Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

DIPLOMOVÁ PRÁCE

Rozšířená webová analytika

Autor: Bc. Jan Bednář

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Karel Mls, Ph.D.

Hradec Králové, 2016

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a uvedl jsem všechny použité prameny a literaturu.

V Hradci Králové dne 18. srpna 2016 Jan Bednář

Poděkování

Rád bych poděkoval vedoucímu práce za jeho odborné vedení, podporu a věcné připomínky, které přispěly ke vzniku této diplomové práce.

Anotace

Práce se zabývá webovou analytikou se zaměřením na uživatelské interakce s webovou stránkou. Mapuje existující systémy a historický vývoj řešení. Hlavní část diplomové práce je potom tvořena návrhem a implementací systému. Systém je navržen tak, aby umožňoval integraci s Google Analytics. Ke svému účelu používá JavaScript, Java EE, Wildfly, WebSocket, Hibernate, Infinispan a Apache Camel. Součástí práce jsou ukázky použití. Implementované řešení je na základě reálných dat návštěvníků otestováno.

Annotation

The work deals with web analytics with a focus on user interaction with the website. Mapping the existing systems and historical development of solutions. The main part of the thesis is then formed the design and implementation of the system. The system is designed to allow integration with Google Analytics. Its purpose uses JavaScript, Java EE, Wildfly, WebSocket, Hibernate, Apache Camel and Infinispan. The thesis includes demonstration of usages. The implemented solution is tested on real data collected from visitors.

Obsah

1 Úvod

V současnosti je na internetu dostupných velmi mnoho služeb a čím více má uživatel možností výběru, tím více je třeba dbát vedle obsahové stránky také na uživatelský komfort při používání webu. Jedním ze způsobu, jak toho docílit, je odsunutí málo používaných částí webu a naopak vystavení částí, které uživatel používá nejčastěji. Další možností je sledování chování uživatele a vyhodnocení logických chyb, kterých se dopouští. V případě, že uživatelé ve větší míře klikají na prvky webu, které po kliknutí nevyvolají žádnou akci, může jít o chybu návrhu, kdy některý z prvků má vzhled tlačítka nebo odkazu. Právě tímto způsobem optimalizace webu se zabývá tato práce. Díky těmto údajům je potom možné odhadnout, který obsah je pro uživatele nejhodnotnější a případně změnit rozmístění tak, aby uživatelům nejvíce vyhovovalo.

Autor se v práci dále věnuje alternativnímu způsobu identifikace návštěvníka na základě otisku prohlížeče.

1.1 Cíl a metodika práce

Hlavním cílem této práce je navržení systému pro sledování uživatelských interakcí na webových stránkách. Tento systém má za úkol zaznamenání všech akcí, které uživatel na webu vykoná. Data bude možné využívat jak samostatně, tak spárovat s daty, která naměřil Google Analytics.

Cílem teoretické části je zmapování možných řešení tohoto problému. Aby toho bylo docíleno, jsou nejprve definovány důvody, které vedou majitele webu k měření interakcí a následně jsou prozkoumány existující řešení měření interakce uživatelů s webovými stránkami.

Cílem práce není grafická prezentace dat a protože je systém určen pro pokročilé uživatele, kteří si jej nainstalují na vlastní server, zvolil autor cestu přímého přístupu k datům přes relační databázi. Tento přístup umožňuje velmi flexibilní možnost nadefinovat i méně běžné vztahy, které existující nástroje nesledují. Součástí praktické části jsou ukázky SQL dotazů, které vedou ke zjištění vybraných údajů. Zároveň je v práci popsána struktura databáze tak, aby měl uživatel tohoto systému možnost napsat dotazy vlastní. Praktickou část tvoří návrh řešení4, kde jsou popsány jednotlivé použité technologie. Součástí praktické části je i kapitola implementace 5, která obsahuje kon-

krétní použití technologií, popsaných v kapitole 4 včetně ukázek kódu.

Práce také obsahuje vlastní řešení pro identifikaci návštěvníků za použití otisku prohlížeče. Tato metoda je v závěru práce otestována a dochází k rozhodnutí o vhodnosti či nevhodnosti řešení v porovnání s Google Analytics.

1.2 Postup a předpoklady práce

Práce předpokládá pokročilou znalost programovacího jazyku Java a JavaScript. Pro pochopení práce a používání aplikace je nutná znalost dotazovacího jazyku SQL. Současně je předpokládána čtenářova orientace v oblasti webové analytiky. Praktická část se věnuje převážně samotné implementaci a technologie jsou popsány spíše stručně. Pro hlubší seznámení s použitými technologiemi jsou poskytnuty reference na doprovodnou literaturu.

Při vývoji aplikace byly použity následující softwarové technologie:

- Javascript: Programovací jazyk pro implementaci interaktivních prvků stránky.
- WebSocket: Protokol umožňující obousměrnou komunikaci ve webových aplikacích.
- Apache Camel: Integrační framework, implementující Java EE patterny.
- Wildfly: Java EE kontejner, vyvíjený firmou RedHat.
- MariaDB: Relační databáze, nástupce MySQL. Dříve open source projekt, nyní vyvíjený pod záštitou Oracle.

2 Rešerše tématu

Přístupnost webu je důležitý aspekt návrhu webové stránky, který přímo ovlivňuje míru vracejících se návštěvníků. Proto je pro dobrou přístupnost nutné testování designu na uživatelích. Nepřehledné stránky mohou znamenat ztrátu stálého návštěvníka, což se projeví i finančními ztrátami, kvůli nutnosti vyšších investic do propagace. Následky špatně přístupného webu lze změřit téměř každým nástrojem pro webovou analytiku, pomocí ukazatele míry okamžitého opuštění a procentuálního rozložení vracejících se návštěvníků. Příčiny špatně přístupného webu je však často velmi složité identifikovat s daty, které analytické nástroje poskytují.

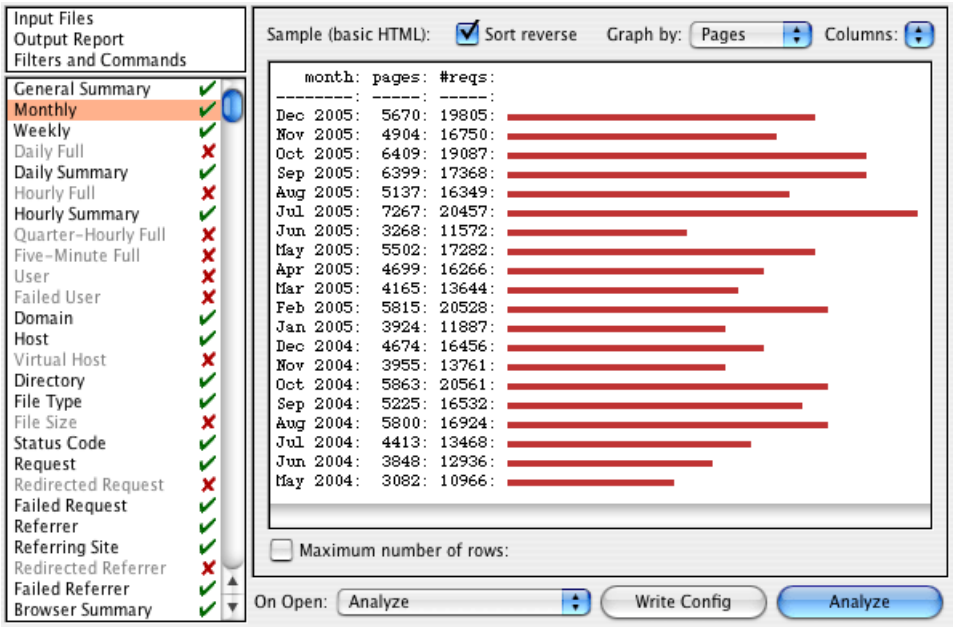
2.1 Historie

2.1.1 Zpracování server logů

První pokusy o webovou analytiku započaly v roce 1993. V té době byly ještě všechny webové stránky statické, bez dynamických prvků a analytické nástroje si tedy vystačily s pojmem „Hit“. JavaScript, ani jiný scriptovací jazyk do prohlížeče neexistoval, proto byly zdrojem informací logy na serveru. Ve chvíli, kdy prohlížeč odeslal požadavek, objevila se informace v textové podobě v logu na serveru a speciální nástroje tyto záznamy počítaly. Nejznámějším z nástrojů byl v té době WebTrends, později i Analog a AWStats.

Analog

Do roku 1995 neexistoval žádný nástroj, jehož provoz by byl zdarma. To se změnilo 14. června 1995 s příchodem nástroje Analog, který vyvinul Dr. Stephen Turner. Pracoval také na principu analýzy logů, ale tato data využíval mnohem efektivněji a kromě záznamu o Hitu zpracovával i informace z HTTP hlaviček a proto bylo možné reportovat i použitý prohlížeč, operační systém a jazyk návštěvníka. Výstupem byly tabulky a grafy vygenerované v HTML. Analog byl konzolová aplikace a reporty se generovaly pomocí příkazů. Pro usnadnění práce byl vytvořen nástroj Analog Helper, který poskytoval GUI 2.1. Vývojáři nikdy neposkytli informace o počtu stažení, ale podle



Obrázek 2.1: Ukázka Analog Helper při generování měsíčního reportu. Převzato z [36]

průzkumu[25], který provedl GVU[1] byl v roce 1998 nejpoužívanějším nástrojem. Žádné další průzkumy neproběhly, proto není možné stanovit vývoj v dalších letech. Poslední úprava, která umožnila detekci Windows 10 proběhla 28. června 2015.

WebTrends

WebTrends byl založen v roce 1993 a šlo o první společnost, která se začala komerčně věnovat webové analytice. V době vzniku používala pro sběr dat analýzu logů s podobným rozsahem jako Analog. Společnost se dokázala velmi dobře přizpůsobit rychle měnícímu se trhu a rychle implementovat nové technologie.

Oproti Analogu, který byl distribuován jako aplikace pro PC, fungoval WebTrends na principu SaaS (Service as a Service). To vývojářům umožnilo rychle implementovat nové způsoby sběru informací, které ve standalone řešení nejsou možné. Během prvního roku umožnili sběr dat pomocí GIF souboru, který byl umístěn na jejich serveru. Protože při načtení tohoto obrázku vzniká podobný požadavek jako při zobrazení stránky a odesílají se i podobné HTTP hlavičky, umožnilo jim to tato data uchovávat a analyzovat a tak vznikl jeden z prvních sběrných serverů, který se obešel bez analýzy logů zákazníka.

[1]Graphic, Visualization, & Usability Center

Zpracování server logů v současnosti

Protože tento způsob sběru informací nevyžaduje úpravu zdrojových kódů, používá se i v současnosti. Kromě Analogu, který zpracovává log v době nahrání, existují i nástroje Webalizer, W3Perl, Piwik a AWStats, které jsou spuštěné jako služba na serveru a zpracovávají záznamy v reálném čase. Metoda zpracování server logů nese několik problémů, které způsobují nepřesnost.

Prvním z problémů je cache. Někteří poskytovatelé připojení implementovali cache, která při prvním požadavku uloží stránku a při dalším požadavku se prohlížeči vrátí stránka z cache ISP [2]. Proto nedochází k požadavku na server, který vkládá záznamy do logu a o návštěvě neexistuje informace. Tento problém se aktuálně vyskytuje pouze u statických stránek, pro dynamické stránky se cache u ISP nepoužívá.

Dalším problémem jsou crawlery, weboví roboti, kteří procházejí webové stránky a hledají informace. Problém se nedotýká crawlerů od vyhledávacích společností, protože ti se během požadavku identifikují jako speciální prohlížeč a lze tyto návštěvy odfiltrovat. Odfiltrovat ale nelze crawlery, které se identifikují jako běžný prohlížeč. Tyto crawlery provozují především jednotlivci při hledání bezpečnostních chyb na webu nebo serveru.

2.1.2 Počítadla návštěvnosti

V roce 1996 již některé větší společnosti používaly pro získání informací o návštěvnících analýzu logů. Pro porozumění datům byl zapotřebí odborník, který je vhodně interpretoval. Zároveň šlo o zdlouhavý proces, který vyžadoval přístup k logům serveru. Drobnější webmasteři neměli ani potřebné znalosti, ani přístup k logům. Začaly proto vznikat počítadla návštěvnosti, která zobrazovala počet zobrazení, nebo počet unikátních návštěvníků. Tato počítadla se vkládají do webové stránky jako obrázek, který se vygeneruje na serveru počítadla. Server počítadla má v době vykreslování k dispozici IP adresu, podle které je schopen určit, zda z této IP adresy proběhla nějaká návštěva a pokud ne, vykreslí obrázek s číslem o jedno vyšším než předchozí.

Počítadla návštěvnosti v současnosti

Implementace byla velmi jednoduchá a služeb, které poskytovaly počítadla přibývalo velmi rychle. Dalším důvodem, který způsobil tak velký počet počítadel bylo budování zpětných odkazů. V té době bylo jedno z hlavních kritérií pro umístění na předních pozicích vyhledávání počet odkazů, které odkazují na web. Jak uvádí Matt Cutts [14] odborník na SEO[3] v interview pro blogera Erica Enge, provozovatelé počítadel po-

[2] Internet service provider - Poskytovatel internetového připojení
[3] Search Engine Optimization - Optimalizace pro vyhledávače

Obrázek 2.2: Ukázka vzhledu počítadla návštěvnosti. Získáno z [22]

užívali nejrůznější techniky pro vložení neviditelného odkazu do počítadla. Proto pokaždé, když webmaster použil počítadlo, zvýšil se provozovateli služby počet zpětných odkazů a tím i umístění ve vyhledávání. Počítadel začalo ubývat až v roce 2011, kdy Google začal bojovat proti nekvalitním zpětným odkazům a aplikoval algoritmus Google Panda [35]. Google Panda nastavil nekvalitním zpětným odkazům nulovou váhu a provoz počítadla se přestal vyplácet.

V současnosti se počítadla používají na webech, kde nelze použít JavaScript, především svoje uplatnění najdou v některých blogovacích systémech či příspěvcích internetových fór.

2.1.3 Vznik JavaScriptu

JavaScript vznikl v roce 1995, kdy společnost Netscape najala Marca Anderseena s požadavkem na vytvoření scriptovacího jazyka pro prohlížeč. Aby si Netscape nápad ochránil, potřeboval rychle prototyp a Marc Anderseen jej připravil během deseti dnů. Tehdy byl ještě vyvíjen pod označením „Mocha“. Následně i další společnosti začaly vyvíjet vlastní implementace a tato nejednotnost rozhranní způsobila nepříliš velkou oblibu u vývojářů. V červenci 1997 byl standardizován asociací ECMA[4] a v srpnu 1998 ISO[5]. Jednotný standard byl pojmenován ECMAScript a jednotná implementace v prohlížečích otevřela dveře širšímu použití i v oblasti webové analytiky. JavaScript způsobil ve webové analytice revoluci. Již nebylo nutné spoléhat se pouze na omezené informace z HTTP požadavku, data bylo možné obohatit o téměř libovolný údaj, který JavaScript API poskytuje.

[4]Nezisková organizace spravující normu ECMAScript - `http://www.ecma-international.org/`
[5]International Organization for Standardization - `http://www.iso.org/iso/home.html`

Použitím JavaScriptu pro sběr informací již nebylo nutné řešit problémy, které se dotýkají zpracovávání logů. ISP Cache nemá na komunikaci vliv, spojení se otevře vždy bez ohledu na zapnutou cache. Vyřešil se i problém s webovými crawlery, protože většina z nich JavaScript nepodporuje.

Coremetrics

Společnost Coremetrics vznikla v roce 1999 a vyvinula „Coremetrics Analytics“, jeden z prvních nástrojů, který pro webovou analytiku využíval JavaScript. V roce 2010 proběhla akvizice společností IBM a nyní je nástroj prezentován pod názvem „IBM Digital Analytics“.

2.1.4 Založení WAA

Do roku 2004 neexistovala žádná jednotná terminologie, týkající se webové analytiky a to byl hlavní důvod, proč Bryan Eisenberg, Andrew Edwans a Jim Sterne založili WAA[6]. Za cíl si kladli nejen sjednocení terminologie, ale i rozšíření webové analytiky do obecného povědomí. Po založení vydali první verzi dokumentu „Web Analytics Definitions“ [3] ze kterého se stal uznávaný standard. Asociace byla roku 2011 přejmenována na „Digital Analytics Association“, aby nebyli svým názvem vázání pouze na webovou analytiku a pokrývali větší oblast analytiky dat [2]. I v současnosti udržují standard, sledují vývoj a pořádají školení, setkání a certifikace na toto téma.

2.1.5 Spuštění Google Analytics

V roce 2005 Google koupil společnost Urchin, která se zabývala webovou analytikou a vyvinula nástroj pro analýzu logů „Urchin on Demand“. Google ve vývoji pokračoval a nabyté zkušenosti investoval do vývoje vlastního řešení, které pojmenoval Google Analytics. Google spustil první veřejnou verzi v listopadu 2005 a o registrace byl tak velký zájem, že po pouhém týdnu pozastavil registrace [28]. Následně postupně navyšoval výkon serverů a registrace opět zprovoznil, ale pouze na pozvánky, které byly generovány náhodně. K plnému spuštění došlo v srpnu 2006. Do roku 2012 ještě probíhal souběžně i vývoj původního programu Urchin, který byl nabízen komerčně, ale 28. března 2012 byl další vývoj a prodej produktu ukončen. V říjnu téhož roku proběhla velká přeměna z Google Analytics na Google Universal Analytics, která mimo jiné umožnila definici uživatelských metrik, dimenzí a událostí. Tím bylo umožněno použití nejen na webu, ale i ve webových aplikacích a dokonce i mobilních aplikacích a hrách. Podle nástroje w3techs.com, který pravidelně monitoruje 10 milionů nejnavště-

[6] Web Analytics Association - Acociace sdružující odborníky na webovou analytiku

vovanějších webů, je Google Analytics použit na 54,5 procentech všech monitorovaných webů, to je 83,3 procent z webů, kde je použit nějaký analytický nástroj. [38]

2.1.6 Spuštění ClickTale

V roce 2006 je spuštěn Clicktale, který umožňuje pomocí JavaScriptu zaznamenávat uživatelské interakce jako pohyb myši, kliknutí a scrollování. Tato data jsou odesílána na server a události lze sledovat v reálném čase. ClickTale byl první nástroj, který se zaměřoval na analytiku uživatelských interakcí. [8]

2.2 Druhy webové analytiky

Webová analytika se vyvíjí dvěma směry. První se spoléhá především na data, která poskytnou samotní návštěvníci, či poskytovatelé internetu a označuje se jako off-site analytika. Druhý ze směrů využívá sběr dat přímo na webových stránkách za pomoci měřícího kódu a označuje se jako on-site analytika.

2.2.1 Off-site analytika

Tento typ analytiky nevyžaduje přístup k měřeným webům. Využívá se tedy především k analýze konkurence, kdy zadavatel nemá možnost úpravy zdrojových kódů měřeného webu. Pro sběr dat se používá více zdrojů a níže jsou popsány ty nejčastější.

Poskytovatelé připojení

Někteří poskytovatelé podepsali smlouvu s některým z agregátorů dat, který částečně anonymizovaná data zpracovává. Tato data jsou z legislativních důvodů vždy anonymizována o údaje, které bv mohly jednoznačně identifikovat návštěvníka. Chybí zde tedy všechny metriky, které se týkají demografických údajů. David Cancel na konferenci „At the Open Data 2007“ uvedl, že data byla od ISP odkupována za cenu kolem 40¢ za uživatele a měsíc. Největší z agregátorů je hitwise.com, který v době psaní tohoto článku disponuje daty od 25 milionů uživatelů.

Panely

Zdrojem panelových dat jsou doplňky do prohlížečů, či jiný software, který zaznamenává veškerou aktivitu na internetu. Výhodou tohoto řešení je vysoká úroveň detailů o návštěvě. Uživatel, který je součástí panelového průzkumu při registraci uvádí kromě věku, pohlaví, náboženství i roční příjem, profesi a bydliště. Nevýhodou je nízká uživatelská základna, proto vznikají chyby, které jsou způsobeny nedostatkem dat. Dal-

ším problémem panelových průzkumů je fakt, že uživatel dostává odměnu, která se odvíjí od času aktivního používání prohlížeče. To způsobilo, že uživatelé začali používat různé nástroje a makra, která tuto činnost automatizovaly. Tyto podvody vyústily v různé typy ověření, které měly za úkol v náhodných intervalech ověřovat, že u počítače sedí reálná osoba. Tyto agresivní techniky odradily i čestné uživatele a tím se problém s malou uživatelskou základnou ještě prohloubil[9]. V době psaní této práce je největší poskytovatel panelových průzkumů comscore.com, který hlásí 2 miliony uživatelů.

Proxy

Další možné řešení nastínili Richard Atterer, Monika Wnuk a Albrecht Schmidt ve své publikaci „Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction"[4] , kteří pro testování použili HTTP proxy. Úkolem HTTP proxy bylo vložení JavaScript kódu do měřené stránky během přenosu mezi zdrojovým a cílovým prvkem sítě a logování aktivit uživatele. Toto řešení je vhodné, pokud při testování není přístup ke zdrojovému kódu webu, ale existuje přístup k počítačům, nebo síti, kde bude přítomný sledovaný návštěvník. Autorovi se nepodařilo dohledat službu, která by tento typ analýzy poskytovala, dohledal pouze vědecké články, popisující princip. Toto řešení se potýká i s dalšími problémy. Například nelze použít na webech, kde se používá SSL šifrování, protože změnu kódu vyhodnotí většina současných prohlížečů jako útok typu man-in-the-middle a znemožní na takovéto stránky přístup.

Statistické metody

Hlavní výhoda, která je zároveň i nevýhodou je ta, že statistické metody analytiky nepoužívají reálné návštěvníky. Místo toho byly vytvořeny algoritmy, které na základě dat od dobrovolníků simulují chování běžného uživatele. Použité algoritmy vycházejí z výzkumu, který provedli Vidhya Novalpkkam a Elizabeth Churchill[32]. Tento přístup je vhodný, pokud je potřeba otestovat použitelnost před nasazením na produkční servery. Nevýhodou je to, že se nejedná o reálné statistiky, nýbrž o predikci, proto mnoho problémů nedokáže identifikovat a některé vyhodnotí nesprávně. Tyto metody se používají pro testování použitelnosti webu a nejznámějším nástrojem je feng-gui.com

2.2.2 On-site analytika

On-site analytika využívá pro sběr informací měřící kód, který se vkládá přímo do zdrojového kódu stránky. To znamená, že je nutné vlastnit přístup k aplikaci, do které je měřící kód vkládán. Protože principy on-site analytiky jsou pro tuto práci klíčové, věnuje se tomuto tématu celá kapitola 3.

3 On-site analytika

Tento druh analytiky se používá nejčastěji a slouží provozovatelům webu ke zjištění přesných informací o návštěvnosti jejich webu a chování návštěvníků. Pro on-site analytiku je důležité, aby měl zadavatel měření přístup k webu a to ze dvou důvodů. Prvním důvodem je nutnost vložení měřícího kódu do zdrojového kódu webu. Druhým důvodem je povinnost vyžadovat souhlas návštěvníka, pokud je k identifikaci používán cookie soubor. Tuto povinnost ukládá směrnice Evropské Unie 2009/136/EC [15].

3.1 Definice pojmů

On-site analytika vychází z pojmů, které definuje asociace Web Analytics Association. V této kapitole jsou stručně popsány nejdůležitější pojmy on-site analytiky. Pro podrobnější informace autor doporučuje čtenáři dokument Web Analytics Definitions [3] zpracovaný výše zmíněnou asociací, ze kterého také tato kapitola čerpá.

3.1.1 Základní pojmy

- Stránka: Analyticky definovatelná část obsahu. Je definována pomocí webové adresy a může obsahovat dynamické prvky.
- Web: Soubor stránek na jedné doméně či subdoméně.
- Zobrazení stránky: Počet zobrazení konkrétní stránky.
- Návštěva: Skládá se z jednoho nebo více požadavků na stránku. Začíná prvním požadavkem konkrétního návštěvníka a končí, pokud po definovaném čase není zaznamenán žádný požadavek.
- Unikátní návštěvník: Osoba nebo prohlížeč, který je unikátní ve zvoleném období.
- Nový návštěvník: Osoba nebo prohlížeč, který během zvoleného časového období přistupuje na web poprvé.

- Opakovaný návštěvník: Osoba nebo prohlížeč, který během zvoleného časového období přistupuje na web opakovaně. To znamená, že vygeneroval 2 a více návštěv.

3.1.2 Charakteristiky návštěvy

- Vstupní stránka: Adresa stránky, která byla zobrazena jako první během návštěvy.
- Výstupní stránka: Adresa stránky, která byla zobrazena jako poslední během návštěvy.
- Doba návštěvy: Časový rozdíl mezi poslední a první uživatelskou interakcí.
- Odkazující stránka (referrer): URL stránky, která způsobila nové zobrazení. To znamená, že na ní byl umístěn odkaz vedoucí na web.
  - Interní referrer: Na stránku sledovaného webu bylo přistoupeno z webu samého v rámci jedné návštěvy.
  - Externí referrer: Na stránku bylo přistoupeno z URL mimo sledovaný web.
  - Vyhledávací referrer: Na stránku bylo přistoupeno z některého webového vyhledávače.
- Počet zobrazení na stránku
- Počet prokliků: Počet kliknutí na konkrétní odkaz
- Míra prokliku: Počet prokliků vydělený počtem zobrazení konkrétní URL

3.1.3 Charakteristiky kontextu

- Poměr opuštění stránky: Počet případů, kdy byla stránka označena jako výstupní, vydělený celkovým počtem zobrazení stránky.
- Jednostránkové návštěvy: Počet návštěv, během kterých bylo zaznamenáno pouze jedno zobrazení stránky.
- Míra opuštění stránky: Počet jednostránkových návštěv vydělený počtem vstupních stránek.

3.1.4 Události

Dokument Web Analytics Definitions [3] definuje událost jako jakoukoliv akci, která je zaznamenána v prohlížeči a má přiřazený konkrétní čas. Protože standard nedefinuje rozsah událostí, které lze měřit, jsou v tomto ohledu existující nástroje velmi rozdílné.

Některé nástroje umožňují definovat vlastní události, některé nikoliv a některé měří události týkající se myši a klávesnice.

3.2 Existující on-site nástroje

V současnosti existují analytické nástroje, které umožňují i sledování uživatelských interakcí. V následujících odstavcích autor popisuje některé vybrané nástroje.

3.2.1 Google Analytics

Google Analytics je velmi mocný nástroj pro webovou analytiku. Bohužel měří pouze omezeně události během návštěvy. Ve výchozím stavu neměří tedy události, které nastanou po načtení stránky. Google Analytics umožňuje zaregistrovat vlastní události a pomocí JavaScriptu tyto události odeslat na vyhodnocovací server. Ve statistikách lze potom tyto události filtrovat a zobrazit v tabulce a grafu. Nelze však nastavit žádný vztah mezi nimi, proto není vhodný k podrobnému sledování uživatelských interakcí.

3.2.2 Mouseflow

Mouseflow je analytický nástroj, umožňující přehrávání chování uživatele na webu, zobrazení heatmapy kliknutí a scrollování. Zároveň umožňuje zobrazení reportů, které jsou však na nižší úrovni, než předchozí zmiňovaný Google Analytics. Funguje na principu, který byl představen v článku „Usability tool for analysis of web designs using mouse tracks“[1].

Feedback

Feedback, neboli hodnocení uživatele, využívá toho, že se na webu zobrazí formulář pro hodnocení webu. K tomuto účelu existuje velké množství nástrojů. U tohoto způsobu je dosahováno velmi velké přesnosti, protože uživatel hodnotí web slovně. Nevýhodou je, že tato data nelze vyhodnotit automaticky a zároveň sběr těchto dat trvá velmi dlouho, proto je tento způsob spíše doplňkový k některému jinému systému měření chování uživatelů na webu.

3.2.3 Loop 11

Loop 11 je nástroj, který ke svému účelu potřebuje interakci s reálným uživatelem webu. Měření probíhá tak, že správce webu nadefinuje některé úkoly, které se potom uživatel, který s tímto způsobem testování souhlasil, snaží splnit. V době jeho testování je měřen pohyb myší a přechod mezi jednotlivými stránkami. Díky těmto datům

lze přesně identifikovat složitost nadefinovaných úkolů, ale vyžaduje loajální návštěvníky, kteří jsou ochotni zúčastnit se průzkumu.

3.2.4 Track:js

Cílem tohoto nástroje je zachytávání a prezentování chyb, které nastanou při vykonávání JavaScriptového kódu. Poskytuje přehledné informace o uživatelských interakcích, které chybě předcházely. Je vhodný pro hledání příčin, proč chyba nastala. Neposkytuje však možnosti pokročilejšího zkoumání dalších ukazatelů, které mohou rozhodnout o závažnosti chyby.

3.2.5 Zhodnocení existujících systémů

Všechny výše uvedené nástroje jsou vhodné pro analytiku chování uživatelů na webu, nicméně problémem je, že rozsah dat není kompletní a v případě, že vznikne nutnost používat více systémů zároveň, je nutné tato data párovat ručně, případně použít některý z externích nástrojů.

Pro webovou analytiku je Google Analytics vhodný nástroj, bohužel nepracuje s údaji uživatelských interakcí, na které se práce zaměřuje. Proto budou v této práci také popsány možnosti integrace nově vytvořeného nástroje s Google Analytics a vybraný způsob implementován.

S webovou použitelností také úzce souvisí sledování chyb v JavaScriptu. Z pohledu autora začíná být v mnoha webových aplikací více JavaScriptového, než HTML kódu a proto je nutné testovat i použitelnost dynamických prvků. Chyba v JavaScriptu je zákeřná, protože taková chyba se může projevovat jen pro kombinaci určitých prohlížečů a operačních systémů a objeví se častokrát až po nasazení na produkci. Existující nástroje jsou vhodné pro odhalování chyb, ale neposkytují informace o důsledcích chyby, které mohou být indikátorem k určení její priority. Informací, která může určit prioritu chyby je například vztah mezi konkrétní chybou a mírou okamžitého opuštění.

Dalším problémem, kterým existující nástroje trpí je nemožnost získat surová data. Vždy jde o data agregovaná a v případě, že analytika zajímají méně běžné vztahy, nemá možnost tyto vztahy nadefinovat.

3.3 Identifikace návštěvníka

Pro webovou analytiku je nutné přesně identifikovat návštěvníka tak, aby bylo možné rozhodnout, zda jde o novou, či opakovanou návštěvu. K tomuto účelu se využívají cookies a otisky prohlížeče.

3.3.1 Cookies

Cookies jsou malé soubory, obsahující textové informace, které webové stránky odkládají do úložiště prohlížeče a odesílají se společně s každým HTTP požadavkem[33]. Tento způsob identifikace návštěvníka využívá většina systémů pro webovou analýzu. Do cookie je typicky během první návštěvy uložen náhodný identifikátor, který se během opakované návštěvy přečte a odešle na analytický server[24].

Druhy Cookie

Cookies se rozdělují z pohledu analytiky podle domény a podle platnosti. Rozdělení cookies definuje RFC2965[27].

Z pohledu domény lze cookies rozdělit na first-party a third-party cookies. First-party cookie platí pouze pro stránku, která jí vystavila. Tyto cookies lze nastavit a přečíst pouze z domény, ze které byly vystaveny a externí scripty k ním nemají přístup. Opačným typem jsou third-party cookie, které může nastavit jakýkoliv externí zdroj a následně je opět přečíst. Toho využívají reklamní systémy, které tak mohou sdílet informace o uživateli napříč všemi weby, kde je daný reklamní systém použitý. Ze stejného důvodu našly cookies uplatnění v nástrojích určených pro webovou analytiku.

Z pohledu platnosti lze cookies dělit na dočasné a trvalé. Dočasné cookies mají krátkou platnost a to pouze do zavření prohlížeče. Neukládají se jako soubor na disk zařízení, ale jsou udržovány pouze v operační paměti. Využívají se především k uchování přihlášení na webových stránkách. Oproti tomu trvalé cookies obsahují pole o časové platnosti ve vteřinách. Protože podle specifikace může toto pole nabývat až hodnoty 2^{31}, je možné nastavit platnost až na 68 let. V analytice se vzhledem k vlastnostem používají trvalé cookies.

Vhodnost cookies pro analytiku

Cookie je nejpoužívanější způsob identifikace návštěvníka. Protože cookies jsou součástí specifikace HTTP [27], není nutné řešit rozdílné implementace v prohlížečích. Další výhodou je minimální náročnost generování identifikátoru.

Zásadní nevýhodou je výhradní přístup uživatele prohlížeče k cookies souborům a tím i možnost jejich editace, mazání, či úplného zakázání v prohlížeči. To je pro analytiku problém, protože po smazání cookie souboru je vytvořen nový, díky kterému může být stejný návštěvník identifikován jako nový.[24] Podle výzkumu, který provedla Amy Weinberger v Austrálii, maže 28 procent internetových uživatelů first-party cookies alespoň jednou měsíčně a third-party cookies dokonce 37 procent. Dále uvádí, že u systémů, které používaly cookie jako jediný způsob identifikace, došlo proto k zjištění 2,7krát více unikátních návštěvníků oproti skutečnosti. [39]. Jak uvádí Eric Fettman

ve svém článku[16], tento problém se dotýká i Google Analytics, který pro identifikaci uživatelů cookies používá. Zároveň v článku uvádí, že výsledky identifikace ovlivňuje i funkce anonymního prohlížení, která je dostupná ve všech moderních prohlížečích.

3.3.2 Otisk zařízení

Princip otisku zařízení, označovaný jako „Device fingerprinting“ využívá informace, které zpřístupňuje JavaScript API a informace, které se odesílají společně s HTTP požadavkem, na základě kterých je určena identita zařízení. Otisk zařízení se skládá z mnoha proměnných, které v kombinaci s ostatními mohou jednoznačně identifikovat prohlížeč. Čtenář si může sám vyzkoušet přesnost, s jakou jej lze identifikovat na webu `https://panopticlick.eff.org`. V následujících kapitolách jsou popsány jednotlivé zdroje informací, které se využívají pro identifikaci návštěvníka. Jednotlivé způsoby otisků se často pro zvýšení přesnosti kombinují. Přesnost identifikace se zvyšuje s počtem proměnných, které jsou do otisku zahrnuty. Zároveň se však přidáním některých proměnných snižují možnosti dlouhodobé identifikace, protože se mohou v čase měnit. Úspěšná implementace otisku zařízení je proto dána vhodným výběrem dostatečného počtu prvků, které se v čase téměř nemění. [31]

HTTP požadavek

Při navázání HTTP spojení odesílá prohlížeč HTTP hlavičky, které obsahují informace o zdrojovém zařízení. HTTP hlaviček odesílá prohlížeč desítky, ale pro identifikaci návštěvníka mají vypovídající hodnotu pouze některé:

- User-Agent: Hlavička, která identifikuje použitý operační systém a prohlížeč včetně použité verze.
- Accept-Language: Obsahuje informace o jazyku použitém v operačním systému.
- Accept-Charset: Preferované kódování znaků klientského zařízení.
- Accept-Encoding: Informace o podporovaných způsobech komprese přenášených dat.

Kromě HTTP hlaviček má webový server k dispozici IP adresu zařízení, kterou lze využít pro zpřesnění identifikace. Tuto techniku lze tedy použít i v případě, že je v prohlížeči vypnutý JavaScript, ale sama o sobě nedokáže dostatečně přesně identifikovat návštěvníka.

JavaScript

JavaScript API poskytuje přístup k velkému množství proměnných, které obsahují informace o použitém prohlížeči a operačním systému. Jednou z nejvíce rozšířených implementací pro výpočet otisku prohlížeče je „Valve/fingerprintjs2“, používající tyto proměnné[37]

- UserAgent: Identifikátor operačního systému a prohlížeče.
- Jazyk: Lokalizace prohlížeče
- Barevná hloubka: Počet bitů pro vykreslení barvy. Závisí na grafické kartě návštěvníka.
- Rozlišení obrazovky
- Podpora WebStorage API [1]
- Podpora IndexedDB API [2]
- Podpora lokální SQL databáze - Pravdivostní hodnota o podpoře metod pro přístup k lokální SQL databázi. Cílí na starší prohlížeče, protože tento typ databáze byl ve W3C specifikaci označen za zastaralý v roce 2010[21]
- Časová zóna
- Platforma operačního sytému
- CPU třída: Proměnná specifikující třídu procesoru. Možné hodnoty jsou x86, 68K, Alpha, PPC, Other.
- Volba DoNotTrack: Proměnná obsahující pravdivostní hodnotu udávající, zda je v nastavení prohlížeče zvolena možnost DoNotTrack [3]
- Seznam nainstalovaných fontů: Proměnná obsahující seznam všech nainstalovaných fontů. Seznam je nesetříděný, protože různé prohlížeče udávají tento seznam v různém pořadí a proto ponechání původního pořadí zvyšuje entropii.
- Seznam nainstalovaných pluginů prohlížeče
- AdBlock: Pravdivostní hodnota určující, zda je AdBlock nainstalovaný.
- Poměr fyzických a logických pixelů obrazovky: Proměnná použitá především u mobilních zařízení.

[1] `https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API`
[2] `https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API`
[3] `https://www.mozilla.org/cs/firefox/dnt/`

HTML5

S příchodem HTML5 se začal používat další prvek pro otisk zařízení, který je označován jako „Canvas fingerprinting“. HTML5 umožňuje vykreslení obrázků, které jsou definovány programově, nikoliv jako sekvence pixelů. K tomuto účelu se používá element canvas, obsahující definice vykreslených tvarů. Pro identifikaci uživatele je důležité, že vykreslený prvek se často nepatrně liší. Svoji roli hraje nejen verze prohlížeče a operačního systému, ale dokonce různé grafické karty a jejich ovladače produkují mírně rozdílný výsledek. Již vykreslený element lze konvertovat do hexadecimální podoby, reprezentující sekvenci pixelů, a v této podobě použít pro zpřesnění identifikace. Technika byla představena vědeckými pracovníky z University of California v práci „Pixel Perfect: Fingerprinting Canvas in HTML5“ [29].

Vhodnost otisku zařízení pro analytiku

Hlavní nevýhoda tohoto řešení je vysoká výpočetní náročnost pro zařízení. Na druhou stranu nepoužívá cookies, eliminuje tak všechny problémy představené v kapitole 3.3.1. Rozdílný výběr proměnných pro sestavení otisku má různou míru úspěšnosti. Výběr proměnných, který je představený v práci „How Unique Is Your Web Browser?“[13], uvádí na základě 470161 měření úspěšnost jednoznačné identifikace v 83,6 procentech případů. Dobrá implementace otisku zařízení tak musí vyvážit výběr proměnných tak, aby byl výpočet co možná nejrychlejší a proměnné co možná stálejší v čase.

3.4 Sledování interakcí

Nejpřesnější způsob, jak měřit zájem uživatele na webových stránkách je využití HW pro sledování pohybu očí. Tento způsob je využíván pro testování použitelnosti nových, či upravených grafických rozhraní. Výsledkem jsou potom informace, do jaké míry je úprava přívětivá pro uživatele. Tato metoda je velmi přesná, ale také nákladná. Je totiž nezbytný drahý HW pro sledování zorničky, ale také skupinka lidí, která je ochotná se danému testování podrobit. V roce 1999 si výzkumníci, používající tuto metodu poprvé všimli, že existuje souvislost mezi pohybem očí a pohybem myši. Výstup tohoto zkoumání je zpracován v článku [26]. Poté se v roce 2001 problému začali věnovat Mon-Chu Chen, John Anderson, a Myeong-Ho Sohn z Carnegie Mellon University. Na toto téma vydali několik prací[30][18][7][34] a jejich výzkum probíhá i v současnosti. Tématu se věnovali i Jeff Huang, Ryen W. White a Georg Busher, kteří ve své práci „User See, User Point“[23] vyslovili zjištění, že změna pozice kurzoru následuje 700ms po změně zaměření zorniček.

V této práci bude předpokládána korelace mezi pohybem myši a pohybem očí, která

existuje také podle výzkumu Vigneshe Raghunatha[5], kde naměřil průměrnou euklidovskou vzdálenost mezi sledovanými souřadnicemi a souřadnicemi kurzoru přibližně 3,7 palce a medián přibližně 3,4 palce.

Důvody pro měření chování uživatele na webu mohou být různé. Ten nejrozšířenější je testování použitelnosti, kdy je podle pohybu kurzoru možné odhadnout, zda je web pro konkrétního uživatele přehledný. Na základě těchto informací lze uspořádat rozložení informací na webu, aby byly poskytované informace nejefektivnější.

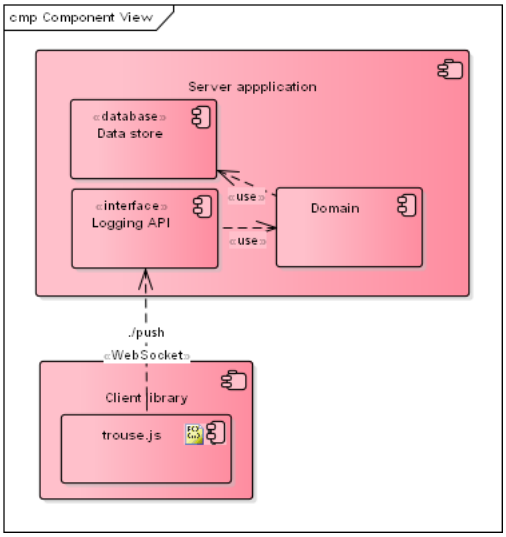
4 Návrh aplikace

Součástí této práce je aplikace, eliminující nedostatky řešení, představených v kapitole 3.2.

Nová aplikace se skládá ze dvou částí:

- Klientský modul: Modul odesílá na server informace o změně polohy kurzoru, scrollbaru, identifikátor a souřadnice objektu při kliknutí a další informace o používání webu.
- Serverový modul: Modul přijímá informace z předchozího modulu a po transformaci tato data ukládá do databáze.

Architektura systému je vyobrazena na obrázku 4.1.



Obrázek 4.1: Architektura systému

4.1 Databáze

Všechna data jsou ukládána do relační databáze MariaDB. Pro veškerou komunikaci mezi databází a jednotlivými moduly byl použit ORM framework Hibernate v kombinaci s Infinispan subsystémem, který zajišťuje cache a snižuje vytížení databáze. Data-

báze se skládá z dvou logických částí, stavové databáze a uživatelské databáze. Databáze je rozdělena na stavovou a uživatelskou z logického pohledu na data. Rozdělení slouží pro přehlednost, protože existuje cizí klíč z tabulky RECORDING_SESSION na tabulku SITE, není vhodné, aby byly tyto dvě části umístěny v jiných databázích.

4.1.1 MariaDB

MariaDB je relační databáze, která vychází ze zdrojových kódů MySQL. Vznikla v době odkupu MySQL firmou Oracle, kdy se vývojáři obávali, že Oracle změní původní GNU GPL licenci. MariaDB až do verze 5.6 obsahuje všechny funkcionality MySQL, další verze již nemají kompatibilitu zaručenou.

4.1.2 Struktura uživatelské databáze

Uživatelská databáze obsahuje informace o zaregistrovaných webech, jejich vazbu na uživatele a API klíč. Uchovávané informace slouží pro validaci požadavků z WebSocket kanálu a pro účely spárování API klíče ke konkrétnímu webu. Struktura uživatelské databáze je zobrazena na diagramu v příloze č. 2.

4.1.3 Struktura stavové databáze

Stavová databáze slouží pro uchování sledovaných uživatelských interakcí. Hlavní tabulkou je RECORDING_SESSION, kde je vytvořen záznam během otevření WebSocket spojení, tedy během načtení stránky. Všechny události, které odesílá klientský modul, obsahují vazbu na tuto tabulku a tím jsou uchovány informace o kontextu návštěvy. Struktura stavové databáze je zobrazena na diagramu v příloze č. 3.

4.1.4 Hibernate

Hibernate je ORM[1] framework, který slouží k mapování mezi Java entitami a tabulkami v relační databázi. Hibernate zajišťuje, že ke každému sloupci v tabulce lze přistoupit přes metodu na entitě a zároveň se postará o propagaci do databáze, když dojde k její změně. To programátorovi umožňuje jednodušší práci s daty, protože je odstíněn od SQL dotazování a umožňuje mu plně využívat principy OOP[2].

4.1.5 Infinispan

Infinispan je nástupce JBoss Cache a jde o NoSQL úložiště dat vyvíjené firmou RedHat. Protože vkládání dat do cache je velmi rychlé, začal se používat pro mnoho účelů. Lze

[1] Objektově relační mapování
[2] Objektově orientované programování

použít jako úložiště pro dočasná data, jako distribuovaná cache mezí více aplikacemi nebo pro komunikaci a sdílení úložiště mezi JVM[3]. Pro účely této práce je využíván v kombinaci s Hibernate jako second level cache. Ta snižuje nároky na výkon databáze, protože většina dat se uchovává v NoSQL a do databáze se propagují až v době, kdy je databázový server méně vytížen jinými SQL dotazy.

4.2 Klientský modul

Klientský modul zachytává změny stavu a odesílá je na server. Zachytávány jsou tyto události:

- Kliknutí myší
- Scrollování
- Změna velikosti okna prohlížeče
- Události klávesnice
- JavaScriptové chyby

4.2.1 Integrace s Google Analytics

Pro účely párování dat mezi nově vyvinutým nástrojem a Google Analytics je nutné, aby existovala jednotná proměnná, podle které se data spárují. V této kapitole jsou nejprve popsány základní metody fronty ga(), které lze k tomuto účelu použít, následně jsou popsána možná řešení a vybráno jedno nejvhodnější.

Fronta ga()

Tato kapitola čerpá z dokumentace analytics.js[11].

Hlavní částí knihovny analytics.js je funkce ga(), která zajišťuje odesílání všech požadavků na server Google Analytics.

Sledovací kód pro Google Analytics používá pro veškerou komunikaci se servery Google Analytics tuto frontu. Fronta je zaregistrována okamžitě, nicméně v čase registrace ještě nejsou načteny všechny závislosti nutné pro běh analytics.js. Všechny příkazy se tedy hromadí ve frontě a až po kompletním načtení webu dojde k jejich odeslání na servery Google Analytics. To vývojářům umožňuje odesílat požadavky okamžitě, bez nutnosti čekání na knihovnu, která se stahuje asynchronně.

Funkce fronty ga() přijímá jako první parametr akci, která bude odeslána na servery Google Analytics. Kompletní výčet akcí je uveden v dokumentaci, ty nejdůležitější jsou:

[3] Java Virtual Machine

- create - vytvoří objekt trackeru a zároveň načte všechny sledované údaje o prohlížeči, navštívené stránce a identifikaci návštěvníka.
- set - nastaví na tracker proměnnou, která se odesílá společně s událostmi.
- send - Odešle předdefinovanou, či uživatelsky definovanou událost.

Možná řešení

Autor uvažoval nad několika možnostmi řešení. Jak již bylo zmíněno, musí existovat proměnná, která bude uložena v obou systémech.

Nejjednodušší možností je odesílat na serverový modul při otevření spojení identifikaci uživatele, kterou lze získat z objektu ga(). Toto řešení nelze použít, protože je v rozporu s licenčními podmínkami Google Analytics, které výslovně zakazují odesílat identifikační údaje externím systémům.

Další možností je využít metody ga('create',..), která umožňuje vložit jako volitelný parametr userId jako identifikaci návštěvníka. To by však znamenalo nepoužívat identifikační algoritmy Google Analytics a nadefinovat si algoritmus vlastní. Zároveň by použití této metody znamenalo, že logika identifikace by byla prováděna pomocí JavaScriptu. Protože jde o poměrně náročnou operaci, rozhodl se autor s ohledem na komfort návštěvníků tuto metodu nepoužít a identifikaci provádět až následně v serverovém modulu. Protože tato práce obsahuje srovnání vlastního řešení pro identifikaci návštěvníků a Google Analytics bude brán jako referenční zdroj informací, mohla by být data přepsáním této proměnné znehodnocena.

Další možností je využití metody ga('set',..), která umožňuje nastavení vlastní proměnné ve formátu dimension<číslo dimenze>. Tato proměnná má definovatelný rozsah, který může nabývat těchto hodnot:

- Hit-level - Proměnné jsou odeslány pouze jednou při odeslání události, následně se smaže.
- Session-level - Proměnné se odesílají společně s každou událostí, která nastane během jedné návštěvy.
- User-level - Proměnné se uloží do cookies uživatele a odesílají se během každé jeho návštěvy konkrétního webu.

Klientský modul by po otevření spojení se serverovým modulem uložil do této proměnné sessionId, které se vygenerovalo v Serverovém modulu a díky tomu by bylo možné v Google Analytics data párovat podle dimenze. Protože se ale každá dimenze

musí korektně nastavit v administraci Google Analytics se správným číslem a správným rozsahem, znamenalo by to nutnost, aby administrátor webu před prvním použitím klientského modulu zasahoval i do konfigurace Google Analytics. Zároveň by chybná konfigurace rozsahu v Google Analytics způsobila znehodnocení těchto dat. Z toho důvodu se autor rozhodl toto řešení nepoužít.

Vybrané řešení

Řešením nedostatků, uvedených v předchozí kapitole 4.2.1 je použití metody ga('event'), která umožňuje odeslání libovolné události na servery Google Analytics bez nutnosti zásahu v administraci. Při odeslání vlastní události se jako parametr metody ga('event',..) nastavuje kategorie události, akce události a popis. Pro účel monitorování a integrace nově vytvořeného nástroje byly nadefinovány tři události, uvedené v tabulce 4.1.

Tabulka 4.1: Definované události klientského modulu

Kategorie	Akce	Popis
trouse	start	
trouse	failed	
trouse	loaded	sessionId

Událost start se odešle okamžitě po načtení klientského modulu, bez čekání na otevření spojení se serverovým modulem. Tato událost je tedy odeslána společně s každou návštěvou a informuje o tom, že prohlížeč návštěvníka klientský modul úspěšně inicializoval.

Událost failed je odeslána v případě, kdy prohlížeč návštěvníka z nějakého důvodu nedokáže otevřít spojení.

Událost loaded se odesílá po úspěšném navázání spojení se serverovým modulem. V případě úspěšného spojení zasílá serverový modul jako odpověď číselné sessionId, které jednoznačně identifikuje současnou návštěvu webu. Tato odpověď se zašle jako třetí parametr události, tedy popis.

4.2.2 Průběh komunikace

Klientský modul používá pro odesílání dat na server WebSocket spojení, které se po inicializaci HTML DOM otevře. Ihned po otevření spojení jsou na server odeslány data, která obsahují informace sloužící k identifikaci návštěvníka. Tyto informace jsou:

- API klíč – slouží ke spárování spojení s konkrétním účtem webmastera
- Protokol, doména, adresa – informace o aktuální URL, kterou návštěvník právě prohlíží

- Velikost okna prohlížeče
- Rozlišení zařízení
- Název prohlížeče
- Operační systém
- URL předchozí navštívené stránky
- Informace o podporovaných funkcích a pluginech prohlížeče.

Tato data jsou na serveru zpracována a jako odpověď na zprávu server vrací identifikační číslo tohoto spojení. Klientský modul tento identifikátor přijme a nastaví jako proměnnou Google Analytics společně s událostí o úspěšném načtení klientského modulu.

Ve chvíli, kdy jsou tyto následující proměnné odeslány, začíná samotné sledování aktivit návštěvníka a to tak, že jsou zaregistrovány EventListenery, které po zachycení konkrétní události odešlou data ve formě JSON na server. Sledovanými událostmi jsou:

- mousemove – návštěvník pohnul kurzorem, odesílají se současné souřadnice X a Y kurzoru relativně k okraji okna.
- resize – návštěvník minimalizoval, maximalizoval, nebo jinak změnil velikost okna. Odesílá se nová velikost okna v pixelech.
- click – návštěvník klikl do oblasti webu v prohlížeči. Odesílají se souřadnice kliknutí relativně k okraji okna a identifikace HTML elementu, na který bylo kliknuto.
- scroll – návštěvník použil scrollbar, nebo kolečko myši. Odesílá se vzdálenost v pixelech, která je právě zakrytá a to jak o horního okraje, tak od levého oraje.
- error – v JavaScriptu byla vyhozena neošetřená vyjímka , na server se odesílá popis vyjímky, číslo řádku a sloupce v kódu webu, kde byla odchycena.
- change – byl upraven některý z formulářových prvků na stránce. Odesílá se identifikace prvku a jeho nová hodnota.

4.3 Serverový modul

Účelem tohoto modulu je zpracování údajů, které odesílá klientský modul. Při prvním požadavku dojde k prohledání databáze na základě identifikačních údajů prohlížeče pro rozhodnutí, zda jde o opakovanou, či novou návštěvu. Tato data jsou následně ukládány do relační databáze. Při uzavření spojení je tato událost zachycena a v databázi je uložena informace o ukončení návštěvy.

4.3.1 Identifikace návštěvníka

Pro identifikaci byly vzhledem k předpokladům, které byly představeny v kapitole 3.3.2, vybrány proměnné, měnící se v čase pouze minimálně. Autor vybral proměnné, které jsou popsány v tabulce 4.2 a s ohledem na rychlost zpracování byla vybrána pouze část z proměnných, představených v kapitole 3.3.2. Z těchto proměnných je v serverovém modulu pomocí algoritmu SHA1 vygenerován hash, který je při historicky první návštěvě vždy unikátní. Při opakované návštěvě je již hash obsažen v databázi a návštěva je označena jako opakovaná.

Tabulka 4.2: Proměnné použité pro identifikaci návštěvníka

Proměnná	Zdroj	Poznámka
Prohlížeč	JavaScript	Proměnná navigator.userAgent v klientském modulu
IP adresa	HTTP	Získáno na serveru z kontextu HTTP spojení
Rozlišení	JavaScript	Kombinace proměnných window.screen.availHeight a window.screen.availWidth
Nastavení prohlížeče	JavaScript	Kombinace metody navigator.javaEnabled() a proměnné navigator.cookieEnabled
Rozšíření prohlížeče	JavaScript	Pole navigator.plugins v nesetříděné podobě
Canvas fingerprint	JavaScript	Získaný technikou popsanou v 3.3.2 a 5.1.2
Platforma systému	JavaScript	Získaný z proměnné navigator.platform

4.4 Použité technologie

V této kapitole jsou popsány použité technologie jednotlivých modulů.

Pro implementaci Serverového modulu byl použit integrační framework Apache Camel a aplikační server Wildfly, vyvíjený firmou RedHat. Pro porozumění je nejprve popsán protokol WebSocket a jeho implementace v Java EE kontejnerech a následně framework Apache Camel.

Pro implementaci klientského modulu autor použil standardní JavaScript API bez frameworku a to především s ohledem na rychlost zpracování a velikost výsledného scriptu.

4.4.1 Protokol WebSocket

Kapitola čerpá z oficiální dokumentace protokolu, jehož standard definuje W3C[20].

WebSocket je protokol na bázi TCP, umožňující obousměrnou komunikaci mezi dvěma síťovými prvky v rámci jediného TCP[4] spojení. Specifikace definuje protokoly ws a wss, první pro běžné spojení, druhý pro spojení šifrované pomocí SSL

Historie

WebSocket byl poprvé definován během definování HTML5 specifikace v roce 2008. První verze dokumentu ještě obsahovala název TCPConnection[19] a v dalších verzích se již objevoval současný název WebSocket. V prosinci roku 2009 byl prohlížeč Google Chrome první, který tento protokol ve výchozím nastavení povoloval[17].

Podpora v prohlížečích

Tato kapitola vychází ze statistik, publikovaných na webu caniuse.com[12], který agreguje data o dostupnosti jednotlivých technologií v prohlížečích.

Podle těchto statistik, podporuje WebSocket 94,47 procent prohlížečů používaných v České Republice a 87,66 procent prohlížečů světově. Ze současně používaných prohlížečů jej nepodporuje pouze Opera Mini a Android Browser verze 4.

Podpora v Java EE kontejnerech

WebSocket je součástí Java EE API od verze 7.0. Pro použití je nutné označit třídu anotací ServerEndpoint[5] a implementovat alespoň jednu metodu označenou anotací OnMessage[6]. Tato metoda přijímá povinný parametr message, který může být typu InputStream nebo String. Volitelným parametrem je Session[7], která obsahuje všechny doplňkové informace včetně HTTP hlaviček.

4.4.2 Apache Camel

Apache Camel je framework, implementující většinu Java EE patternů[8]. Tím usnadňuje jejich implementaci a zároveň udržitelnost kódu. První verze Apache Camel byla vydána v roce 2007 a představila nový způsob, jak vyvíjet integrační aplikace. Tento nový způsob produkoval čistší a čitelnější kód, což znamenalo lepší udržitelnost a méně chyb. Apache Camel definuje následující pojmy:

- Component

[4] Transmission Control Protocol - Sada protokolů pro komunikaci v počítačové síti
[5] javax.websocket.ServerEndpoint
[6] javax.websocket.OnMessage
[7] javax.websocket.Session
[8] Soubor ustálených a přenositelných návrhových vzorů při vývoji Java EE aplikací

- Endpoint
- Route
- Exchange

Komponenta je implementace jednotlivých protokolů. Apache Camel zastřešuje implementace jednotlivých protokolů do komponent, které jsou továrny pro vytváření endpointů. Pro účely této práce byly použity tyto komponenty

- bean pro zavolání třídy zaregistrované v DI[9] kontejneru
- direct pro přímé volání route
- log pro logování vstupu a výstupu
- seda pro frontu požadavků, které není nutné obsloužit ihned

Endpoint je pojmenovaná instance jednotlivých komponent. To umožňuje, aby byla komponenta zaregistrována vícekrát v jednom kontextu.

Route je samotná definice chování aplikace, obsahující informace o vstupních endpointech, použitých transformacích, řídících prvcích a výstupních endpointech.

Exchange je hlavní objekt, který mezi sebou sdílí jednotlivé endpointy. Obsahuje vstupní zprávu a výstupní zprávu. Dalšími informacemi jsou hlavičky, které slouží k uchovávání relevantních informací o průběhu zpracování a zároveň umožňují programátorovi nastavit proměnné, které budou dostupné po celou dobu zpracování zprávy.

Pro podrobnější popis odkazuje autor čtenáře do dokumentace Apache Camel, která je velmi kvalitně zpracována[10].

SEDA fronta

SEDA je akronymem k „Staged Event-Driven Architecture“ a je navržena jako mechanismus k regulaci výkonu mezi jednotlivými fázemi zpracování zprávy[10]. K tomuto účelu je využívána BlockingQueue z balíčku java.util.concurrent. Hlavní výhodou použití tohoto principu je zpracování zpráv v jiných vláknech. Díky tomu lze zprávy zpracovávat paralelně a asynchronně, takže samotné zpracování zprávy neblokuje hlavní vlákno. Praktická ukázka použití je popsána v kapitole 5.2.1.

[9] Dependency Injection - Technika pro definování závislostí mezi třídami tak, aby mezi nimi nevznikala silná závislost, která by ztěžovala testování.

[10] `http://camel.apache.org/architecture.html`

5 Implementace

Tato kapitola se věnuje samotné implementaci řešení, které bylo popsáno v kapitole 4.

5.1 Klientský modul

Jak již bylo zmíněno, pro implementaci tohoto modulu bylo použito JavaScript API. V této kapitole jsou popsány jednotlivé metody a použité principy. Klientský modul se skládá ze dvou souborů trouse_loader.js a trouse.js. Script trouse_loader.js zajišťuje asynchronní registraci scriptu trouse.js tak, aby inicializace scriptu neblokovala načítání stránky, ve kterých je klientský modul zaregistrován.

5.1.1 Načtení klientského modulu

Protože při inicializaci klientského modulu se vykonávají poměrně náročné operace, které by mohly zpomalovat načítání stránky, rozhodl se autor k asynchronní registraci scriptu. Registraci zajišťuje jednoduchý script, který pomocí HTML DOM[1] manipulace zaregistruje klientský modul až ve chvíli, kdy jsou ostatní scripty v HTML hlavičce načteny.

```
var TROUSE_LOADER = TROUSE_LOADER || (function(){
        return {
            load : function(initData) {
                var callback = function () {
                    TROUSE.start(initData)
                }
                var script = document.createElement("script")
                script.type = "text/javascript";

                if (script.readyState){ //IE
                    script.onreadystatechange = function(){
                        if (script.readyState == "loaded" ||
                            script.readyState == "complete"){
                            script.onreadystatechange = null;
```

[1] Document Object Model – objektový model dokumentu. DOM je API umožňující přístup k tomuto modelu a jeho modifikaci.

```
                        callback();
                    }
                };
            } else { //Others
                script.onload = function(){
                    callback();
                };
            }

            script.src = "http://janbednar.eu:8080/trouse.js?v4";
            document.body.appendChild(script);
        }
    }());
```

Ukázka kódu 5.1: Globální funkce pro asynchronní zaregistrování klientského modulu

Samotné vložení do sledovaného webu se provede zavoláním funkce load() na globálním objektu TROUSE_LOADER s parametry trouseKey a analyticsKey. Parametr trouseKey se musí shodovat s nějakým záznamem v tabublce SITE, protože se používá pro autorizaci WebSocket spojení. Parametr analyticsKey je nepovinný. V případě že není vyplněný, je komunikace s Google Analytics vypnuta a klientský modul neodesílá žádné události, které se týkají párování návštěv v Google Analytics.

```
<script type="text/javascript"
    src="http://janbednar.eu:8080/trouse_loader.js"></script>
<script>TROUSE_LOADER.load({trouseKey: "ABCDEFGHIJKLMN",
    analyticsKey: "UA-XXXXXXXX-YY"})</script>
```

Ukázka kódu 5.2: Použití Klientského modulu v HTML webu.

5.1.2 LoggingService

LoggingService se objekt, který zpracovává veškeré sledované události, komunikaci se serverovým modulem a Google Analytics. Jednotlivé metody LoggingService jsou popsány v následujících kapitolách.

Registrace služby pro zachytávání událostí

```
var TROUSE = TROUSE || (function(){
    var _loggingService;

    return {
        init : function(initData) {
```

```
            if (initData.trouseKey){
                _loggingService = new
                    LoggingService(initData.trouseKey,
                    initData.analyticsKey, initData.dimensionMapping);
                _loggingService.registerGA();
                _loggingService.ga().event('trouse','start');
                _loggingService.connect();
            }
        },
        register : function() {
            if (_loggingService !== null) {
                    _loggingService.registerEvents(window);
            }
        },
        start : function(initData) {
                    this.init(initData);
                    this.register();
        }
    };
}());
```

Ukázka kódu 5.3: Registrace LoggingService

Odeslání události

Metody zmiňované v této kapitole obsahují funkcionalitu pro odeslání zprávy pomocí WebSocket spojení. Protože existuje určitá časová prodleva mezi zaregistrováním událostí a otevřením spojení, rozhodl se autor pro implementaci jednoduché fronty zpráv. Fronta je datového typu Array a během odesílání zprávy je kontrolováno, zda existuje aktivní WebSocket spojení. V případě, že ke spojení ještě nedošlo, vloží se daná zpráva pomocí metody Array.push() na konec pole. V okamžiku navázání spojení je fronta v metodě processQueue() zpracována a každá jednotlivá zpráva je odeslána do serverového modulu. Následně se fronta označí jako zpracovaná. Následující zprávy se již za předpokladu, že fronta byla označena jako zpracovaná, odesílají v okamžiku, kdy nastanou. Samotné odeslání zajišťuje metoda send() na objektu typu WebSocket.

```
this.send = function (action, data) {
    var json = JSON.stringify({action: action, data: data});
    if (_self.ws !== undefined && _self.ws.readyState ==
        _self.ws.OPEN && (_queueProcessed || action=="load")) {
        _self.ws.send(json);
    }else if (!sessionId && _commandQueue.length<1000){
            _commandQueue.push(json)
```

```
    }
};
```

Ukázka kódu 5.4: Odeslání události na Server modul

```
this.processQueue = function () {
    var query;
    while(query = _commandQueue.shift())
    {
        _self.ws.send(query);
    }
    _queueProcessed = true;
};
```

Ukázka kódu 5.5: Fronta požadavků

Identifikace návštěvníka

Tato metoda vychází z požadavků, definovaných v kapitole 4.3.1 a prezentuje, jakým způsobem jsou proměnné definované v téže kapitole získány. Výstupem je JSON, který je odesílán na serverový modul během události load a kromě identifikačních údajů obsahuje i jednotlivé části URL a klíč pro validaci požadavku.

```
this.getBrowserData = function () {
    var data = {
        apiKey: apiKey,
        protocol: window.location.protocol,
        host: window.location.host,
        path: window.location.pathname,
        size: _self.getSize(),
        canvasFingerPrint: _self.getCanvasFingerPrint(),
        lang: navigator.language || navigator.userLanguage,
        resolution: {
            height: window.screen.availHeight,
            width: window.screen.availWidth
        },
        userAgent: navigator.userAgent,
        referrer: document.referrer,
        platform: navigator.platform,
        support: {
            java: navigator.javaEnabled(),
            cookie: navigator.cookieEnabled
        }
    }, i;
```

```
    data.plugins = [];
    for (i = 0; i < navigator.plugins.length; i++) {
        data.plugins.push(navigator.plugins[i].name);
    }
    return data;
};
```

Ukázka kódu 5.6: Zajištění informací pro otisk zařízení

Součástí informací, které jsou následně použity pro identifikaci návštěvníka, je i vygenerování údajů technikou canvas fingerprinting, která je popsána v kapitole 3.3.2. Tato proměnná je generována pomocí metody getCanvasFingerprint() a výkonná část byla převzata a upravena z browserleaks.com[6].

```
this.getCanvasFingerPrint = function () {
    var canvas = document.createElement('canvas');
    if (!canvas || !canvas.getContext){
        return null;
    }
    var ctx = canvas.getContext('2d');
    if (!ctx){
        return null;
    }
    // https://www.browserleaks.com/canvas#how-does-it-work
    var txt = 'trouse.js';
    ctx.textBaseline = "top";
    ctx.font = "14px 'Arial'";
    ctx.textBaseline = "alphabetic";
    ctx.fillStyle = "#f60";
    ctx.fillRect(60,1,62,20);
    ctx.fillStyle = "#069";
    ctx.fillText(txt, 2, 15);
    ctx.fillStyle = "rgba(102, 204, 0, 0.7)";
    ctx.fillText(txt, 4, 17);

    return canvas.toDataURL();
}
```

Ukázka kódu 5.7: Canvas fingerprinting

Vytvoření WebSocket spojení

Funkce connect zajišťuje WebSocket připojení k serveru a odesílá do Google Analytics události představené v tabulce 4.1. V případě, že prohlížeč nepodporuje WebSocket,

odesílá se událost failed. V ukázce je vidět také odeslání události loaded, která se odesílá během první odpovědi z serverového modulu a také dojde ke zpracování fronty požadavků, která byla popsána v kapitole 5.1.2.

```
this.connect = function(){
    if (!("WebSocket" in window)){
        //document.body.style.backgroundColor = "red";
        _self.ga().event('trouse','failed');
        return;
    }
    try {
        _self.ws = new WebSocket(_wsUri);
        _self.ws.onerror = function(evt){
            console.log("TROUSE: The following error occurred: " +
                evt);
        }
        _self.ws.onmessage = function(evt){
            if (!sessionId){
                sessionId = evt.data;
                //document.body.style.backgroundColor = "blue";
                _self.ga().set(dimensionMapping.trouseSessionId,sessionId);
                _self.ga().event('trouse','loaded',sessionId);
                _self.processQueue();
            }
        }
        _self.ws.onopen = function () {
            _self.ga().set(dimensionMapping.websocketSupport,"enabled");
            _self.send('load', _self.getBrowserData());
        };}
```

Ukázka kódu 5.8: Vytvoření WebSocket spojení na Server modul

Zachytávání událostí

Tato funkce registruje EventListenery pro každou zachytávanou událost a je volána při registraci LoggingService metodou register(). Události mousemove, resize, click a scroll jsou zaregistrovány pomocí metody addEventListener na objektu Window, který je součástí JavaScript API. Metoda addEventListener zajišťuje, aby byly zachovány všechny předchozí zaregistrované události a nebyla použitím modulu porušena funkcionalita stránky.

Jiná situace je u zachycení události error, kde se funkce pro obsloužení události registruje přímo nastavením na objektu Window a tím dochází k jeho přepsání. Proto je

nutné předchozí hodnotu Window.onError nejprve uložit a v implementaci nové metody zavolat i tu předchozí.

Pro zachycení změny formulářových prvků neexistuje v JavaScript API žádný globální listener. Existuje pouze metoda addEvenntListener na objektu Element. Proto jej bylo třeba zaregistrovat na každý formulářový element. K tomu byla vytvořena funkce registerChangeListenerOnTags, která zajistí zaregistrování v cyklu pro všechny elementy typu input, keygen, select a textarea. Pro zajištění soukromí nejsou zachytávány události nad elementy, které slouží k zadávání hesla.

```
this.registerEvents = function(windowInstance){
    if (_self.ws !== undefined) {
        windowInstance.addEventListener('mousemove',
            _self.mouseMonitor);
        windowInstance.addEventListener('resize',
            _self.resizeMonitor);
        windowInstance.addEventListener('click', _self.clickMonitor,
            false);
        windowInstance.addEventListener('scroll',
            _self.scrollMonitor);
        var oldOnError = windowInstance.onerror || function(msg,
            url, line, col, error){};
        windowInstance.onerror = function (msg, url, line, col,
            error) {
            _self.errorMonitor(msg, url, line, col, error);
            oldOnError(msg, url, line, col, error);
        };
        _self.registerChangeListenerOnTags(windowInstance,
            ['input','keygen','select','textarea'],
            _self.inputMonitor);
    }else {
        console.log("Socket closed, ignoring trouse session");
    }
};
```

Ukázka kódu 5.9: Ukázka registrace událostí

Událost mousemove

Událost mousemove se odesílá na serverový modul ve chvíli, kdy je pohnuto kurzorem nad oblastí pro vykreslení webové stránky v prohlížeči. Logiku zajišťuje funkce mouseMonitor na objektu LoggingService. Odesílaný JSON obsahuje souřadnice na ose X, souřadnice na ose Y a všechny atributy elementu, na který právě ukazuje kurzor myši.

Protože některé prohlížeče vracely souřadnice s desetinnými místy, byly souřadnice zaokrouhleny na celá čísla.

```
this.mouseMonitor = function (e) {
    if(e.pageX == x && e.pageY == y) return;
    x = Math.round(e.pageX);
    y = Math.round(e.pageY);
    attr = _self.getElementAttributesFromEvent(e);
    _self.send('mouseMove', {mouseX: x, mouseY: y,
        elementAttributes: attr});
    //_self.ga('mouseMove', {mouseX: e.pageX, mouseY: e.pageY,
        elementAttributes: attr});
```

Ukázka kódu 5.10: Zachycení události mousemove

Událost resize

Událost resize se odesílá během změny velikosti okna prohlížeče. Odesílaný JSON obsahuje výšku a šířku okna. Protože prohlížeč v iOS spouštěl tuto funkci i během scrollování, musela být přidána podmínka kontrolující, že se velikost okna opravdu změnila.

```
this.resizeMonitor = function (e) {
    //iOS triggering this event on scroll, so we wil check the size
        is changed
    var oldSize = _windowSize;
    var newSize = _self.getSize();
    if (oldSize.width != newSize.width || oldSize.width !=
        newSize.width){
        _self.send('resize', _self.getSize());
    }
};
```

Ukázka kódu 5.11: Zachycení události resize

Událost click

Událost click je odeslána ve chvíli, kdy uživatel kliknul myší na desktopovém prohlížeči, nebo při dotyku na dotykovém zařízení. Některé mobilní prohlížeče ohlašovaly souřadnice s desetinnými místy, proto byly zaokrouhleny na celé pixely.

```
this.clickMonitor = function (e) {
    e = e || window.event;
            _self.send('click', {mouseX: Math.round(e.pageX), mouseY:
                Math.round(e.pageY), elementAttributes:
```

```
                _self.getElementAttributesFromEvent(e)});
};
```

Ukázka kódu 5.12: Zachycení události click

Událost error

Odesílá se ve chvíli, když je JavaScriptu zachycena neošetřená výjimka. Výsledný JSON obsahuje popis chyby, URL, řádek a sloupec, kde chyba nastala. Jsou ignorovány chyby s popisem "Script error.", protože tato výjimka neobsahuje žádné relevantní informace. Jde o chyby, které nastanou ve scriptu umístěném na jiné doméně. Tyto chyby by mohly obsahovat citlivé informace a proto je z bezpečnostních důvodu prohlížeče nepropagují.

```
this.errorMonitor = function (msg, url, line, col, error) {
    if (msg.indexOf('Script error.') > -1) {
        return;
    }
    _self.send("error", {message: msg, url: url, line: line, column:
        col, error: error});
    console.log(error);
};
```

Ukázka kódu 5.13: Zachycení události error

Událost scroll

Událost obsluhuje metoda scrollMonitor a je vyvolána při scrollování na desktopovém prohlížeči, nebo posouvání na dotykovém prohlížeči. Ve zjištění levého a horního posunu se implementace prohlížečů liší. V některých se k těmto hodnotám přistupuje přes proměnnou document.documentElement a v některých přes document.body. Na serverový modul se odesílá nový levý a pravý posun v pixelech.

```
this.scrollMonitor = function () {
    var left = document.documentElement.scrollLeft ||
        document.body.scrollLeft;
    var top = document.documentElement.scrollTop ||
        document.body.scrollTop;
    _self.send('scroll', {
        top: Math.round(top),
        left: Math.round(left)
    });
```

```
};
```

Ukázka kódu 5.14: Zachycení události scroll

Událost change

Je vyvolána při změně formulářového prvku, na kterém je listener zaregistrovaný. Na serverový modul se odesílají všechny atributy elementu, obsahující identifikaci a jeho novou hodnotu.

```
this.inputMonitor = function (evt) {
    console.log(evt);
    console.log(_self.getElementAttributesFromEvent(evt));
    _self.send("change",{elementAttributes:
        _self.getElementAttributesFromEvent(evt)})
};
```

Ukázka kódu 5.15: Zachycení události change

5.2 Serverový modul

5.2.1 RouteBuilder

Třída MainRouteBuilder obsahuje celkovou definici logiky aplikace. Díky použití integračního frameworku Apache Camel je definice logiky chování velmi krátká. Konkrétní implementace je popsána v následujících kapitolách.

```
    public void configure() throws Exception {
        getContext().getTypeConverterRegistry()
      .addTypeConverters(inputMessageTypeConverters);
        getContext().setAllowUseOriginalMessage(false);
        getContext().setMessageHistory(false);
        errorHandler(defaultErrorHandler().maximumRedeliveries(3));

        final String SEDA_URI =
            "seda:queue?waitForTaskToComplete=Never"
            +"&concurrentConsumers=100";
      from("direct:entry")
            .unmarshal().json(JsonLibrary.Jackson, InputMessage.class)
            .choice()
            .when().simple("${body.action} == 'load'")
                .to("direct:perform")
            .otherwise()
```

```
                .to(SEDA_URI)
                .setBody(constant(new HashMap<>())).end();

        from("direct:close").to("bean:closeAction");

        from("direct:perform")
                .to("bean:databaseLogger")
                .to(uriUtils.getLoggerUri("entryInput",true))
                .log(LoggingLevel.DEBUG, "About to call
                   bean:${body.action}Action")
                .recipientList().simple("bean:${body.action}Action")
                .to(uriUtils.getLoggerUri("entryOutput"));

        from(SEDA_URI)
                .log(LoggingLevel.DEBUG, "Seda processing
                   bean:${body.action}Action")
                .to("direct:perform");

        from("timer:sedaChecker?period=60s")
      .process(exchange -> log.info(
            "Seda size = "+
            ((SedaEndpoint)getContext().getEndpoint(SEDA_URI))
            .getExchanges().size())
            )
      .end();
    }
```

Ukázka kódu 5.16: Implementace RouteBuilder serverového modulu

Nastavení kontextu

První řádek zaregistruje logiku pro odchytávání chyb. Lze nadefinovat vlastní třídu, nebo použít builder pattern. V tomto konkrétním případě byl použit builder pattern a nastavuje, aby se Apache Camel pokusil zprávu po zachycení vyjímky zpracovat znovu, maximálně však třikrát.

První řádek přidává do registrů uživatelsky definované konvertory. Konvertor je třída, implementující rozhraní org.apache.camel.TypeConverters a slouží k definování uživatelských konverzi mezi objekty. Díky konvertorům lze následně zavolat metodu getBody s parametrem typu Class<?> a framework se postará o zavolání konvertoru, pokud je pro danou třídu k dispozici.

Další řádky vypínají archivaci původní zprávy a uchování historie zpráv. Originální nastavení je velmi vhodné pro vývojové účely, protože usnadňuje ladění, ale pro pro-

dukční servery s odladěnou aplikací je vhodné tyto dvě funkce vypnout a ušetřit tak velkou část operační paměti serveru.

Vstupní cesta

Vstupní cesta je definována v route direct:entry. Klientský modul odesílá přes WebSocket zprávu ve formátu JSON, proto je vhodné provést unmarshalling a zprávu překonvertovat do objektu, se kterým se bude snáze pracovat. V Apache Camel stačí vložit příkaz unmarshal, který zajistí, že původní JSON zpráva se nahradí objektem.

Zbytek kódu vstupní cesty je rozhodovací blok, zajišťující, aby zpráva s akcí load byla zpracována ihned a ostatní se pouze vložily do SEDA fronty pro následné asynchronní zpracování.

Zavolání JavaBean

Bean je v pojetí Apache Camel jakákoliv třída, kterou má uloženou v registrech a má alespoň jednu metodu, která je beznávratová s parametrem typu Exchange. Beana lze zaregistrovat příkazem v RouteBuilderu, nebo označením anotací @Named při použití v kontejneru, nebo @Component, při použití Spring frameworku.

Je mnoho způsobů jak v Apache Camel docílit zavolání konkrétní beany. Autor použil metodu simple(), která složí její název tak, že použije parametr action ze vstupního JSON objektu a za ten vloží konstantu Action. Výsledkem metody simple je už přímo URL, která obsahuje informaci o použité komponentě, v tomto případě bean a názvu beany. Takto složenou URL přijímá metoda recipientList(), která zajistí zavolání.

5.2.2 Zpracování akcí klienského modulu

Postup, jakým je pomocí Apache Camel zavolána konkrétní beana byl popsán v předchozích dvou kapitolách a v této kapitole bude popsána ukázka implementace zpracování akce mousemove, kterou klientský modul odesílá při zjistění pohybu kurzoru. Pro každou z akcí, které jsou popsány v seznamu 4.2.2 je implementována samostatná beana, ale protože všechny obsahují podobnou logiku pro vytvoření JPA entity[2], je zde ukázka pouze pro MouseMoveAction.

```
public void perform(Exchange exchange) {
    final MessageData data =
       exchange.getIn().getBody(MessageData.class);
    Utils.checkMandatoryHeaders(exchange.getIn());
    MouseMove mm = new MouseMove();
```

[2]JPA entita je třída, respektující konvence obsažené v Java Persistence API tak, aby mohl zvolený ORM framework korektně namapovat data z relační databáze na entitu.

```
        mm.setTime(exchange.getIn().getHeader(
            ExchangeHeaderKeys.DATE_RECEIVED,Date.class));
        mm.setMouseX(data.get("mouseX",Integer.class).shortValue());
        mm.setMouseY(data.get("mouseY",Integer.class).shortValue());
        Map<String,String> attributes =
            data.get("elementAttributes",Map.class);
        mm.setElementId(Utils.buildElementId(attributes));
        mm.setElementName(attributes.get("name"));

        RecordingSession recordingSession =
            loggingApiDao.find(RecordingSession.class,
    exchange.getIn().getHeader(ExchangeHeaderKeys.SESSION_ID));

        mm.setRecordingSession(recordingSession);
        loggingApiDao.persist(mm);
    }
```

Ukázka kódu 5.17: Implementace MouseMoveAction

5.3 Sestavení modulů

Pro správu závislostí, sestavení modulů a testy byl použit Apache Maven[3]. Aplikace se skládá ze tří Maven modulů:

- loggingApi: Serverový modul, obsahující logiku zpracování WebSocket zpráv.
- loggingClient: Klientský modul, který obsahuje obsahuje JavaScript pro nasazení na měřený web.
- domain: Doménový modul, ve kterém jsou obsaženy JPA entity a DAO pro přístup k databázi.

Zmíněné moduly jsou součástí modulu parent. Modul parent neobsahuje žádný zdrojový kód, jeho hlavním účelem je obalit ostatní moduly tak, aby bylo možné definovat společné konfigurace na jednom místě. Maven modul obsahuje především sekci dependency management, kde jsou nadefinovány projektové závislosti a jejich verze z Maven repozitáře. Další sekcí je pluginManagement, kde je nastavení procesu sestavení. Především použitý Java kompilátor a konfigurace maven-war-plugin pro sestavení balíčku.

Doménový modul (domain) je použitý jako knihovna, na které má závislost serverový modul. Obsahuje JPA entity pro ORM mapování pomocí Hibernate. Další částí je DAO, obsahující metody s HQL dotazy pro přístup k datům

[3] Apache Mavenje nástroj pro správu, řízení a automatizaci buildů aplikací

Serverový modul (loggingApi) je sestavován do balíčku war, který je přímo určený pro nasazení na aplikační server.

Klientský modul (loggingClient) není sestavován do žádného balíčku. Úkolem modulu je spuštění testů nad scriptem trouse.js a vytvoření komprimované verze scriptů pro snížení výsledné velikosti. Pro otestování autor zvolil Maven plugin jasmine-maven-plugin. Testy jsou spuštěny během každého procesu sestavení a testují, zda lze úspěšně zaregistrovat LoggingService a zda script obsahuje všechny nutné metody pro sledování událostí. První zmíněný test byl implementován z toho důvodu, aby případný další vývojář neudělal chybu tím, že by metody init() a register() přejmenoval. Protože ve chvíli přejmenování se již metody mohou používat na více webech, jejich přejmenování by způsobilo nefunkčnost aplikace. Druhý zmíněný test kontroluje, že jsou implementovány všechny metody pro sledování událostí tak, aby se nestalo, že v průběhu implementace dojde k jejich přejmenování. Změna názvu metody způsobí částečnou nefunkčnost aplikace pro konkrétní přejmenovanou událost. Pro sestavení komprimované verze trouse.js vybral autor yuicompressor-maven-plugin. Zmíněný plugin během procesu sestavení zkontroluje syntaktickou správnost podle striktních definic, aby byla zajištěna funkčnost ve všech prohlížečích. Dalším krokem, který tento plugin provádí je zkomprimování kódu tím, že jsou lokální proměnné přejmenovány na kratší a také tím, že odstraní nadbytečné mezery a odřádkování. Výsledkem je script, který má menší datovou velikost.

Celá aplikace se sestaví příkazem mnv clean install který vytvoří soubory loggingApi.war a trouse.js. Balíček war je určený pro nasazení na aplikační server. JavaScriptové scripty byly během vývoje nakopírovány do složky welcome-context v umístění aplikačního serveru. Vystavení scriptu na webu tedy zajišťoval Wildfly. Pro produkční prostředí autor doporučuje umístění scriptů k nějakému poskytovateli CDN[4] pro snížení vytížení serveru.

5.4 Konfigurace aplikačního serveru

Pro aplikaci autor zvolil aplikační server Wildfly ve verzi 9.0.2.Final, který byl v době vývoje aplikace aktuální. V době psaní práce je aktuální verze 10.0.0.Final, pro kterou však nebyla aplikace testována. Veškerá konfigurace je umístěna v souboru standalone.xml ve složce ./standalone/configuration/. Konfigurace některých subsystémů byla ponechána ve výchozím nastavení, upraveny byly pouze tyto:

- jboss:domain:datasources:3.0: Subsystém pro konfiguraci databázových přístupů

[4]Content Delivery Network - Síť serverů, rozmístěných v různých lokalitách tak, aby byl statický obsah dostupný vždy s co nejnižší latencí.

- jboss:domain:infinispan:3.0: Subsystém pro cache.

Kompletní obsah standalone.xml je uveden na přiloženém CD.

5.5 Interpretace dat

Protože součástí této práce není GUI, které by prezentovalo data, obsahuje tato kapitola ukázky SQL dotazů, vedoucí ke zjištění některých metrik, které byly definovány v kapitole 3.

5.5.1 Počet unikátních návštěvníků

Uvedený dotaz vrací počet unikátních návštěvníků za den. Pro omezení na konkrétní časové období je možné přidat podmínku do klauzule WHERE.

```
SET @siteId = 1;
SELECT DATE(START_TIME), COUNT(DISTINCT BROWSER_ID)
FROM RECORDING_SESSION
WHERE SITE_ID=@siteId
GROUP BY DATE(START_TIME)
```

Ukázka kódu 5.18: SQL pro počet unikátních návštěvníků za den

5.5.2 Nejčastější externí referrer

Výstupem dotazu je seznam externích webů, ze kterých bylo na web přistoupeno.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT REFERRER, COUNT(*) as pocet
FROM RECORDING_SESSION
WHERE REFERRER NOT LIKE CONCAT('%', (SELECT ROOT_URL FROM SITE
   WHERE SITE_ID = @siteId),'%') AND SITE_ID = @siteId
GROUP BY REFERRER
ORDER BY pocet desc
```

Ukázka kódu 5.19: SQL pro nejčastější externí referrer

5.5.3 Nejčastější interní referrer

Dotaz vrací seznam interních odkazujících stránek. Tedy těch, které jsou uloženy na doméně měřeného webu. Seznam je setříděn podle počtu výskytů.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT REFERRER, COUNT(*) as pocet
```

```
FROM RECORDING_SESSION
WHERE REFERRER LIKE CONCAT('%', (SELECT ROOT_URL FROM SITE WHERE
    SITE_ID = @siteId),'%') AND SITE_ID = @siteId
GROUP BY REFERRER
ORDER BY pocet desc
```

Ukázka kódu 5.20: SQL pro nejčastější interní referrer

5.5.4 Nejčastější vstupní stránky z vyhledávače Google

Výstupem dotazu je výpis URL, které byly jako vstupní stránky přistoupeny z vyhledávače Google.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT URL, COUNT(*) as pocet
FROM RECORDING_SESSION
WHERE REFERRER REGEXP '^(http|https)://www\.google\.[a-z.]+/.*$'
    AND SITE_ID = @siteId
GROUP BY URL
ORDER BY pocet desc
```

Ukázka kódu 5.21: SQL pro nejčastější vstupní stránky z vyhledávače Google

5.5.5 Nejčastější vstupní stránky z Facebook

Výstupem dotazu je výpis URL, které byly jako vstupní stránky přistoupeny ze sociální sítě facebook.com.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT URL, COUNT(*) as pocet
FROM RECORDING_SESSION
WHERE REFERRER REGEXP
    '^(http|https)://[a-zA-Z0-9]{1,3}\.facebook\.com.*$' AND SITE_ID
    = @siteId
GROUP BY URL
ORDER BY pocet desc
```

Ukázka kódu 5.22: SQL pro nejčastější pro nejčastější vstupní stránky Facebook

5.5.6 Počet zobrazení na stránku

Dotaz vrací počet zobrazení stránky pro každou stránku webu, která patří pod záznam v tabulce SITE. Seznam je seřazeny sestupně podle počtu zobrazení.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT URL, COUNT(*) as pocet
FROM RECORDING_SESSION
WHERE SITE_ID = @siteId
GROUP BY URL
ORDER BY pocet desc
```

Ukázka kódu 5.23: SQL pro počet zobrazení

5.5.7 Statistika kliknutí na HTML elementy

Dotaz vrací pro každou stránku webu seznam elementů, setříděný podle počtu kliknutí. Protože výsledkem jsou všechny elementy, na které bylo v historii webu kliknuto, je vhodné tento výstup omezit na konkrétní URL.

```
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT rs.URL, mc.ELEMENT_ID, mc.ELEMENT_NAME
FROM MOUSE_CLICK mc
LEFT JOIN RECORDING_SESSION rs ON rs.ID = mc.SESSION_ID
WHERE rs.SITE_ID=@siteId;
```

Ukázka kódu 5.24: SQL statistiku kliknutí na HTML elementy

5.5.8 Statistika kliknutí podle souřadnic

Tento SQL dotaz slouží k výpisu počtu kliknutí ve všech čtvercích obrazovky. Obrazovka je rozdělena na čtverce o velikosti strany v pixelech, definovaných v proměnné @i. Pro každou URL jsou vypsány všechny čtverce, setříděné podle počtu kliknutí.

```
SET @i = 10;/*rozměr čtverce v pixelech*/
SET @siteId = 1;/*ID stránky z tabulky SITE*/
SELECT
rs.URL as url,
CONCAT(CEIL(MOUSE_X/@i)*@i,'-', CEIL(MOUSE_X/@i)*@i+@i) as X,
CONCAT(CEIL(MOUSE_Y/@i)*@i,'-', CEIL(MOUSE_Y/@i)*@i+@i) as Y,
COUNT(*) as pocet
FROM MOUSE_CLICK mc
LEFT JOIN RECORDING_SESSION rs ON rs.ID = mc.SESSION_ID
WHERE rs.SITE_ID=@siteId
GROUP BY url, X, Y
ORDER BY url, pocet desc, X,Y;
```

Ukázka kódu 5.25: SQL statistiku kliknutí podle souřadnic

6 Dosažené výsledky

Pro otestování byl použit web magazín jenpromlade.cz. Testování probíhalo od 30.6.2016 do 30.7.2016 a tato kapitola hodnotí dosažené výsledky. Zaměřuje se na podíl návštěv, které nemohly být změřeny z důvodu absence podpory pro WebSocket. Dalším kritériem je porovnání naměřených údajů s údaji naměřenými pomocí Google Analytics.

6.1 Chyby způsobené absencí podpory pro WebSocket

V daném sledovaném období byla do Google Anylytics odesílána vlastní událost failed značící absenci podpory pro WebSocket. Při vyfiltrování těchto událostí v Google Analytics vyšlo najevo, že podpora pro WebSocket chyběla v 1,09% zobrazení webu. Ve zbylých 98,91% zobrazení bylo spojení úspěšně otevřeno. To je překvapivě více, než je uvedeno na webu caniuse.com[12], kde se uvádí podpora v 94,75% případů. Tento rozdíl může být způsoben tématikou magazínu, na kterém byl klientský modul testován, protože se zaměřuje především na mladší populaci, která používá často nové technologie.

6.1.1 Prohlížeče bez WebSocket podpory

Již bylo zjištěno, že WebSocket byl podporován v 98,91% zobrazení stránky. Tato kapitola se zaměřuje na rozložení prohlížečů ve zbylých 1,09% zobrazení, tedy těch, kdy byla zjištěna absence podpory. Tabulka 6.1 čerpá ze sekce chování a záložky nejčastější události v Google Analytics.

6.2 Test změřených unikátních návštěv

Pro účely tohoto testu byl v daném období pro každý den sledován počet unikátních návštěv změřených jak pomocí Google Analytics, tak pomocí otisku prohlížeče. Naměřené údaje jsou prezentovány v tabulce 6.2 a jsou kombinací údajů z Google Analytics a výsledků dotazu 5.5.1. Protože Google Analytics používá ve výchozím nastavení časovou zónu GMT-8:00 a MariaDB server GMT, bylo třeba dotaz upravit a přidat funkci INTERVAL pro normalizaci výsledků.

6.2.1 Cíl testu

Cílem je určit, jestli je počet návštěv změřený pomocí Google Analytics srovnatelný s počtem naměřených návštěv pomocí metody otisku prohlížeče. K tomu účelu byly stanoveny následující proměnné:

- X: Počet unikátních návštěv podle Google Analytics.
- Y: Počet unikátních návštěv podle metody otisku prohlížeče.
- $D = X - Y$: Nová náhodná veličina - rozdíl počtu návštěv podle Google Analytics a podle metody otisku prohlížeče.

Testuje se hypotéza, že střední hodnota (resp. medián) náhodné veličiny je roven nule.

6.2.2 Test normality

Aby bylo možné zvolit správný test, je nejprve nutné otestovat normalitu náhodné veličiny D.

- H_0: D pochází z normálního rozdělení.
- H_A: D nepochází z normálního rozdělení.

Autor zvolil Shapiro-Wilkův test normality, který je vhodný pro menší datové soubory (n<50), což je splněno.

Podle statistického programu R je p-hodnota $4,579 * 10^{-8} < 0.05$. Proto je na hladině významnosti $\alpha = 0.05$ zamítnuta hypotéza H_0 ve prospěch alternativy.

6.2.3 Test středních hodnot

Protože bylo zamítnuto, že by D pocházelo z normálního rozdělení, je nutné použít neparametrický test. Autor zvolil znaménkový test pro otestování mediánu náhodné veličiny.

- $H_0 : \mu = 0$
- $H_A : \mu \neq 0$

Autor pro otestování použil program R (funkci SIGN.test z knihovny BSDA).

Dle tohoto testu je $p - hodnota = 0.136 > 0.05$, proto nedochází na hladině významnosti $\alpha = 0.05$ k zamítnutí hypotézy H_0.

6.2.4 Zhodnocení

Pozorovaný počet rozdílů r^+ nebyl dostatečně malý, ani velký, aby se prokázalo, že medián náhodné veličiny D je různý od 0. Tedy nelze zamítnout, že obě metody jsou srovnatelné.

Tabulka 6.1: Prohlížeče bez WebSocket podpory

Prohlížeč	Verze prohlížeče	Počet událostí	Podíl z celku (%)
Android Browser	4.0	500	75,76
Chrome	27.0.1453	33	5
Internet Explorer	9.0	24	3,64
Opera Mini	17.0.2211	22	3,34
Opera Mini	4.2.22228	10	1,52
Opera Mini	7.1.32448	9	1,36
Android Browser	4.1.2016	7	1,06
Opera Mini	4.5.40312	6	0,91
Opera Mini	15.0.2125	5	0,76
Opera Mini	7.6.40234	5	0,76
UC Browser	10.1.4.573	5	0,76
Android Browser	534.30	4	0,6
Chrome	11.0.696.34	3	0,45
Nokia Browser	7.3.1.33	3	0,45
Nokia Browser	7.3.1.37	3	0,45
Opera Mini	11.0.1912	3	0,45
Opera Mini	16.0.2168	3	0,45
Opera Mini	7.5.35199	3	0,45
Safari	5.1.2016	3	0,45
Nokia Browser	7.4.2.6	2	0,30
Opera Mini	13.0.2036	2	0,30
Opera Mini	4.2.22537	2	0,30
Opera Mini	7.5.33361	1	0,15
Safari	5.0.2	1	0,15
Safari	525	1	0,15

Tabulka 6.2: Naměřené unikátní návštěvy

Datum	Google Analytics	Otisk prohlížeče
30.6.2016	1838	1699
1.7.2016	3803	3946
2.7.2016	3560	3184
3.7.2016	1516	1517
4.7.2016	488	413
5.7.2016	65	52
6.7.2016	62	55
7.7.2016	26	27
8.7.2016	37	36
9.7.2016	991	1012
10.7.2016	1155	1104
11.7.2016	1235	1215
12.7.2016	641	589
13.7.2016	816	822
14.7.2016	606	577
15.7.2016	1305	1322
16.7.2016	1036	961
17.7.2016	1305	1310
18.7.2016	358	348
19.7.2016	40	37
20.7.2016	29	25
21.7.2016	33	30
22.7.2016	22	18
23.7.2016	20	23
24.7.2016	23	23
25.7.2016	13	12
26.7.2016	25	21
27.7.2016	13	12
28.7.2016	21	22
29.7.2016	13	15
30.7.2016	21	21

7 Závěr

Cílem práce bylo navržení a implementace systému pro sběr a uchování údajů uživatelských interakcí. Samotnému vývoji předcházel průzkum již existujících řešení a definování pojmů, které jsou pro webovou analytiku klíčové. Práce shrnula historický vývoj webové analytiky.

Práce zkoumala vhodný způsob, jakým je možné párovat návštěvy mezi Google Analytics a nově vyvinutým systémem. Proto bylo nutné, aby mezi oběma systémy existoval jednotný identifikátor, podle kterého lze návštěva v tom druhém systému vyhledat. Autor zvolil odesílání uživatelské akce do Google Analytics s informací obsahující číslo, specifikující číslo zobrazení vygenerované v novém systému. Tím bylo zajištěno, že v Google Analytics se toto číslo uložilo a v databázi nového systému bylo možné podle tohoto identifikátoru dohledat více informací.

Výsledný systém se skládá ze dvou částí, klientského a serverového modulu. S ohledem na rychlost zpracování a co nejnižší výslednou velikost scriptu byl použit čistý JavaScript bez přidaných knihoven. Pro implementaci serverového modulu autor použil integrační framework Apache Camel, který vede k čistému a udržitelnému kódu. Modul byl implementován takovým způsobem, aby případné přidání sledované události, nebo úprava struktury JSONu způsobila minimální zásah do zdrojového kódu. V práci byla popsána architektura systému a popsány konkrétní použité technologie.

Moduly používají pro komunikaci WebSocket, který není ve starších prohlížečích podporovaný. Proto autor v kapitole 6.1 zjišťoval, v kolika procentech zobrazení byl WebSocket podporován. Dospěl k tomu, že WebSocket byl podporován v 98,91% zobrazení.

Práce se také zabývala vlastní implementací alternativního způsobu identifikace návštěvníka za použití otisku zařízení. Pro ověření tohoto způsobu autor otestoval návštěvy změřené pomocí Google Analytics a otisku zařízení. Výsledkem tohoto testu bylo nezamítnutí tvrzení, že obě metody jsou srovnatelné.

Bibliografie

[1] Ernesto Arroyo, Ted Selker a Willy Wei.
"Usability tool for analysis of web designs using mouse tracks".
In: *CHI '06 extended abstracts on Human factors in computing systems - CHI EA '06*.
Association for Computing Machinery (ACM), 2006. ISBN: 1595932984.
DOI: 10.1145/1125451.1125557.
URL: http://dx.doi.org/10.1145/1125451.1125557.

[2] Web Analytics Association. *Mission/Vision*. URL:
http://www.digitalanalyticsassociation.org/mission-vision.

[3] Web Analytics Association. *Web Analytics Definitions – Version 4.0*. STANDARD.
Web Analytics Association, 2008, s. 1–34.
URL: http://www.digitalanalyticsassociation.org/Files/PDF_
standards/WebAnalyticsDefinitionsVol1.pdf.

[4] Richard Atterer, Monika Wnuk a Albrecht Schmidt.
"Knowing the user's every move".
In: *Proceedings of the 15th international conference on World Wide Web - WWW '06*.
Association for Computing Machinery (ACM), 2006. ISBN: 1595933239.
DOI: 10.1145/1135777.1135811.
URL: http://dx.doi.org/10.1145/1135777.1135811.

[5] MelissaO Braxton et al. "Mouse cursor movement and eye tracking data as an indicator of pathologists' attention when viewing digital whole slide images".
In: *Journal of Pathology Informatics* 3.1 (2012), s. 43. ISSN: 2153-3539.
DOI: 10.4103/2153-3539.104905.
URL: http://dx.doi.org/10.4103/2153-3539.104905.

[6] BrowserLeaks. *Canvas Fingerprinting — BrowserLeaks.com*.
https://www.browserleaks.com/canvas#how-does-it-work.

[7] Mon Chu Chen, John R. Anderson a Myeong Ho Sohn.
"What can a mouse cursor tell us more?"
In: *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*.
Association for Computing Machinery (ACM), 2001. ISBN: 1581133405.

DOI: 10.1145/634067.634234.
URL: http://dx.doi.org/10.1145/634067.634234.

[8] Clicktale. *Why Choose Clicktale*.
https://www.clicktale.com/company/why-clicktale/.

[9] Brian Clifton. *Successful Analytics*. s.l: Advanced Web Metrics Ltd, 2015.
ISBN: 978-1910591000.

[10] Scott Cranton. *Apache Camel developer's handbook solve common integration tasks with over 100 easily accessible Apache Camel recipes*.
Birmingham, UK: Packt Pub, 2013. ISBN: 1782170308.

[11] Google Developers. *Analytics for Web (analytics.js) | Google Developers*.
https://developers.google.com/analytics/devguides/collection/analyticsjs/. 2016.

[12] Alexis Deveria. *Can I use... Support tables for HTML5, CSS3, etc*.
http://caniuse.com/#search=websocket. 2009.

[13] Peter Eckersley. *How Unique Is Your Web Browser?*
https://panopticlick.eff.org/static/browser-uniqueness.pdf. 2010.

[14] Eric Enge. "Matt Cutts Interviewed by Eric Enge". In: ().

[15] Council of the European Union European Parliament.
Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services.
http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32009L0136. 2009.

[16] Eric Fettman.
A Sweet Treat, But Users Delete: Cookies and Cookie Deletion in Google Analytics.
https://www.e-nor.com/blog/google-analytics/cookies-and-cookie-deletion-in-google-analytics. 2015.

[17] Yuzo Fujishima. *Web Sockets Now Available In Google Chrome*.
http://blog.chromium.org/2009/12/web-sockets-now-available-in-google.html. 2009.

[18] Qi Guo a Eugene Agichtein.
"Towards predicting web searcher gaze position from mouse movements".
In: *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*.

Association for Computing Machinery (ACM), 2010. ISBN: 9781605589305.
DOI: 10.1145/1753846.1754025.
URL: http://dx.doi.org/10.1145/1753846.1754025.

[19] Ian Hickson. *The WebSocket API*. Working Draft.
https://www.w3.org/TR/2008/WD-html5-20080610/. W3C, červ. 2008.

[20] Ian Hickson. *The WebSocket API*. Candidate Recommendation.
http://www.w3.org/TR/2012/CR-websockets-20120920/. W3C, zář. 2012.

[21] Ian Hickson a Inc. Google. *Web SQL Database*. Working Group Note.
https://www.w3.org/TR/webdatabase/. W3C, lis. 2010.

[22] hitwebcounter. *Free Counter,Page Counter,Web Counter,Web Counter Code, Webpage Counters, Free internet counter, WebSite Counter, html counter, PHP, ASP, webtracker*.
http://www.hitwebcounter.com/.

[23] Jeff Huang, Ryen White a Georg Buscher. "User see, user point".
In: *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. Association for Computing Machinery (ACM), 2012.
ISBN: 9781450310154. DOI: 10.1145/2207676.2208591.
URL: http://dx.doi.org/10.1145/2207676.2208591.

[24] Avinash Kaushik. *A Primer On Web Analytics Visitor Tracking Cookies*.
http://www.kaushik.net/avinash/web-analytics-visitor-tracking-cookies/. 2009.

[25] Colleen Kehoe et al. "GVU's 10th WWW User Survey". In: ().

[26] Wendy Kellogg. "CHI 2005 : technology, safety, community : conference proceedings : Conference on Human Factors in Computing Systems : Portland, Oregon, USA, April 2-7". In: (2005).

[27] David M. Kristol. *HTTP State Management Mechanism*. Standards Track.
https://www.ietf.org/rfc/rfc2965.txt. IETF, říj. 2000.

[28] FARNEY & MCHALE. *MAXIMIZING GOOGLE ANALYTICS*.
ALA Editions, 2014. ISBN: 0838958923.
URL: http://www.amazon.com/MAXIMIZING-GOOGLE-ANALYTICS-FARNEY-MCHALE/dp/B00KIFSUR2%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB00KIFSUR2.

[29] Keaton Mowery a Hovav Shacham.
Pixel Perfect: Fingerprinting Canvas in HTML5.
http://w2spconf.com/2012/papers/w2sp12-final4.pdf. 2012.

[30] Florian Mueller a Andrea Lockerd. "Cheese".
In: *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*.
Association for Computing Machinery (ACM), 2001. ISBN: 1581133405.
DOI: 10.1145/634067.634233.
URL: http://dx.doi.org/10.1145/634067.634233.

[31] Martin Mulazzani et al.
Fast and Reliable Browser Identification with JavaScript Engine Fingerprinting.
http://www.w2spconf.com/2013/papers/s2p1.pdf. 2013.

[32] Vidhya Navalpakkam a Elizabeth Churchill. "Mouse tracking".
In: *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. Association for Computing Machinery (ACM), 2012.
ISBN: 9781450310154. DOI: 10.1145/2207676.2208705.
URL: http://dx.doi.org/10.1145/2207676.2208705.

[33] Cram101 Textbook Reviews. *e-Study Guide for: Web Application Architecture: Principles, Protocols and Practices: Computer science, Computer science*.
Cram101, 2014. URL: http://www.amazon.com/-Study-Guide-Application-Architecture-Principles-ebook/dp/B008VB8QLA%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB008VB8QLA.

[34] Kerry Rodden et al.
"Eye-mouse coordination patterns on web search results pages".
In: *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08*.
Association for Computing Machinery (ACM), 2008.
DOI: 10.1145/1358628.1358797.
URL: http://dx.doi.org/10.1145/1358628.1358797.

[35] SEOMoz. *Google Algorithm Change History*.
https://moz.com/google-algorithm-change.

[36] Sig Software. *Analog Helper - Mac GUI for Web Log Analysis*.
http://www.sigsoftware.com/analoghelper/screenshots.html.

[37] Valve. *GitHub - Valve/fingerprintjs2: Modern & flexible browser fingerprinting library, a successor to the original fingerprintjs*.
https://github.com/Valve/fingerprintjs2.

[38] W3Techs. *Usage of traffic analysis tools for websites*. https://w3techs.com/technologies/overview/traffic_analysis/all.

[39] Amy Weinberger.
The Impact of Cookie Deletion on Site-Server and Ad-Server Metrics in Australia.
`http://www.comscore.com/content/download/7251/125689/file/Impact+of+Cookie+Deletion+Australia_January+2011.pdf`. 2011.

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

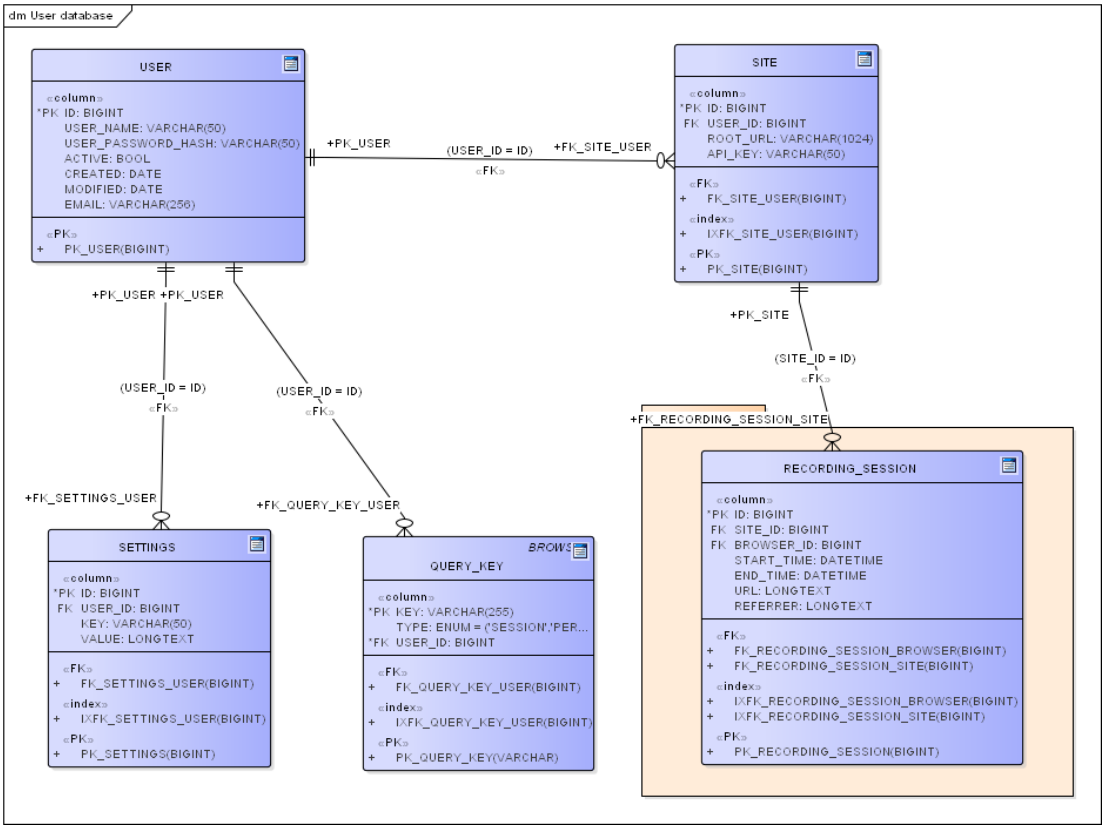
Přílohy

Příloha č. 1: CD-ROM

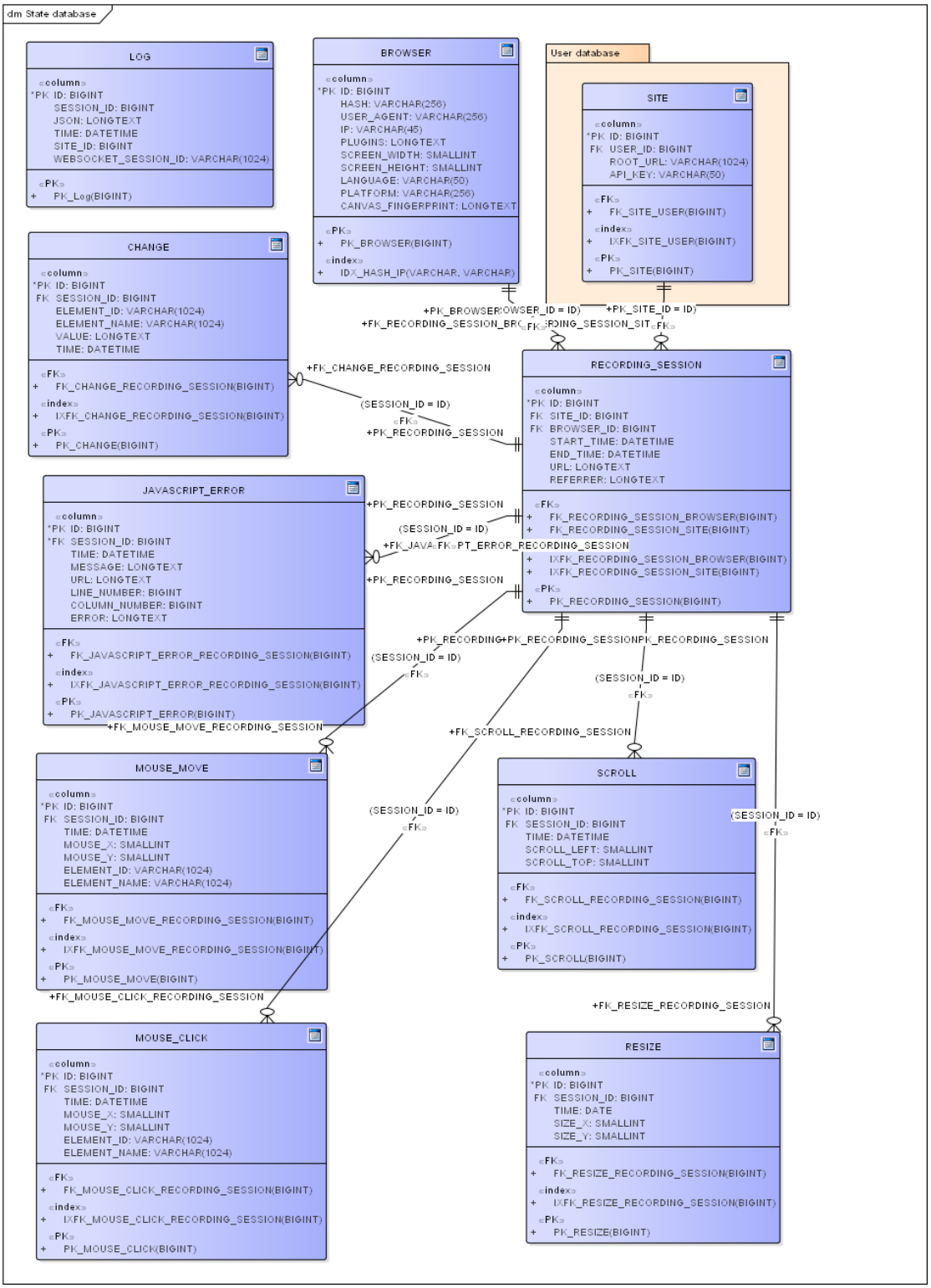
Přiložený CD-ROM obsahuje tyto soubory:

- standalone.xml: Konfigurace Wildfly 9.0.2.Final.
- prace.pdf: Tato diplomová práce v PDF formátu.
- zdrojove_kody: Zdrojové kódy klientského a serverového modulu.
- trouse-ddl.sql: DDL databáze pro MariaDB.
- EA.xml: XMI soubor určený k importu do programu Enterprise Architect

Příloha č. 2: Uživatelská databáze



Příloha č. 3: Stavová databáze



Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bednář Jan	Lovčice 55, Lovčice	I1301691

TÉMA ČESKY:

Rozšířená webová analytika

TÉMA ANGLICKY:

Extended web analytics

VEDOUCÍ PRÁCE:

Ing. Karel Mls, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Návrh a implementace aplikace pro sběr dat o chování uživatelů na webových stránkách skládající se z klientské a serverové části.
Osnova:
Úvod
Cíl práce, volba metodologie, způsob řešení
Rešerše tématu
Zhodnocení možných přístupů a možnosti integrace s Google analytics
Návrh a implementace klientské a serverové části aplikace
Shrnutí výsledků
Závěry a doporučení
Seznam použité literatury

SEZNAM DOPORUČENÉ LITERATURY:

CLIFTON, Brian. Advanced web metrics with Google Analytics. John Wiley & Sons, 2012.
THEUWISSEN, Albert; ROKS, Edwin. Building a better mousetrap. SPIE's OE magazine, 2001, 29-32.
WADKAR, Sameer; SIDDALINGAIAH, Madhu; VENNER, Jason. Pro Apache Hadoop. Apress, 2014.
CRANTON, Scott, Korab, Jakub. Apache Camel Developer's Cookbook. Packt Publishing, 2013.

Podpis studenta: ..... **Datum:** 14.10.2015

Podpis vedoucího práce: ..... **Datum:** 14.10.2015