formou obrázku, ktorý bude zobrazovať stránku v čase analýzy.

3.3.2.7 Generovanie štruktúry stránky v predloženej kategórii

Hlavnou funkčnou požiadavkou systému je generovanie štruktúry webovej stránky na základe predloženej kategórie webovej stránky a URL stránky, ktorá má byť inšpiráciou. Vygenerovaná štruktúra bude zložená z dátových elementov, ktoré sú súčasťou výslednej webovej stránky.

[1] `https://www.w3.org/Addressing/rfc1808.txt`

3.3.3 Nefunkčné požiadavky

3.3.3.1 Výkon

Navrhovaný systém musí spĺňať základné podmienky škálovania, aby mohol byť využívaný množstvom redakčných systémov. Ďalej musí spĺňať podmienky, ktoré sú predpísané pre REST API:

- jednotné rozhranie (z angl. uniform interface),
- bezstavovosť (z angl. stateless),
- kešovateľnosť (z angl. cacheable),
- model Client-Server,
- vrstvený systém (z angl. layered system).

Analýza štruktúry webovej stránky môže byť v závislosti od veľkosti stránky výkonovo náročná, preto je nutné zabezpečiť, aby implementácia výkonovo náročných častí mala dostatočné rezervy a zvolená technológia, aby poskytovala dostatočné možnosti pre výkonovú optimalizáciu.

3.3.3.2 Jednoduché použitie

Aby boli redakčné systémy podstrčené do voľby práve využitia externého systému pre generovanie štruktúr webových stránok, je potrebné aby použitie tohto systému bolo jednoduché. Jednoduché použitie zabezpečí implementácia REST API. Zo strany redakčných systémov bude teda potrebné implementovať iba komunikáciu s API a následné preklopenie vygenerovanej štruktúry webovej stránky do webovej stránky, ktorá bude spravovateľná v ich systéme.

3.3.3.3 Dokumentácia REST API

API, ktoré bude systém poskytovať musí byť zdokumentované, aby osoba zodpovedná za napojenie redakčného systému na tento nástroj nebola nútená k emailovej komunikácii s autorom tejto práce. Keďže redakčné systémy môžu byť implementované v rôznych technológiách budú súčasťou dokumentácie aj príklady dotazov na API v rôznych technológiách.

Kapitola 4
Návrh

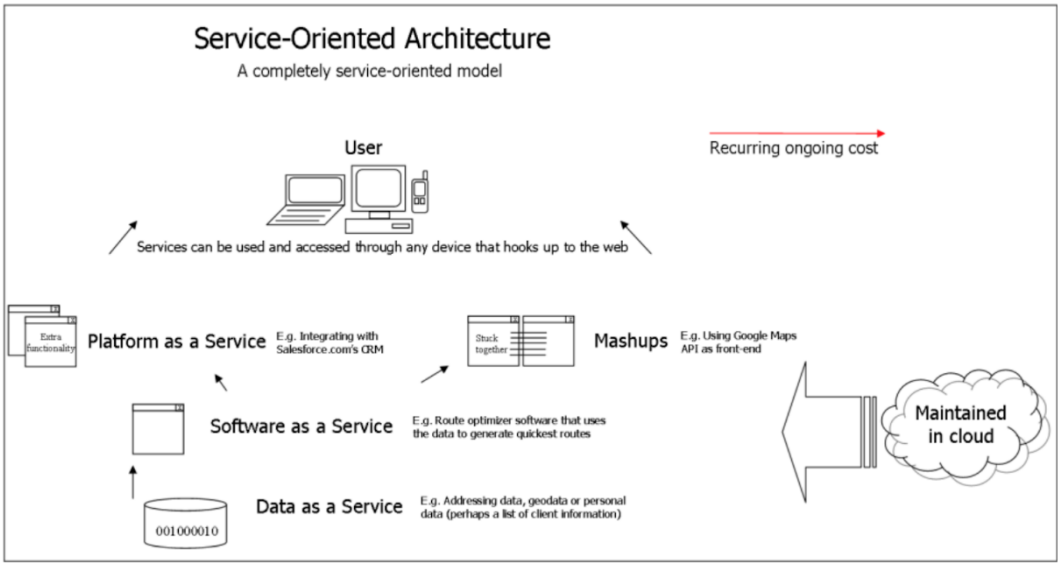
V tejto etape vývoja softvéru sa pracuje s výsledkami z etapy analýza a navrhujú sa riešenia.[21] Návrh a analýza by sa mali striedať dovtedy, kým nebude návrh kompletný.[21] Niektoré postupy v skutočnosti spájajú analýzu a návrh do jednej etapy.[21] Nižšie teda bude bližšie predstavená architektúra aplikácie a navrhované riešenie.

4.1 Architektúra aplikácie

Pre architektúru bol zvolený model Client - Server. Tento model bol vybraný ako predpísaná podmienka v REST API, o ktorej je možné sa dočítať v kapitole 3.3.3. Nástroj bude dostupný formou softvéru ako služby (SaaS - z angl. Software as a Service).

4.1.1 Softvér ako služba

Ponúkanie softvéru online bolo dlhým snom mnohých spoločností a CIO.[22] Koncept takéhoto ponúkania softvéru je veľmi jednoduchý. Nevyžaduje žiadne inštalácie zo strany používateľa. Ďalším veľkým plusom je, že infraštruktúru pre prevádzku softvéru zabezpečuje prevádzkovateľ softvéru. SaaS sa zameriava na oddelenie vlastníctva softvéru od jeho používania. Poskytovanie softvéru, ako súbor distribuovaných služieb, ktoré je možné nakonfigurovať v čase dodania, môže prekonať mnohé obmedzujúce súčasné obmedzenia používanie softvéru, nasadenie a vývoj.[23] SaaS je metodológia pre poskytovanie počítačových služieb cez internet.[24] SOA (z angl. Software Oriented Architecture) je softvérová architektúra, ktorá ovláda SaaS aplikáciu. V SOA architektúre sú najčastejšie používané formáty, ktoré budú predstavené neskôr v tejto kapitole. Obrázok nižšie zobrazuje rolu SOA v SaaS.



Obrázok 4.1. Rola SOA v SaaS[23]

4.1.1.1 JSON

Formát JSON sa často používa na serializáciu a prenos štruktúrovaných údajov pomocou sieťového pripojenia. Prináša alternatívu k XML, ktoré môže byť za určitých podmienok objemnejšie.

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                {"value": "New", "onclick": "CreateNewDoc()"},
                {"value": "Open", "onclick": "OpenDoc()"},
                {"value": "Close", "onclick": "CloseDoc()"}
            ]
        }
    }
}
```

Obrázok 4.2. Náhľad formátu JSON[1]

4.1.1.2 XML

Návrhové ciele XML zdôrazňujú jednoduchosť, všeobecnosť a použitie v oblasti internetu. Jedná sa o textový formát, aj keď ktorého návrh cielil na dokumenty, je široko používaný na zobrazenie rôznych dátových štruktúr napríklad vo webových službách.

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Obrázok 4.3. Náhľad formátu XML[2]

4.1.1.3 ATOM

Názov Atom sa vzťahuje na dvojicu súvisiacich štandardov. Formát pre atómovú syndikáciu (z angl. The Atom Syndication Format) je XML jazyk používaný pre webové zdroje, zatiaľ čo zverejňovací protokol atom (z angl. Atom Publishing Protokol) je jednoduchý protokol založený na protokole HTTP pre vytváranie a aktualizáciu webových zdrojov.

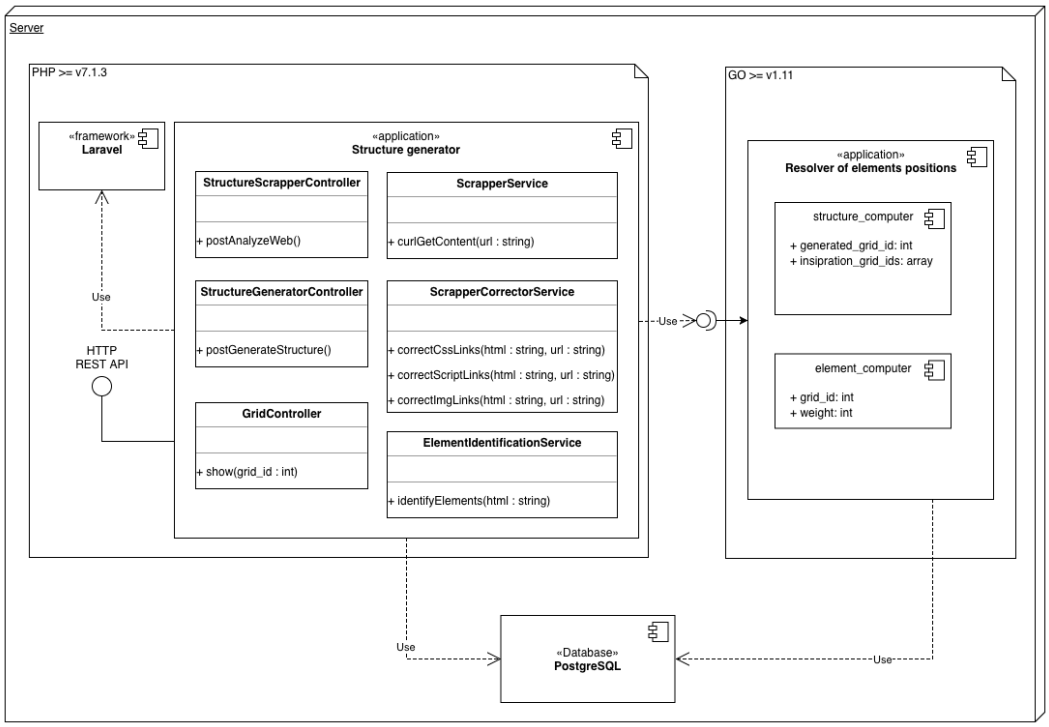
[1] zdroj: `https://json.org/example.html`
[2] zdroj: `https://json.org/example.html`

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

    <title>Example Feed</title>
    <link href="http://example.org/"/>
    <updated>2003-12-13T18:30:02Z</updated>
    <author>
        <name>John Doe</name>
    </author>
    <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

    <entry>
        <title>Atom-Powered Robots Run Amok</title>
        <link href="http://example.org/2003/12/13/atom03"/>
        <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
        <updated>2003-12-13T18:30:02Z</updated>
        <summary>Some text.</summary>
    </entry>
</feed>
```

Obrázok 4.4. Náhľad formátu v technológii ATOM[1]

4.1.2 Oddelenie výpočtovej časti

Systém je rozdelený na dve nezávislé časti. Prvá časť systému je hlavnou obslužnou časťou celého systému. Zodpovedá za aktualizáciu dát v databáze, za spúšťanie výpočtovej časti systému a za prezentáciu dát prostredníctvom REST API. Druhou časťou je už spomenutá výpočtová časť systému. Diagram nižšie bližšie znázorňuje samotné rozdelenie systému na jednotlivé nezávislé funkčné časti. Obe časti majú vlastné pripojenie k databáze, aby nebolo nutné prenášanie veľkého objemu dát medzi nimi. Podrobnému popisu samotných častí sa venuje práca nižšie.



Obrázok 4.5. Rozdelenie systému na samostatné časti

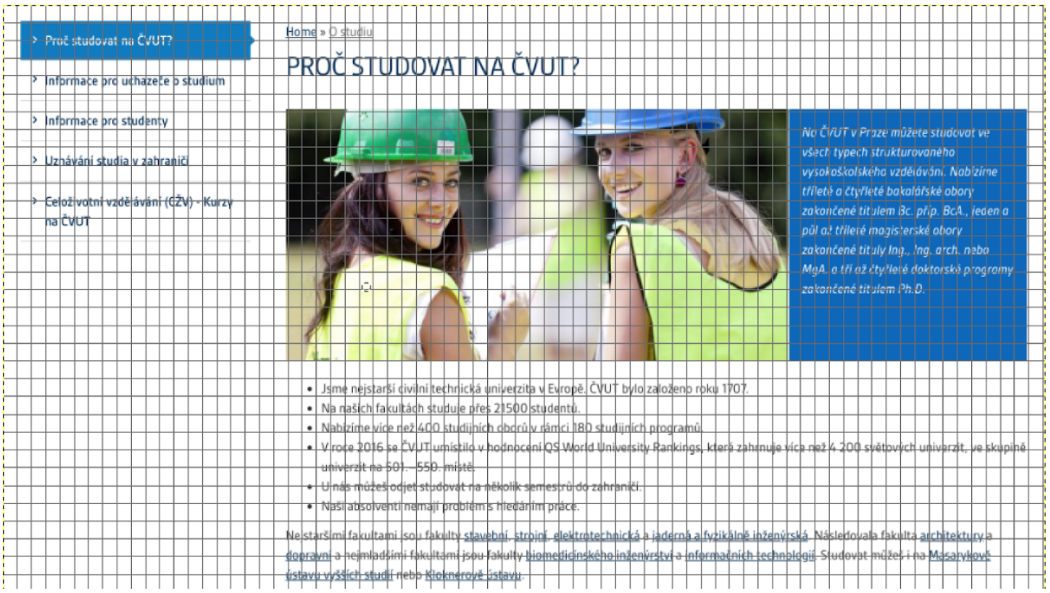
[1] zdroj: `https://tools.ietf.org/html/rfc4287`

4.2 Analýza štruktúry

Analýza štruktúry webovej stránky sa bude skladať z nasledujúcich krokov:

- stiahnutie zdrojového kódu analyzovanej stránky,
- kvalifikácia elementov v zdrojovom kóde, ktoré budú následne merané,
- spustenie analyzovanej stránky lokálne v nástroji,
- odmeranie a uloženie nameraných údajov z analyzovanej stránky,
- analýza odmeraných údajov a vytvorenie štruktúry elementov webovej stránky.

Z vyššie uvedených krokov bude posledný krok vykonávaný v oddelenej výpočtovej časti systému. Analýza odmeraných údajov bude prebiehať formou rozdelenia stránky na mriežku rovnako veľkých blokov a následný výpočet koľko a akých elementov sa v každom bloku nachádza. Rozdelenie stránky do mriežky zobrazuje obrázok nižšie.



Obrázok 4.6. Rozdelenie stránky do mriežky

Pod vyššie zmieneným blokom si predstavte práve jeden štvorček na obrázku vyššie. V každom z týchto blokov sa určí počet jednotlivých typov elementov. Pre uvedenie príkladu si pohľadom vyberte blok v mriežke, ktorú vidíte na obrázku vyššie. V prípade ak blok, ktorý ste si vybrali je nad obrázkom, ktorý sa nachádza v strede analyzovanej stránky, bude v tomto bloku zvýšený počet elementov typu obrázok o jedna.

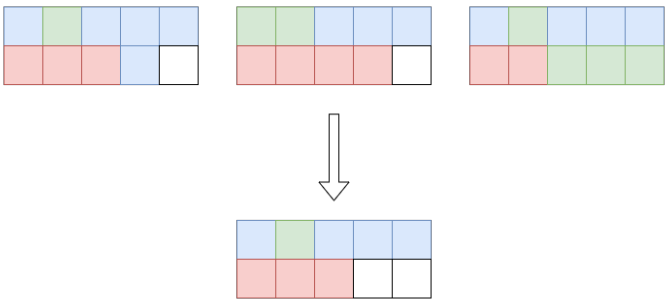
Takýto druh analýzy je potrebný z dôvodu, že v elemente obrázok môže byť vložený napríklad element typu nadpis, a teda v takto analyzovanom bloku nebude iba počet elementov typu obrázok jedna, ale aj počet elementov typu nadpis jedna. Túto situáciu bližšie znázorňuje obrázok nižšie, na ktorom je výrez zo stránky, kde práve takáto situácia nastáva tým, že text “# DENTista, zubař pro všechny” je vložený akoby “do” obrázku.

Obrázok 4.7. Nadpis nad obrázok

Po takto analyzovanej stránke bude do databázy vložená matica, ktorá bude popisovať analyzované bloky a počet jednotlivých typov elementov v nich. Náhľad takejto štruktúry nájdeme v prílohe E.

4.3 Generovanie štruktúry

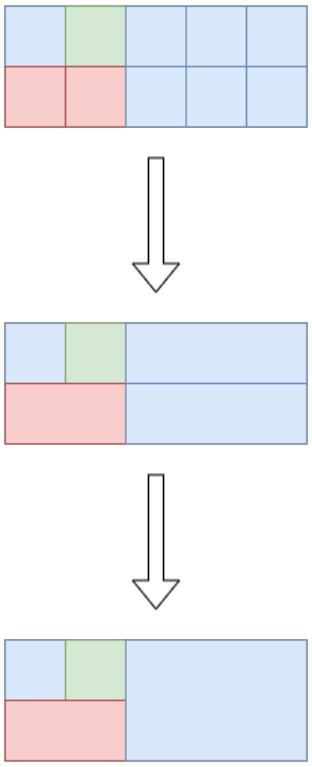
Generovanie štruktúry spočíva na preložení jednotlivých analyzovaných mriežok cez seba a následnom výbere najpočetnejších typov elementov na jednotlivých pozíciách. Zároveň je nutnou podmienkou, aby počet daného typu elementu na danej pozície bol väčší alebo rovný polovici z počtu analyzovaných mriežok, inak je dané miesto označené ako prázdne. Ak teda v dvoch mriežkach sa na pozícii [0,0] nachádza element typu obrázok a v tretej mriežke sa na pozícii [0,0] nachádza text, tak vo vygenerovanej mriežke bude na pozícii [0,0] element typu obrázok. V prípade rovnosti početností viacerých typov elementov bude prioritný rebríček nasledovný: video, obrázok, nadpis a text. Takýto druh rebríčku bol zvolený z dôvodu, aby vygenerované štruktúry stránok obsahovali viac mediálneho obsahu oproti tomu textovému. Počas priemernej návštevy priemernej webovej stránky majú užívatelia čas na prečítanie najviac 28% slov, ktoré sa na nej nachádzajú.[25] Pravdepodobnejšie je však, že prečítajú len 20% slov.[25] Obrázok 4.8 jednoducho znázorňuje, ako prekrývanie mriežok funguje. Pre jednoduchosť je v každom analyzovanom bloku iba jeden typ elementu (typ elementu je znázornený farbou). V skutočnosti však v každom analyzovanom bloku môže byť viac typov elementov, ako to bolo vysvetlené v kapitole 4.2.



Obrázok 4.8. Spájanie analyzovaných mriežok

Po vytvorení štruktúry z najpočetnejších typov elementov nasleduje ešte spájanie jednotlivých blokov. Pri pohľade na obrázok rozdelenia stránky do mriežky, je možné

vidieť, že obrázok v strede stránky je rozdelený do viacerých blokov. Tieto bloky potrebujú byť naspäť spojené do jedného veľkého bloku, ktorý bude predstavovať práve element typu obrázok s presnými rozmermi a pozíciou. Takéto spájanie musí byť vykonané horizontálne aj vertikálne. Následne vznikne pole objektov, ktoré jednoznačne popisuje pozície a typy dátových elementov tvoriace vygenerovanú štruktúru webovej stránky. Náhľad takéhoto poľa je možné vidieť v prílohe F tejto práce. Obrázok 4.9 zobrazuje postupne horizontálne a vertikálne spájanie analyzovaných blokov.



Obrázok 4.9. Horizontálne a vertikálne spájanie analyzovaných blokov

Je nutné poznamenať, že vertikálne spojenie blokov sa vykoná iba v prípade absolútnej rovnosti po sebe nasledujúcich blokov. Tak, ako je to zobrazené na obrázku vyššie, v prvom a druhom riadku sa nachádzajú absolútne rovnaké horizontálne spojené bloky, a teda môžu byť spojené aj vertikálne. Takéto obmedzenie je nutné z dôvodu, aby napríklad nevznikali nepravidelné tvary obrázkov.

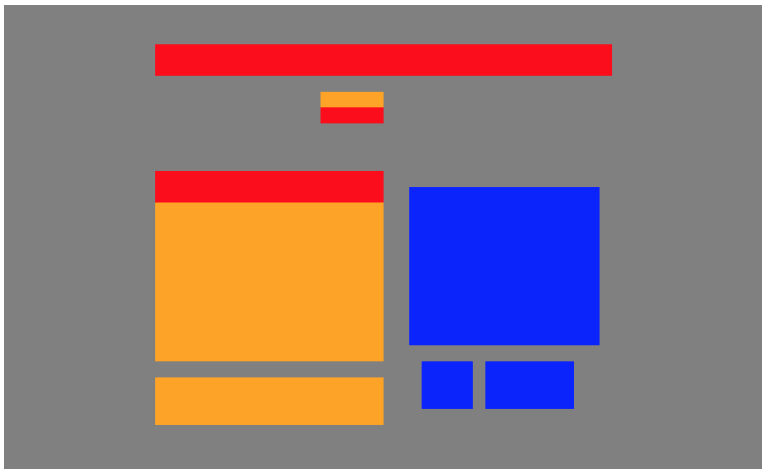
V prípade generovanie štruktúry budú navzájom cez seba preložené mriežky analyzovaných stránok, ktoré majú byť inšpiráciou a zároveň aj predvolená mriežka pre danú kategóriu stránky. Táto predvolená mriežka bude generovaná z najlepších 5 stránok v danej kategórii podľa rebríčka Alexa, ktorý hodnotí stránky podľa návštevnosti.

Obrázok 4.10 predstavuje vizualizáciu dát, ktoré sú obsiahnuté v poli hodnôt vytvorenom po generovaní štruktúry webovej stránky. Náhľad takého poľa je možné nájsť v prílohe F tejto práce. Na obrázku sa vyskytujú štyri rôzne farby, ktoré predstavujú nasledujúce typy elementov:

- nadpis - červená farba,
- text - oranžová farba,

- obrázok - modrá farba,
- prázdne miesto - sivá farba.

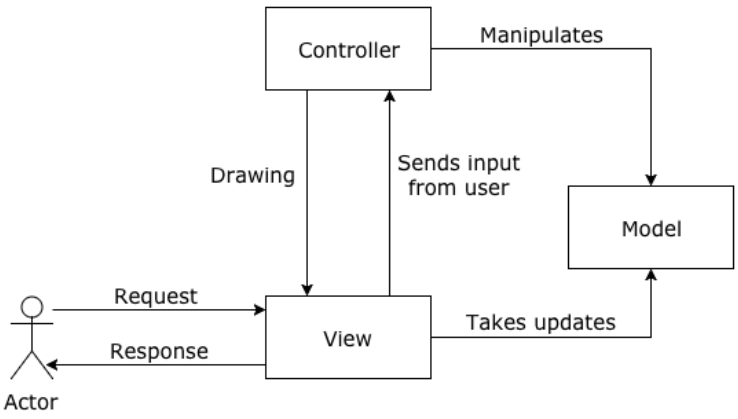
Na prvý pohľad sa môže zdať, že ukladanie prázdnych miest nie je potrebné, no opak je pravdou. Technické riešenia niektorých redakčných systémov môžu potrebovať explicitné pozície medzier medzi elementami a preto, aby neboli nútení tieto pozície prepočítavať sú obsiahnuté v tomto poli.



Obrázok 4.10. Vizualizácia vygenerovanej štruktúry stránky

4.4 Obslužná časť systému

Ako už bolo spomenuté táto časť je hlavnou časťou celej aplikácie. Pre túto časť systému bola zvolená Model View Controller (ďalej len MVC) architektúra. Táto architektúra pozostáva z nasledujúcich troch častí. Model - to je objekt, ktorý reprezentuje prácu s dátami, logiku a pravidlá aplikácie. Je to dynamická dátová štruktúra nezávislá na užívateľskom rozhraní. Obrazovka (z angl. View) - reprezentuje prezentáciu samotných dát aplikácie. Kontrolér (z angl. Controller) - je časť, ktorá reprezentuje definíciu interakcie medzi užívateľom a aplikáciou.



Obrázok 4.11. MVC architektúra

Táto architektúra oddeľuje obrazovky prezentované užívateľovi od modelu vytvorením komunikačného rozhrania medzi nimi. Aktuálny stav modelu prezentovaný na obrazovkách, ktoré zabezpečujú korektnosť prezentovaných dát. Obrazovky sú zmenené

v momente, kedy sa zmení aj model. Takýto prístup umožňuje pripojenie viacerých obrazoviek k rovnakému modelu. Samostatné uživateľské rozhranie a logika umožňujú jednoduchú údržbu a prehľadnosť kódu.

Do architektúry bola pridaná vrstva interných mikro-služieb. Tieto služby sú volané v jednotlivých metódach kontrolérov, ktoré sú zodpovedné za obsluhu požiadaviek na REST API, alebo prostredníctvom uživateľského rozhrania. Tým bola splnená podmienka REST API o vrstvenom modeli a zároveň bolo dosiahnuté oddelenie zodpovednosti za prácu s dátami. Takáto pridaná vrstva v aplikácii môže byť jednoducho testovaná pomocou jednotkových testov, ktorými sa môže zabezpečiť väčšie pokrytie aplikácie testami.

4.4.1 Kvalifikácia meraných elementov

Zdrojový kód webovej stránky obsahuje veľke množstvo nepotrebných údajov, ktoré musí vedieť nástroj odfiltrovať, aby bola štruktúra webovej stránky korektne analyzovaná. Nižšie budú dve veľmi časté situácie, s ktorými musí nástroj počítať.

Častou situáciou, ktorá môže zapríčiniť nesprávnu analýzu štruktúry webovej stránky je obaľovanie dátových elementov štrukturálnymi. Na obrázku nižšie môžeme vidieť, ako je dátový element obrázok obalený dvomi tagmi div. Nástroj musí takúto situáciu rozoznať a na konečne meranie veľkosti elementov označit práve obrázok a nie štrukturálne elementy div. Obalenými elementami môžu byť aj iné dátové elementy ako nadpis, video a pod.

```
<div>
    <div>
        <img src="zdroj_obrazku.png" alt="Obrazok">
    </div>
</div>
```

Obrázok 4.12. Obaleny dátový element

Ďalšia situácia, ktorá pri analýze štruktúry webovej stránky môže zapríčiniť značne nekorektné výsledky, je viac dátových elementov v štrukturálnom elemente, ktorý ich obaľuje. Takáto situácia musí byť riešená odmeraním veľkostí jednotlivých dátových elementov. Následne bude určený pomer miesta, aké zaberajú jednotlivé dátové elementy v štrukturálnom elemente. Ak niektorý dátový element bude zaberať viac ako polovicu miesta v štrukturálnom elemente, bude sa brať v úvahu tento dátový elemente. Toto riešenie bolo zvolené z dôvodu komplexnosti výpočtu veľkosti jednotlivých dátových elementov. Najhoršie je to v prípade textového elementu, ten totiž môže vystupovať voľne v štrukturálnom elemente, ako tomu je aj na obrázku nižšie.

```
<div>
    <div>
        <img src="zdroj_obrazku.png" alt="Obrazok">

        Nejaký text, ktorý môže mať rôznu veľkosť.
    </div>
</div>
```

Obrázok 4.13. Viac dátových elementov v jednom štrukturálnom elemente

4.4.2 Odmeranie kvalifikovaných elementov

Po kvalifikácii elementov je potrebné tieto elementy odmerať. Odmeranie spočíva v určení pozície týchto elementov voči ľavému hornému rohu webovej stránky a samot-

nej veľkosti elementu. Po odmeraní všetkých kvalifikovaných elementov je výsledkom pole dátových elementov s ich pozíciami a veľkosťami, ktoré tvoria štruktúru webovej stránky. Takéto pole môžeme nájsť v prílohe D tejto práce. Korektné odmeranie elementov sa musí vykonať na zobrazenej stránky, preto je nutné, aby bola stránka lokálne zobrazená. Na zobrazenej stránke následne vykoná zdrojový kód napísaný v javascripte meranie a odošle ich pomocou AJAX požiadavku na uloženie.

4.5 Výpočtová časť systému

Systém obsluhuje viac nezávislých requestov naraz, a preto bolo potrebné zabezpečiť plynulú rýchlu odozvu systému. Z toho dôvodu bola náročná výpočtová časť oddelená od hlavnej časti systému. Výber technológie pre túto časť mal svoje jasné kritériá a nimi boli:

- efektívna práca s vláknami,
- výborné výkonové výsledky v práci s poliami,
- jednoduchá práca a nasadenie na serveri,
- moderný jazyk.

Na základe vyššie spomenutých kritérií sa do užšieho výberu voľby technológie použitej vo výpočtovej časti dostali nasledujúce:

- MATLAB,
- Go,
- Python,
- Java.

4.5.1 MATLAB

Táto technológia sa dostala do užšieho výberu práve z dôvodu jej vzniku. Bola vyvinutá pre analýzy dát, strojové učenie a rôzne iné náročné matematické operácie.[26] Táto technológia umožňuje náročné výpočtové operácie riešiť jednoduchým zápisom v jej programovacom jazyku založenom na veľkom použití matic. Keďže môžeme povedať, že matica sa skladá z poľa polí, tak môžeme konštatovať, že práca s poliami nebude problém.

4.5.1.1 Práca s vláknami

MATLAB sám o sebe je implementovaný tak, že volanie jednotlivých matematických funkcií je interne riešené viac-vláknovo. Zjednodušene MATLAB umožňuje jednoduchý zápis matematického problému, ktorý následne sám interne vykonáva na viacerých vláknach.[27]

4.5.1.2 Práca s poliami

Táto technológia bola vytvorená za účelom hlavne riešenia matematických problémov, a teda toto kritérium bolo aj kritériom pri vytváraní samotnej technológie.

4.5.1.3 Práca a nasadenie na serveri

Pre nasadenie tejto technológie je potrebná konzultácia s firmou MathWorks, ktorá danú technológiu poskytuje.[28] Pre vzdelávacie účely by pravdepodobne umožnili jej bezplatné použitie, ale keďže by sa tento systém mal používať aj komerčne určite by sa nenašlo bezplatné riešenie.

4.5.1.4 Modernosť jazyka

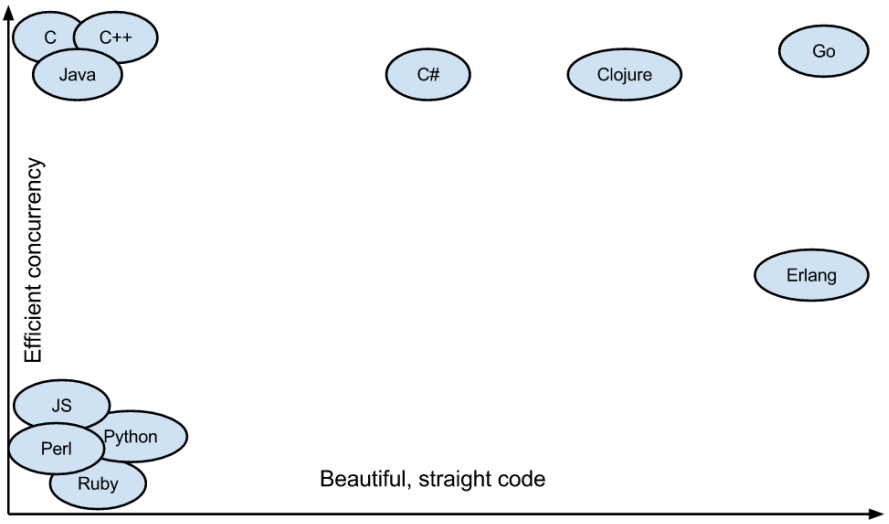
Hodnotiť modernosť tohto jazyka sa dá len veľmi ťažko. Jazyk sa sám o seba skladá z matematických výrazov a tie sa nedajú posúdiť ako moderné alebo zastaralé.

4.5.2 Go

Táto technológia sa dostala do užšieho výberu z dôvodu, že je najmladšou z uvedených technológií. Verzia 1.0 bola zverejnená v marci 2012 firmou Google, ktorá stojí za jej vytvorením.[29] Technológia vznikla v dobe, keď už boli vo svete rozšírené viac-vláknové prostredia, zatiaľ čo všetky ostatné spomínané technológie vznikali v ére jednovláknových prostrediach, a teda neboli nikdy priamo stavané na viac-vláknové použitie.[30]

4.5.2.1 Práca s vláknami

Technológia bola vyvinutá s vedomím súbežného vykonávania procesov.[30] Poskytuje tzv. "Goroutines", ktoré umožňujú viacero súbežne bežiacich procesov pri nízkej spotrebe zdrojov ako je pamäť.[30] Porovnanie čitateľnosti a prehľadnosti kódu pri práci s vláknami je znázornená na obrázku nižšie.



Obrázok 4.14. Porovnanie programovacích jazykov v ich čitateľnosti a práci s vláknami[30]

4.5.2.2 Práca s poliami

Go disponuje štandardnou prácou s poliami, ako aj ostatné spomenuté technológie okrem MATLABu. Avšak, na rozdiel od ostatných technológií, poskytuje možnosti práce s pointrami, čo by mohlo byť neskôr výhodou, preto môžeme povedať, že po Matlabe je v tomto kritériu na druhej pozícii.

4.5.2.3 Práca a nasadenie na serveri

Nasadenie tejto technológie na server nie je komplikované a neprináša so sebou žiadne potencionálne problémy. Autor tejto práce má s touto technológiou minimálne skúsenosti, preto práca s ňou nesie so sebou povinnosť si túto technológiu najprv naštudovať.

4.5.2.4 Modernosť jazyka

Ako už bolo spomenuté skôr, verzia 1.0 tejto technológie bola zverejnená v roku 2012. Jedná sa teda o mladú modernú technológiu.

4.5.3 Python

Je to jedna z najjednoduchších technológií a zároveň aj najvýkonnejších zo skupiny interpretovaných jazykov. To bol hlavný dôvod, prečo sa dostala do užšieho výberu. Výkon je totiž veľmi dôležitý hlavne, ak sa jedná o matematické výpočty.

4.5.3.1 Práca s vláknami

Ak hovoríme o referenčnej verzii Pythonu, tak je to nemožné. Sám interpreter totiž nie je thread-safe.[31] Toto kritérium by sa teda dalo splniť iba pomocou iných druhov interpreterov. To by znamenalo ďalšie inštalácie na server a tie by už mohli so sebou priniesť potencionálne problémy. Tento jazyk, sa ako jediný zo zástupcov interpretovaných jazykov dostal do užšieho výberu a tie ponúkajú v porovnaní s kompilovanými jazykmi nižšie výkony.

4.5.3.2 Práca s poliami

V Pythone sú polia reprezentované formou listov. Avšak môžeme povedať, že nijak zvlášť nevyčnieva v práci s poliami od ostatných technológií.

4.5.3.3 Práca a nasadenie na serveri

Práca s touto technológiou je aj pre úplných začiatočníkov veľmi jednoduchá. Je to technológia najrýchlejšia na naučenie.[32] Inštalácia tejto technológie nenesie so sebou žiadne potencionálne problémy, a preto je toto kritérium viac než splnené.

4.5.3.4 Modernosť jazyka

Python bol vytvorený v roku 1991.[32] Je to teda 27 rokov stará technológia. Stále sa síce teší svojej obľube,[33] no ako o modernom jazyku sa o ňom hovoriť nedá.

4.5.4 Java

Do užšieho výberu sa dostala najmä vďaka skúsenostiam, ktoré s ňou autor tejto práce má. Práca s ňou teda bude pre autora veľmi jednoduchá a vývoj systému teda nebude podliehať štúdiu novej technológie.

4.5.4.1 Práca s vláknami

Táto technológia poskytuje štandardnú prácu s vláknami, avšak vytvorenie nového vlákna nie je pamäťovo efektívne.[30]

4.5.4.2 Práca s poliami

Java disponuje štandardnou prácou s poliami, ako aj ostatné spomenuté technológie okrem MATLABu.

4.5.4.3 Práca a nasadenie na serveri

Z vyššie spomínaných technológií má práve s touto technológiou autor tejto práce najviac skúseností. Nasadenie na server taktiež nie je problémom, keďže technológia je dominantnou technológiou v oblasti enterprise systémov.[27]

4.5.4.4 Modernosť jazyka

Bola vyvinutá firmou Sun Microsystems v roku 1995[34], a teda nedá sa povedať, že je to mladá technológia, avšak jej popularita je stále vysoká.[33] V tomto kritériu teda Java pôsobí neutrálne.

4.5.5 Zhrnutie

Hlavným dôvodom, pre oddelenie výpočtovej časti systému, bolo výkonnostné zaťaženie, a preto tou správnou voľbou bola technológia Go, ktorá najviac spĺňala kritérium práce s vláknami. Táto voľba niesla so sebou nutnosť ďalšieho štúdia technológie. Tabuľka nižšie zobrazuje bodové hodnotenie splnenia kritérií pre jednotlivé technológie. Bodový interval bol <0,3>, pričom viac bodov znamená lepšie splnenie kritéria.

	Vlákna	Polia	Nasadenie	Modernosť	**Spolu**
Matlab	2	3	0	1	6
Go	**3**	**2**	**3**	**3**	**11**
Python	1	2	3	2	8
Java	2	2	3	2	9

Tabuľka 4.1. Bodové hodnotenie kritérií pre výber technológie

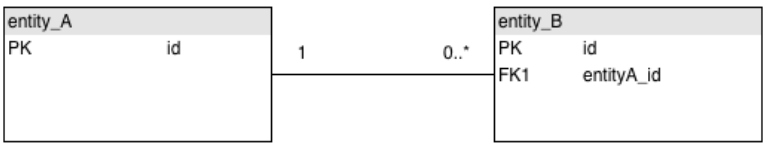
4.6 Dátový model

Pre popis dátového modelu bude použitý ER diagram (z angl. Entity Relationship Diagram). Takýto diagram popisuje jednotlivé entity a vzťahy medzi nimi. Entita je "vec", ktorá sa dá jednoznačne identifikovať.[35] Výbornými príkladmi sú človek, udalosť alebo spoločnosť. Vzťah je určité prepojenie medzi jednotlivými entitami, ktoré z niečoho vyplývajú. Napríklad, vzťah medzi dvoma entitami človek je otec-syn. Do databázy nie je možné zaznamenať úplný opis vzťahu medzi dvoma entitami.[35] Je nemožné (a možno zbytočné) zaznamenávať všetky potenciálne dostupné informácie o entitách a ich vzťahoch.[35] Informácie o entite, alebo vzťahu sa získajú pozorovaním, alebo meraním a sú vyjadrené pomocou súboru párov atribút - hodnota.[35] Pre uvedenie príkladov použijeme znova entitu človek, ktorá može byť popísaná atribútom vek, ktorého hodnota môže byť 40.

Všetky entity v tomto systéme budú na databázovej úrovni ukladané v samostatných tabuľkách a budú obsahovať atribúty created_at (dátum a čas vytvorenia), updated_at (dátum a čas poslednej úpravy) a unikátny automatické identifikátor id. Tento identifikátor bude vždy unikátny v rámci jednej entity, to znamená, že dve entity rovnakého typu nikdy nebudú mať rovnaké id. Väzby medzi entitami budú realizované v databáze pomocou cudzích kľúčov (z angl. Foreign keys). Takéto kľúče realizujú vzťah medzi entitami na úrovni databázy.

4.6.1 Väzba 1:n

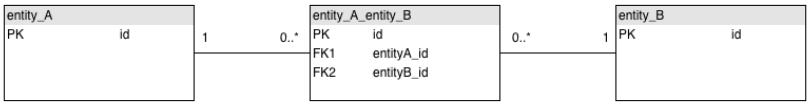
Tento druh väzby sa používa ak jedna entita typu A má väzbu k "n" entitám typu B, pričom n môže nadobúdať kladných celočíselných hodnôt vrátane nuly. Entita B obsahuje atribút cudzí kľúč, ktorého hodnota sa bude rovnať hodnote identifikátora entity typu A.



Obrázok 4.15. Väzba 1:n

4.6.2 Väzba m:n

Tento druh väzby sa používa ak jedna entita typu A má väzbu k "n" entitám typu B a zároveň entita typu B môže mať väzbu k "m" entitám typu A, pričom m a n môžu nadobúdať kladných celočíselných hodnôt vrátane nuly. Takáto väzba sa na databázovej úrovni realizuje pridaním väzbovej tabuľky, v ktorej sa budú nachádzať cudzie kľúče oboch entít, ktoré majú medzi sebou vzťah.



Obrázok 4.16. Väzba m:n

ER diagram dátového modelu aplikácie je možné nájsť v prílohe C.

4.6.3 Užívatelia

Táto entita je v databáze reprezentovaná tabuľkou "users". Entita predstavuje užívateľov, ktorí aplikáciu používajú. Užívateľom môže byť napríklad redakčný systém, ktorý využíva tento nástroj. Užívateľ obsahuje tieto atribúty:

- name - názov pod ktorým chce užívateľ vystupovať,
- email - email, ktorým sa bude užívateľ prihlasovať,
- email_verified_at - dátum a čas, kedy bol email verifikovaný,
- password - heslo, ktorým sa bude užívateľ prihlasovať,
- remember_token - token, ktorý bude slúžiť k udržiavaniu prihláseného stavu.

4.6.4 Mriežky

Táto entita je v databáze reprezentovaná tabuľkou "grids". Táto entita bude obsahovať údaje o analyzovaných a generovaných štruktúrach webových stránok. Entita mriežka obsahuje nasledujúce atribúty:

- site_id - cudzí kľúč k entite stránka, ktorý definuje, že mriežka patrí k danej stránke,
- category_id - cudzí kľúč k entite kategória ktorý definuje predvolenú mriežku pre danú kategóriu,
- elements - tento atribút obsahuje textový reťazec vo formáte JSON, v ktorom sú údaje o pozíciách elementov v čase analýzy stránky,
- merged_same_elements - atribút obsahuje textový reťazec vo formáte JSON, ktorý obsahuje pozície a typy elementov na určených pozíciach,
- matrix_prob - atribút obsahuje textový reťazec vo formáte JSON, v ktorom sú údaje o početnosti výskytu elementu daného typu na určenej pozícii,
- generated - atribút, ktorý obsahuje informáciu o tom, či mriežka bola vygenerovaná,
- invalid - atribút, ktorý obsahuje informáciu o tom, či počas spracovávania mriežky došlo k nejakej chybe.

4.6.5 Kategórie

Táto entita je v databáze reprezentovaná tabuľkou “categories”. Táto entita bude v aplikácii enumeračnou a bude predstavovať kategórie stránok, ktoré môžu byť v aplikácii analyzované a vygenerované. Entita obsahuje nasledujúci atribút:

- name - názov kategórie.

4.6.6 Stránky

Táto entita je v databáze reprezentovaná tabuľkou “sites”. Entita predstavuje samotné analyzované, alebo generované webové stránky. Obsahuje nasledujúce atribúty:

- category_id - cudzí kľúč k entite kategória ktorý definuje predvolenú mriežku pre danú kategóriu,
- o_auth_access_token_id - cudzí kľúč k systémovej entite token, ktorý sa používa pri autorizovaní požiadavkou na REST API,
- client_uid_id - unikátny identifikátor, pod ktorým je evidovaná stránka v redakčnom systéme, pre ktorý je stránka analyzovaná, alebo generovaná,
- url - URL adresa stránky.

4.6.7 Ostatné entity

Ostatné entity, ktoré sa v systéme vyskytujú slúžia pre jeho korektnú funkčnosť a zodpovedajú napríklad za korektnú autorizáciu požiadaviek na REST API. Jedná sa o pomocné entity a keďže nie sú predmetom tejto práce nebude im venovaný priestor v tejto práci.

4.7 Užívateľské rozhranie

Užívateľské rozhranie tejto aplikácie slúži predovšetkým na prezentáciu dokumentácie aplikácie, preto bola zvolená jednoduchá designová šablóna v aplikačnom rámci Bootstrap[1]. Tento aplikačný rámec je voľne dostupný a umožňuje rýchly vývoj užívateľského rozhrania, pričom je zameraný aj na to, aby bolo rozhranie použiteľné na mobilných zariadeniach. Užívateľské rozhranie tejto aplikácie slúži predovšetkým na popis práce s REST API. Náhľady uživateľského rozhrania nájdeme v prílohe H tejto práce.

[1] `https://getbootstrap.com/`

Kapitola 5
Implementácia

Po dostatočnej analýze a návrhu by sa úsek implementácie mal realizovať ľahko a rýchlo.[21] Táto kapitola opisuje, ako je nástroj implementovaný. Súčasťou tejto kapitoly je aj podkapitola, ktorá sa bude venovať nasadeniu systému.

5.1 Technológie

Systém pre generovanie štruktúr webových stránok bol vytvorený pomocou viacerých technológií, ktoré zabezpečovali rôzne fázy a časti projektu. V tejto kapitole sú postupne predstavené najdôležitejšie technológie, ktoré umožnili úspešný vývoj tohto softvéru.

5.1.1 Git

Keďže vývoj tohto softvéru sa niesol v duchu agilného vývoja, bolo potrebné uchovávať jednotlivé stavy riešenia s možnosťou vrátiť sa späť k všetkým verziám. Ako verzovací nástroj bol teda použitý Git[1]. Repozitár, ktorý uchováva všetky verzie vyvíjaného softvéru, sa nachádza v aplikácii Bitbucket[2].

5.1.2 PostgreSQL

Aplikácia využíva databázu na ukladanie všetkých potrebných údajov. Pre databázu bola zvolena technológia PostgreSQL. Pred najbežnejšie využívanou databázou MySQL má PostgreSQL niekoľko výhod, ktoré rozhodli o tejto voľbe. Prvou, veľmi dôležitou výhodou je, ukladanie JSONu v stĺpci typu jsonb. Takýto typ uloží vstup vo formáte string v jeho binárnej forme. Takéto uloženie JSONu je oveľa efektívnejšie, ako jeho ukladanie vo formáte textového reťazca. Dokumentácia PostgreSQL odporúča väčšine aplikácií ukladať JSON práve pomocou typu stĺpca jsonb, keďže to ponúka značne lepšie výkonové výsledky.[36] PostgreSQL ďalej podporuje na rozdiel od MySQL aj iné pokročilé funkcie, ktoré môžu byť neskôr využité. Za zmienku stoja napríklad materializované náhľady (z angl. Materialized view), ktoré umožňujú vytvorenie "dočasných" tabuliek pre veľké a časté príkazy SELECT v databáze.

5.1.3 Laravel

Aplikačný rámec Laravel[3] sa stal základom pre obslúžnu časť aplikácie. Laravel je v súčasnosti najobľúbenejším PHP[4] aplikačným rámcom.[37] Laravel poskytuje kostru, do ktorej vývojár postupne dopĺňa funkcionalitu. Kostra obsahuje veľké množstvo funkcionality, ktorú vývojár nemusí znova vyvíjať. Takýmito funkciami, ktoré boli použité aj v tejto práci, sú napríklad: prihlásenie, registrácia, obnova hesla, ORM a MVC architektúra. Veľkou výhodou stavby aplikácie, nad overeným aplikačným rámcom, je veľká

[1] https://git-scm.com/
[2] https://bitbucket.org
[3] https://laravel.com
[4] http://php.net

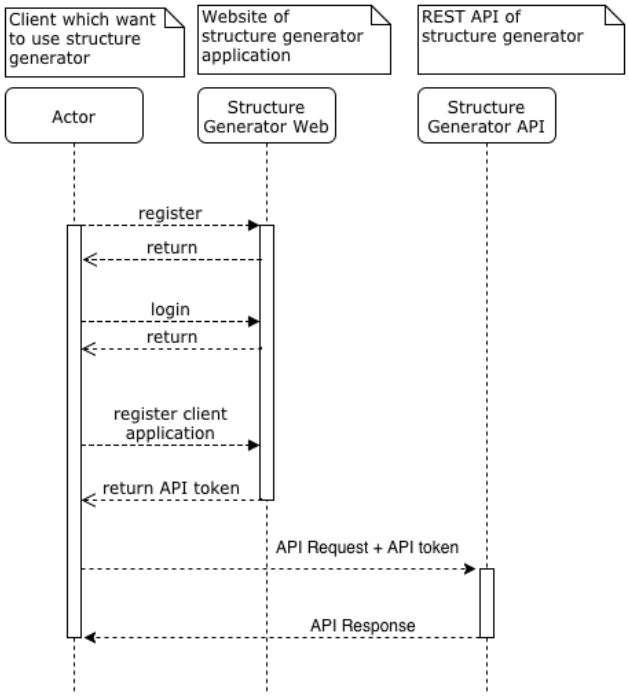
miera zabezpečenia, ktorú aplikačný rámec poskytuje. Vývojár sa teda nemusí zapodievať zabezpečením aplikácie. Laravel poskytuje niekoľko zabezpečení, za zmienku stoja tie najdoležitejšie:

- hashovanie uložených hesiel,
- ochrana proti CSRF útokom,
- SQL injekcia a ochrana proti XSS.

Tento aplikačný rámec využíva nástroj Composer[1] na správu svojich závislostí. Takýto nástroj umožňuje jednoduché pridávanie a odoberanie závislosti do projektu.

5.1.3.1 Autorizácia v REST API

Keďže bola komunikácia s aplikáciou navrhnutá pomocou REST API je potrebné takúto komunikáciu aj zabezpečiť. Laravel ponúka kompletné riešenie serveru OAuth 2, ktorý môže byť použitý na zabezpečenie API. V tomto prípade bola zvolená metóda osobných tokenov, ktorými sa budú požiadavky na API podpisovať. Sekvenčný diagram nižšie znázorňuje celý proces vytvorenia tokenu a jeho následné použitie.



Obrázok 5.1. Registrácia a použite tokenu pre REST API

Registrovaný token musí byť súčasťou každej požiadavky na REST API. Ak token nebude súčasťou požiadavky, alebo bude neplatný, požiadavka nebude realizovaná. Obrázok nižšie znázorňuje zápis tokenu do HTTP hlavičky Authorization.

[1] `https://getcomposer.org/`

```
POST /api/site/analyze HTTP/1.1
Host: sg.lukasfigura.sk
Content-Type: application/json
Authorization: Bearer <token>
Cache-Control: no-cache
```

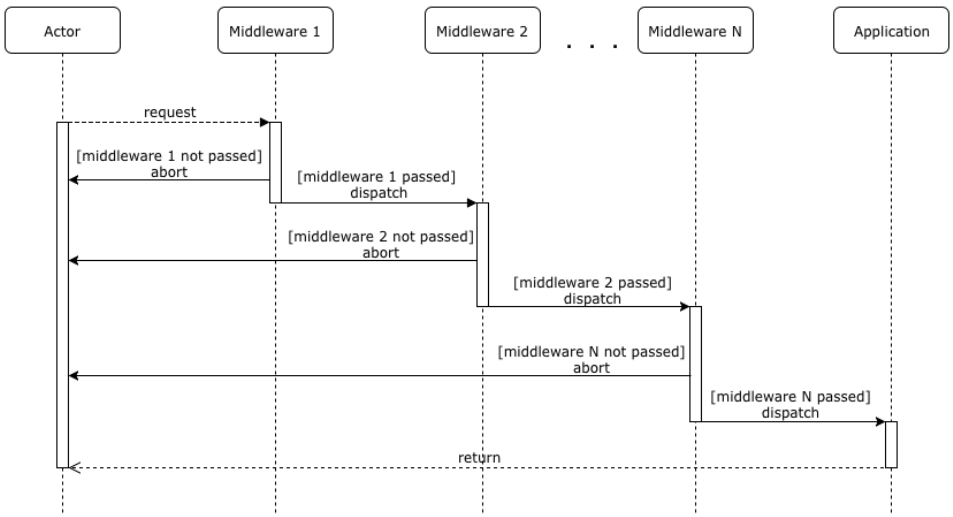
Obrázok 5.2. Použitie tokenu v požiadavke na REST API

5.1.3.2 Servisná vrstva

Servisná vrstva obsahuje interné mikro služby, ktoré boli popísané v podkapitole 4.4. Triedy, ktoré tvoria tieto mikro služby, sa skladajú z metód, ktoré majú vždy jeden účel a sú navzájom nezávislé. Tieto metódy vykonávajú špecifické operácie s dátovým modelom aplikácie. Takto postavené triedy umožňujú výbornú možnosť testovania jednotlivých funkcií systému. Zároveň je zavedením takejto vrstvy zvýšená čitateľnosť kódu.

5.1.3.3 Filter

Filter je mechanizmus, ktorým je možné filtrovať požiadavky, ktoré budú aplikáciou obslúžené. V Laraveli sa filter nazýva middleware.

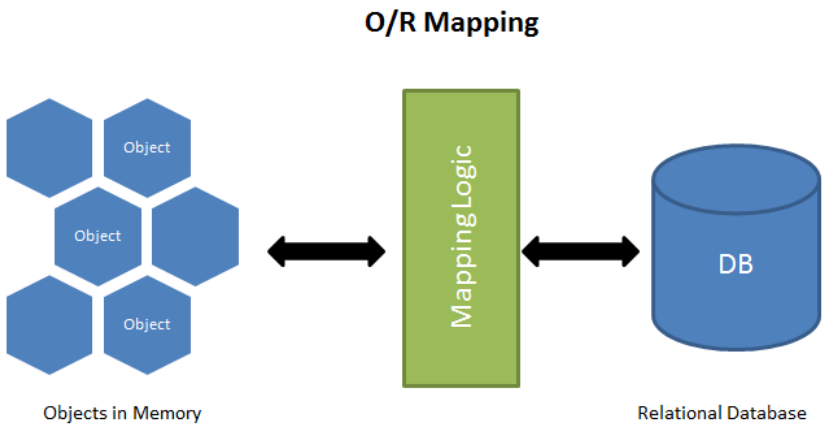


Obrázok 5.3. Filtrovanie požiadaviek v aplikácii

V prípade, že požiadavka neprejde úspešne všetkými preddefinovanými filtrami bude odmietnutá. Takéto filtre používa Laravel interne napríklad pri zabezpečení proti CSRF útokom. Pomocou tohto mechanizmu bola v aplikácii vytvorená časť, ktorá môže byť spustená iba samotnou aplikáciou. Táto časť je bližšie vysvetlená v podkapitole 5.2.

5.1.3.4 ORM

Objektovo-relačné mapovanie (z angl. Object-relational mapping) je technika, ktorá umožňuje dotazovanie a manipuláciu s údajmi z databázy, pomocou objektovo orientovanej formy. Aplikačný rámec Laravel obsahuje ORM, ktoré sa nazýva Eloquent. Toto ORM poskytuje implementáciu návrhového vzoru Aktívny záznam (z angl. Active Record). Jednotlivé inštancie modelov v ORM reprezentujú odpovedajúci záznam v databáze. Návrhový vzor ďalej zabezpečuje, že pri akejkoľvek operácii s inštanciou modelu v ORM sa daná operácia vykoná aj v databáze.



Obrázok 5.4. Objektovo-relačné mapovanie zdroj:[1]

5.1.3.5 Migrácie

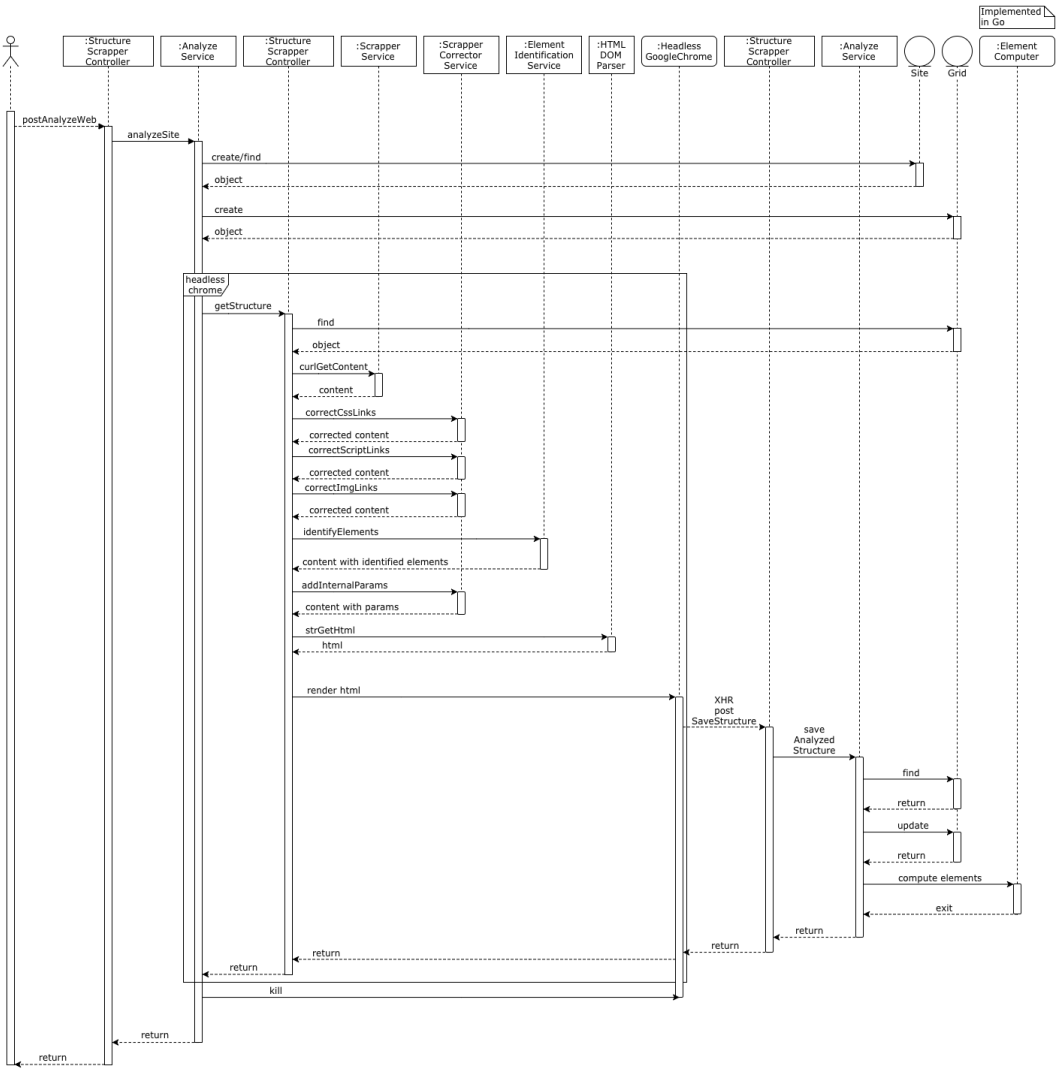
Pomocou migrácií sa v aplikáciách využívajúcich aplikačný rámec Laravel spravuje štruktúra databázy. Každá zmena štruktúry databázy je zapísaná v jednej migrácii. Tento prístup umožňuje jednoduché verzovanie štruktúry databázy. Migrácie zľahka odstraňujú potrebu verzovacieho systému akým je Git, avšak s použitím Gitu umožňuje jednoduché zdieľanie verzovaných zmien štruktúry databázy pre celý vývojársky tím.

```
Migration table not found.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2016_06_01_000001_create_oauth_auth_codes_table
Migrated:  2016_06_01_000001_create_oauth_auth_codes_table
Migrating: 2016_06_01_000002_create_oauth_access_tokens_table
Migrated:  2016_06_01_000002_create_oauth_access_tokens_table
Migrating: 2016_06_01_000003_create_oauth_refresh_tokens_table
Migrated:  2016_06_01_000003_create_oauth_refresh_tokens_table
Migrating: 2016_06_01_000004_create_oauth_clients_table
Migrated:  2016_06_01_000004_create_oauth_clients_table
Migrating: 2016_06_01_000005_create_oauth_personal_access_clients_table
Migrated:  2016_06_01_000005_create_oauth_personal_access_clients_table
Migrating: 2018_10_14_142650_create_table_sites
Migrated:  2018_10_14_142650_create_table_sites
Migrating: 2018_10_14_145054_create_table_grids
Migrated:  2018_10_14_145054_create_table_grids
Migrating: 2018_10_14_145245_create_table_categories
Migrated:  2018_10_14_145245_create_table_categories
Migrating: 2018_10_14_161728_add_foreigns
Migrated:  2018_10_14_161728_add_foreigns
Database seeding completed successfully.
```

Obrázok 5.5. Migrácie spustené pre vytvorenie databázy

[1] https://medium.com/@mithunsasidharan/should-i-or-should-i-not-use-orm-4c3742a639ce

5.2 Analýza stránky



Obrázok 5.6. Sekvenčný diagram analýzy stránky

Sekvenčný diagram (obrázok 5.6) zobrazuje, ako prebieha analýza stránky v nástroji. Analýza stránky začína požiadavkou na REST API o analýzu webovej stránky. Následne obslúžna časť systému vytvorí príslušné záznamy v databáze, s ktorými sa bude pracovať. Dôležitým objektom je Mriežka, ktorá bude výsledkom celej analýzy. Časť procesu sa vykonáva v tzv. bezhlavičkovom prehliadači (z angl. headless browser). Technológia bezhlavičkového prehliadača je predstavená nižšie v tejto kapitole. Takéto riešenie je nutné z toho dôvodu, že analýza štruktúry musí prebehnúť na zobrazenej webovej stránke, avšak nie je možné, aby si nástroj v cloude otvoril plnohodnotný prehliadač. Takto vykonávanú časť ohraničuje rámček "headless chrome". Bezhlavičkový prehliadač je spustený vykonaním nasledujúceho príkazu v príkazovom riadku.

```
google-chrome --headless
```

Jedným z parametrov tohto príkazu je aj URL stránky, ktorá má byť načítaná. Nižšie je uvedený vzorový príkaz, ktorý spúšťa obslúžna časť systému.

```
google-chrome --headless --disable-gpu --hide-scrollbars \
```

```
--window-size=1920,1080 --screenshot=grid10.png \
http://sg.lukasfigura.sk/10
```

Vlajka `--hide-scrollbars` zneviditeľní posuvník (z angl. scrollbar), aby sa nezobrazoval na prípadnej snímke zobrazenej stránky. Ďalšou použitou vlajkou je `--window-size` umožňuje nastavenie rozmeru, v akom má byť stránka zobrazená. Následne je možnosť uvedenia vlajky `--screenshot`, ktorou vieme ovplyvniť názov súboru, v ktorom bude uložená snímka obrazovky. Práve touto vlajkou je splnená funkčná požiadavka o náhľade analyzovanej stránky v podkapitole 3.3.2. Posledným parametrom je práve URL stránky, ktorá má byť zobrazená. V tomto prípade to teda je interná URL, ktorú je možné načítať iba lokálne. Obsluha tejto URL spustí interný proces analýzy stránky.

Ako už bolo spomenuté v kapitole 6.2 samotná analýza sa skladá z viacerých krokov. Stiahnutie zdrojového kódu stránky sa uskutočňuje volaním metódy `curlGetContent` v mikroslužbe `ScrapperService`. Následne sa vykoná úprava zdrojového kódu, aby bolo možné stránku spustiť mimo jej pôvodného serveru. O tieto úpravy sa starajú metódy `correctCssLinks`, `correctScriptLinks` a `correctImgLinks` volané v mikroslužbe `ScrapperCorrectorService`. Tieto metódy postupne upravia odkazy na kaskádové štýly, javascripty a obrázky, ktoré sa nachádzajú v zdrojovom kóde stránky. Nasleduje kvalifikácia meraných elementov, ktorú zabezpečuje metóda `identifyElements` v mikroslužbe `ElementIdentificationService`. Postup kvalifikácie elementov bol popísaný v kapitole 4.4.1. Poslednou prácou so samotným zdrojovým kódom je pridanie parametrov, ktoré vyžaduje javascript vykonávajúci meranie elementov a prevedenie textového reťazcu na HTML kód, ktorý je zobrazený v bezhlavičkovom prehliadači. V ňom javascript odmeria kvalifikované elementy a odošle namerané hodnoty pomocou XHR požiadavku. Po uložení nameraných hodnôt je iniciovaná analýza štruktúry stránky vo výpočtovej časti nástroja napísanej v programovacom jazyku Go. Skompilovaná verzia tohto programu je spustená v príkazovom riadku s dvoma parametrami. Prvým parametrom je ID mriežky, ktorá má byť analyzovaná a druhým nepovinným parametrom je váha s akou majú byť určené početnosti elementov. Váhový parameter vynásobuje početnosť typu elementu v analyzovanom bloku. Takto sa dá docieliť vyššia dôležitosť mriežky pri následnom spájaní mriežok počas generovania. Nižšie je uvedený vzorový príkaz, ktorý spustí analýzu štruktúry stránky.

```
./element_computer 14 1
```

Výpočtová časť systému zabezpečí analýzu štruktúry stránky,ktorá je špecifikovaná v kapitole 4.2, a uloženie výsledkov do databázy. Po vykonaný vzorového príkazu vyššie je ukončená XHR požiadavka, ktorá bola inicializovaná javascriptom. Tým sa ukončí aj načítanie webovej stránky a teda aj samotné spustenie bezhlavičkového prehliadača. Aby bezhlavičkový prehliadač neostával v operačnej pamäti je potrebné ešte ukončenie procesu v operačnom systéme. Potom je celý proces analýzy stránky ukončený a užívateľovi je vrátené ID mriežky, ktorá bola vytvorená pre analyzovanú stránku.

5.2.1 Bezhlavičkový prehliadač

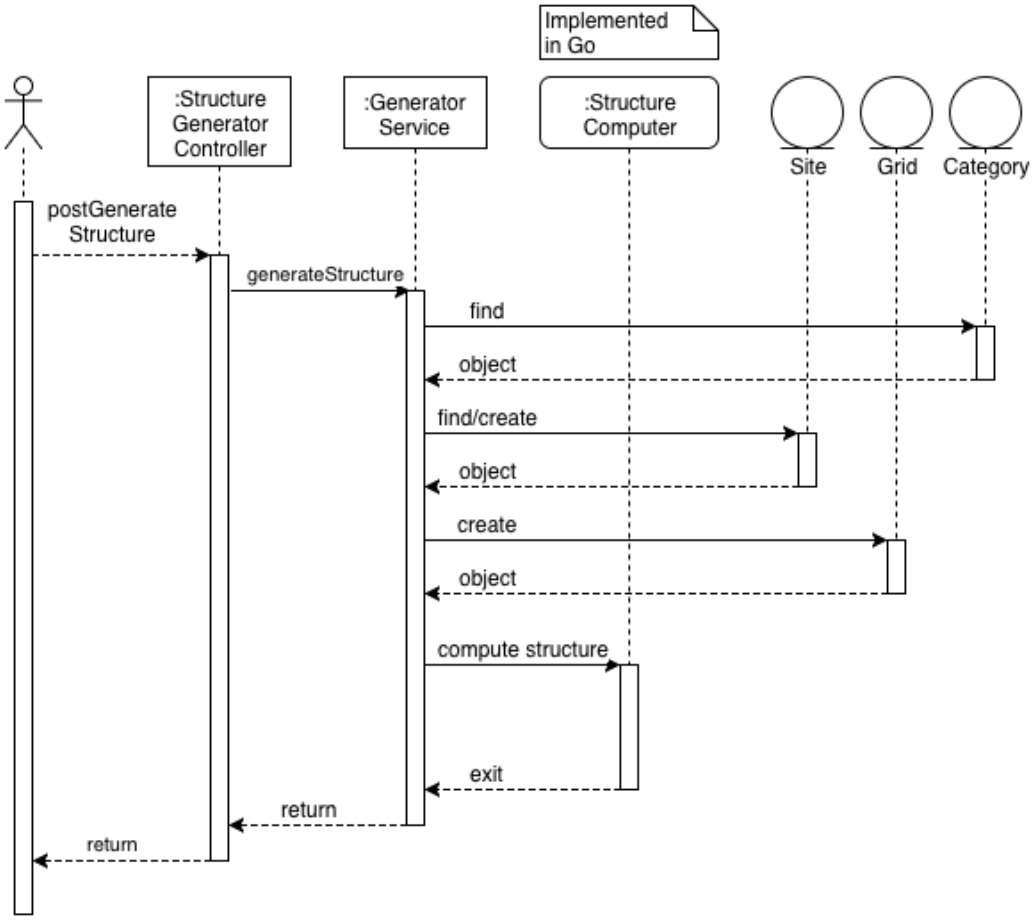
Bezhlavičkový prehliadač sa môže zdať ako zvláštny termín, ale je to jednoducho názov pre prehliadač, alebo simuláciu prehliadača bez rozpoznateľného grafického rozhrania.[38] Bezhlavičkové prehliadače sú často používané na testovanie:

- web stránok a webových aplikácií,
- javascriptových knižníc,
- javascriptové simulácie a interakcie,

- pomocou na pozadí bežiacich automatických UI testov.[38]

Autor tejto práce zvolil bezhlavičkový prehliadač Headless Chrome. Tento bezhlavičkový prehliadač od iných umožňuje vytvorenie snímky načítanej stránky.

5.3 Generovanie štruktúry



Obrázok 5.7. Sekvenčný diagram generovania štruktúry stránky

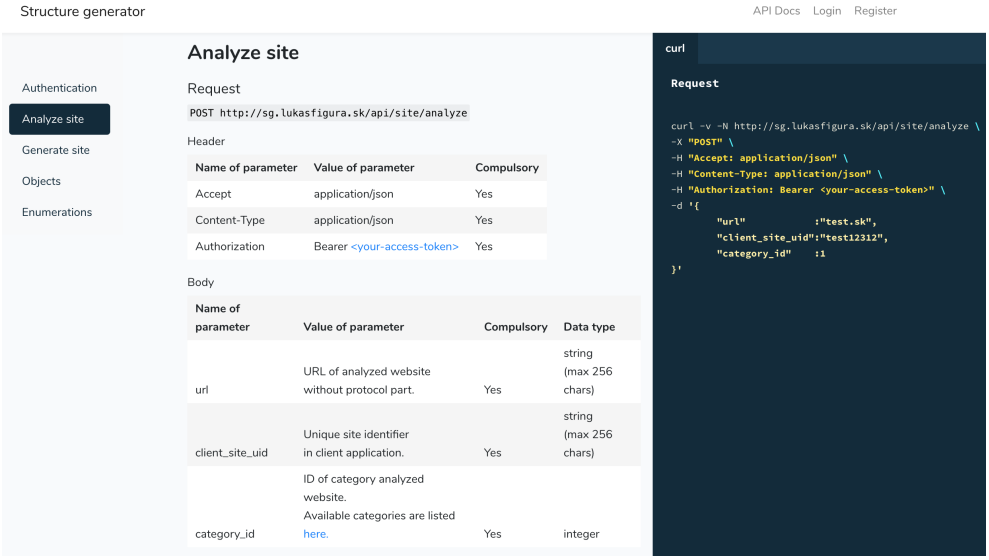
Vyššie zobrazený sekvenčný diagram znázorňuje priebeh generovania štruktúry stránky v obslúžnej časti systému. Oproti analýze je táto funkcia značne jednoduchšia. Pred generovaním štruktúry je vytvorená mriežka, ktorá bude predstavovať konečnú vygenerovanú štruktúru stránky. Samotné generovanie štruktúry stránky, ako bolo popísané v kapitole 4.3, je vykonané vo výpočtovej časti systému, ktorá je spustená vzorovým príkazom uvedeným nižšie.

```
./structure_computer 10 34 56
```

Prvým parametrom programu je ID mriežky, v ktorej má byť vygenerovaná štruktúra uložená. Ďalej nasledujú IDčka mriežok, ktoré majú byť použité pri generovaní štruktúry. Počet mriežok nie je obmedzený, no vyšší počet mriežok znamená dlhší proces generovania. Po skončení generovania je užívateľovi vrátené ID mriežky, ktorá obsahuje vygenerovanú štruktúru.

5.4 Užívateľské rozhranie

Užívateľské rozhranie nástroja je prioritne zamerané na dokumentáciu REST API, pomocou ktorého je možné používanie nástroja. Na obrázku nižšie je možné vidieť náhľad dokumentácie. V hornej časti je menu, na ktorom sa nachádza odkaz na dokumentáciu API, prihlásenie a registráciu. Po prihlásení sa, ešte zobrazuje odkaz na registráciu nového tokenu. V ľavej časti zobrazenej stránky sa nachádza rýchle menu dokumentácie. V pravej časti sa nachádzajú príklady, ako je možné pomocou technológie cURL[1] pracovať s API. Stredná časť sa venuje popisu jednotlivých druhov požiadaviek, ich parametrom a odpovediam, ktoré API vracia. Celé užívateľské rozhranie je implementované tak, aby bolo použiteľné aj na mobilných zariadeniach. Podľa rozmerov obrazovky zariadenia, na ktorom je dokumentácia zobrazená, sa postupne skryje ľavá a pravá časť dokumentácie.



Obrázok 5.8. Náhľad užívateľského rozhrania dokumentácie

5.5 Testovacia prevádzka

Pre testovaciu prevádzku bola dohodnutá spolupráca so spoločnosťou Web Klikem, ktorá prevádzkuje systém Klikem.cz. Spoločnosť umožnila autorovi tejto práce implementáciu triedy na úrovni služieb v architektúre systému Klikem, ktorá zabezpečí komunikáciu s týmto nástrojom. Súčasťou tejto implementácie boli aj metódy, ktoré zabezpečili zber testovacích dát pre obe testovacie metodiky. Testovacie metodiky sú popísané v kapitole 8, ktorá sa venuje testovaniu. Spoločnosť bola ďalej zodpovedná za implementáciu metód, ktoré prevedú vygenerovanú štruktúru do ich redakčného systému. Spoločnosť vyjadrila potrebu ukladania zmien štruktúry v databáze, ktoré sú súčasťou jednej z metodík testovania. Z tohto dôvodu bola pre tieto dáta vytvorená tabuľka v ich databáze. Následne zverejnili autorovi tejto práce dáta z obdobia testovacej prevádzky tohto nástroja. Tabuľka obsahuje nasledujúce stĺpce:

- id - unikátny automaticky generovaný identifikátor,
- site_id - cudzí kľúč ku stránke ktorej sa zmeny týkajú,
- change - názov zmeny,

[1] `https://curl.haxx.se/`

- value - hodnota zmeny.

V prípade zmien typu pridanie alebo zmazanie elementu sa hodnota v stĺpci value rovná `null`. Ak bola zmenou v štruktúre presun elementu, tak hodnota v stĺpci value rovná Manhattanskej vzdialenosti presunu elementu. Bližšie sa výpočtu tejto vzdialenosti venuje kapitola 7.2.

Dáta z druhej testovacej metodiky (dotazník) sú ukladané do CSV súboru, ktorý po ukončení testovacej prevádzky bude odovzdaný autorovi tejto práce. Všetky odpovede, ktoré boli uložené v tomto súbore je môžné nájsť v prílohe G tejto práce.

5.6 Nasadenie

Pre nasadenie nástroja bolo zvolené cloudové riešenie, ktoré umožňuje výborné možnosti v oblasti škálovania. Nástroj bol nasadení v cloude DigitalOcean[1]. Po výbere virtuálneho stroja bola nutná inštalácia nasledujúcich závislostí:

- Nginx - server,
- PostgreSQL - databáza,
- PHP 7.2,
- Google Chrome - súčasťou je bezhlavičkový mód,
- Composer - nástroj na správu závislosti pre PHP,
- Go 1.11.

Závislosti boli inštalované cez príkazový riadok. Po vytvorení virtuálneho stroja boli dostupné iba základné funkcie, ktorými disponuje operačný systém Ubuntu 18.04 a SSH pomocou ktorého je možná vzdialená správa. DigitalOcean umožňuje monitorovanie využitia hardvérových prostriedkov, ktoré dáva výborné informácie k tomu či je potrebné rozšírenie hardvéru. Systém je dostupný na doméne `http://sg.lukasfigura.sk`. Návod, ako korektne nainštalovať aplikáciu je uvedený v kapitole 6.

[1] `https://www.digitalocean.com/`

Kapitola 6
Inštalácia

Zdrojové súbory aplikácie sú na priloženom CD v tejto práci. Pre úspešnú inštaláciu aplikácie je potrebné vykonať nasledujúce kroky.

6.1 Závislosti a ich konfigurácia

Predpokladom pre úspešnú inštaláciu aplikácie je server s operačným systémom, ktorý je postavený na Linuxovom jadre. Autor tejto práce odporúča operačný systém Ubuntu vo verzii 18.04, ktorý bol použitý pri testovacej prevádzke systému. Pri čistej inštalácii tohto operačného systému je potrebné doinštalovať nasledujúce závislosti.

V prvom rade je potrebné rozbehnúť server. Autor tejto práce odporúča ako server použiť Nginx. Inštaláciu spustíme nasledujúcim príkazom, ktorý vykonáve v príkazovom riadku.

```
sudo apt install nginx
```

Vykonaním tohto príkazu by sa mal Nginx úspešne nainstalovať a uvítacia stránka by mala byť dostupná na doméne `http://IP_adresa_servera`. Následne je potrebné nainštalovať PHP a nakonfigurovať Nginx, aby ho používal. Najnovšiu verziu PHP nainštalujeme vykonaním tohto príkazu.

```
sudo apt install php php-pgsql
```

Časť príkazu `php-pgsql` nainštaluje knižnicu, ktorá umožňuje prácu PHP s databázou PostgreSQL. Teraz je potrebné dokončiť konfiguráciu Nginxu, aby používal PHP. Nasledujúcim príkazom vytvoríme nový konfiguračný súbor.

```
sudo nano /etc/nginx/sites-available/vasa_domena.com
```

Textový reťazec `vasa_domena.com` nahraďte doménou, pre ktorú bude táto konfigurácia platná. Úpravou nového konfiguračného súboru namiesto pôvodného dosiahneme jednoduchý prechod do pôvodných nastavení serveru. Do vytvoreného súboru pridáme nasledujúci obsah.

```
server {
        listen 80;
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name vasa_domena.com;

        location / {
                try_files \$uri \$uri/ =404;
        }

        location ~ \.php\$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
```

```
        }

        location ~ /\.ht {
                deny all;
        }
}
```

Po uložení a zatvorení súboru je potrebné ešte vytvoriť prelinkovanie na nový konfiguračný súbor. To zabezpečíme príkazom:

```
sudo ln -s /etc/nginx/sites-available/vasa_domena.com \
/etc/nginx/sites-enabled/
```

Následne zrušíme prelinkovanie na pôvodný konfiguračný súbor nasledujúcim príkazom:

```
sudo unlink /etc/nginx/sites-enabled/default
```

Svoje preklepy otestujeme vykonaním nasledujúceho príkazu:

```
sudo nginx -t
```

Ak nám neboli oznámené žiadne chyby reštartujeme server.

```
sudo systemctl reload nginx
```

Po vykonaní tohto príkazu by mal byť plne funkčný server s podporou pre PHP. Otestovať to môžeme vytvorením súboru `index.php`, ktorý umiestnime do adresára `/var/www/html`. Obsah súboru bude nasledovný:

```
<?php
phpinfo();
```

Po zadaní adresy `http://your_server_domain_or_IP/index.php` do webového prehliadača by sme mali vidieť informácie o nainštalovanom PHP. Ak inštalácia bola úspešná dočasný súbor index.php zmažeme.

Teraz ešte potrebujeme nainštalovať PostgreSQL, ktoré bude zabezpečovať databázovú časť aplikácie. Oficiálny návod, ako nainštalujeme PostgreSQL nájdeme na adrese `https://www.postgresql.org/download/`. Zároveň si aj vytvoríme novú databázu, ktorú bude naša aplikácia používať.

Poslednou technológiou, ktorú potrebujeme nainštalovať na serveri je Go. Inštaláciu uskutočníme vykonaním nasledujúcich príkazov.

```
sudo apt install wget
wget https://dl.google.com/go/go1.11.2.linux-amd64.tar.gz
tar -C /usr/local -xzf go1.11.2.linux-amd64.tar.gz
export PATH=\$PATH:/usr/local/go/bin
```

V tomto momente by sme mali mať kompletne nainštalované a pripravené všetky technológie, ktoré sú potrebné pre korektnú prácu aplikácie.

6.2 Nasadenie a konfigurácia aplikácie

V prvom kroku skopírujeme všetky súbory zo zložky `/app`, ktorá sa nachádza na pribalenom CD, do nejakého pracovného adresára na serveri. Otvoríme príkazový riadok a prejdeme do zložky `/prob_computer`, v ktorej sa nachádzajú zdrojové súbory pre výpočtovú časť aplikácie napísanu v Go. Následne potrebujeme upraviť prihlasovacie

údaje do databázy v konfiguračnom súbore `/src/prob_computer/constants.go`. Obsah súboru by mal teda vyzerať nasledovne:

```
package prob_computer

const (
host       = "localhost"
port       = 5432
user       = "uzivatel"
password   = "heslo"
dbname     = "nazov_databazy"
widthStep  = 32
heightStep = 40
totalWidth = 1920
)
```

Položky `widthStep, heightStep a totalWidth` slúžia na konfiguráciu analýzy a generovania štruktúr webových stránok. Autor tejto práce neodporúča zmenu týchto hodnôt. Pre vysvetlenie, položky `widthStep a heightStep` upravujú veľkosť jednotlivých analyzovaných blokov. Položka `totalWidth` udáva šírku stránky v akej je daná stránka analyzovaná. Autor práce upozorňuje, že výsledok podielu položiek `totalWidth a widthStep` by mal byť celočíselný, aby bol zabezpečený celočíselný počet analyzovaných a generovaných stĺpcov.

Teraz potrebujeme skompilovať zdrojové súbory, aby boli spustiteľné z príkazového riadku na našom servri. To docielime vykonaním nasledujúcich príkazov v zložke `/prob_computer`:

```
env GOOS=linux GOARCH=amd64 go build element_computer.go
env GOOS=linux GOARCH=amd64 go build structure_computer.go
```

Tieto príkazy nám vygenerujú spustiteľné súbory `element_computer` a `structure_computer`. Dané súbory prekopírujeme do zložky `/var/www/html`. Ďalej prekopírujeme obsah zložky `/structure_generator`, ktorá sa nachádza v zložke `/app` na pribalenom CD. Zložka `/var/www/html` musí obsahovať súbor `.env`, ktorý obsahuje konfiguráciu obslúžnej časti aplikácie. Jeho obsah by mal vyzerať s korektne doplnenými údajmi pre prihlásenie do databázy a cestami k spustiteľným súborom výpočtovej časti aplikácie nasledovne.

```
APP_NAME="Structure generator"
APP_ENV=prod
APP_KEY=
APP_DEBUG=true
APP_URL=http://vasa_domena.com

LOG_CHANNEL=daily

DB_CONNECTION=pgsql
DB_HOST=localhost
DB_PORT=5432
DB_DATABASE=nazov_databazy
DB_USERNAME=uzivatel
DB_PASSWORD=heslo

BROADCAST_DRIVER=log
```

```
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

GO_PATH=/var/www/html
GO_ELEMENT_COMPUTER_SCRIPT_NAME=element_computer
GO_STRUCTURE_COMPUTER_SCRIPT_NAME=structure_computer

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="\${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="\${PUSHER_APP_CLUSTER}"
```

Položka `GO_PATH` obsahuje cestu k spustiteľným súborom výpočtovej časti. Posledným úkonom je vygenerovanie kľúča aplikácie, ktorý sa používa napríklad pri zabezpečení hesiel. To zabezpečíme vykonaním príkazu `php artisan key:generate` v zložke `/var/www/html`.

Kapitola 7
Testovanie

Súčasťou vývoja každého softvéru je testovanie. Keďže ma softvér, ktorý je výsledkom tejto práce, pomôcť hlavne koncovým užívateľom, je veľmi dôležité, aby testovanie prebehlo s užívateľmi. Testovanie vyhodnotilo nielen či softvér generuje relevantné štruktúry stránok, ale aj to, ako je koncový užívateľ s výslednou štruktúrou spokojný. Softvér bol na dočasnú dobu v testovacej prevádzke v partnerskom systéme, ktorý zabezpečil dostatočné množstvo testovacích dát.

7.1 Testovacie metodiky

Testovanie pozostávalo z dvoch metodík. Tieto metodiky vyhodnotili, či softvér generuje relevantné štruktúry stránok a zároveň, či je koncový užívateľ s výsledkom spokojný. Očakávaným výsledkom testovania je veľká obľúbenosť u koncových užívateľov.

Prvá metodika testovania je určená k empirickému vyhodnoteniu relevantnosti vygenerovaných štruktúr webových stránok pomocou klasifikácie zmien, ktoré užívateľ vykonal po vygenerovaní štruktúry webovej stránky. Táto metodika je podrobne vysvetlená v kapitole 7.2.

Druhá metodika cieli na užívateľa. Dotazník, ktorý je podstatou tejto časti je popísaný v podkapitole 7.3. Spätná väzba od užívateľa je veľmi dôležitá a môže pomôcť pri výbere smeru, ktorým má vývoj pokračovať, aby bol zaručený jeho úspech.[39]

7.2 Klasifikácia zmien

Po vygenerovaní štruktúry stránky má užívateľ možnosť ďalej stránku upravovať. Podľa počtu a závažnosti úprav v štruktúre stránky vieme povedať či daná štruktúra skutočne užívateľovi vyhovovala. Budeme rozlišovať tri nasledujúce zmeny:

- pridanie elementu,
- vymazanie elementu,
- presun elementu.

Pre určenie závažnosti presunu elementu bude použitá Manhattanská vzialenosť medzi pôvodnou pozíciou elementu a novou pozíciou. Každý element vieme jednoznačne popísať jeho pozíciou voči ľavému hornému okraju webovej stránky. Teda X bude pôvodná pozícia elementu vyjadrená vzdialenosťou od ľavého okraja webovej stránky a Y bude pôvodná pozícia elementu vyjadrená vzdialenosťou elementu od horného okraja webovej stránky. Obdobne to platí aj pre X' a Y', ktoré už ale budú predstavovať novú pozíciu presunutého elementu. Obe vzdialenosti budú vyjadrené v pixeloch. Manhattanskú vzdialenosť presunu elementu A vypočítame pomocou nasledujúceho vzorca:

$$d_A = |X_A - X'_A| + |Y_A - Y'_A| \tag{1}$$

Následne bude Manhattanská vzdialenosť presunu elementu d_A porovnaná s Manhattanskou vzdialenosťou ľavého horného rohu stránky s pravým dolným rohom webovej

stránky d_{ALL}, aby bola odstránená relatívna miera presunu elementu v rámci webovej stránky.

$$m = d_A/d_{ALL} \tag{2}$$

Závažnosť zmeny bude vyjadrená stupnicou od 0 po 4. Pričom vyššie číslo predstavuje väčšiu vážnosť zmeny štruktúry stránky. Tabuľka nižšie definuje závažnosť štrukturálnej zmeny webovej stránky.

Zmena	Závažnosť zmeny
Vymazanie elementu	SCH = 4
Pridanie elementu	SCH = 4
Presun elementu	m $\geq$ 0.5 $\Rightarrow$ SCH = 4
	m $\geq$ 0.3 $\Rightarrow$ SCH = 3
	m $\geq$ 0.2 $\Rightarrow$ SCH = 2
	m $\geq$ 0.1 $\Rightarrow$ SCH = 1
	m $\geq$ 0 $\Rightarrow$ SCH = 0

Tabuľka 7.1. Bodové hodnotenie závažnosti zmien

Výsledná priemerná závažnosť zmien v štruktúre webovej stránky je daná nasledujúcim vzorcom:

$$ASCH = sum_{D \in E}(SCH_D)/c \tag{3}$$

- ASCH - priemerná závažnosť zmien v štruktúre webovej stránky,
- E - množina zmien v štruktúre webovej stránky,
- SCH_D - závažnosť zmeny v štruktúre webovej stránky určená podľa tabuľky vyššie,
- c - počet zmien v štruktúre webovej stránky.

Pre túto metodiku bola stanovená hypotéza, ktorá je určená vyššie uvedeným vzorcom pre výpočet priemernej závažnosti zmien v štruktúre webovej stránky. Cieľom je dosiahnuť najmenšiu možnú priemernú závažnosti zmien v štruktúre webovej stránky.

7.2.1 Výsledok metodiky

Tabuľka nižšie zobrazuje postupne počty zmien, sumu závažností zmien a priemernú závažnosť zmien pre jednotlivé stránky, ktoré boli vygenerované počas testovacej prevádzky systému.

Číslo stránky	c	$\text{sum}_{D \in E}\ (\text{SCH}_D)$	ASCH
1	0	0	0
2	8	1	0,125
3	15	27	1,8
4	1	4	4
5	13	24	1,85
6	3	12	4
7	0	0	0
8	2	8	4
9	2	8	4
10	0	0	0
11	4	12	3
12	4	16	4
13	8	32	4
14	2	8	4
15	0	0	0

Tabuľka 7.2. Výsledok metodiky klasifikácie zmien

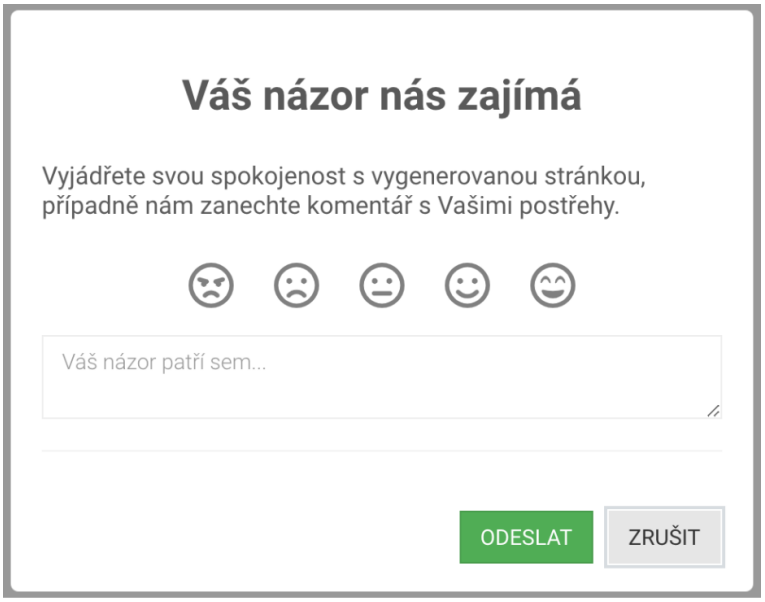
Priemerne sa teda závažnosť zmien na webových stránkach rovná 34,775 / 15 = **2,32**. Ak sa teda riadime stupnicou uvedenou vyššie a hodnota 4 predstavuje veľmi vážnu zmenu v štruktúre, tak môžeme povedať, že užívatelia robili stredne závažné zmeny. Avšak z tabuľky vyššie môžeme subjektívne zhodnotiť, že ak už museli užívatelia robiť vážne zmeny, tak to boli tie najzávažnejšie. Mohlo sa však ešte stať, že užívateľ neurobil žiadnu zmenu, pretože vygenerovaná štruktúra bola veľmi nevhodná a užívateľ s danou štruktúrou odmietol pracovať. Daná situácia nastala v prípade užívateľa číslo 15, ktorý hodnotil vygenerovanú stránku známkou 1 a v štruktúre už neurobil žiadne zmeny.

7.3 Dotazník

Cieľom dotazníka, ktorý je užívateľovi ponúknutý po vygenerovaní stránky, je zistiť či je užívateľ s vygenerovanou štruktúrou stránky spokojný. Dotazník zároveň ponúka užívateľovi možnosť vyjadriť svoj názor. Vďaka tomuto dotazníku by mal autor tejto práce zistiť všeobecný názor koncových užívateľov na tento softvér. Zozbierané názory užívateľov môžu pomôcť v ďalšom vývoji softvéru. Pre tento dotazník bola stanovená nasledujúca stupnica:

- 0 - veľmi negatívny výsledok,
- 1 - negatívny výsledok,
- 2 - neutrálny výsledok,
- 3 - pozitívny výsledok,
- 4 - veľmi pozitívny výsledok.

Cieľom je dosiahnuť čo najviac pozitívny výsledok. Daná stupnica je užívateľovi prezentovaná formou smajlíkou. Čím sa smajlík viac usmieva tým je aj výsledok pozitívnejší. Pod smajlíkmi nasleduje textové pole, v ktorom môže užívateľ prezentovať svoj názor.



Obrázok 7.1. Dotazník po vygenerovaní stránky

Nasledujúca tabuľka zobrazuje číselné vyjadrenie odpovedí, ktoré užívatelia odoslali počas testovacej prevádzky.

Užívateľ	Odpoveď
1	3
2	3
3	4
4	4
5	2
6	2
7	4
8	2
9	3
10	2
11	2
12	3
13	2
14	1
15	1

Tabuľka 7.3. Odpovede v dotazníku

Tabuľka nižšie obsahuje vypočítané hodnoty priemeru, mediánu a priemernej odchylky z hodnôt, ktoré sú uvedené v tabuľke vyššie.

Priemer	2,53
Medián	2
Priemerná odchylka mediánu	0,84

Tabuľka 7.4. Vypočítané hodnoty z výsledkov dotazníka

Z priemeru a podľa stupnici uvedenej vyššie môžeme povedať, že užívatelia hodnotili generovanie štruktúr na hrane pozitívneho a neutrálneho výsledku. Medián určuje neutrálny výsledok, avšak odchýlka jasne naznačuje, že sa hodnotenia značne líšili. Je však nutné zobrať do úvahy aj komentáre, ktoré užívatelia mohli vložiť pri vypĺňaní dotazníka. Z nich môžeme povedať, že ak už užívateľ hodnotil systém menej pozitívne poznamenal, že sa mu myšlienka automatického generovania páči a je potrebné ju ďalej rozvíjať. Objavili sa aj reakcie, ktoré nesúvisia so samotným vygenerovaním stránky, ale s následnou prácou v redakčnom systéme. Tieto komentáre nám naznačujú, že užívateľova reakcia môže byť ovplyvnená aj celkovou funkčnosťou redakčného systému. Práve najnižšie hodnotenie 1 bolo v momente, kedy sa užívateľ potýkal s problémom správy webovej stránky vo webovom prehliadači Firefox. Kompletný zoznam odpovedí v dotazníku sa nachádza v prílohe tejto práce.

Kapitola 8
Záver

Testovacia prevádzka tohto systému dokázala poukázať na silné stránky systému, ale aj na nedostatky, ktorými nástroj trpí. Veľkou škodou bolo, že spoločnosť Web Klikem nedokázala do termínu testovacej prevádzky pripraviť ich systém na redakciu webových stránok typu blog, preto bolo testovanie tohto systému iba pre webové prezentácie. Pred spustením testovacej prevádzky, bol autor tejto práce upozornený, že by bolo potrebné venovať väčšie množstvo času implementácii, ktorá zabezpečuje prevod vygenerovanej štruktúry. Spoločnosť sa ďalej vyjadrila, že aj v prípade nižšej spokojnosti užívateľov majú o tento nástroj záujem.

8.1 Zhodnotenie výsledkov

Na základe testovania môžeme povedať, že užívateľom sa možnosť automatického generovania stránok páčila a hodnotili ju ako skvelý nápad. Avšak je nutné byť aj do značnej miery kritický voči nástroju, ktorý je predmetom tejto práce. Autor práce už počas vývoja tohto nástroja zaznamenal, že rozmanitosť webových stránok je oveľa väčšia než sa na prvý pohľad zdalo. Už počas implementácie, bolo zistených niekoľko problémov, ktoré sú vysvetlené nižšie. Na základe problému popísaného v kapitole 8.1.1 bolo nutné obmedziť testovaciu prevádzku nástroja na generovanie štruktúry iba z jednej inšpiračnej stránky.

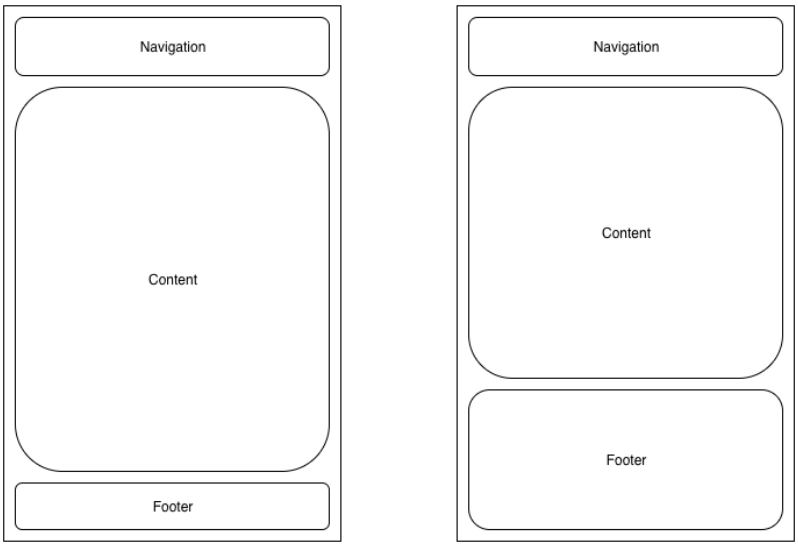
Z testovacej metodiky uvedenej v kapitole 7.2 vyplýva, že užívatelia boli nútení robiť stredne závažné zmeny vo vygenerovanej štruktúre ich webovej stránky. To znamená, že vygenerované štruktúry neboli úplne podľa predstáv užívateľov. Príčinou mohli byť problémy spomenuté nižšie.

Avšak, ako už bolo spomenuté vyššie, nápad automatického generovania webových stránok sa koncovým užívateľom páčil a takúto možnosť by zdá sa radi privítali. Preto je autor tejto práce motivovaný aj zo strany koncových užívateľov k ďalšiemu vývoju a poskytnúť nástroj, ktorý dokáže automaticky vygenerovať štruktúru webovej stránky.

8.1.1 Nekorektné prekladanie mriežok

Práve rozmanitosť a rôzne veľkosti stránok sú príčinou tohto problému. Pre bližšie pochopenie tohto problému si predstavme dve rôzne webové stránky, ktoré majú štruktúry zobrazené na obrázku nižšie. Veľmi často sa totiž môže stať, že inšpiračné stránky a aj stránky, z ktorých sa generuje predvolená štruktúra v danej kategórii, môžu mať proporcionálne iné rozmery jednotlivých kontextových blokov, ako si to je možné všimnúť na obrázku 8.1. K tomu môže prispieť aj celková výška stránky, ktorá môže byť medzi jednotlivými stránkami rozdielna a teda, pozície jednotlivých kontextuálnych blokov sa budú líšiť. Z toho vyplýva, že napríklad pozícia pätičky môže byť na jednotlivých stránkach rozdielna a to má za následok nekorektné preloženie jednotlivých mriežok. Samozrejme, že tento problém sa netýka len pätičky stránky, ktorá bola použítá iba ako príklad pre ľahšie pochopenie problému, ktorý vzniká. Tento problém by mohol byť čiastočne vyriešený proporcionálnym premeriavaním, avšak takéto premeriavanie by

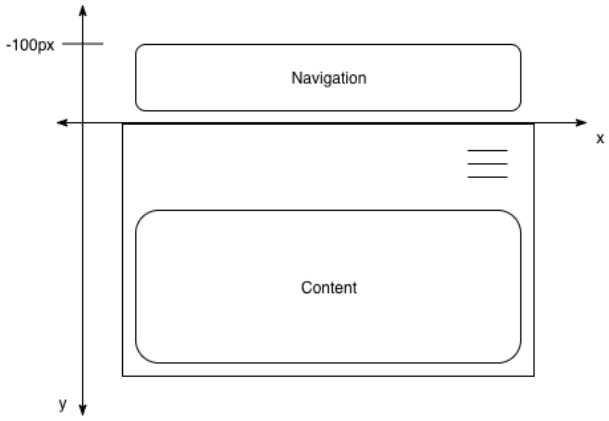
neodstránilo problém s rôznymi pozíciami jednotlivých kontextuálnych blokov. Autor tejto práce uvádza ako kontextuálny blok webovej stránky časť jej štruktúry, ktorá sa viaže k určitému kontextu, napríklad popis ponúkaných služieb, zloženie tímu, kontakt a iné. Preto je dôležité poznamenať, že by stálo za úvahu premyslieť oveľa komplexnejšie riešenie. To môže byť zároveň výborným podnetom pre ďalší vývoj tohto nástroja.



Obrázok 8.1. Príklad proporcionálne inej štruktúry

8.1.2 Absolútne pozície elementov

Pri zadávaní pozície jednotlivým elementom môžu byť použité aj absolútne hodnoty, ktoré sa niekedy môžu zdať ako absurdné. Na obrázku nižšie je predstavená situácia, v ktorej sa navigačný panel nachádza mimo zobrazenej webovej stránky.

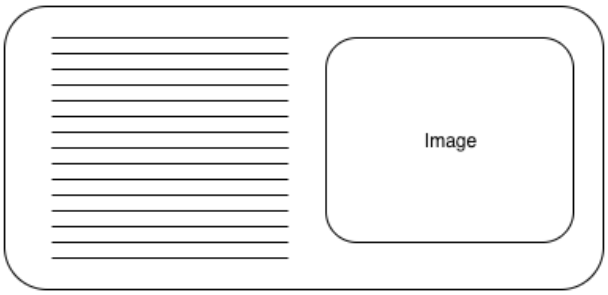


Obrázok 8.2. Navigačný panel mimo zobrazenej webovej stránky

Začiatkom v mriežkach s ktorými pracuje tento nástroj, je ľavý horný roh webovej stránky. Avšak v praxi môže nastať situácia, kedy sú elementy za hranicami zobrazenej

časti webovej stránky a do tejto časti sa dostanú až po určitej akcii zo strany užívateľa, ktorou v tomto prípade je kliknutie na tzv. “Hamburger” menu, ktoré je symbolicky zobrazené pomocou troch rovnobežných priamok v pravej hornej časti stránky. Po kliknutí sa navigačný panel dostane do zobrazenej časti webovej stránky. Takýmto prístupom môžu byť docielené rôzne efekty, ktorých cieľom nemusí byť len navigačný panel, ale aj iné elementy. Nástroj však nedokáže simulovať správanie užívateľa prípadne odhadnúť interakciu s webovou stránkou. Preto môžu byť takéto “skryté” elementy pri analýze prehliadnuté a neovplyvnia vygenerovanú štruktúru.

8.1.3 Zarovnávanie textového obsahu

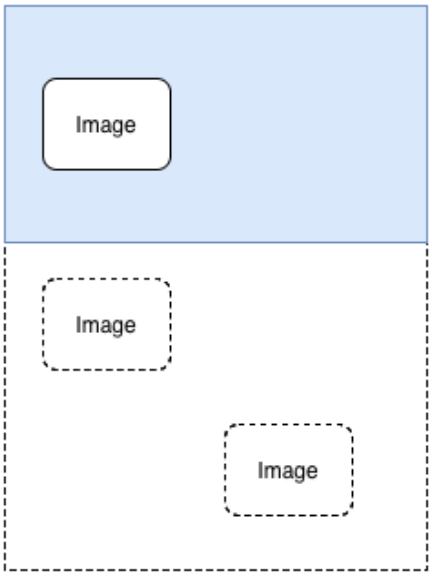


Obrázok 8.3. Zarovnanie textového obsahu

Obrázok vyššie znázorňuje veľmi častý problém, ktorý má za následok ovplyvnenie vygenerovanej štruktúry, nesprávnou analýzou. Tento problém spočíva v zarovnávaní textu, alebo použití tagu br, ktorý vkladá explicitne nový riadok v texte. Text, ktorý je symbolicky vyznačený rovnobežnými priamkami sa nachádza v elemente, ktorého súčasťou je aj obrázok. Textový obsah je len v ľavej časti, no počas analýzy je to vyhodnotené ako textový element s plnou šírkou a obrázok v pravej časti. To má za následok, že v pravej časti sa v analyzovanej mriežke berie v úvahu ako aj textový obsah tak aj obrázok.

8.1.4 Javascriptové načítanie obrázkov

Za účelom optimalizácie načítania stránok sa dosť často môžeme stretnúť s technikou oneskoreného javascriptového načítania obrázkov. Takéto načítanie môže výrazným spôsobom ovplyvniť rýchlosť načítania stránok napríklad na mobilných zariadeniach. Obrázok nižšie znázorňuje prípad, kedy sa napríklad takáto technika využíva. Zvýraznená modrá časť stránky je zobrazená vo webovom prehliadači užívateľa a zvyšná časť je dostupná až po zrolovaní webovej stránky nižšie. Preto je načítanie obrázkov, ktoré sú mimo zobrazenej časti, nutné až v prípade, kedy sa dostanú do zobrazenej časti webovej stránky. Takže pri načítaní webovej stránky sa zároveň načítajú iba obrázky v zobrazenej časti. To má za následok nulové rozmery obrázkov v nezobrazenej časti webovej stránky.

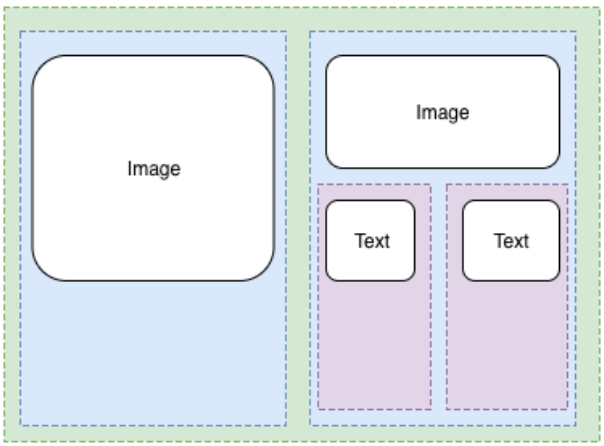


Obrázok 8.4. Obrázky mimo zobrazenej časti webovej stránky

Obrázky s nulovými rozmermi v podstate ani obrázkami niesú, a preto sa to neprejaví vo vygenerovanej štruktúre. Tento problém by sa dal odstrániť pridaním funkcie, ktorá bude kontrolovať načítanie obrázkov a ich rozmery by boli odoslané až v momente, kedy sú všetky obrázky načítané. Zároveň by však bolo nutné implementovať simuláciu skrolovania stránky v prípade ak sa načítanie obrázka viaže na to či je zobrazený. Avšak zobrazenie obrázka nemusí byť viazané iba na aktuálne zobrazenú časť webovej stránky a preto by bola nutná hlbšia analýza, ktorá by zabezpečila kvalitnejšie riešenie tohto problému.

8.1.5 Prevod generovanej štruktúry do redakčného systému

Ďalší problém, ktorý už ale ovplyvnil konečný výsledok prezentovaný užívateľovi, sa objavil pri prevode štruktúry webovej stránky z JSONu do podoby webovej stránky v konkrétnom redakčnom systéme. Vygenerovaná štruktúra, ktorá je dostupná prostredníctvom REST API vo formáte JSONu obsahuje konkrétne pozície a popis aký druh elementu sa na danej pozícii nachádza. Bližšie sa tomu venovala kapitola 6.3. Systém Klikem používa na členenie štruktúry webovej stránky riadky a stĺpce, avšak delenie stĺpcov je možné len do prvej úrovne. Ak teda je riadok rozdelený na konečný počet stĺpcov tak dané stĺpce nemôžu byť rozdelené na ďalšie stĺpce. Toto technické riešenie limituje prevod vygenerovanej štruktúry na webovú stránku pretože môže nastať prípad, ktorý je zobrazený na obrázku nižšie.



Obrázok 8.5. Delenie stĺpcu na ďalšie stĺpce

Zelenou farbou je vyznačený riadok v štruktúre webovej stránky, ktorý obsahuje dva modré stĺpce. Avšak systém Klikem nedokáže technicky vygenerovať fialové stĺpce v pravom modrom stĺpci. Tento problém nemusí byť iba v redakčnom systéme Klikem, ale aj v iných systémoch, ktoré pracujú na podobnom princípe.

8.2 Možnosti budúceho vývoja

Ďalší vývoj tohto nástroja môže obsahovať nasledujúce funkcie:

- vytvoriť komplexnejšie riešenie pri generovaní a analýze stránok, ktoré odstráni problémy uvedené v kapitole 8.1,
- príklady práce s API v iných technológiách - veľké množstvo redakčných systémov zabezpečujú aj veľkú variabilitu v použitých technológiách, preto pre väčšiu obľúbenosť použitia nástroja navrhujeme uvedenie príkladov práce s API v technológiách akými sú Java, PHP a Python,
- inštaláciu HTTPS certifikátu pre vyššiu mieru zabezpečenia REST API,
- pridanie možnosti generovania štruktúry pre webovú stránku kategórie internetového obchodu,
- skrášlenie webovej stránky, ktorá prezentuje tento nástroj.

Literatúra

[1] *Total number of Websites*.
http://www.internetlivestats.com/total-number-of-websites/.

[2] Cisco. *The Zettabyte Era: Trends and Analysis*. 2017.
https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html_Toc484556821.

[3] Betsy McLeod. *Do I need a website for my small business? Yes. Yer, you do.* 2018.
https://www.bluecorona.com/blog/do-i-need-a-website.

[4] Nick Schäferhoff. *How Much Does a Website Cost?* 2018.
https://websitesetup.org/website-cost/.

[5] Diana Budds. *Can You Teach AI To Design? Wix Thinks So*. 2016.
https://www.fastcompany.com/3060934/can-you-teach-ai-to-design-wix-is-trying.

[6] Nigel Bevan. *Usability issues in web site design*. In: *HCI (2)*. 1997. 803–806.

[7] *Website*.
https://www.techopedia.com/definition/5411/website.

[8] Joshua Frisby. *Designing Websites*. 2018.
https://www.websitebuilderexpert.com/designing-websites/.

[9] Jakob Nielsen. Top 10 guidelines for homepage usability. *Neilsen Norman Group. Google Scholar*. 2002,

[10] Renee Chambers. *10 steps for better website navigation*.
https://www.butterfly.com.au/blog/design/10-steps-for-better-website-navigation.

[11] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information.. *Psychological review*. 1956, 63 (2), 81.

[12] John Medina. *Brain rules*. 2008.

[13] Bogdan Sandu. *How to Keep Users Happy When Designing a Website With Video*.
https://www.bittbox.com/general-web-design/using-video-in-web-design.

[14] *Sound on websites: A sensitive subject*. 2009.
http://www.everydaylistening.com/articles/2009/6/15/sound-on-websites-a-sensitive-subject.html.

[15] Tim O'reilly. *What is web 2.0*. 2005.

[16] Ogi Djuraskovic. *What is a Blog? – The Definition of Blog, Blogging, and Blogger*. 2018.
https://firstsiteguide.com/what-is-blog/.

[17] Dan Shewan. *5 Hot Blog Design Trends in 2017*. 2018.
https://www.wordstream.com/blog/ws/2017/05/30/blog-design-trends.

[18] Ralph Stair a George Reynolds. *Principles of information systems*. Cengage Learning, 2013.

[19] *The Future of Website Creation: Wix Artificial Design Intelligence*. 2016. https://www.wix.com/blog/2016/06/wix-artificial-design-intelligence/.

[20] *Salted Password Hashing - Doing it Right*. 2018. https://crackstation.net/hashing-security.htm.

[21] Ľudmila Jašková. *Vývoj IS*. http://edi.fmph.uniba.sk/~jaskova/InformacneSystemy/tema10/tema10.html.

[22] Abhijit Dubey a Dilip Wagle. Delivering software as a service. *The McKinsey Quarterly*. 2007, 6 (2007), 2007.

[23] Mark Turner, David Budgen a Pearl Brereton. Turning software into a service.. *Computer.*. 2003, 36 (10), 38–44.

[24] Haojie Hang a Ogheneovo Dibie. *Software as a Service.* In: *Configuration and Customization Perspectives," in 2008 IEEE Congress on Services Part II (services-2 2008), IEEE*. 2008.

[25] Jakob Nielsen. *How Little Do Users Read?* 2008. https://www.nngroup.com/articles/how-little-do-users-read/.

[26] MathWorks. *What is MATLAB?* https://www.mathworks.com/discovery/what-is-matlab.html.

[27] *What is best language for enterprise solutions software development and why?* https://www.quora.com/What-is-best-language-for-enterprise-solutions-software-development-and-why.

[28] MathWorks. *Matlab*. https://www.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab.

[29] Google. *The Go Project*. https://golang.org/project/.

[30] Keval Patel. *Why should you learn Go?* 2018. https://medium.com/@kevalpatel2106/why-should-you-learn-go-f607681fad65.

[31] Lance Diduck. *Is Python and multithreading a good idea?* 2018. https://www.quora.com/Is-Python-and-multithreading-a-good-idea.

[32] Jamie Spencer. *Learn to Code: What's the Best Programming Language to Learn First?* https://makeawebsitehub.com/which-programming-language/.

[33] Michael Garbade. *Top 3 most popular programming languages in 2018*. 2018. https://hackernoon.com/top-3-most-popular-programming-languages-in-2018-and-their-annual-salaries-51b4a7354e06.

[34] Oracle. *What is Java technology and why do I need it?* https://www.java.com/en/download/faq/whatis_java.xml.

[35] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*. 1976, 1 (1), 9–36.

[36] PostgreSQL. *PostgreSQL 9.4.20 Documentation*. https://www.postgresql.org/docs/9.4/datatype-json.html.

[37] *Most Popular PHP Frameworks Of 2018*. https://lunarpages.com/most-popular-php-frameworks-of-2018/.

[38] Cody Arsenault. *6 Popular Headless Browsers for Web Testing*. https://www.keycdn.com/blog/headless-browsers.

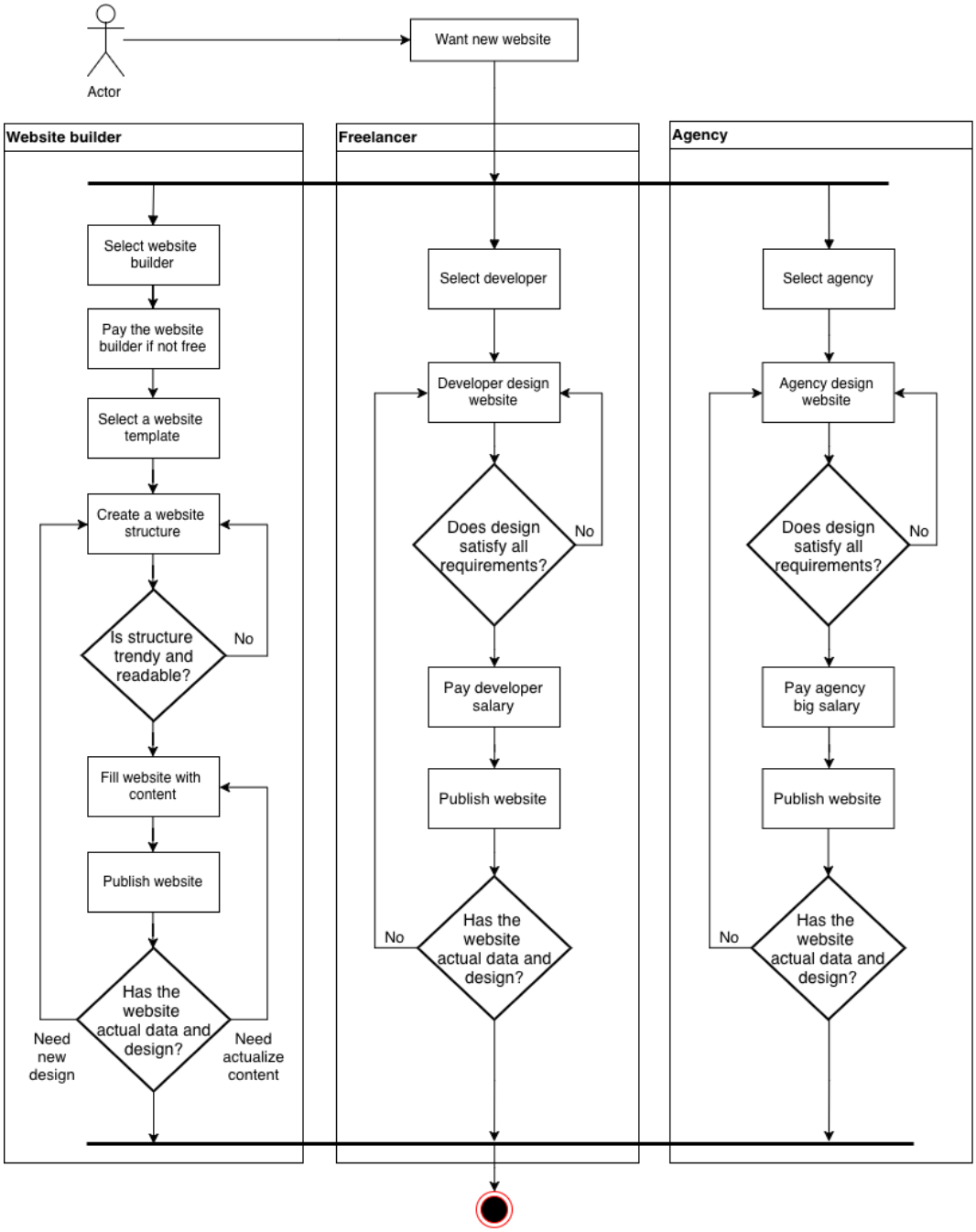
[39] Boardroom Metrics. *The Importance of Relevant User Input for IT Project Planning Success*. 2013.

https://www.boardroommetrics.com/blog/the-importance-of-relevant-user-input-for-it-project-planning-success-20131110.htm.

Príloha A
Slovník

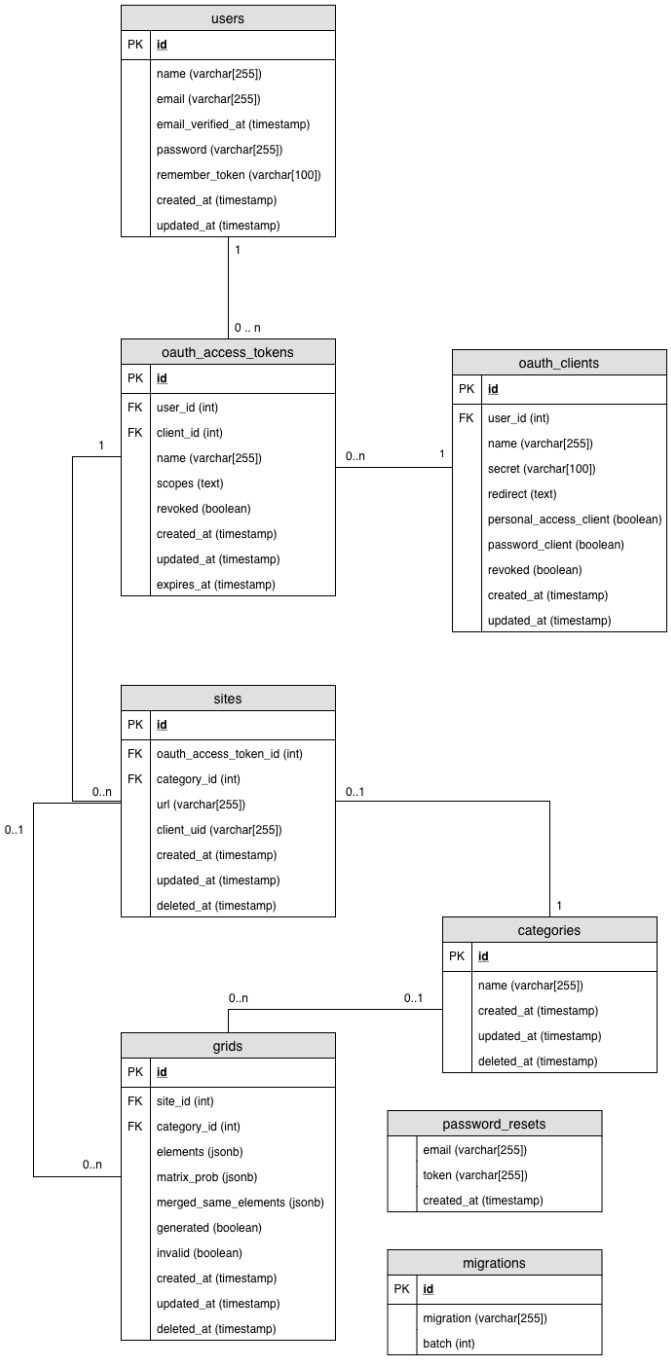
AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
CD	Compact disc
CIO	Chief information officer
CSRF	Cross-site request forgery
CSS	Cascading Style Sheets
CSV	Comma Separated Values
CZK	Česká koruna
EB	Exabyte
ER	Entity Relationship
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
MVC	Model View Controller
ORM	Object Relation Model
RFC	Request For Comments
SaaS	Software as a Service
SEO	Search Engine Optimization
SOA	Service Oriented Architecture
SQL	Structured Query Language
SSH	Secure Shell
UI	User Interface
URL	Unified Resource Locator
UX	User Experience
XHR	XMLHttpRequest
XML	Extensible Markup Language
XSS	Cross-site scripting

Príloha B
Scenár tvorby webovej stránky



Obrázok B.1. Scenár vytvorenia webovej stránky pre novú firmu

Príloha C
ER diagram



Obrázok C.2. ER diagram dátového modelu nástroja

Príloha D
Náhľad odmeraných hodnôt v javascripte


```
{
   "top":1044,
   "left":975,
   "type":"text",
   "width":485,
   "height":30
},
{
   "top":1134,
   "left":1075.828125,
   "type":"text",
   "width":283.328125,
   "height":57
},
{
   "top":1322,
   "left":460,
   "type":"text",
   "width":313.328125,
   "height":22
},
{
   "top":0,
   "left":445,
   "type":"image",
   "width":192,
   "height":85
},
{
   "top":0,
   "left":445,
   "type":"image",
   "width":192,
   "height":85
},
{
   "top":85,
   "left":1152,
   "type":"image",
   "width":694.375,
   "height":400
},
{
   "top":1325.46875,
   "left":1318.328125,
   "type":"image",
   "width":171.5625,
   "height":15
},
```


Obrázok D.3. Náhľad JSONu odmeraných hodnôt v javascripte

Príloha E
Náhľad početnosti v analyzovanej mriežke


```
{
   "top":40,
   "left":928,
   "width":32,
   "height":40,
   "navItemCount":0,
   "textItemCount":0,
   "imageItemCount":0,
   "headingItemCount":0
},
{
   "top":40,
   "left":960,
   "width":32,
   "height":40,
   "navItemCount":0,
   "textItemCount":1,
   "imageItemCount":0,
   "headingItemCount":0
},
{
   "top":40,
   "left":992,
   "width":32,
   "height":40,
   "navItemCount":0,
   "textItemCount":1,
   "imageItemCount":0,
   "headingItemCount":0
},
{
   "top":40,
   "left":1024,
   "width":32,
   "height":40,
   "navItemCount":0,
   "textItemCount":2,
   "imageItemCount":0,
   "headingItemCount":0
},
{
   "top":40,
   "left":1056,
   "width":32,
   "height":40,
   "navItemCount":0,
   "textItemCount":1,
   "imageItemCount":0,
   "headingItemCount":0
},
```


Obrázok E.4. Náhľad početnosti v analyzovanej mriežke

Príloha F
Náhľad vygenerovanej mriežky


```
{
    "top":4200,
    "left":384,
    "type":"heading",
    "width":192,
    "height":40,
    "verticallyMerged":false
},
{
    "top":4200,
    "left":576,
    "type":"text",
    "width":192,
    "height":40,
    "verticallyMerged":false
},
{
    "top":4200,
    "left":768,
    "type":"heading",
    "width":768,
    "height":40,
    "verticallyMerged":false
},
{
    "top":4200,
    "left":1536,
    "type":"",
    "width":384,
    "height":40,
    "verticallyMerged":true
},
```

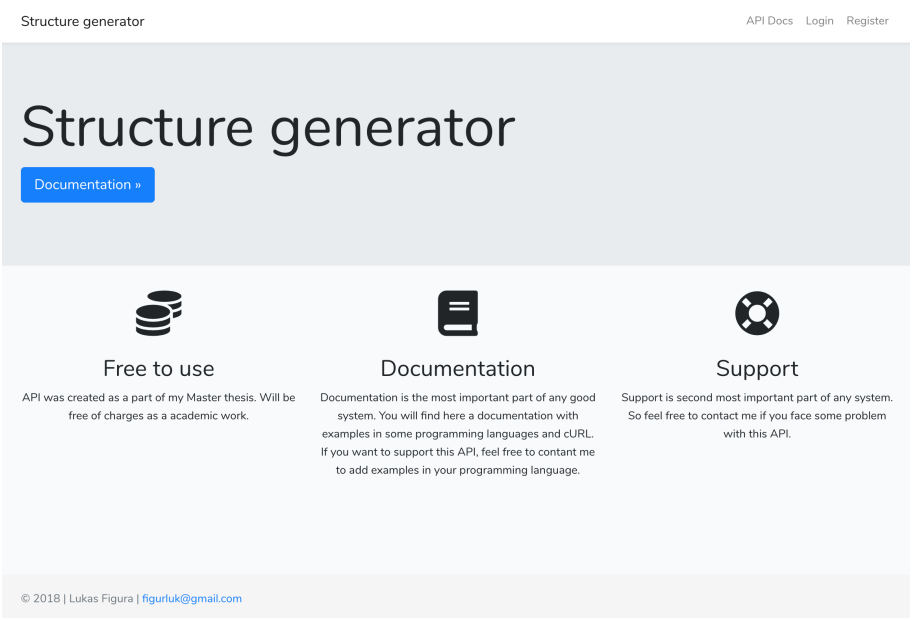

Obrázok F.5. Náhľad JSONu vygenerovanej mriežky

Príloha G
Odpovede v dotazníku

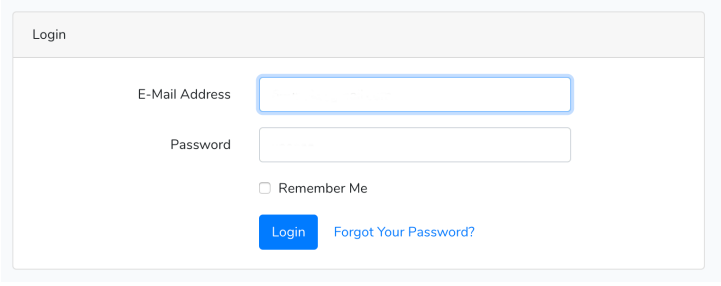
Smile	Comment
3	Trebalo by to ešte asi trochu doladiť ale super nápad na pomoc pre amatérov v tvorení stránok.
3	Vytvoření stránky bylo velmi jednoduché. Nic komplikované. Zvládl by to i méně zdatný člověk, co s týče počítačů. Líbilo se mi vytvořit si svou vlastní stránku, podle mých představ. Pozitivy jsou velký výběr šablon, písma, přidávání obrázků. Negativa jsem nenašla žádné.
4	Bola som veľmi milo prekvapená z vytvorenia mojej stránky. Bolo to skvelé! Ďakujem.
4	Vytvorenie stránky bolo jednoduché a rýchle. Páčila sa mi široká ponuka šablón a celkový vzhľad vytvorenej stránky.
2	
2	Páči sa mi tento nápad a ľahká práca na tejto stránke. Škoda, že mi nevygenerovalo stánku, ktorú som žiadala.
4	
2	
3	veľmi user friendly stranka, možno viac vymakať grafiku by to chcelo, funkčne ale je to uplne v pohode
2	Mi to hodilo inú šablónu ako som si zvolil
2	
3	Základ pre stránku bol vytvorený podľa mojich predstáv, jednoducho si môžem posunúť jednotlivé políčka, doplniť text alebo obrázok.
2	
1	Na Mozille mi to nefungovalo skoro vobec, sablona sa sekala, pri pretahovani ikoniek zlava sa mi to vzdy zaseklo a neslo s tym pohnut. Ak som tym pohla, tak neslo ulozit zmeny, hlasilo to "pracujem...".Potom som skusila chrome, kde to bolo lepsie. Obrazky, texty apod. som vlozila bez problemov. Avsak, prvotna sablona, ktora na mna vyskocila mala x nadpisov, obrazkov...Ak som to chcela upravit, musela som to mazat po jednom, alebo som neprisla na spravny sposob, co ma nebavilo. Ich radenie vedla seba bolo nerealne. webovu stranku si myslim, ze by som vytvorila, ale myslim si, ze by som musela robit ustupky voci tomu, co naozaj chcem. Vlastne nechapem podstatu sablony, kludne by som prvotne mala prazdnu stranku nez nieco upravovala.
1	Som z toho dost zmäteny. Mam slabu predstavu ako by to vyzeralo chybaju mi tam obrazky. Aspon keby som videl ako to vyzera s asistentom a bez asistenta.

Obrázok G.6. Všetky odpovede v dotazníku

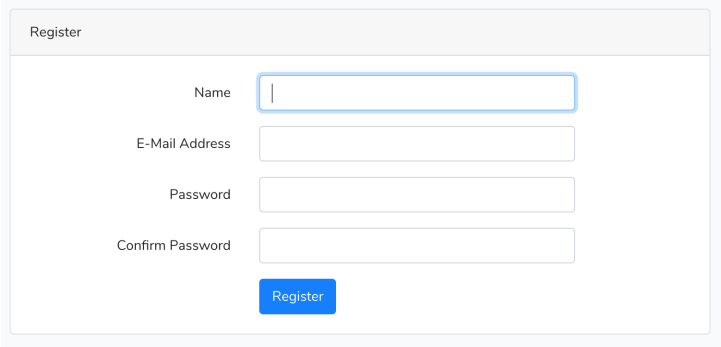
Príloha H
Náhľady UI



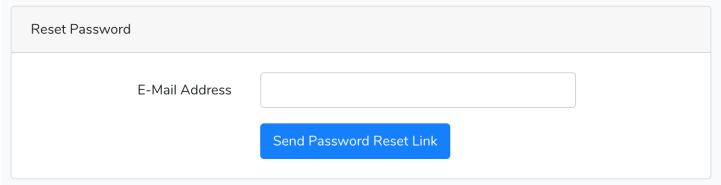
Obrázok H.7. Domovská stránka



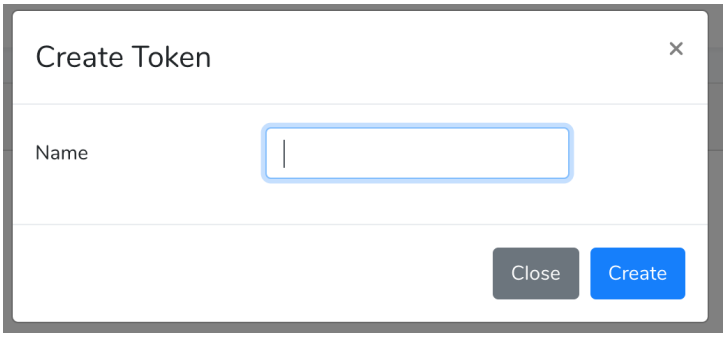
Obrázok H.8. Prihlásenie



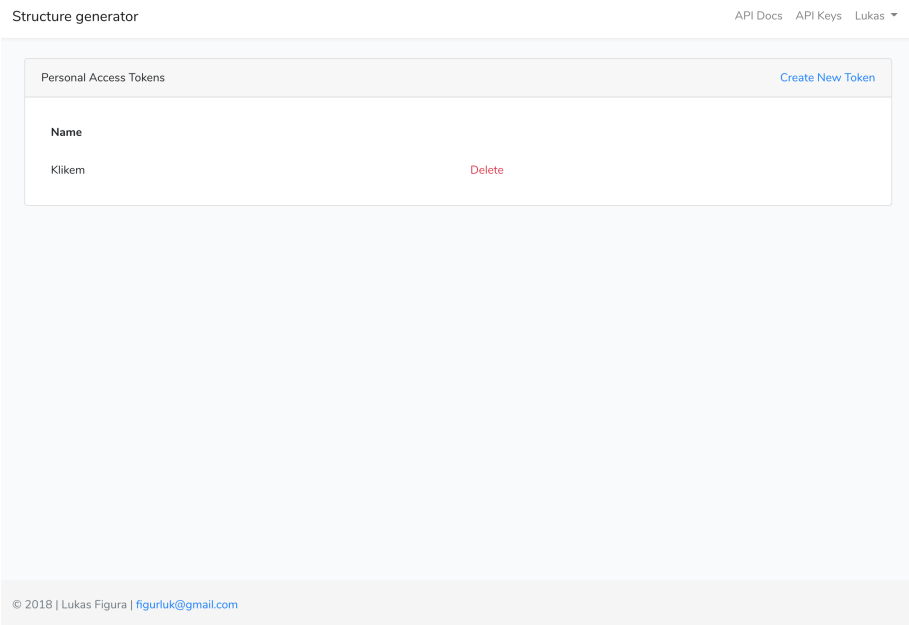
Obrázok H.9. Registrácia



Obrázok H.10. Obnova hesla



Obrázok H.11. Vytvorenie tokenu



Obrázok H.12. Správa tokenov

Príloha I
Obsah pribaleného CD

–app
|–prob_computer
| |–build_script.sh
| |–element_computer.go
| |–pkg (knižnice)
| |–src
| | |–github.com (knižnice)
| | |–prob_computer
| | |–computer.go
| | |–constants.go
| | |–db_operations.go
| | |–helpers.go
| | |–types.go
| |–structure_computer.go
|–structure_generator
| |–app
| | |–Classes
| | |–Console
| | | |–Kernel.php
| | |–Exceptions
| | | |–Handler.php
| | |–Http
| | | |–Controllers
| | | | |–Auth
| | | | | |–ForgotPasswordController.php
| | | | | |–LoginController.php
| | | | | |–RegisterController.php
| | | | | |–ResetPasswordController.php
| | | | | |–VerificationController.php
| | | | |–CRONController.php
| | | | |–Controller.php
| | | | |–GridController.php
| | | | |–StructureGeneratorController.php
| | | | |–StructureScrapperController.php
| | | |–Kernel.php
| | | |–Middleware
| | | |–Authenticate.php
| | | |–CheckForMaintenanceMode.php
| | | |–EncryptCookies.php
| | | |–LocalMiddleware.php
| | | |–RedirectIfAuthenticated.php
| | | |–TrimStrings.php

```
| | | |–TrustProxies.php
| | | |–VerifyCsrfToken.php
| | |–Models
| | | |–Category.php
| | | |–Grid.php
| | | |–Site.php
| | | |–User.php
| | |–Providers
| | | |–AppServiceProvider.php
| | | |–AuthServiceProvider.php
| | | |–BroadcastServiceProvider.php
| | | |–EventServiceProvider.php
| | | |–RouteServiceProvider.php
| | |–Services
| | |–AnalyzeService.php
| | |–ElementIdentificationService.php
| | |–GenerateService.php
| | |–ScrapperCorrectorService.php
| | |–ScrapperService.php
| |–artisan
| |–bootstrap
| | |–app.php
| | |–cache
| | |–packages.php
| | |–services.php
| |–composer.json
| |–composer.lock
| |–config
| | |–app.php
| | |–auth.php
| | |–broadcasting.php
| | |–cache.php
| | |–cors.php
| | |–database.php
| | |–element-types-break-dfs.php
| | |–element-types.php
| | |–filesystems.php
| | |–generator.php
| | |–hashing.php
| | |–ide-helper.php
| | |–logging.php
| | |–mail.php
| | |–queue.php
| | |–services.php
| | |–session.php
| | |–view.php
| |–cp_go_app.sh
| |–database
| | |–factories
```

| | | |–UserFactory.php
| | |–migrations
| | | |–2014_10_12_000000_create_users_table.php
| | | |–2014_10_12_100000_create_password_resets_table.php
| | | |–2018_10_14_142650_create_table_sites.php
| | | |–2018_10_14_145054_create_table_grids.php
| | | |–2018_10_14_145245_create_table_categories.php
| | | |–2018_10_14_161728_add_foreigns.php
| | |–seeds
| | |–DatabaseSeeder.php
| |–node_modules (knižnice)
| |–package-lock.json
| |–package.json
| |–phpunit.xml
| |–public
| | |–css
| | | |–app.css
| | |–favicon.ico
| | |–fonts
| | | |–vendor (knižnice)
| | |–index.php
| | |–js
| | | |–app.js
| | | |–layout_scrapper.js
| | |–mix-manifest.json
| | |–robots.txt
| | |–svg
| | | |–403.svg
| | | |–404.svg
| | | |–500.svg
| | | |–503.svg
| | |–web.config
| |–readme.md
| |–resources
| | |–js
| | | |–app.js
| | | |–bootstrap.js
| | | |–components
| | | |–ExampleComponent.vue
| | | |–passport
| | | |–AuthorizedClients.vue
| | | |–Clients.vue
| | | |–PersonalAccessTokens.vue
| | |–lang
| | | |–en
| | | |–auth.php
| | | |–pagination.php
| | | |–passwords.php
| | | |–validation.php

```
| | |–sass
| | | |–_variables.scss
| | | |–app.scss
| | |–views
| | |–analyze.blade.php
| | |–auth
| | | |–login.blade.php
| | | |–passwords
| | | | |–email.blade.php
| | | | |–reset.blade.php
| | | |–register.blade.php
| | | |–verify.blade.php
| | |–doc.blade.php
| | |–home.blade.php
| | |–layouts
| | | |–app.blade.php
| | |–oauth.blade.php
| |–routes
| | |–api.php
| | |–channels.php
| | |–console.php
| | |–cron.php
| | |–web.php
| |–server.php
| |–storage (cache)
| |–tests
| | |–CreatesApplication.php
| | |–Feature
| | | |–ExampleTest.php
| | |–TestCase.php
| | |–Unit
| | |–ExampleTest.php
| |–vendor (knižnice)
| |–webpack.mix.js
–thesis
| |–Technika-Bold.pfb
| |–Technika-BoldItalic.pfb
| |–Technika-Book.pfb
| |–Technika-BookItalic.pfb
| |–Technika-Italic.pfb
| |–Technika-Regular.pfb
| |–analyza.tex
| |–ctulogo-bw-new.pdf
| |–ctulogo-new.pdf
| |–ctustyle-ts.tex
| |–ctustyle2.tex
| |–encxvlna.tex
| |–glosdata.tex
| |–img
```

| | |–1-n.png
| | |–SOA.png
| | |–SWOT-Analysis.png
| | |–abs_values.png
| | |–ajax-load-img.png
| | |–all-answers.png
| | |–analyzeSiteSEQ.png
| | |–application-component-diagram.png
| | |–atom.png
| | |–blog_structure.png
| | |–case-study-uml.png
| | |–countedAnalyzed.png
| | |–countedJS.png
| | |–create-structure-diff.png
| | |–create-structure.png
| | |–dentist.png
| | |–designProblem1.png
| | |–designProblem2.png
| | |–diff_structures.png
| | |–doc-ui.png
| | |–er-diagram.png
| | |–generate-structure.png
| | |–generate-structure1.png
| | |–generate-structure2.png
| | |–generateStructureSEQ.png
| | |–generatedJSON.png
| | |–goGraph.png
| | |–gridWeb.png
| | |–homepage.png
| | |–json.png
| | |–limited-cols.png
| | |–login.png
| | |–m-n.png
| | |–middleware.png
| | |–migration.png
| | |–mvc.png
| | |–numberOfWebsites.png
| | |–orm.png
| | |–printedGeneratedStructure.png
| | |–questions.png
| | |–recur-ele.png
| | |–register.png
| | |–registerAPI-token.png
| | |–requestAPI.png
| | |–reset-pass.png
| | |–text-align.png
| | |–token.png
| | |–tokens.png
| | |–transferData.jpg

| | |–xml.png
| | |–zav_prace1.png
| | |–zav_prace2.png
| |–implementacia.tex
| |–instalacia.tex
| |–master-thesis.log
| |–master-thesis.pdf
| |–master-thesis.ref
| |–master-thesis.tex
| |–mybase.bbl
| |–mybase.bib
| |–navrh.tex
| |–prilohy.tex
| |–resers.tex
| |–technika-bf.tfm
| |–technika-bi.tfm
| |–technika-bk.tfm
| |–technika-bkit.tfm
| |–technika-it.tfm
| |–technika.tfm
| |–testovanie.tex
| |–uvod.tex
| |–xl2tech.enc
| |–zav_prace.pdf
| |–zaver.tex