Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Visitortrack

Dokumentácia k inžinierskemu dielu
Tím 14

Vedúci tímu: Ing. Peter Krátky
Členovia: Bc. Tomáš Gaššo, Bc. Peter Paška, Bc. Michal Vantuch, Bc. Jakub Ďaďo, Bc. Dávid Slezák, Bc. Dávid Spišák, Bc. Erik Dzurňak
Názov tímu: tím 14
Akademický rok: 2015/2016

Základné informácie o dokumente

Názov dokumentu:	VisitorTrack - Dokumentácia k inžinierskemu dielu
Verzia:	3.0
Stav:	~~Draft~~ / ~~Tímové pripomienkovanie~~ / ~~1. odovzdanie/ Odovzdanie za zimný semester/~~ Finálne odovzdanie
Autor:	Tím 14

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Úvod

1.1 Účel dokumentu

Cieľom dokumentu je informovať o technických vlastnostiach a implementačných postupoch použitých počas troch projektových iterácií (šprinty) pri realizácii aplikácie VisitorTrack. Dokument bol zostrojený pre účely prvého kontrolného bodu projektu Tímový Projekt I, kde je jedným z dvoch odovzdávaných dokumentov.

1.2 Motivácia projektu

Väčšina vlastníkov webových stránok sa v súčasnosti snaží monitorovať návštevnosť svojej stránky a taktiež činnosť jednotlivých návštevníkov. Túto aktivitu vyvíjajú s cieľom zlepšiť a rozvíjať svoju stránku a napĺňať požiadavky verejnosti. Pre tento účel môžu využiť množstvo dostupných nástrojov, ktoré im na základe získaných údajov od návštevníka vytvoria výstupy rôznych typov a kvalít. Práve v kvalite, resp. presnosti výstupov majú súčasné nástroje značné nedostatky. Táto nepresnosť vyplýva zo spôsobu zberu dát, kedy prevažná väčšina nástrojov využíva ako majoritný zdroj informácií cookies. Najnovšie prieskumy dokázali, že až 30% používateľov internetu cookies pravidelne odstraňuje, resp. ich ani nepovoľuje vytvárať. Dôsledkom toho je, že jednotlivé monitorovacie nástroje nedokážu rozoznať, či daný návštevník webovú stránku už navštívil alebo nie, čím dochádza k duplikácii údajov a teda k nepresným poskytovaným výstupom. Ďalším dôležitým faktorom, ktorý znižuje kvalitu výstupov a ktorý nie všetky monitorovacie nástrojov dokážu ošetriť, sú boti. Aktivity botov na webe nie sú plnohodnotnými ľudskými činnosťami a pri získavaní údajov ich treba z tohto dôvodu ignorovať.

Naším hlavným cieľom je preto vytvoriť aplikáciu, ktorá dokáže zo získaných údajov rozpoznať vracajúceho sa, resp. nového návštevníka (aj bez použitia cookies) a taktiež správne identifikovať činnosti botov. Z takto získaných surových dát následne budú vlastníkovi webu poskytnuté relevantné výstupy (tabuľkové prehľady, grafy, heatmapy a pod.) v najvyššej možnej kvalite a presnosti.

1.3 Ciele projektu na zimný semester

Cieľom nášho projektu na zimný semester je vytvoriť webovú aplikáciu, ktorá bude schopná monitorovať aktivitu (najmä pohyby myšou) návštevníkov na vybraných webových stránkach. Na základe týchto informácií aplikácia dokáže vytvoriť a vhodne uchovávať modely návštevníkov, ktoré budú slúžiť ako porovnávacie kritérium pre rozlíšenie nového a vracajúceho sa návštevníka. Samotnému používateľovi (majiteľovi / správcovi monitorovanej stránky) následne budú poskytnuté najpresnejšie možné informácie o počtoch nových a vracajúcich sa návštevníkoch. Výsledky týchto štatistík o návštevnosti bude aplikácia zobrazovať v jednoduchých a ľahko čitateľných grafoch.

Okrem spomínanej funkčnej stránky sa v zimnom semestri zameriame aj na určenie úspešnosti párovanie modelov návštevníkov našej aplikácie, tzn. s akou pravdepodobnosťou dokáže naša aplikácia správne určiť, či návštevník je nový alebo nie.

Žiadaným bonusom našej snahy v zimnom semestri by bolo úspešné nasadenie aplikácie na externú webovú stránku (ktorá, nie je pod správou školy). Tento cieľ je ale podmienený implementovaním dostatočnej funkcionality, ktorá by bola lákavá pre cieľového používateľa.

1.4 Ciele projektu na letný semester

Cieľom letného semestra je doplniť ďalšiu funkcionalitu do existujúcej webovej aplikácie. Hovoríme tu o nových grafoch, filtrovania dát v grafoch. Taktiež sa chceme zamerať na integráciu nášho riešenia s existujúcim analytickým nástrojom, čím by sme používateľovi priniesli omnoho viac analytických pohľadov. Okrem grafov a integrácie s externým systémom sa pozrieme na animáciu pohybu sledovaných návštevníkov na stránkach. Keďže produkt bude takmer dokončený na konci semestra, chceme sa zúčastniť konferencie na škole, počas, ktorej by sme mohli prezentovať naše riešenie ľuďom z praxe a získať od nich dôležitú spätnú väzbu.

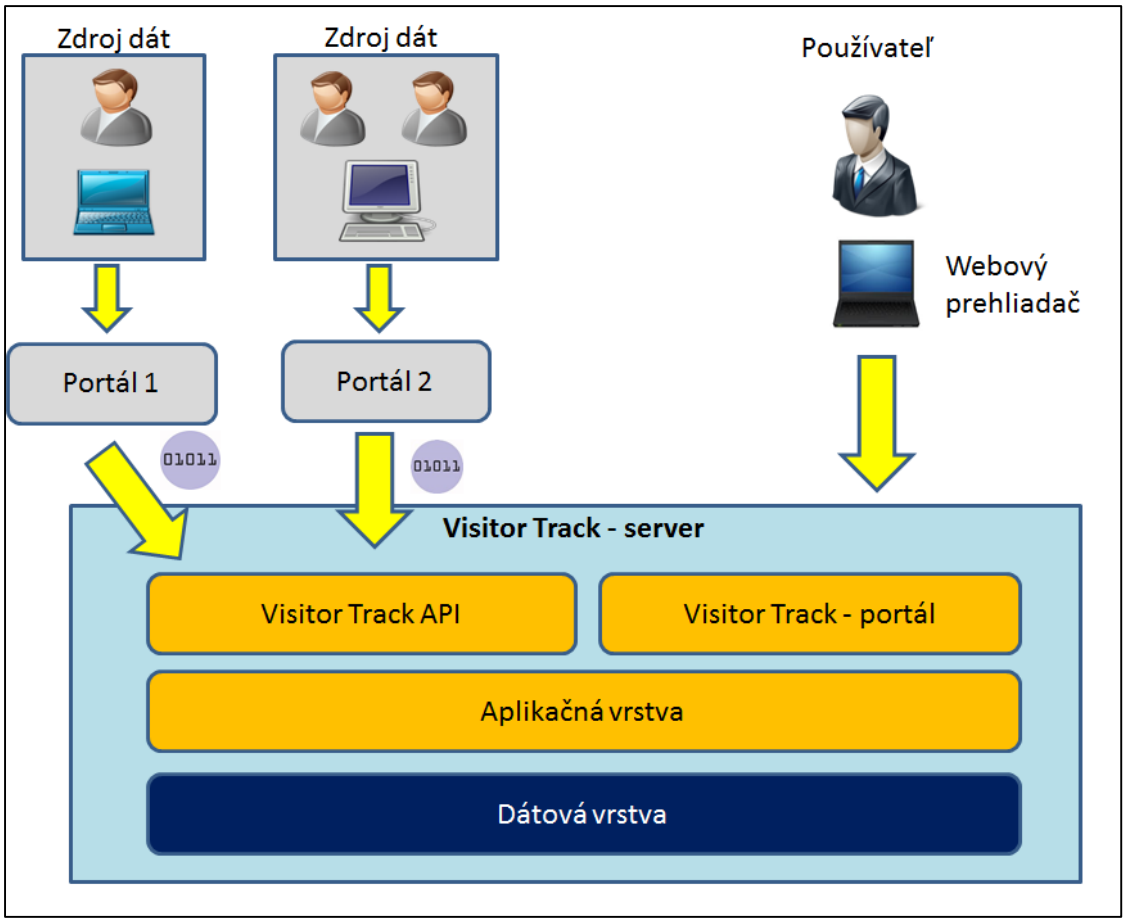
Z pohľadu backendu systému sa chceme zamerať na optimalizáciu precesu deduplikácie, ktorá vyhodnocuje podobnosť používateľov. Chceme ju optimalizovať z hľadiska škálovatelnosti ale aj presnosti.

Chceli by sme sa pozrieť aj na súčasný server, ktorý nám svojím výkonom nepostačuje a premigrovať všetky dáta na ďalšiu mašinu.

2 Globálny pohľad na systém

2.1 Architektúra systému

Naša aplikácia je postavená ako klient – server riešenie. Jednotliví používatelia pristupujú prostredníctvom webových prehliadačov (tenký klient) k službám, ktoré ponúka samotná aplikácia na serveri.
Dvojvrstvová architektúra klient – server (na trojvrstvovú nemáme dostatočné HW prostriedky a v súčasnej fáze projektu nie je ani potrebná) umožňuje jednoduchý prístup k aplikácii z každého počítača, ktorý má prístup na internet. Pretože je klient riešený ako tenký, hardvérové požiadavky sú kladené len na stranu servera, kde beží samotná aplikácia a preto nie sú naši používatelia obmedzený výkonom ich počítača. Štýl klient – server taktiež umožňuje realizovať aktualizácie systému bez nutnosti niečo meniť na strane klienta a preto môžeme aplikáciu nasadiť do produkčného prostredia už so základnou funkcionalitou. Ako budú následne doimplementovávané ďalšie funkcie, nie je potrebné (vo väčšine prípadov) navštíviť používateľa a niečo u neho meniť.



Obrázok 1- HL architektúra aplikácie VisitorTrack

2.2 Biznis architektúra

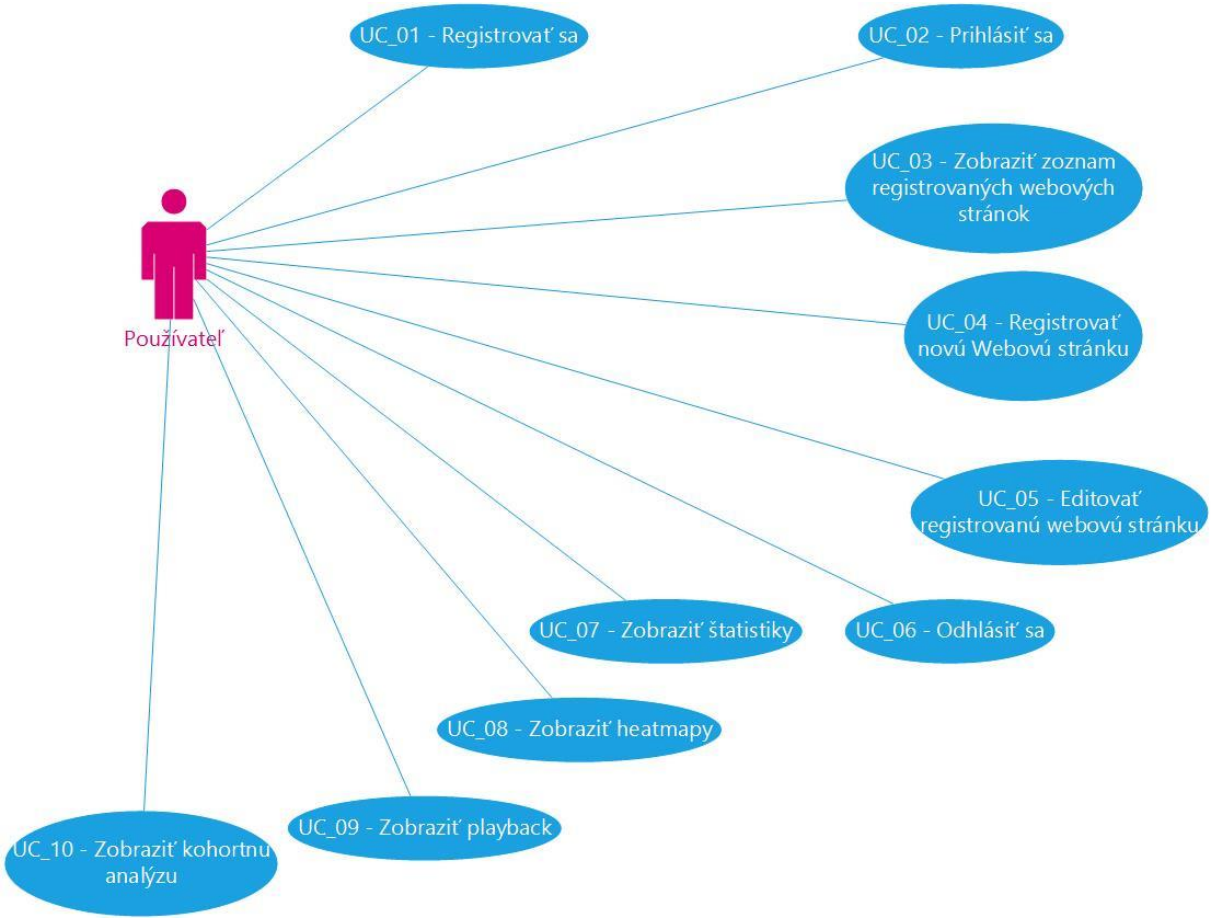
Z biznisového hľadiska našaaplikáciaposkytuje služby pre majiteľov, resp. správcov webových stránok, ktoré im umožňujú monitorovať návštevníkov svojej stránky a vďaka tomu prispôsobovať a zlepšovať svoju webovú stránku, či už z používateľského alebo dizajnového hľadiska.

V našom projekte rozpoznávame dva typy účastníkov:

1. **Používateľ** – majiteľ / správca webovej stránky, ktorý bude cielene používať našu aplikáciu s cieľom monitorovať svoju webovú stránku, resp. webové stránky.
2. **Návštevník** – v kontexte nášho projektu sú návštevníci nepriamymi účastníkmi a pre našu aplikáciu predstavujú len zdroj dát, na základe ktorých budeme vytvárať výstupy. Našou snahou je práve monitorovať činnosti návštevníkov na webových stránkach.

Prehľad základných prípadov použitia, ktoré aplikácia momentálne poskytuje:

- Registrovať sa
- Prihlásiť sa
- Zobraziť zoznam registrovaných webových stránok
- Registrovať novú webovú stránku
- Editovať registrovanú webovú
- Zobraziť štatistiky
    - Zobraziť počet návštevníkov za zvolené obdobie



Obrázok 2 - UseCase diagram

Stručný popis prípadov použitia zobrazených naObrázok 2 - UseCase diagram:

UC_01 - Registrovať sa

Majiteľovi, resp. správcovi webového portálu (ďalej len používateľ) aplikácia umožní vytvoriť si vlastný účet, prostredníctvom ktorého bude môcť sledovať štatistiky o návštevníkoch, ktoré naša aplikácia poskytuje. Registrácia vyžaduje zadanie e-mailu a hesla.

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Prístup na internet. • Účet s používateľovým e-mailom ešte nie je vytvorený.
Dôsledky	• Používateľ sa bude môcť prostredníctvom zadaných prihlasovacích údajov (e-mail a heslo) prihlásiť do aplikácie VisitorTrack.
Základný tok – Úspešná registrácia	1. Používateľ do webového prehliadača zadá URL adresu aplikácie VisitorTrack. 2. Webový prehliadač zobrazí úvodnú stránku aplikácie VistorTrack. 3. Používateľ zvolí možnosť „*Signup*“ pre vytvorenie nového účtu. 4. Webový prehliadač zobrazí registračný formulár.

	5. Používateľ vyplní registračný formulár zadaním e-mailu, hesla a opätovným zadaním hesla. Registráciu následne potvrdí možnosťou „*Register*“. 6. Aplikácia VisitorTrack vykoná validáciu zadaných registračných údajov, ktorá skončí úspešne. 7. Webový prehliadať po úspešnej registrácii zobrazí dashboard novo registrovaného používateľa.
Alternatívny tok 1 – Účet s mailom existuje	Alternatívny tok 1 nastane, ak používateľ pri registrácii zadá e-mailovú adresu, ktorá je už v aplikácii registrovaná. Alternatívny tok 1 sa spustí, ak krok 6 v Základnom scenári skončí neúspešnou validáciou. Vykonajú sa kroky 1-5 základného scenára. Následne: 1. Aplikácia VisitorTrack vykoná validáciu zadaných registračných údajov, ktorá skončí neúspešne. 2. Webový prehliadač používateľa informuje, že účet s danou e-mailovou adresou už existuje a vyžiada zadanie nového e-mailu. 3. Používateľ zadá do registračného formulára registračné údaje s novou e-mailovou adresou a zvolí možnosť „Register“. Alternatívny tok 1 pokračuje krokom 6 Základného toku.
Alternatívny tok – Nesprávne overenie hesla	Alternatívny tok 2 nastane, ak používateľ pri registrácii zadá rozdielne heslá. Alternatívny tok 2 sa spustí, ak krok 6 v Základnom scenári skončí neúspešnou validáciou. Vykonajú sa kroky 1-5 základného scenára. Následne: 1. Aplikácia VisitorTrack vykoná validáciu zadaných registračných údajov, ktorá skončí neúspešne. 2. Webový prehliadač používateľa informuje, že zadané heslá sa nezhodujú a vyzve používateľa na ich opätovné zadanie. 3. Používateľ znovu zadá do registračného formulára registračné údaje a zvolí možnosť „Register“. Alternatívny tok 1 pokračuje krokom 6 Základného toku.

Tabuľka 1 - UC_01 - Registrovať sa

UC_02 - Prihlásiť sa

Každému registrovanému používateľovi aplikácia VisitorTrack umožňuje opätovné prihlásenie k svojmu účtu a to zadaním e-mailu a hesla, ktoré si zvolil pri registrácii. V prípade zadania nesprávnych prihlasovacích údajov je používateľ informovaný chybovou správou. Počet pokusov o prihlásenie nie je obmedzený, preto nie je možné aby si používateľ zablokoval už existujúci účet.

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Prístup na internet. • Používateľ je už registrovaný v aplikácii VisitorTrack. • Používateľ pozná svoje prihlasovacie údaje.
Dôsledky	• Používateľ je úspešne prihlásený do aplikácie VisitorTrack.
Základný tok – Úspešné prihlásenie	1. Používateľ do webového prehliadača zadá URL adresu aplikácie VisitorTrack. 2. Webový prehliadač zobrazí úvodnú stránku aplikácie VistorTrack. 3. Používateľ zvolí možnosť „*Sign in*“ pre prihlásenie sa do

	aplikácie. 4. Webový prehliadač zobrazí prihlasovací formulár. 5. Používateľ do prihlasovacieho formuláru zadá svoje prihlasovacie údaje a potvrdí ich možnosťou „*Log in*“. 6. Aplikácia VisitorTrack vykoná validáciu zadaných prihlasovacích údajov, ktorá skončí úspešne. 7. Webový prehliadač po úspešnom prihlásení zobrazí dashboard používateľa.
Alternatívny tok 1 – Nesprávne prihlasovacie údaje	Alternatívny tok 1 nastane, ak používateľ pri prihlasovaní zadá nesprávne prihlasovacie údaje. Alternatívny tok 1 sa spustí, ak krok 6 v Základnom scenári skončí neúspešnou validáciou. Vykonajú sa kroky 1-5 základného scenára. Následne: 1. Aplikácia VisitorTrack vykoná validáciu zadaných prihlasovacích údajov, ktorá skončí neúspešne. 2. Webový prehliadač používateľa informuje, že zadal nesprávny e-mail alebo heslo a vyzve ho na opätovné vloženie údajov. Alternatívny tok 1 pokračuje krokom 5 Základného toku.

Tabuľka 2 - UC_02 - Prihlásiť sa

UC_03 – Zobraziť zoznam registrovaných webových stránok

Aplikácia prostredníctvom webového prehliadača umožní prihlásenému používateľovi zobraziť webové stránky, ktoré má používateľ pre daný účet registrované. Jednotlivé webové stránky sú zobrazené v tabuľke, so stĺpcami:

1.stĺpec – poradové číslo webovej stránky.

2.stĺpec „*Web page*“ – názov stránky.

3.stĺpec „*API key*“ – vygenerovaný unikátny kód pre registrovanú stránku.

4.stĺpec„*Active*“ – informuje, či aplikácia VisitorTrack zaznamenáva informácie o návštevníkoch stránky alebo nie.

5.stĺpec „*Registred*“ – dátum a čas, kedy bola stránka zaregistrovaná.

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack.
Dôsledky	• Používateľovi aplikácia VisitorTrack zobrazí prostredníctvom webového prehliadača zoznam jeho registrovaných webových stránok.
Základný tok – Zobrazenie zoznamu stránok	1. Používateľ klikne na svoj prihlasovací e-mail v pravom hornom rohu obrazovky. 2. Webový prehliadač zobrazí rolldown menu s možnosťami „*UserProfil*“, „*Settings*“ a „*Log out*“. 3. Používateľ zvolí možnosť „*Settings*“. 4. Webový prehliadač používateľovi zobrazí webové stránky, ktoré má registrované v aplikácii VisitorTrack.

Tabuľka 3 - UC_03 – Zobraziť zoznam registrovaných webových stránok

UC_04 - Registrovať novú webovú stránku

Každá webová stránka, z ktorej používateľ požaduje zaznamenávať informácie o návštevníkoch, musí byť v aplikácii VisitorTrack registrovaná. Pri registrácii je webovej

stránke vygenerovaný unikátni API kľúč. Tento je potrebné zadať do javascriptového kódu, ktorý si používateľ stránky musí vložiť do kódu svojej stránky. Toto je prvá podmienka pre monitorovanie návštevníkov. Druhou podmienkou je mať registrovanú webovú stránku označenú ako aktívnu v aplikácii VisitorTrack (možnosť „*Active*“). Túto možnosť je možné nastaviť pri registrácii (*UC_4 – Registrovať nový web*) alebo pri editácii stránky (*UC_05 – Editovať registrovaný web*).

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack. • Používateľ sa nachádza na zozname registrovaných webových stránok pre jeho účet.
Dôsledky	• Pridanie novej stránky do aplikácie VisitorTrack. Stránku je možné vidieť v zozname registrovaných webových stránok s vygenerovaným unikátnym API kľúčom.
Základný tok – Registrovanie novej webovej stránky	1. Používateľ klikne na možnosť „*Actions*“ v pravo hore nad zoznamom registrovaných webových stránok. 2. Webový prehliadač zobrazí rolldown menu s možnosťami „*Add web*“. 3. Používateľ zvolí možnosť „*Add web*“. 4. Webový prehliadač používateľovi zobrazí formulár pre pridanie, resp. zaregistrovanie novej webovej stránky. 5. Používateľ zadá názov stránky a zvolí možnosť *„Active“,* ak požaduje zaznamenávať informácie o návštevníkoch z registrovanej stránky. Registráciu stránky potvrdí možnosťou „*Create*“. 6. Aplikácia VisitorTrack presmeruje používateľa na zoznam registrovaných stránok, kde sa už nachádza aj práve zaregistrovaná stránka.

Tabuľka 4 - UC_04 - Registrovať novú webovú stránku

UC_05 - Editovať registrovanú webovú stránku

Aplikácia VisitorTrack umožňuje prihlásenému používateľovi editovať *názov* a „*active*“ stav registrovaných webových stránok.

Názov stránky je len informačný pre používateľa na nemá žiadnu ďalšiu funkcionalitu. **„Active“ stav** stránky určuje, či používateľ požaduje zaznamenávať, resp. monitorovať návštevníkov na zvolenej stránke.

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack. • Používateľ sa nachádza na zozname registrovaných webových stránok pre jeho účet.
Dôsledky	• Zmena názvu stránky alebo jej aktivácia / deaktivácia.
Základný tok – Editovanie webovej stránky	1. Používateľ klikne na názov stránky, ktorú požaduje editovať. 2. Webový prehliadač používateľovi zobrazí formulár pre editovanie názvu stránky, resp. jej aktiváciu / deaktiváciu. 3. Používateľ upraví názov stránky a prípadne stránku aktivuje, resp. deaktivuje. Vykonané zmeny potvrdí možnosťou „*Save*

	web“. 4. Aplikácia VisitorTrack presmeruje používateľa na zoznam registrovaných stránok, používateľ môže vidieť stránku už so zmeneným názvom, resp. jej aktualizovaný „*active*“ stav.

Tabuľka 5 - UC_05 - Editovať registrovanú webovú stránku

UC_06 – Odhlásiť sa

Aplikácia VisitorTrack umožňuje prihlásenému používateľovi odhlásiť sa zo svojho účtu prostredníctvom možnosti „*Log out*“.

Stručný popis prípadu použitia:

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack.
Dôsledky	• Odhlásenie prihláseného používateľa z aplikácie VisitorTrack.
Základný tok – Odhlásenie	1. Používateľ klikne na svoj prihlasovací e-mail v pravom hornom rohu obrazovky. 2. Webový prehliadač zobrazí rolldown menu s možnosťami „*UserProfil*“, „*Settings*“ a „*Log out*“. 3. Používateľ zvolí možnosť „*Log out*“. 4. Aplikácia VisitorTrack odhlási používateľa a presmeruje ho na úvodnú stránku, kde sa môže opätovne prihlásiť.

Tabuľka 6 - UC_06 – Odhlásiť sa

UC_07 - Zobraziť štatistiky

Prihlásený používatelia si môžu v našej aplikácii prezerať rôzne štatistiky o návštevníkoch na jeho registrovaných webových stránkach a to vo forme prehľadných a do istej miery filtrovateľných grafov.

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ)
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack. • Používateľ má registrované nejaké webové stránky v aplikácii VisitorTrack.
Dôsledky	• Zobrazenie grafov.
Základný tok – Odhlásenie	1. Používateľ klikne na záložku „Statistics“. 2. Aplikácia VisitorTrack prostredníctvom webového prehliadača zobrazí štatistické grafy.

Tabuľka 7 - UC_07 - Zobraziť štatistiky

UC_07–Zobraziť štatistiky – Alternatívny tok - Zobraziť počet návštevníkov za zvolené obdobie

Prihlásený používateľ môže pri grafoch sumarizujúcich počty návštevníkov meniť časový úsek, ktorý graf zachytáva. Po zmenení časového obdobia bude graf automaticky upravený a zobrazený používateľovi.

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ).
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack a má otvorenú záložku „Statistics“.

	• Používateľ má registrované nejaké webové stránky v aplikácii VisitorTrack.
Dôsledky	• Modifikácia grafu podľa zvoleného obdobia.
Základný tok – Odhlásenie	1. Používateľ na záložke „Statistics“ zvolí možnosť pre zmenu zobrazovaného času. 2. Webový prehliadač používateľovi ponúkne zoznam možných časových úsekov, pre ktoré je možné graf vykresliť. 3. Používateľ zvolí požadovaný časový úsek a potvrdí zobrazenie grafu. 4. Aplikácia VisiorTrack prostredníctvom webového prehliadača zobrazí graf pre zvolené časové obdobie.

Tabuľka 8 - UC_08 - Zobraziť počet návštevníkov za zvolené obdobie

UC_08 - Zobraziť heatmapy

Prihlásený používateľ si môže popri štatistikách zobraziť heatmapy konkrétnej stránky vo forme heatmapy. Na heatmape vidí miesto a intenzitu klikania. Aj na toto zobrazenie sa dá aplikovať časový filter.

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ).
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack a má otvorenú záložku „Behavior“. • Používateľ má registrované nejaké webové stránky v aplikácii VisitorTrack. • Používateľ má vo filtri vybranú stránku • Používateľ je v záložke heatmapa
Dôsledky	• Zobrazenie obrázku vybranej stránky so správaním používateľov
Základný tok – Generovanie Heatmapy	1. Používateľ zvolí stránku pre generáciu Heatmapy „Generovať Heatmapu“. 2. Používateľ zvolí časoé obdobie pre generáciu Heatmapy. 3. Používateľ klikne na tlačidlo „Generovať Heatmapu“

UC_09 – Zobraziť playback

Prihlásený používateľ si môže popri štatistikách zobraziť animáciu pohybu používateľov na vopred vybranej stránke.

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ).
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack a má otvorenú záložku „Behavior“. • Používateľ má registrované nejaké webové stránky v aplikácii VisitorTrack. • Používateľ má vo filtri vybranú stránku • Používateľ je v záložke playback
Dôsledky	• Zobrazenie obrázku vybranej stránky s animáciou pohybu návštevníkov používateľovej stránky
Základný tok – Playback	1. Používateľ zvolí stránku, na ktorej sa bude prehrávaťsprávanie používateľov 2. Používateľ zvolí časové obdobie pre ktoré bude používateľovi prehraná animácia. 3. Používateľ klikne na tlačidlo „Vykresliť pohyb návštevníkov“

UC_10 – Zobraziť kohortnú analýzu

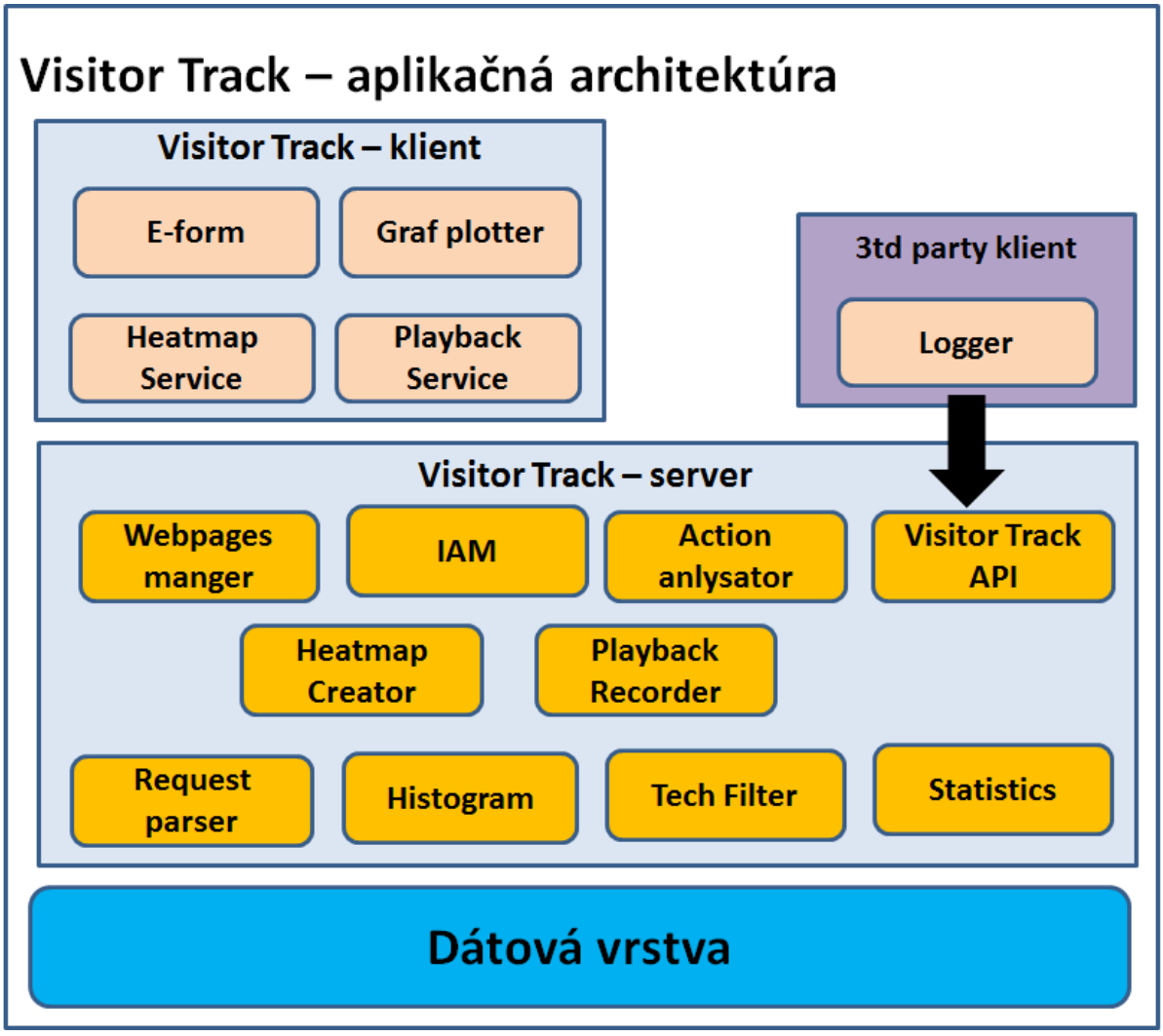
Používateľ má možnosť zobraziť kohortnú analýzu jeho registrovanej stránky v našom systéme.

Primárny používateľ	• Majiteľ / správca webového portálu (ďalej len používateľ).
Predpoklady	• Používateľ je prihlásený v aplikácii VisitorTrack a má otvorenú záložku „Statistics“. • Používateľ má registrované nejaké webové stránky v aplikácii VisitorTrack.
Dôsledky	• Prehľaď spôsobom kohortnej analýzy
Základný tok – Playback	1. Používateľ na záložke „Statistics“ zvolí možnosť pre zmenu zobrazovaného času. 2. Webový prehliadač používateľovi ponúkne zoznam možných časových úsekov, pre ktoré je možné vykresliť kohortnú analýzu. 3. Používateľ zvolí požadovaný časový úsek a potvrdí zobrazenie kohortnej analýzy. 4. Aplikácia VisiorTrack prostredníctvom webového prehliadača zobrazí kohortnú analýzu.

2.3 Aplikačná architektúra

Samotnú aplikáciu VisitorTrack je možné rozdeliť na niekoľko spolupracujúcich komponentov / modulov. Prevažná časť komponentov a teda aj samotnej aplikačnej logiky sa nachádza na strane servera, zatiaľ čo klientská časť slúži len na zobrazovanie výsledkov v požadovanej forme (text, tabuľky, grafy, ikony a pod.).

Poznámka: Aplikačný pohľad na našu aplikáciu, ktorý je predstavený v tejto kapitole je založený na zoskupení rovnakých funkcionalít do aplikačných komponentov na určitej úrovni abstrakcie. Aj keď vo väčšine prípadov je možné v aplikácii presne identifikovať popisované komponenty, nie vždy je to možné. Ako príklad môžeme uviesť komponent E-form, ktorý neobsahuje a nespravuje všetky dostupné elektronické formuláre, s ktorými používateľ pracuje. Pretože sú ale takéto elektronických formuláre súčasťou našej aplikácie a majú rovnakú funkcionalitu, bolo vhodnejšie vytvoriť abstraktný komponent, ktorý formuláre zastrešuje.



Obrázok 3 - Aplikačná architektúra

2.3.1 Klient VisitorTrack - komponenty

Prehľad realizovaných komponentov nachádzajúcich sa na strane klienta:

E-form - úlohou komponentu je prijímať základné vstupy od používateľa. Momentálne *E-form* obsahuje elektronické formuláre pre registráciu a prihlásenie používateľa ako aj pre registráciu a editáciu webovej stránky. *E-form* taktiež bude realizovať aj základné validácie vložených údajov, ktoré ale nie sú momentálne implementované.
Graf plotter –komponent zabezpečuje zobrazovanie grafov z údajov nachádzajúcich sa v databáze aplikácie. Graf plotter bude časom umožňovať aj meniť typ a mierku grafu ako aj dynamicky nastaviť zobrazované obdobie (aktuálne je zabezpečené len výber z ponúkaných časových intervalov).
Heatmap Service – komponent umožňuje klientov zobraziť vytvorené heatmapy, ktoré sú uložené na servery.
Playback Service - úlohou komponentu je na základe získaných koordinátov prehrať pohyb myši na vybranej podstránke a pre vybraného návštevníka.

2.3.2 Server VisitorTrack - komponenty

Prehľad realizovaných komponentov nachádzajúcich sa na strane servera:

VisitorTrack API – nie je plnohodnotnýmkomponentom (je súčasťou komponentu *Requestparser*) ale skôr rozhraním, ktoré ponúkaREST službu. Táto služba je volaná komponentom *Logger*z klienta tretích strán (webová stránka používateľa).
Requestparser – komponent zabezpečujúci spracovanie prijatých údajov z requestu. Každý request obsahuje JSON, ktorý tento komponent dokáže spracovať a následne uložiť do databázy, kde už k týmto dátam majú prístup iné komponenty.
Actionanalysator – úlohou tohto komponentu je správne spracovať jednotlivé akcie používateľov, ktoré boli získané z monitorovaných webových stránok. Komponent pristupuje k dátam v databáze, ktoré pripravil komponent *Requestparser*.
IAM – IAM je skratka pre identifikačný a autentifikačný modul, resp. komponent. IAM zabezpečuje registráciu, identifikáciu a autentifikáciu jednotlivých používateľov našej aplikácie.
Histogram – zabezpečuje vytvorenie modelu návštevníka na základe záznamov z aktivít jednotlivých návštevníkov monitorovanej webovej stránky. Model používateľa je jedným z identifikačných parametrov, pomocou ktorého dokážeme určiť, či už používateľ navštívil stránku alebo nie. Komponent Histogram zabezpečuj aj deduplikáciu, resp. párovanie nových návštevníkov s tými, ktorí sa úž nachádzajú v databáze.
Tech Filter – komponent zmenšuje množinu návštevníkov, ktorý sa následne budú porovnávať v komponente *Histogram.Tech Filter*hľadá zhody na základe vybraných technických parametrov ako sú platforma, prehliadač, jazyk a pod.
Webpagesmanger – umožňuje základnú správu monitorovaných webových stránok pre registrovaného používateľa. Poskytuje funkcionalitu registrovania novej stránky a editovania už existujúcej stránky.

Statistics–pripravenie vstupov pre komponent *Graf plotter*. Komponent vykonáva rôzne štatistické selecty z databázy, ktorých výstupy sú následne odoslané klientovi, kde sú zobrazené prostredníctvom *Graf plotter*-u.
VisitorSessions – komponent z uložených informácií o aktivitách jednotlivých návštevníkov identifikuje jednotlivé sedenia návštevníka pre celú doménu ako aj pre konkrétne podstránky. Tieto údaje následne vhodne uloží do databázy a tak poskytne komponentu *Statistics,* na ďalšie spracovanie.
Heatmap creator – komponent plní dve funkcionality. Prvou je na základe vybranej URL vytvoriť screen požadovanej stránky. Tieto screeny sú následne dočasne uložené na server vo formáte png. Druhou funkcionalitou je z uložených koordinátov pre odfotenú stránku vybrať tie, ktoré predstavujú kliky návštevníkov a tie zobraziť na uloženom png screene.
Playback recorder – komponent spolupracuje podobne ako komponent Heatmap creator, len s tým rozdielom, že vyberá koordináty, ktoré predstavujú pohyb mišy a nie kliky a tie následne pripraví pre komponent **Playback service**.

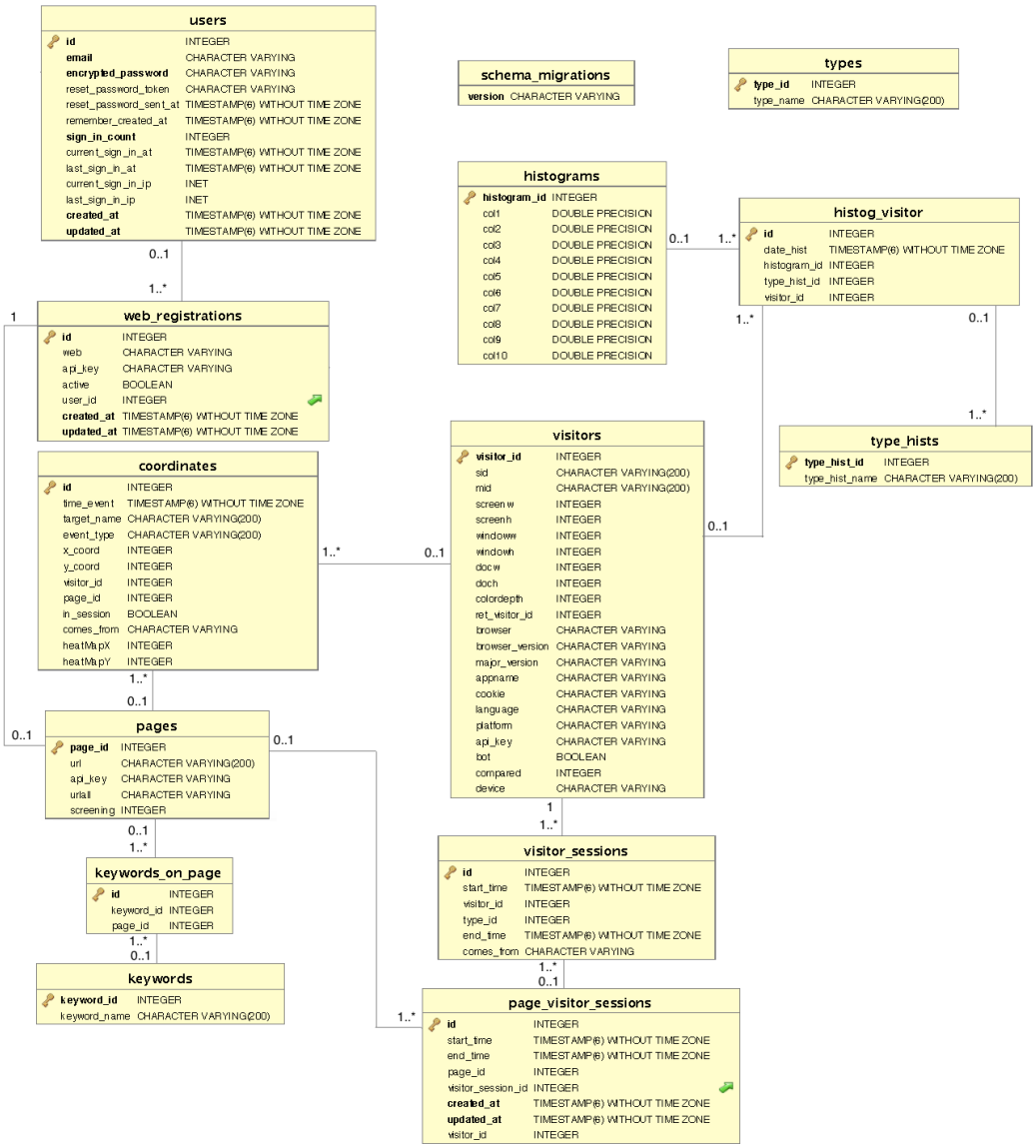
2.3.3 Klient 3. strana – komponenty

Prehľad komponentov nachádzajúcich sa na strane klienta 3. strán (webové stránky používateľov našej aplikácie):

Logger – komponent zaznamenáva aktivitu jednotlivých návštevníkov na webovej stránke. Takto získané informácie sú upravené do formátu JSON, kedy následne komponent zavolá REST službu, ktorú poskytuje komponent *VisitorTrack API* na strane servera.

2.4 Dátový model

Dátový model našej aplikácie bol vytváraný v niekoľkých iteráciách podľa aktuálnych potrieb a požiadaviek na zabezpečenie funkčnej stránky aplikácie. Súčasný dátový model, nad ktorým pracuje naša aplikácia je zobrazený na*Obrázok 4 - Dátový model*. Pod obrázkom sa nachádzajú stručné popisy najdôležitejších entít nášho dátového modelu.



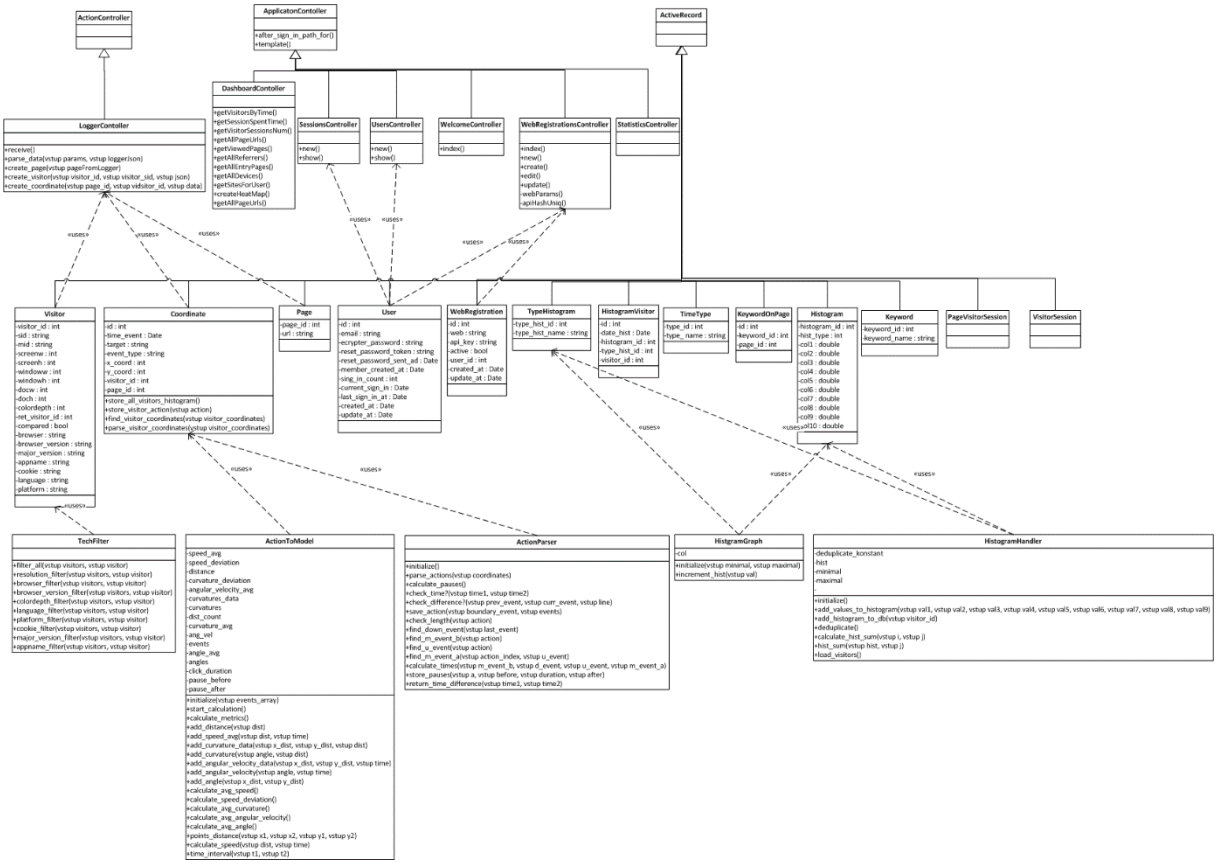
Obrázok 4 - Dátový model

Stručný popis jednotlivých entít dátového modelu:
User –používateľ aplikácie VisitorTrack.
WebRegistration- zákaznícka registrácia web stránok pre získanie API kľúča potrebného pre funkčnosť komponentu *Logger*.
Coordinate- raw dáta prichádzajúce z komponentu *Logger*.
Keyword- zaznamenané kľúčové slová, pre rozšírenejšiu identifikáciu návštevníka.

KeywordonPage- vytvorené prepojenie medzi kľúčovými slovami a stránkou.
Page- zaznamenané web stránky zákazníkov, na ktorých beží náš softvér a loguje dáta o návštevníkoch.
Time- dáta z komponentu *Logger* zadelené pre jednotlivých návštevníkov, stránky a typy udalostí, potrebných pre správne vytváranie histogramov(komponent *Histogram*).
TimeType- typy udalostí rozložených do časových okien.
TypeHistogram- typy histogramov (číselník).
Histogram- model návštevníka vyjadrený stĺpcami v histograme, získaný/vypočítavaný na základe dát z komponentu *Logger* (z pohybového zariadenia).
HistogramVisitor- vytvorené prepojenie medzi návštevníkom a jeho histogramami.
Visitor- návštevník stránky, ktorému sa logujú akcie vykonané pohybovým zariadením, a jeho preferencie pri používaní stránky (dáta z cookies).

2.5 Diagram tried

Na obrázku je zobrazený aktuálny diagram tried po letnom semestri.



Obrázok 5 - Diagram tried

Diagram obsahuje aj triedy, ktoré neboli zatiaľ ešte použité. Príkladom takýchto tried je: Keyword a KeywordOnPage, ktoré boli vytvorené s predstihom a používať sa budú až v nasledujúcich fázach projektu.

2.6 Prehľad e-dokumentov

V súčasnej fáze projektu neboli vytvorené žiadne elektronické dokumenty, ktoré by boli využité pri tvorbe tohto dokumentu.

3 VisitorTrack – komponenty

Kapitola poskytuje stručný popis komponentov, ktoré boli doteraz implementované.

3.1 Logger

Logger je komponent, ktorý slúži na zachytávanie aktivity používateľa na webe. Používa sa na webovej stránke klienta, ktorý komunikuje so serverom našej aplikácie prostredníctvom REST služby.

Analýza
Komponent *Logger* nebol implementovaný naším tímom v celom rozsahu. Ako základ bol použitý existujúci kód, ktorý náš produktový vlastník používal už na niektorých predchádzajúcich projektoch a pre potreby tohto projektu nám ho sprístupnil. Pretože samotná funkcionalita sprístupneného loggera nebola úplne zhodná s našimi potrebami, bolo nutné vykonať implementačné úpravy.

Logger má slúžiť na zber dát, pozostávajúcich hlavne z akcií myši, ktoré používateľ počas pobytu na webovej stránke vykoná. Súčasne by sme mali jeho prostredníctvom zistiť, čo najviac informácií a dát o používateľovi z webového prehliadača. Môžeme tu zaradiť rozlíšenie okna, rozlíšenie obrazovky, typ a verziu prehliadača, platformu, sýtosť farieb, odkiaľ na sledovanú prišiel a pod. Toto nám pomôže lepšie analyzovať a namodelovať používateľa. Logger má byť efektívny a rýchly, aby nezaťažoval stránku klienta. Súčasne musí byť jednoducho a rýchlo implementovaný, tak aby to zvládol aj neskúsený používateľ.

Tabuľka 9 - Zoznam logovaných dát na webovej stránke

Názov udalosti (v anglickom jazyku)	Popis udalosti
mousemove	Pohyb myši
mouseover	Pohyb myši cez element stránky
mouseout	Vykoná sa po opustení elementu stránky
mousedown	Vykoná sa, pri prejdení po elemente stránky so súčasne stlačením tlačidlom myši
mouseup	Vykoná sa, pri prejdení

	po elemente stránky so súčasne pustením tlačidlom myši
click	Kliknutie myši
dblclick	Dvojité kliknutie myši
blur	Zaznamená odchod zo stránky
focus	Zaznamená príchod na stránku
scroll	Zaznamená posúvanie stránky
size	Zaznamená veľkosť prehliadača, rozlíšenie počítača
referrer	Zaznamená referujúcu (predošlú) stránku

Na zistenie typu používaného prehliadača a v akom „tabe“ stránky sa používateľ nachádza používame lokálne úložisko prehliadača (ang. localstorage).

Návrh

Dáta z *Loggera* sa posielali ako čistý text oddelené od seba oddelovačom „|“ a na strane servera sa ukladali do textového súboru. Čo naším požiadavkám – ukladanie do databázy úplne nevyhovoval. Pre odosielanie zaznamenaných údajov z *Loggera* sme sa nakoniec rozhodlipoužiť formát JSON, ktorý v súčasnosti patrí medzí štandardy vo webovej komunikácií. Použitím tohto formátu sa vyhneme parsovaniu dát pomocou čistého textu, čím odľahčíme procesor servera, čo sa pri množstve prijímaných údajov určite prejaví. Zároveň dokážeme jednoduchšie a efektívnejšie ukladať dáta do databázy pomocou serializácii a deserializácii JSON formátu. Navrhnutá a používaná štruktúra JSON-u je zobrazená na*Obrázok 6 - Návrh JSON formátu.*

```
{
	"visitor" : {
		"mid" : "",
		"sid" : "",
		"page" : ""
	},
	"size" : {
		"screenw" : "",
		"screenh" : "",
		"windoww" : "",
		"windowh" : "",
		"docw" : "",
		"doch" : "",
		"colordepth" : "",
		"timezone" : "",
		"browser" : "",
		"browser_versiom" : "",
		"major_version" : "",
		"appname" : "",
		"cookie" : "",
		"language" : "",
		"platform" : "",
		"comes_from" : "",
		"bot" : "",
		"api_key" : "",

	},
	"data" : [{
			"datetime" : "",
			"eventtype" : "",
			"corx" : "",
			"cory" : "",
			"targetName" : ""
		}
```

Obrázok 6 - Návrh JSON formátu

Visitor

- mid – unikátny identifikátor používateľa.
- sid – na akom tabe sa používateľ nachádzal.
- page – z akej stránky/podstránky mi prišli zalogované údaje.

Size

- screenw – šírka monitora.
- screenh – výška monitora.
- windoww – šírka okna webového prehliadača.
- windowh – výška okna webového prehliadača.
- doch – výška dokumentu, často krát je to <body> element.
- docw – šírka dokumentu, často krát je to <body> element.
- colordepth – hĺbka farieb obrazu (24bit, 32bit).
- timezone – časová zóna z ktorej boli údaje zozbierané.
- browser – názov prehliadača.
- browser_version – presná verzia prehliadača.
- major_version – hlavná verzia prehliadača.
- cookie – informácia a povolení cookies.
- language – jazyk prehliadača.
- platform – platforma, na ktorej je nainštalovaný prehliadač.
- comes_from – adresa z ktorej sa návštevník dostal na stránku
- bot – informácia či stránku navštívil bot, respektíve crawler
- api_key – apikľúčA

Data– pole, ktoré tvoria x,y súradnice udalosti vykonanej a zalogovanej na stránke:

- datetime – čas vykonanie udalosti
- eventtype – názov udalosti
- corx – súradnice x (pixel na osi X)
- cory – súradnice y (pixel na osi Y)
- targetName – element, na ktorom bola vykonaná udalosť

Udalosti, ktoré boli zachytené sa automaticky uložia do formátu JSON (obr. 1) v reálnom čase a sú pripravené na odoslanie. Odosielanie dát sme museli vylepšiť, pretože webové prehliadače správne nepodporovali funkcionalitu, ktorou bol tento logger implementovaný. Pomocou funkcie „*sendBeacon*“ odosielame dáta na server po dávkach. Ak je používateľ aktívny na stránke, tak každé 2 sekundy odošleme JSON s dátami. Tento spôsob implementácie bol nutný z dôvodu funkcie „*sendBeacon*“, ktorá nepodporuje odoslanie viac ako 64KB dát naraz (v jednej dávke). Interval odosielania dát, sa bude optimalizovať počas procesu vývoja projektu a v závislosti od vyťaženia databázy nášho servera.

Realizácia

Logger je implementovaný v javascripte. Implementácia sa nachádza v jednom súbore, ktorý je nutné skopírovať do stromovej štruktúry webovej stránky klienta. Po jeho nakopírovaní, sa názov súboru loggera zahrnie do hlavičky webovej stránky nasledovne pomocou html tagov: *<script>Cesta k súboru</script>*

Logger a jeho správna funkcionalita je podporovaná najrozšírenejšími prehliadačmi– Chrome, Firefox a Safari.

Jeho ďalšou funkcionalitou je pridanie overenia či stránku navštívil bot, respektíve crawler alebo používateľ. Vytvorili sme zoznam známych botov a crawlerov, ktorý sa nachádza v súbore *bot_cawler.txt*, ktoríboli aktívny za posledné 3 roky. Pomocou informácie z *userAgent*dokážeme zistiť jeho názov a nastaviť príznak *bot* na hodnotu 1, ktorý sa odosiela na server rovnakým spôsobom ako je to v prípade bežného používateľa.

Testovanie

Testovanie komponentu *Logger* prebiehalo pri jeho implementačných zmenách na localhoste, kde bolo jednoduché vykonať základné unit testy pozostávajúce z kontroly prijatých údajov. Následne prebehlo aj testovanie komponentu na testovacom prostredí po nasadení na webovú stránku nášho projektu. Z funkčného hľadiska *Logger* pracuje správne, čo ale neviem povedať o výkonnostnej stránke komponentu. V súčasnosti nedokážeme vykonať plnohodnotné výkonnostné a záťažové testy, na základe ktorých by sme dokázali rozhodnúť, či bude, resp. nebude potrebné *Logger*optimalizovať za účelom zefektívnenia zaznamenávania a odosielania údajov o aktivitách návštevníkov. Z tohto dôvodu je komponent *Logger*stále otvoreným komponentom, ktorý plnohodnotné testovanie ešte len čaká.

3.2 E-form

Ako už bolo spomenuté, komponent *E-form*je abstraktným komponentom, pod ktorý sme zaradili všetky elektronické formuláre prostredníctvom, ktorých budú používatelia našej aplikácie zasielať dáta na stranu servera.

Analýza
Funkcionalita, ktorá je v súčasnosti poskytovaná našou aplikáciou využíva 4 elektronické formuláre, ktoré pracujú s dvomi komponentmi a to *IAM*a*Webpages manager*.
Elektronické formuláre komponentu *IAM*:

- Registračný formulár
- Prihlasovací formulár

Elektronické formuláre komponentu *Webpages manager:*

- Formulár pridania novej webovej stránky
- Formulár editovania existujúcej stránky

Návrh
Pre elektronické formuláre komponentu *IAM* boli navrhnuté tieto typy grafických prvkov:

- Textové pole (text field),
- Password,
- Tlačidlo (Button),
- Checkbox.

Prehľad navrhnutých grafických prvkov pre - Registračný formulár:

Názov prvku	Popis	Typ	Statická validácia	Dynamická validácia	Príklad validnej hodnoty
E-mail	e-mailové konto používateľa / prihlasovacie meno do aplikácie	textové pole	nesmie byť už registrovaný	nie je	aaaa@mail.com
Password	voliteľné heslo používateľa	password	minimálne 8 znakov	nie je	123Asdfghj
ConfirmationPassword	potvrdenie správneho zadanie prvého hesla, slúži ako prevencie pred chybnými klikmi	password	musí sa zhodovať s Password	nie je	123Asdfghj
Register	odoslanie zadaných informácií na server	tlačídlo	nie je	nie je	-

Tabuľka 10 - Grafické prvky registračného formuláru (AIM)

Prehľad navrhnutých grafických prvkov pre - Prihlasovací formulár:

Názov prvku	Popis	Typ	Statická validácia	Dynamická validácia	Príklad validnej hodnoty
E-mail	prihlasovací e-mail používateľa	textové pole	musí byť registrovaný	nie je	aaaa@mail.com

Password	heslo používateľa	password	musí patriť k e-mailu	nie je	123Asdfghj
Log in	odoslanie zadaných informácií na server	tlačídlo	-nie je	nie je	-

Tabuľka 11 - Grafické prvky prihlasovacieho formuláru (AIM)

Pre elektronické formuláre komponentu *Webpagesmanger* boli navrhnuté tieto typy grafických prvkov:

- Textové pole (text field),
- Tlačidlo (Button),
- Popisok (Label),
- Neviditeľné pole (Hiddenfield)
- Checkbox.

Prehľad navrhnutých grafických prvkov pre - Formulár pridania novej webovej stránky:

Názov prvku	Popis	Typ	Statická validácia	Dynamická validácia	Príklad validnej hodnoty
API keypopisok	text pri prvku API key text: API key:	label	nie je	nie je	API key:
API key	vygenerovaná unikátna hodnota pre webovú stránku	label	nesmie byť priradený inému webu	nie je	333044975
Web namepopisok	text pri pvku Web name text: Enter web name:	label	nie je	nie je	Enter web name:
Web name	zadanie informačného názvu pre registrovanú webovú stránku	text field	nie je	nie je	www.visitortrack.sk
Activepopisok	text pri prvku Active text: Active:	label	nie je	nie je	Active:
Active	aktivovanie, resp. deaktivovanie monitorovania webovej stránky	checkbox	nie je	nie je	-
Active	zadanie api kľúča Google Analitics	text field	nie je	nie je	UA-45312138-1
GA_Api	text pri prvku GA_Api text: Enter google analytics api key:	label	nie je	nie je	Enter google analytics api key:
GA_Api popisok	aktivovanie, deaktivovanie posielanie Google Analitics	checkbox	nie je	nie je	-
GA_Active	text pri prvku GA_Api text: With google analytics:	label	nie je	nie je	With google analytics:
Log in	odosielanie vložených dát	button	nie je	nie je	-
Back	návrat na zoznam registrovaných webov bez pridania nového webu	button	nie je	nie je	-

Tabuľka 12 - Grafické prvky formuláru pre pridanie webovej stránky

Prehľad navrhnutých grafických prvkov pre - Formulár editovania existujúcej stránky:

Názov prvku	Popis	Typ	Statická validácia	Dynamická validácia	Príklad validnej hodnoty
Web namepopisok	text pri prvku Web name text: Web	label	nie je	nie je	Web:
Web name	textové pole obsahuje aktuálny názov webovej stránky	text field	nie je	nie je	www.visitortrack.sk
Activepopisok	text pri prvku Active text: Active:	label	nie je	nie je	Active:
Active	aktivovanie, resp. deaktivovanie monitorovania webovej stránky	checkbox	nie je	nie je	-
GA_Api	zadanie api kľúča Google Analitics	text field	nie je	nie je	UA-45312138-1
GA_Api popisok	text pri prvku GA_Api text: Enter google analytics api key:	label	nie je	nie je	Enter google analytics api key:
GA_Active	aktivovanie, deaktivovanie posielanie Google Analitics	checkbox	nie je	nie je	-
GA_Active popisok	text pri prvku GA_Api text: With google analytics:	label	nie je	nie je	With google analytics:
Save Web	aktualizovanie editovaného webu	button	nie je	nie je	-
Back	návrat na zoznam registrovaných webov bez pridania nového webu	button	nie je	nie je	-

Tabuľka 13 - Grafické prvky formuláru pre editovanie existujúcej webovej stránky

Realizácia

Každý elektronický formulár, nachádzajúci sa v našej aplikácii je postavený na voľne dostupnej front-end šablóne, ktorú sme zahrnuli do našej aplikácie. Každá následná úprava, resp. vytvorenie nového elektronického formuláru si vyžaduje len narenderovanie pripraveného layoutu do nového okna. Následne použitím štandardných príkazov jazykov HTML, CSS, Javacsricpt a Ruby je možné customizovať zobrazenie elektronických formulárov a aj celého front-endu podľa našich potrieb.

Testovanie

Testovanie komponentu *E-form* nie je možné vykonať samostatne, nakoľko každý elektronický formulár slúži na zadanie vstupných údajov do ďalšieho komponentu (napr. IAM). Z tohto dôvodu prebehlo testovanie jednotlivých elektronických formulárov pri unit testoch samotných komponentoch, ku ktorým sa elektronické formuláre viažu.

Jediná funkcionalita, ktorú je možné testovať na úrovni *E-formu*sú dynamické validácie zadaných údajov, ktoré ale v súčasnej fáze projektu nie sú implementované a v zimnom semestri sa s ich realizáciou ani nepočíta.

3.3 Graf plotter

Komponent *Graf plotter* slúži na vykreslenie štatistík návštevnosti pre používateľa ktorý má v našej aplikácií zaregistrovanú svoju webovú stránku. Údaje, na základe ktorých sa budú grafy vykresľovať sú získavané z komponentu *Statistics*.

Analýza
Jedným z hlavných cieľov našej aplikácie je poskytnúť registrovaným používateľom presné informácie o návštevnosti monitorovaných webových stránok. V snahe dosiahnuť čo najväčšiu čitateľnosť a výpovednú hodnotu je možné použiť dva spôsoby na zobrazenie týchto výstupov prihlásenému používateľovi:

1. Tabuľka
2. Graf

Výhody tabuľky:

- Ľahké kopírovanie a export (napr. do Excelu).
- Možné ďalšie spracovania dát zo strany používateľa.
- Bezproblémové zobrazenie na všetkých prehliadačoch a platformách.
- Jednoduchší proces návrhu.
- Jednoduchšia implementácia.

Výhody grafu

- Vysoká čitateľnosť.
- Oproti tabuľke lepší Look and Feel efekt – používateľsky príťažlivejšie.
- Človek potrebuje menej času na pochopenie grafu než tabuliek.
- Moderné.
- Pri zobrazovaní časovej závislosti údajov omnoho vhodnejšie ako tabuľky.

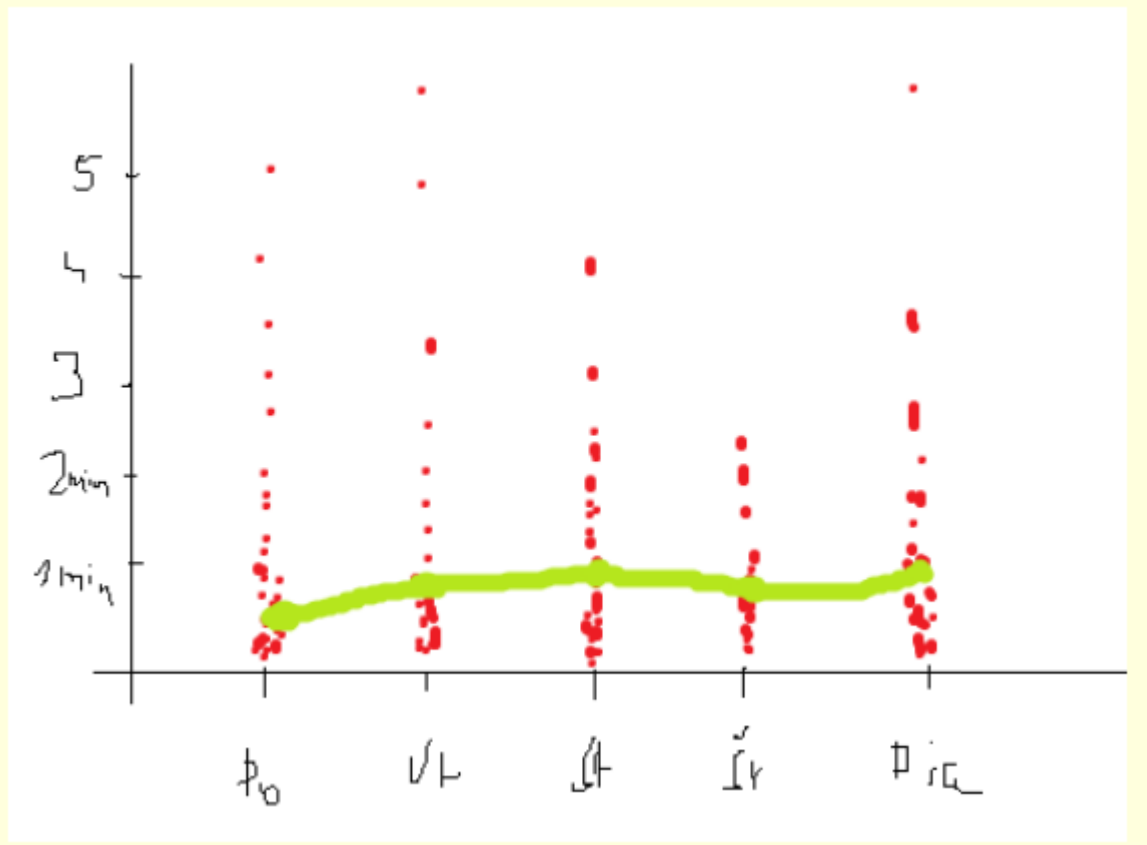
Po tímovej diskusii bol zvolený ako spôsob zobrazenia výsledkov pre používateľov graf. Náš productowner dokonca na tejto možnosti zobrazenia trval a preto nemalo zmysel hlbšie analyzovať a porovnávať tabuľky a grafy.

Návrh
Naša aplikácia disponuje niekoľkými grafmi, konkrétne sa jedná o:

- Visitors - Graf zobrazenia nových a vracajúcich sa návštevníkov
- Referrers - Graf zobrazujúci stránky, z ktorých návštevníci prišli
- Entry Pages - Graf informujúci o prvej stránke, ktorú návštevník zobrazil
- Devices - Graf zobrazujúci zariadenia, ktoré návštevníci použili pri prezeraní
- Spent time– Graf zobrazuj čas návštevníkov strávený na stránke
- Viewed Pages – Graf zobrazuje počet stránok, ktoré návštevníci prezreli pri jednom sedení
- Visitor Sessions – Graf informuje o počte sedení návštevníkov na deň

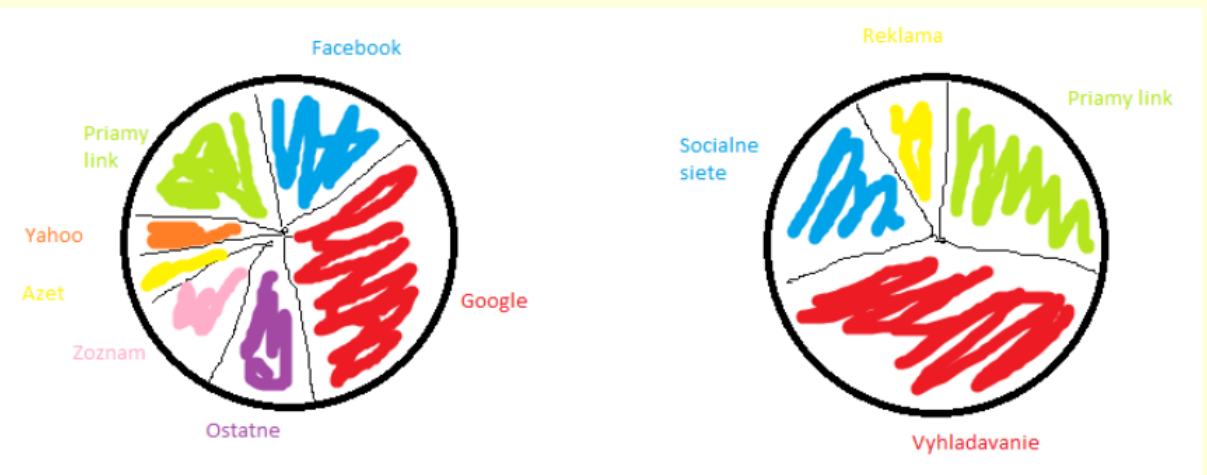
Od nášho product ownera sme dostali približný návrh pre jednotlivé grafy. Na nasledujúcich obrázkoch sú zobrazené prvé návrhy, ktoré poslúžili ako podklady pre jednotlivé grafy.

Na obrázku nižšie je návrh pre graf Spent Time:



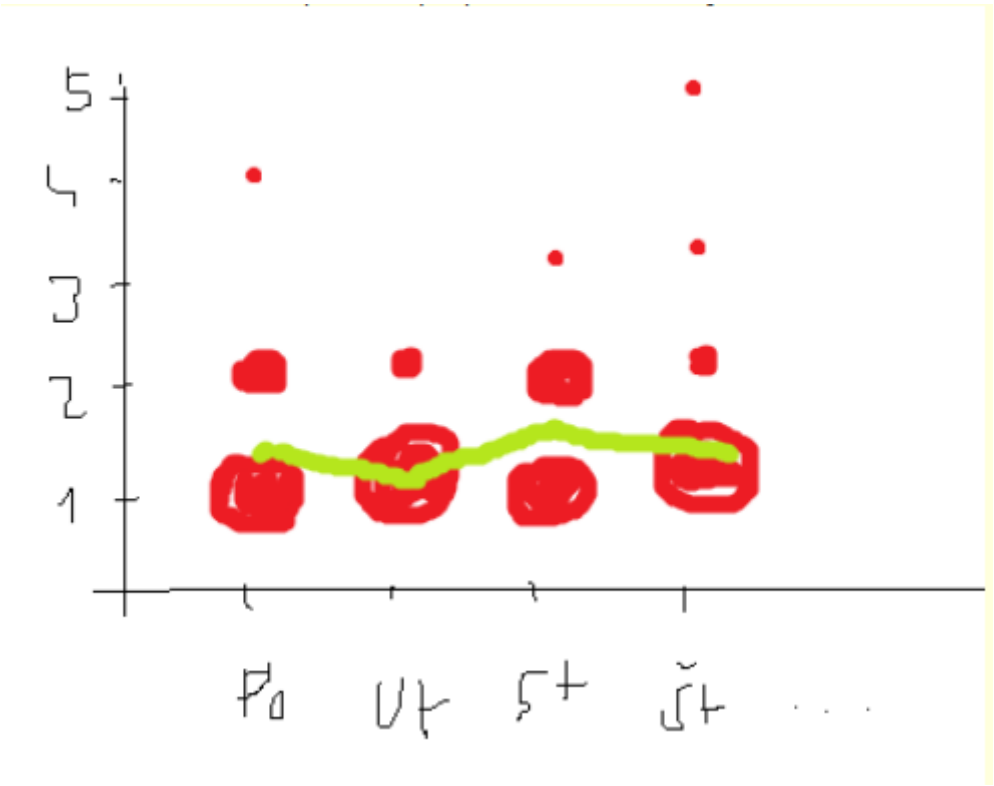
Obrázok 7 - Návrh grafu Spent time

Na obrázku je zobrazený návrh koláčového grafu, ktorý mal byť použitý pre grafy Referrers, Entry Pages a Devices:



Obrázok 8 - Návrh koláčovéh grafu pre Referrers, Entry Pages a Devices

Posledný navrhovaný graf je typu bubble a je určený pre grafy **Visitors Sessions a Viewed Pages:**



Obrázok 9 - Návrh grafu pre Visitors Sessions a Viewed Pages

Pre základný graf Prvým zobrazujúci nových a vracajúcich sa nštevníkov bol použitý obyčajný stĺpcový graf, ktorého návrh nebol nakreslený.

Realizácia

Pre implementáciu komponentu *Graf plotter*bolo použité už hotové riešenie s názvom *Highcharts*. Okrem toho, že *Highcharts* spĺňal všetky požiadavky na vizuál grafu, je jednoduchý na použitie a je implementovaný javascriptom, čím je zaručená jeho spustiteľnosť takmer na všetkých prehliadačoch a platformách bez dodatočnej inštalácie.
Na zobrazenie grafu je potrebné na stránku aplikácie VisitorTrack umiestniť HTML kód pomocou ktorého sa zavolajú príslušné JavaScript súbory.

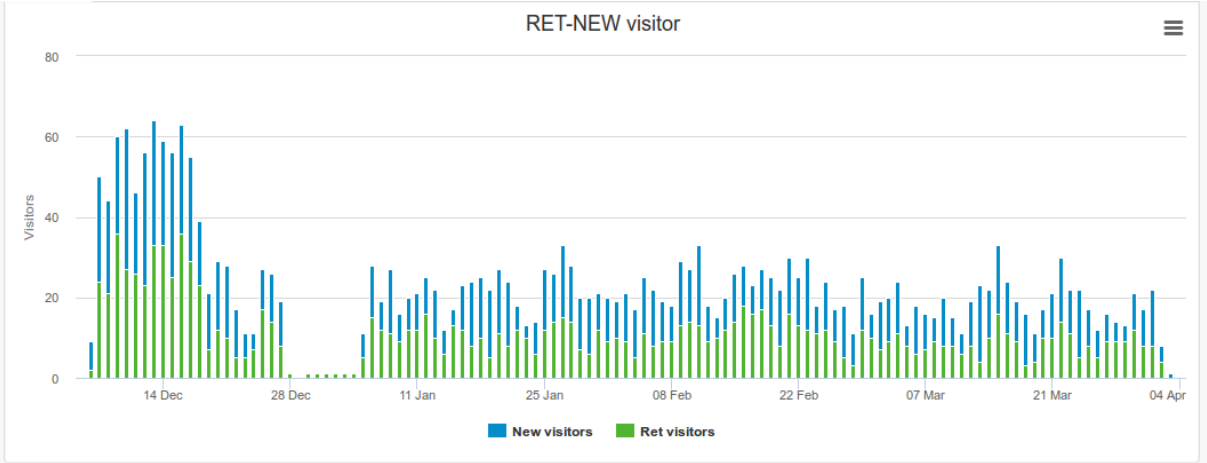
Príklad použitia riešenia *Highcharts*v našej aplikácii:

```
<scriptsrc="https://code.highcharts.com/highcharts.js"></script>
<scriptsrc="https://code.highcharts.com/modules/exporting.js"></script>
<div id="container" style="min-width: 310px; height: 400px; margin: 0 auto"></div>
```

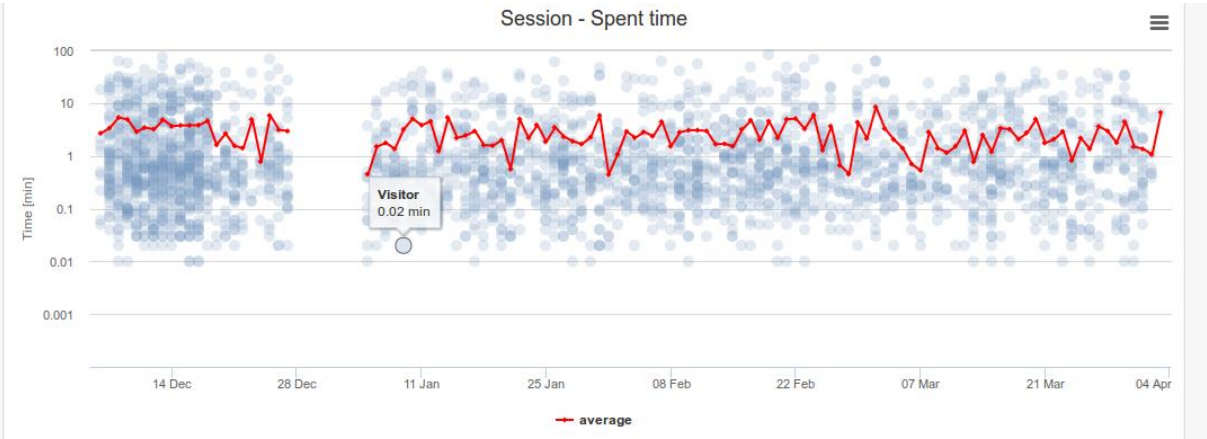
Potrebné dáta a informácie, na základe ktorých sa dáta vykresľujú sú prijímané vo formáte JSON, ktorý pripravuje a odosiela komponent *Statistics*. Vďaka takémuto riešeniu je strana klienta čo najmenej zaťažená výpočtami a preto nároky na HW u klienta sú minimálne.

Jednotlivé grafy, ktoré boli implementované podľa dodaného návrhu nakoniec vyzerajú nasledovne:

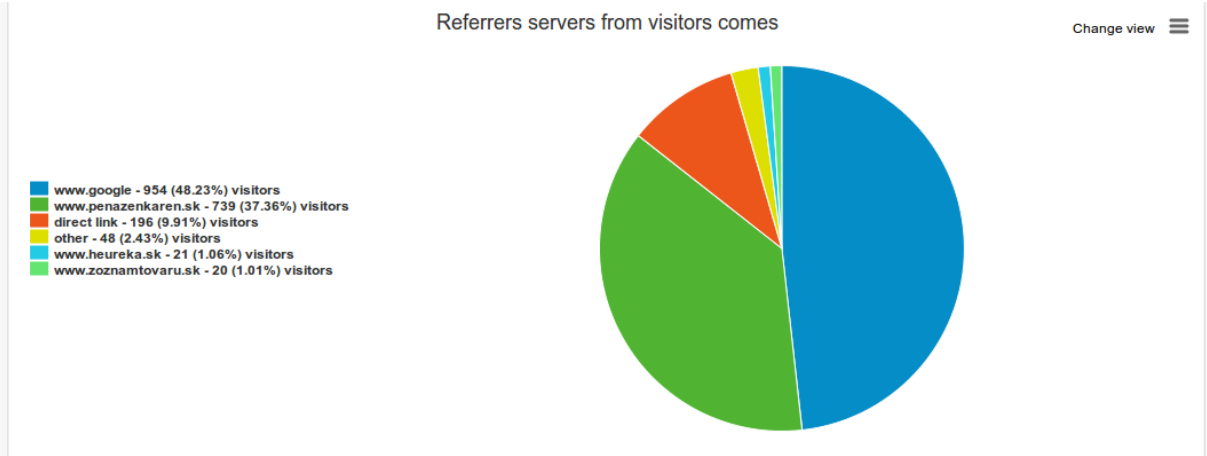
Graf Visitors:



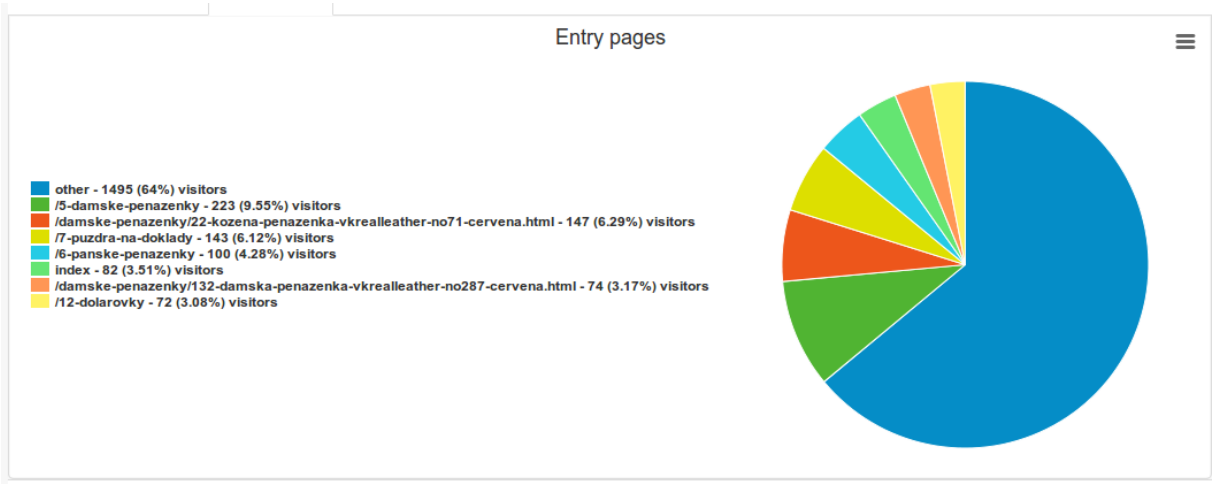
Graf Spent Time:



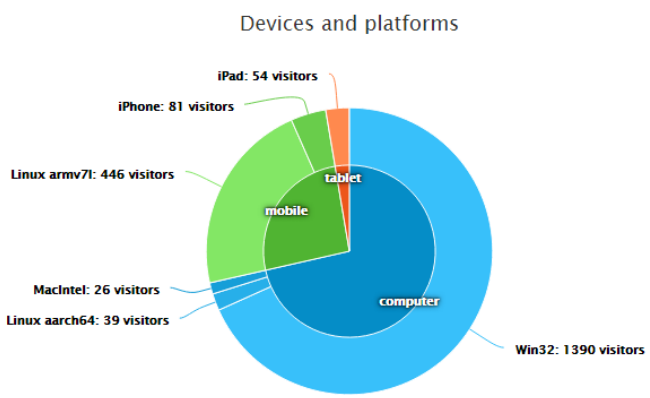
Grafy Referrers



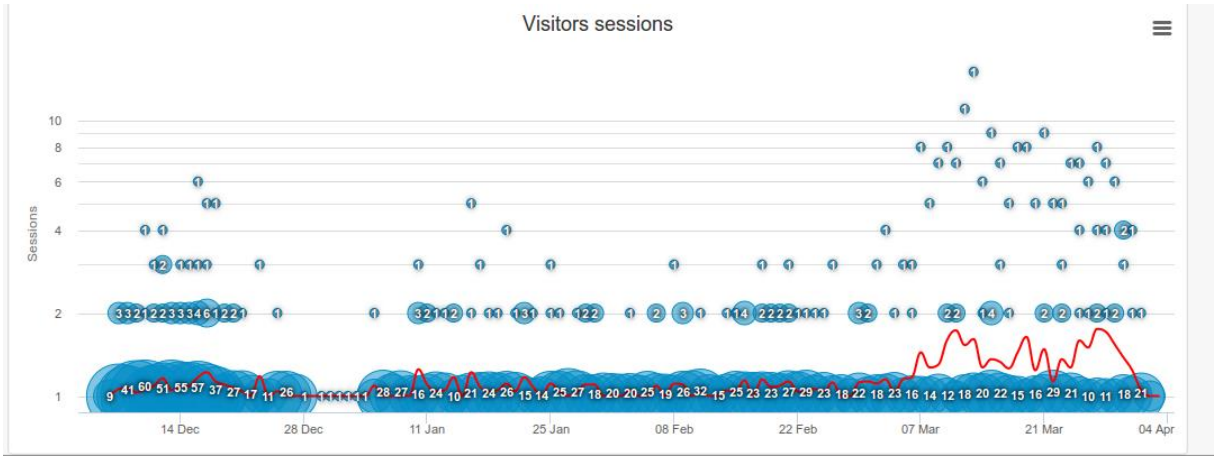
Graf Entry Pages



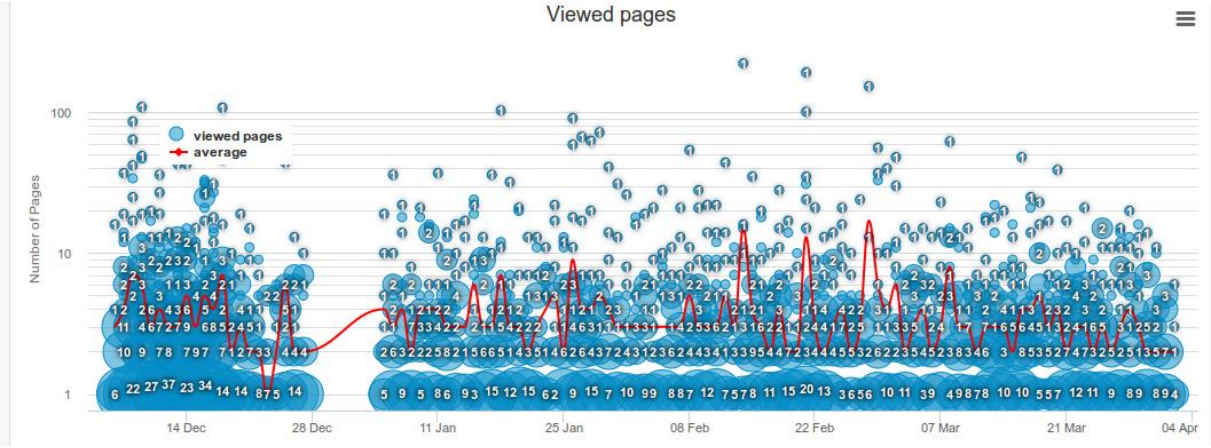
Graf Devices:



Graf Visitors Sessions



Graf Viewed Pages



Testovanie

Komponent bol testovaný vizuálnou validáciou na základe pripravenej testovacej vzorky, ktorú spracoval a odoslal komponent *Statistics*. Tento komponent neplánujeme testovať automatizovaným testovaním, pretože sa jedná o známe riešenie tretej strany, ktoré je celosvetovo používané a verifikované množstvom vývojárov a používateľov.

3.4 Requestparser a VisitorTrack API

Komponenty *Requestparser* a *VisitorTrack API* sú úzko spolupracujúce komponenty, pričom ich implementácia prebiehala súbežne a keďže nie je možné ich jednoznačne oddeliť sú popísané v spoločnej kapitole.

Poznámka: Zjednodušene by bolo možné povedať, že VisitorTrack API je len RESTová služba, ktorú poskytuje Requestparser. Pretože sa ich funkcionalita odlišná, boli ponechané ako dva rozdielne komponenty.

Analýza

Jednou zo základných funkcionalít, ktoré musela naša aplikácia disponovať bolo prijímať správy z monitorovaných webov konkrétne z komponentu *Logger*. Do procesu analýzy vstupovali dva možné spôsoby implementácie a to ako REST alebo SOAP služba. Počas analýzy oboch možností sme ako vhodnejší spôsob zvolili práve RESTovú službu, a to ako z pohľadu lepšej podpory v Ruby on Rails (v čom je aplikácia implementovaná) tak aj z pohľadu jednoduchosti implementácie.

Po prijatí requestu s dátami je potrebné dáta správne spracovať a uložiť. Toto spracovanie a ukladanie dát zabezpečuje komponent *Request parser*.Počas analýzy sme sa snažili určiť najvhodnejší formáte pre posielané dáta, či už z pohľadu komponentu *Logger* u klienta tak aj z pohľadu komponentu *Requestparser*na strane serveru. Nakoniec sa ako najvhodnejšia voľba ukázalo použiť formát JSON, ktorého podpora v Ruby on Rails je na vysokej úrovni a preto výrazne klesla implementačná náročnosť pre komponent *Requestparser*.

Návrh

Na to aby bolo možné implementovať komponent *Requestparser*bolo potrebné navrhnúť dátový model. Práve pri návrhu komponentu *Requestparser*bola navrhnutá prevažná časť dátového modelu, ktorý môžeme vidieť v kapitole *2.4Dátový model*. Navrhli sa bázické entity medzi ktoré môžeme zaradiť:

- Visitor,
- Page,
- Coordinates,
- Histogram
- Time

Realizácia

RESTová služba implementovaná v našej aplikácii je metóda, ktorá je prístupná pre vonkajšie systémy. Aby ju bolo možné korektne zavolať bolo potrebné správne nastaviť routing k tejto metóde. Momentálne je metóda prístupná na:
147.175.149.192:443/logger

Implementácia *Requestparsera*je orientovaná na ľahké a efektívne spracovanie prijatých údajov vo formáte JSON. V kapitole *3.1Logger* je zobrazený formát requestu, ktorý je posielaný *Loggerom*, resp. prijímaný komponentom *Visitor Trak API*. A presne v takejto podobne je následne posunutý *do Requestparseru*. Keď sa pozrieme na*Obrázok 6 - Návrh JSON formátu* môžeme vidieť presné údaje, ktoré musia byť spracované. *Requestparser*preto ako prvé vytvorí (ak sa v databáze ešte nenachádza) inštanciu triedy „*Page*“, ktorá predstavuje webovú stránku, z ktorej request prišiel.
Druhý krok, ktorý sa následne vykoná je overenie atribútu „*mid*“, ktorý slúži ako jednoznačný identifikátor návštevníka. V prípade, že sa „*mid*“ s rovnakou hodnotou už nachádza v databáze, je vrátený záznam tohto používateľa, inak je vytvorený nový používateľ. Vrátenému, resp. novo vytvorenému návštevníkovi sú následne v treťom kroku priradené všetky aktivity, ktoré boli doručené v dátovej časti prijatéhorequestu.

Testovanie

Samotné testovanie oboch komponentov prebehlo po ich nasadení na testovacie prostredie, kde sme mohli následne sledovať a kontrolovať prijaté záznamy a spôsob ich uloženia do DB. Údaje boli odosielané zo webovej stránky nášho tímu a zaznamenávané boli na tímovom servery, kde bola aplikácia nasadená.

3.5 Integrácia Google Analitics API

Komponent pre integráciu Google Analytics API (GA) sme zapracovali do systému aby používatelia, ktorí už využívajú analytický softvér od Googlu vedeli priradiť vracajúceho sa používateľa na základe nášho unikátneho algoritmu deduplikácie.

Analýza
Cieľom modulu je sprostredkovať pre aktuálneho návštevníka, ktorému GA priradilo identifikátor náš identifikátor, ktorý je v prípade vracajúceho sa používateľa s vymazanými Cookies pre GA viacnásobný, no našou deduplikáciou ho vieme zjednotiť.

Návrh
Počas návrhu modulu sme zistili, že je potrebné od používateľa nášho systému vedieť sledovacie číslo webovej stránky v tvare UA-XXXXXXXX-X, ktorým vieme poslať žiadosť na servery GA a ktorými vieme priradiť aktuálnemu identifikátoru, ktorý GA používa náš. Navrhli sme, že z našej strany najvhodnejšie bude posielať náš *mid* identifikátor, ktorý zjednocuje vracajúceho sa a nového používateľa.

Realizácia
V rámci nášho loggera sme vytvorili callback interface, ktorý v prípade, ak má webová stránka zaznačené sledovanie GA, použije v našom systéme rozhranie GA na vlastnú identifikáciu návštevníkov. Pri príchode dát z loggera je skontrolovaný návštevník v našej databáze a ak existuje prepojenie na predchádzajúceho používateľa, je tento údaj poslaný späť do JS sledovanej webovej stránky, odkiaľ je cez GA API poslané na spracovanie.

Testovanie
Testovanie bolo vykonávané na reálnych dátach a bolo v reálnom čase kontrolované s hodnotami v Google Analytics. Bola vytvorená testovacia web stránka. na ktorú bol priradený GA a náš tracker. V prípade vymazania Cookies nevedelo GA rozpoznať vracajúceho sa používateľa, no po spracovaní nášho požiadavku na zjednotenie používateľov nastalo v GA vizualizácii správne zobrazenie.

3.6 Heatmap creator

Analýza
Existuje viacero typov heatmap, ktoré sú používané vkonkurenčných aplikáciách. Keďže naša aplikácia obsahuje dva podobné moduly, rozhodli sme sa, že každý z modulov bude zameraný na inú stránku správania používateľov, pre zefektívnenie výkonu aplikácie. Zatiaľ čo playback komponent sa zaoberá hlavne pohybom myši návštevníka, komponent heatmap rieši prioritne jeho klikanie. Takto dokážeme našim používateľom dodať vyfiltrované informácie podľa ich chuti.

Návrh
Počas návrhu modulu sme sa rozhodli znázorňovať jednotlivé klikania používateľov pomocou čiastočne priesvitných kruhov s zväčšujúcim sa alfa levelom so vzdialenosťou od stredu, čím sa docieli hlavný efekt heatmap, tj. na miestach s najviac kliknutiami v tesnej blízkosti je farebná vrstva najvýraznejšia.

Realizácia
Ako prvý krok je vykonané fotenie cieľovej strány. Vo webovom rozhraní v sekcii Heatmapy si používateľ vyberie podstránku, pre ktorú ich chce zobraziť. Požiadavka sa odošle na server, kde je spustený skript obsahujúci fotiacu sekvenciu. Hlavným problémom je autentifikácia používateľa na cieľovej stránke. V prípade, že sa náš používateľ snaží získať Heatmapy pre podstránku, na ceste ku ktorej je potrebné sa autentifikovať, systém , ktorý používaa proxy server nie je schopný rozpoznať prekážku na ceste a ak je mu zobrazená na fotenej URL iba výzva pre prihlásenie spracuje tú. Odfotená stránka je dočasne uložená na serveri ako obrázok vo formáte PNG, ktorý bude spracovaný.
V druhom kroku sú z databázy vybrané záznamy používateľov pre žiadanú web stránku. Jedná sa o položky v tabuľke Coordinates, ktorých event_type je mouse_down. Pre nás zaujímavé atribúty sú koordináty odpovedajúce klikom na web stránke. Pomocou grafického frameworku ImageMagick sú tieto dáta spracované do obrazovej podoby a je z nich vytvorený layer, ktorý je nanesený na odfotenú webovú stránku.
Výsledný obrázok je dočasne uložený na server a posunutý používateľovi na zobrazenie.

Testovanie
Testovanie priebiehalo kontrolou správnosti a dĺžky vytvárania Heatmap nad skutočnými stránkami a podstránkami používateľov. Pri testovaní sa podarilo zredukovať pôvodný čas vytvárania Heatmap na 25% na úkor zvýšenej spotreby pamäte.

3.7 Playback

Analýza

Tento komponent aplikácie je založený na komponente Heatmap creator. Takisto ako pri vytváraní heatmapy je prvým krokom načítanie obrázka konkrétneho webu, respektíve jeho podstránky. Po prvotnom výbere obrázka má používateľ možnosť výberu konkrétnej session reprezentujúcej správanie používateľa na konkrétnej podstránke. Po jej výbere sa automaticky spustí záznam pohybu myši pre konkrétneho návštevníka podstránky. Id konkrétneho návštevníka ale zostáva pre používateľa našej aplikácie neviditeľné, aj z dôvodu ochrany súkromia návštevníkov.

Návrh

V návrhu komponentu sme sa snažili vybrať súradnice konkrétnych typov akcií (events), ktoré najlepšie reprezentujú skutočný pohyb myši (prípadne touchpadu) návštevníka webu. Tieto boli potom použité pri vykreslení pohybu. Bol taktiež navrhnutý spôsob kreslenia na HTML plátno (canvas).

Realizácia

Realizácia spočívala v niekoľkých krokoch. Najprv sa analyzoval a mierne upravil kód pre generovanie obrázku webu. Z uloženého obrázka sa vytiahla jeho dĺžka potrebná pre účely druhého kroku. V druhom kroku sa pod element obrázka webu pridal element plátna pomocou kombinácie relatívneho a absolútneho poziciovania. Rozmer tohto plátna (dĺžka) sa nastaví podľa zistenej veľkosti obrázka. Toto plátno je priehľadné kvôli potrebe vidieť obrázok webu pod ním. Po vybratí konkrétnej session používateľom aplikácie sa načítajú koordináty pre konkrétnu podstránku, session a používateľa. Vyberú sa iba koordináty potrebné pre plynulý pohyb myšou. Po tomto kroku sa informácie v JSON formáte predajú z komponentu používajúcom Angular, skriptu určenému na vykresľovanie súradníc. V tomto skripte sa nachádza funkcia ktorá vykresľuje pohyb po oneskorení určenom časovým rozdielom medzi dvoma súradnicami. Táto funkcia využíva na tento účel rekurzívne volania kvôli spôsobu akým sa v JavaScripte používajú časovače. Nie je teda možné jednoduchým spôsobom zistiť ukončenie volania tejto funkcie. Po vykreslení pohybu sa plátno pred ďalším použitím zmaže. Používateľovi aplikácie sa dočasne vypnú možnosti pre načítavanie obrázku pre web alebo výber inej session, aby ho táto funkcionalita zbytočne pri vykresľovaní neplietla. Toto je realizované pomocou časovača na trvanie prvej a poslednej akcie. Používateľovi aplikácie ponúkame iba session s dostatočnou dĺžkou trvania aspoň nad 20 sekúnd.

Testovanie

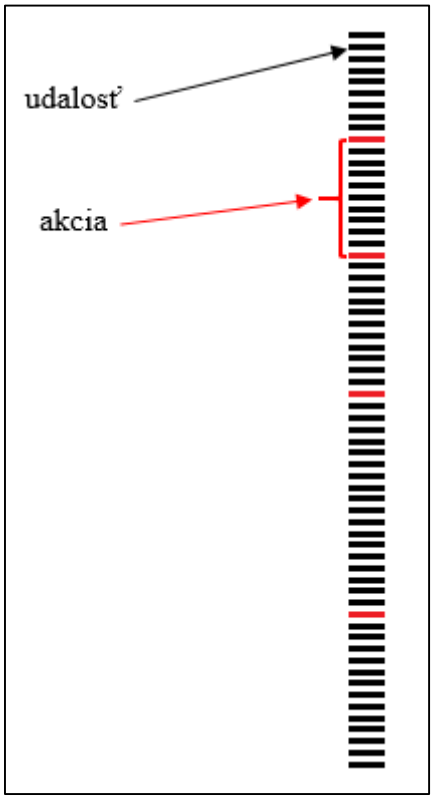
Testovanie riešenia prebiehalo na dátach z webu penazenkaren.sk. Boli vybraté približne 3 podstránky na ktorých sa spustilo vykreslenie pohyby trvajúce aspoň minútu. Z tohto sa tiež zistilo, ktoré typy koordinátov (súradníc) nie sú pre plynulý pohyb akcií potrebné.

3.8 Actionparser a Histogram

Komponent *Actionparser* spracováva surové dáta, ktoré sú prijaté komponentom *VisitorTrack API* a následne uložené do databázy komponentom *Requestparser*.
Komponent *Histogram*využíva dáta, ktoré sú výstupom *Actionparsera*. Oba tieto komponenty úzko spolupracujú a preto boli analyzované, navrhované a implementované súbežne. Z tohto dôvodu sú popísané v jednej spoločnej kapitole.

Analýza
Logger zaznamenáva udalosti (move, move_down, move_up, ...), ktoré sa združujú do akcií (súbor udalostí). Akcia môže byť ohraničená dvoma spôsobmi:

- klik – klik,
- udalosť (rôzny typ – prvá respektíve posledná udalosť, udalosť po dlhšom časovom úseku) – klik.



Obrázok 10 - Záznam z Loggera

Pre získanie modelu používateľa potrebujeme z dát získaných z *Loggera* vypočítať nasledovné hodnoty, z ktorých následne budeme získavať jednotlivé histogramy:
Trvanie kliku

$$t = |t_{move\_down} - t_{move\_up}|$$

Pauza pred klikom

$$t = |t_{move} - t_{move\_down}|$$

Pauza po kliku

$$t = |t_{move} - t_{move\_up}|$$

Priemerná rýchlosť (v)

$$v = \frac{s}{t}$$

Štandardná odchýlka rýchlosti (σ)

$$\bar{v} = \frac{1}{n}\sum v_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum (v_i - \bar{v})^2}$$

Vzdialenosť (s)

$$s = s + |s_{i-1} - s_i|$$

Štandardná odchýlka zakrivenia (c)

$$\alpha_i = \text{arc}tg(s_i)$$

$$s_i = y_{rozdiel}/x_{rozdiel}$$

$$\bar{c} = \frac{1}{n}\sum \frac{\alpha_i}{s_i}$$

$$\sigma = \sqrt{\frac{1}{n}\sum (\frac{\alpha_i}{s_i} - \bar{c})^2}$$

Priemerná uhlová rýchlosť (w)

$$w = \frac{1}{n}\sum \frac{\alpha_i}{t_i}$$

$$\alpha_i = \text{arc}tg(s_i)$$

Priemerný smer ($\bar{\alpha}$)

$$\bar{\alpha} = \frac{1}{n}\sum \alpha_i$$

Dáta z *Loggera* vyzerajú nasledovne:

User	Action	Time	x	Y	url	resolution	
userID	m	1398348029997	1061	163	/pexeso/	1920	1080
userID	m	1398348030004	1057	165	/pexeso/	1920	1080
userID	m	1398348030011	1049	168	/pexeso/	1920	1080

Tabuľka 14 - Dáta z Loggera

User – predstavuje identifikátor používateľa
Action – udalosť ktorá sa vykonala (move, move_down, move_up, ...)
Time – v akom čase udalosť nastala - t_i
X – x-ová súradnica x_i
Y – y-ová súradnica y_i
url – url adresa webovej stránky
resolution – rozlíšenie displeja používateľa
x a y súradnice predstavujú hodnoty s_i.

Histogram sa skladá z hodnôt jedného bodu (napr. 1 – trvanie kliku), ktoré sú zozbierané za celý pobyt používateľa na stránke a vypočítané. Pre histogram sa v praxi zistilo, že pri danom probléme je výhodné mať stanovené dve hodnoty min a max (Tabuľka 3), ktoré reprezentujú minimálnu respektíve maximálnu hodnotu z priemerných meraných hodnôt o používateľoch.

Akcia	**MIN**	**MAX**
trvanie kliku	47,85	297,33
pauza pred klikom	38,39	1574,34
pauza po kliku	0	26812,63
priemerná rýchlosť	107,35	2963,09
štandardná odchýlka rýchlosti	77,62	6776,66
vzdialenosť	146,3	12917,54
štandardná odchýlka zakrivenia	-0,348	1,42
priemerná uhlová rýchlosť	-0,015	0,114
priemerný smer	-1,95	2,23

Tabuľka 15 - MIN/MAX hodnoty

Návrh

Po zaznamenaní údajov z loggera budú prebiehať tri hlavné udalosti:

1. Zapísanie údajov z RequestParsersa do DB a následné vybratie týchto údajov na spracovanie.
2. Program vypočíta hodnoty (1 – 9 uvedené v analýze) zo zaznamenaných údajov. Jednotlivé hodnoty 1-9 analýzy sa počítajú pre hodnoty nachádzajúce sa medzi dvoma klikmi, medzi prvým pohybom a klikom alebo klikom a posledným pohybom. Tieto hodnoty je potrebné ukladať, pretože z aktivity celého pobytu používateľa na stránke vzniká veľa udalostí a následne všetky hodnoty napríklad bodu jedna (trvanie kliku) budú vstupom pre vytvorenie histogramu trvania kliku v nasledujúcom bode.
3. Program z vypočítaných hodnôt zostrojí histogramy a uloží ich ako model používateľa (súbor histogramov). Histogram bude pozostávať z ôsmych častí, ktoré budú ohraničené zadefinovanými pevnými hodnotami min a max, kde medzi týmito hodnotami min a max rozdelíme os na osem častí (intervalov) do ktorých budeme priraďovať vypočítané hodnoty podľa toho či patria do daného intervalu. Zostávajúce dva intervaly (hodnoty menšie ako min respektíve väčšie ako max) budú dopĺňať celý histogram

Realizácia

Tvorba akcií sa deje pomocou dvoch tried v adresári ./lib. Trieda *ActionParser*má na starosti sekanie udalostí na akcie. Takisto sa tu počítajú pauzy pred/po kliku a trvanie kliku. Trieda *ActionToModel* predstavuje akciu. Základné hodnoty (okrem páuz) sa tu vypočítajú hneď po inicializácii objektu. K hodnotám akcie sa dá pristupovať pomocou zodpovedajúcich 9 atribútov:

- click_duration
- pause_before

- pause_after
- speed_avg
- speed_deviation
- distance
- curvature_deviation
- angular_velocity_avg
- angle_avg

V modeli *Coordinates* sú pridané 2 metódy. Prvá vyhľadá udalosti (find_visitor_coordinates) pomocou id visitora. Druhá ich parsuje a vracia akcie (parse_visitor_coordinates).

Zapisovanie hodnôt do histogramov je realizované pomocou dvoch tried – *HistogramGraph.rb* a *HistogramHandler.rb* ktoré sa nachádzajú v ./lib. Trieda *HistogramHandler* obsahuje pole histogramov (inštancie triedy *HistogramGraph*) a nachádza sa tu metóda *add_values_to_histograms(val1, val2, val3, val4, val5, val6, val7, val8, val9),* ktorá po jej zavolaní inkrementuje pole histogramov. 9 parametrov tejto funkcie predstavuje 9 vypočítaných hodnôt (viď analýza) a pre každú túto hodnotu existuje samostatný histogram. Trieda *HistogramGraph* po inicializácii (zadefinovanie intervalov) obsahuje metódu *add_values(val)*, ktorá nájde pre danú hodnotu *val* miesto v interval a inkrementuje stĺpec tohto interval v histograme.

Po vypočítaní hodnôt pre všetky akcie daného návštevníka, sa zavolá metóda add_histogram_to_db(visitor_id) triedy HistogramHandler. Táto metóda ukladá histogramy do databázy. Jej atribútom je visitor_id, čo je identifikátor návštevníka v databáze, pre ktorého boli hodnoty počítané. Metóda najskôr ukladá konkrétny histogram do databázy. Následne získa jeho ID z databázy a vytvorí väzobnú entitu, ktorá spája tabuľky databázy: typ histogramov, návštevník a histogram.

V Modeli *Coordinate* sa nachádza metóda *store_all_visitors_histograms*, ktorá spája obidve funkcionality. Z koordinátov sa vytiahnu jedinečné identifikátory visitorov. Pre tieto sa získajú koordináty udalostí. Tieto sa posekajú na akcie. Hodnoty z akcií sa uložia v metóde *store_visitor_action(action)* ktorej parameter predstavuje akciu, tá potom zavolá *add_values_to_histogram*s príslušnými hodnotami akcie. Na konci sa ešte zavolá *add_histogram_to_db*s prílušným id visitora, pre ktorého histogramy ukladáme.

Trieda *HistogramHandler* obsahuje aj ďalšiu funkcionalitu, ktorá predstavuje porovnávanie návštevníkov, takzvanú deduplikáciu na základe histogramov. Metóda *deduplicate* volá metódu *load_visitors*, ktorá načíta z databázy návštevníkov a rozdelí ich pomocou stĺpca compared na už porovnaných (historických) a neporovnaných (nových – napr. za posledný deň). V rámci deduplikácie sa používajú filtre na filtrovanie používateľov podľa parametrov získaných z loggera ako napríklad *browser, browser_version, appName* a podobne. Metódakaždého nového návštevníka porovná so všetkými historickými. Porovnanie funguje na základe porovnania rovnakých typov histogramov tak, že histogram je prechádzaný cyklom od jeho začiatku po koniec (metóda *hist_sum*). Prechádza sa od nula po

jeho dĺžku čo predstavuje premenná *i*. Vždy sa spočíta prvých *i* stĺpcov histogramu. Z tohto dostaneme absolútnu hodnotu rozdielu pre každý i-ty výpočet. Najväčšia hodnota z týchto hodnôt je výsledný rozdiel dvoch histogramov. Táto hodnota je sčítaná s ostatnými rozdielmi zostávajúcich histogramov a takto vznikne hodnota, ktorá určuje mieru podobnosti návštevníkov.
Najlepšia hodnota spomedzi všetkých porovnaní medzi novým návštevníkom a všetkými historickými sa porovná s konštantou. Ak je táto určujúca hodnota menšia ako konštanta, je id tohto historického návštevníka zapísané do stĺpca ret_visitor_id nového návštevníka. Následne keď je návštevník porovnaný a vyhodnotený, je zapísaný do databázy so zmenou v stĺpci compared na true, čo označuje, že už bol daný návštevník porovnaný.

Testovanie
Oba komponenty boli testované najskôr na testovacích vstupoch dodaných od produktového vlastníka. Pretože sa jedná o zložité a komplikované výpočty, taktiež boli realizované automatizované testy, ktoré sú spustiteľné v prípade potreby. Model *Coordinate*bol testovaný v súbore *coordinate_spec.rb.* Tu sa testuje správne načítanie koordinátov, ukladanie akcie, pridávanie hodnôt do histogramov a čiastočne aj výstup z parsovanýchkoordinátov. Ďalšie testovanie sa nachádza v súbore *histogram_handler_spec.rb*. Tu sa testuje správne rozdelenie ohraničení histogramov podľa MIN/MAX konštánt. Testuje sa tu aj inkrementácia správneho stĺpca histogramu.

3.9 IAM

Identifikačný a autentifikačný komponent našej aplikácie. Umožňuje používateľovi registrovať sa a taktiež overuje jeho totožnosť pri prihlasovaní. Momentálne nie je v aplikácii využívaná správa prihlasovacích údajov (gem Devise túto funkcionalitu pri inštalovaní vygeneroval, tá ale nie je zatiaľ customizovaná pre naše potreby).

Analýza
Pre potreby našej aplikácie je nutné sprístupniť používateľom funkcionalitu pre vytvorenie vlastného účtu a prihlasovanie sa prostredníctvom neho. V súčasnosti sú dostupné pre Ruby on Rails rôzne riešenia, ktoré uľahčujú implementáciu IAM komponentov. Pre naše potreby sme sa rozhodli použiť jedeno z nich - gem *Devise*.

Návrh
V aktuálnej fáze projektu je pre nás postačujúce zabezpečenie funkcionality:

- Prihlásenie používateľa
- Registrácia používateľa
- Odhlásenie používateľa

Ku každej z týchto funkcií bolo potrebné navrhnúť elektronický formulár (návrh formulárov pre IAM s nachádza v kapitole 0*E-form*). Taktiež boli navrhnuté základné chybové situácie, ktoré IAM musí ošetrovať (momentálne sa jedná o náš jediný bezpečnostný komponent) a aj text jednotlivých chybových správ.

Prehľad identifikovaných chybových situácií:

Názov chyby	Popis chyby	Reakcia	Zobrazená správa
Zadaný už registrovaný e-mail	Používateľ zadá pri registrácii e-mail, ktorý je už v aplikácii VisitorTrack registrovaný	Používateľ je informovaný chybovou správou, že ním zadaný e-mail už sa nachádza v aplikácii.	Email has alreadybeentaken.
Zadané krátke heslo	Používateľ zadá pri registrácii heslo, ktoré neobsahuje minimálne 8 znakov.	Používateľ je informovaný chybovou správou, že ním zadané heslo je príliš krátke.	Passwordistooshort (minimum is 8 characters).
Neúspešné potvrdenie hesla	Používateľ zadá pri registrácii do textových polí pre heslo a potvrdenie hesla rozdielne hodnoty.	Používateľ je informovaný chybovou správou, že ním zadané heslá sa nezhodujú.	ConfirmationpassworddoesnotmatchwithPassword.
Nesprávne prihlasovacie údaje	Používateľ pri prihlasovaní zadá nesprávne prihlasovacie meno alebo heslo.	Používateľ je informovaný chybovou správou o zadaní nesprávneho prihlasovacieho mena alebo hesla.	Invalid email or password.

Tabuľka 16 - Identifikované chybové situácie komponentu IAM

Realizácia

Väčšina potrebného zdrojového kódu ako aj entít (napr. user) v databáze boli vygenerovanésamotným gemom*Devise*. Pre naše potreby bolo následnenutné už len nastavenie správneho routingy a upravenie views, resp. elektronických formulárov, ktoré IAM komponent používa.

Testovanie

Pri testovaní sme sa okrem kladných testovacích scenárov, t.j. úspešná registrácia a prihlásenie zamerali najmä na chybové situácie popísané v časti návrh. Každá chybová situácia bola testovaná podľa *Tabuľka 16 - Identifikované chybové situácie komponentu IAM,* kde sme pri vložení chybového vstupu očakávali požadovanú reakciu aplikácie s korektnou chybovou správou.

3.10 Statistics

Úlohou komponentu *Statistics*je pripraviť dáta, na základe ktorých budú používateľovi zobrazované grafy, tzn. komponent *Statistics* pripravuje číselné vstupy pre komponent *Graffplotter*.

Analýza
V aktuálnej fáze projektu, je komponent *Statistics* len prototypom, ktorý sa v ďalších iteráciách bude priamo úmerne rozširovať s množstvom grafických výstupov (grafov), ktoré si bude môcť používateľ zobraziť. Prvé štatistiky (vstupy pre graf), ktoré sme sa v našej aplikácii rozhodli realizovať boli zvolené na základe vysokej výpovednej hodnoty, častého využitia z pohľadu používateľa a jednoduchšej implementácie.

Prvé štatistické údaje, ktoré má komponent pripravovať:
Časové obdobie:

- posledný deň - záznamy pre čas medzi aktuálnym časom (Time.now) a 00:00 v aktuálny deň,
- posledný mesiac - záznamy pre čas medzi aktuálnym časom Time.now a 00:00 v prvý deň aktuálneho mesiaca,
- predchádzajúci mesiac - záznamy pre čas medzi 00:00 v prvý deň predchádzajúceho mesiaca a 23:59 v posledný deň predchádzajúceho mesiaca.

Z pohľadu návštevníkov musí rozlišovať:

- nových návštevníkov - zhoduje sa im položka ID a Ret_Visitor_ID,
- vracajúcich sa návštevníkov: položka ID a Ret_Visitor_ID je rozdielna.

Návrh
Každý používateľ našej aplikácie musí mať možnosť zobraziť si grafové výstupy, ktoré ho budú informovať o návštevnosti jeho webovej stránky. Z tohto pohľadu musí komponent *Statistics* správne pripraviť údaje pre každého používateľa a každú jeho registrovanú stránku zvlášť. Tzn., že komponent musí poskytovať vhodné metódy, po zavolaní ktorých vráti korektné údaje pre požadovaného používateľa, pre ním vybranú registrovanú stránku o a pre zvolený časový interval.
Návrh možného volania metódy pre získanie štatistických údajov:

si = Statistics.grafInput(user, web, from, to)

Realizácia
Základom správnej funkcionality komponentu je korektné vytvorenie databázových selectov , ktoré bude komponent spúšťať. Okrem selectov je dôležité aj jednoznačne určiť premenné selectu.

Premenné používanékomponentom *Štatistiky*:
Časové obdobia

- time = Time.now
- today = time - time.sec - time.min*60 - time.hour*3600
- this_month = today - (time.day-1)*3600*24
- last_month = this_month - 3600*24*30-3600

Selecty realizované komponentom *Statistics* a ich názov premennej, ktorá obsahuje dáta:
Stránky používateľa

- @web_users_pages = *WebRegistration.all.select("web_registrations.web").*
  where("web_registrations.user_id=?","#{current_user.id}")

Aktuálny deň

- @new_visitors_1 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ?
  and visitors.id =
  ret_visitor_id","#{web_user_pages[1]},#{today}").count
- @old_visitors_1 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ?
  and visitors.id <>
  ret_visitor_id","#{web_user_pages[1]},#{today}").count

Aktuálny mesiac

- @new_visitors_2 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ?
  and visitors.id = ret_visitor_id","#{web_user_pages[1]},
  #{this_month}").count
- @old_visitors_2 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ? and visitors.id
  <>ret_visitor_id","#{web_user_pages[1]},
  #{this_month}").count

Minulý mesiac

- @new_visitors_3 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ? and
  time_vs.Time_event < ? and visitors.id = ret_visitor_id","
  #{web_user_pages[1]},#{last_month},#{this_month}").count
- @old_visitors_3 = *Pages.joins(:times).joins(:visitors).select("visitors.*").*
  where("pages.url=? and time_vs.Time_event > ? and
  time_vs.Time_event < ? and visitors.id <>
  ret_visitor_id","#{web_user_pages[1]},#{last_month},
  #{this_month}").count

Dáta získané z databázy je nutné po získaní z databázy previesť do formátu JSON. To je zabezpečené metódou „to_json“:

```
@chart_data = [{ name: "Nový", data:
                    [@new_visitors_1,@new_visitors_2,@new_visitors_3] },
```

```
{ name: "Vracajúci sa", data:
        [@old_visitors_1,@old_visitors_2,@old_visitors_3]
}].to_json
```

3.11 VisitorSessions

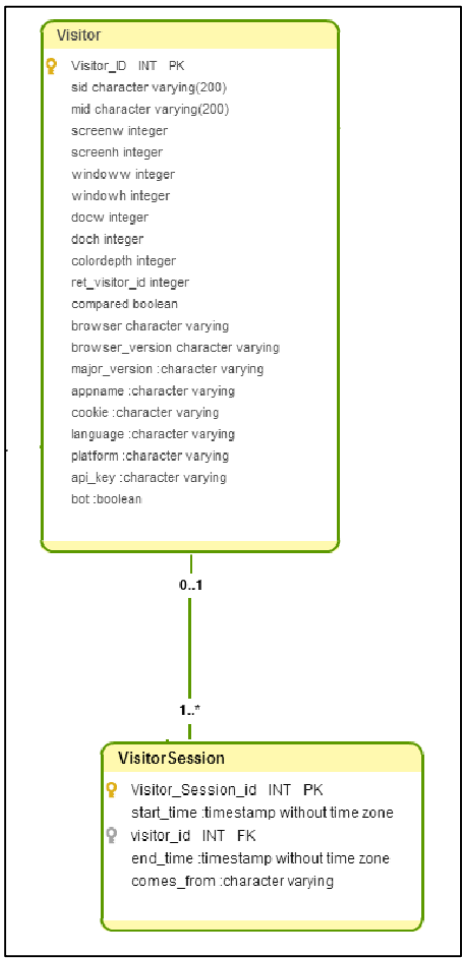
Analýza
Súčasťou aplikácie je požadované identifikovať sedenia (sessions) jednotlivých návštevníkov. Sedenie je možné definovať ako súvislú činnosť návštevníka na monitorovanej stránke. Pod súvislou činnosťou návštevníka v tomto prípade rozumie sled za sebou nasledujúcich aktivít (pohyby a kliky myšou), kde rozdiel medzi časmi ich vykonávanie nie je viac, ako zvolená konštanta.

Vzorec podmienky sedenia:
akcia[n].cas_vzniku - akcia[n-1].cas_vzniku<= KONST.minutes

Návrh
Návrh komponentu bol riešený v dvoch fázach:

1. fáza – identifikácia sedení návštevníkov
2. fáza – 2. fáza pozostáva len z jednoduchého rozšírenia už existujúceho algoritmu o identifikovanie sedenia na konkrétnu podstránku a doplnenie dátového modelu, ktoré umožní ukladanie týchto sedení realizovaných na konkrétnej podstránke.

1. fáza
Pre získanie informácii o jednotlivých sedeniach je potrebné prechádzať záznamy akcií všetkých návštevníkov a na základe timestampu ich vzniku identifikovať, ktorá akcia začína nové sedenie vybraného návštevníka a ktorá ho končí pomocou vzorca, ktorý je spomenutý vyššie. Tento proces je potrebné opakovať v periodických časových intervaloch (v našom prípade raz denne). Aby sme nemuseli počas každého intervalu spúšťať program vždy nad všetkými zaznamenanými akciami, je nutné odlíšiť, ktoré akcie už boli začlenené do sedení v predchádzajúcich dňoch. Na tento spôsob je vhodné každej akcii nastavovať flag, či už je začlenená do nejakého sedenia alebo nie.
Po každom behu tohto identifikačného procesu, je potrebné zistené informácie uložiť do databázy.
Pre tento účel bola databáza aktualizovaná nasledovane:



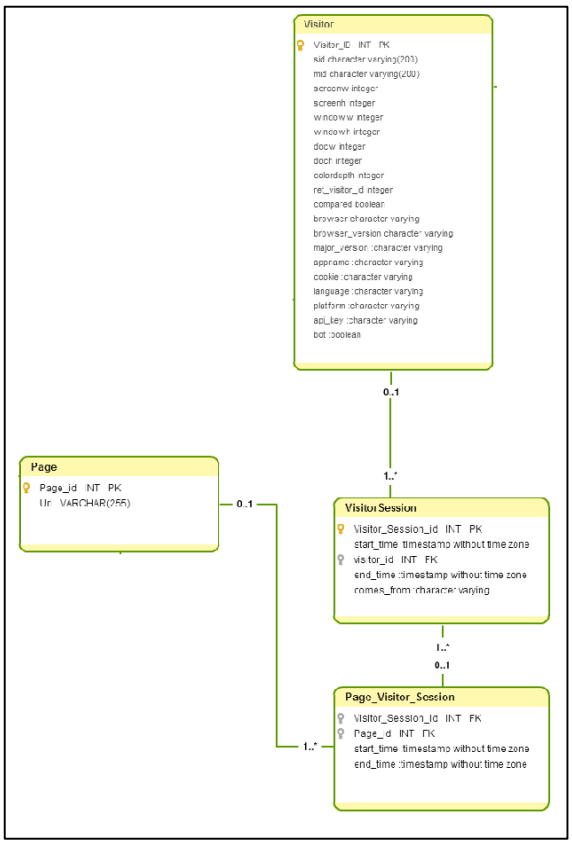
Obrázok 11 - Aktualizácia DM pre sedenia návštevníka

Takto navrhnutý algoritmus spolu s dátovým modelom našej aplikácii umožní identifikovať a ukladať jednotlivé sedenia v priamej väzbe na návštevníka.

2. fáza

Základom vzniku druhej fázy bol návrh, okrem entity zaznamenávajúcej sedenia návštevníka celkovo, doplniť aj entitu, ktorá bude zaznamenávať sedenie pre konkrétnu podstránku. Pre identifikáciu stránky, na ktorom sedenie bolo realizované je potrebné len kontrolovať, na ktorej stránke, resp. podstránke návštevník akciu vykonal a následne vykonať nový záznam v určenej tabuľke v databáze.

Návrh podmienky pre zistenie sedenia na podstránke:
akcia[n-1].stránka != akcia[n].stránka
Aktualizáciadátovéhomodelu pre potreby 2. fázy:



Obrázok 12 - Aktualizácia DM pre sedenia návštevníka na podstránke

Poznámka: konštanta pre oddelenie sedení bola stanovená na 30 minút.

Realizácia

Realizácia jednotlivých fáz neprebiehala paralelne, ale najskôr bola implementovaná funkcionalita 1. fázy a následne bolo doplnené rozšírenie pre 2. fázu. Tento postup bol zvolený pre splnenie dodanie požadovanej funkcionality v stanovenom termíne, pričom 1. fáza bola na rozdiel od tej druhej požadovaná.

Implementácia 1. fázy

Základom algoritmu pre identifikáciu sedenia je cyklus, ktorý prechádza všetky akcie, návštevníka pre vybranú monitorovanú webovú stránku (každý návštevník má atribút v podobe API kľúča, na základe ktorého je možné ho jednoznačne priradiť k stránke, ktorú navštívil) a porovnáva časy ich vzniku

Cyklus pre identifikáciu sedenia:

```
#coords predstavujú koodrinaty jednotlivých akcií návštevníka
coords.each_with_index do |c, index|
    c.update(in_session: true)     #nastavenieflagu, koordinát je užsúčasťousedenia
    if(!coords[index+1].nil?)
        if(is_new_sessions?(coords[index], cords[index+1]))
            create_new_session(cords, index, visitor_id)
        end
    else
```

```
            save_session_end(cords, index, visitor_id)
        end
end
```

Implementácia 2. fázy

Pre zrealizovanie druhej fázy, tzn. identifikovanie sedení návštevníka pre konkrétnu podstránku, polo potrebné už len doplniť nasledujúci fragment kódu hneď za: *if(!coords[index+1].nil?).*

Fragment kódu zabezpečujúci vytvorenie sedenia pre podstránku:

```
    if (is_new_page_session?(cords[index], cords[index+1]))
        create_new_page_session(cords, index)
    end
```

Samozrejme základom oboch implementačných fáz bol vhodne navrhnutý dátový model, ktorý je zobrazený v návrhu,

Testovanie

Testovanie tohto modulu spočívalo v napísaní jednotkových (Unit) testov pre metódy predstavujúce hlavnú funkcionalitu. Testovanie je rozdelené do celkov podľa jednotlivých metód, začína sa testovaním tzv. metód triedy. Pre metódu*max_visitor_sessions*satestuje vrátenie najväčšieho počtu sedení patriacich konkrétnemu návštevníkovi pomocou vytvorenia 2 návštevníkov a k nim prislúchajúcich sedení. Metóda *longest_session* sa testuje pomocou vytvorenia návštevníka a k nemu prislúchajúcich sedení. Predpokladá sa vrátenie časovo najdlhšieho sedenia. Metóda *visited_pages_num* sa testuje vytvorením návštevníka a k nemu prislúchajúcich sedení. K sedeniam sa vytvoria prislúchajúce *page_visitor_sessions*. Testuje sa vrátenie správneho počtu navštívených stránok pre konkrétneho návštevníka. Metóda *visitor_sessions_num*je otestovaná pomocou jednoduchej kontroly počtu vrátených sedení pre návštevníka. Metóda *visited_pages_session* testuje počet navštívených stránok pre konkrétne sedenia jednoduchou kontrolou očakávaného počtu vrátených hodnôt. V nasledujúcom bloku sa testuje hlavná funkcionalita modulu.

Patria tam metódy *create_all_sessions, create_web_sessions, create_visitor_sessions*,takisto privátne metódy *create_new_page_session, create_new_session, save_session_ends, is_new_page_session?, is_new_session?*. Prvé tri spomínané metódy sú na sebe závislé, naviac privátne metódy využívajú inštančné premenné. Z tohto dôvodu sa táto časť testuje ako celok. Pre vstupné dáta sa kontroluje vytvorenie *visitor_sessions*, označenie použitých koordinátov, vytvorenie *page_visitor_sessions*. Metódy sa testovali aj pre negatívne prípady.

3.12 Tech Filter

Komponent *Tech Filter,* má za úlohu spresniť vstupnú množinu pre komponent *Histogram,* ktorý zabezpečuje párovanie zhodných návštevníkov.

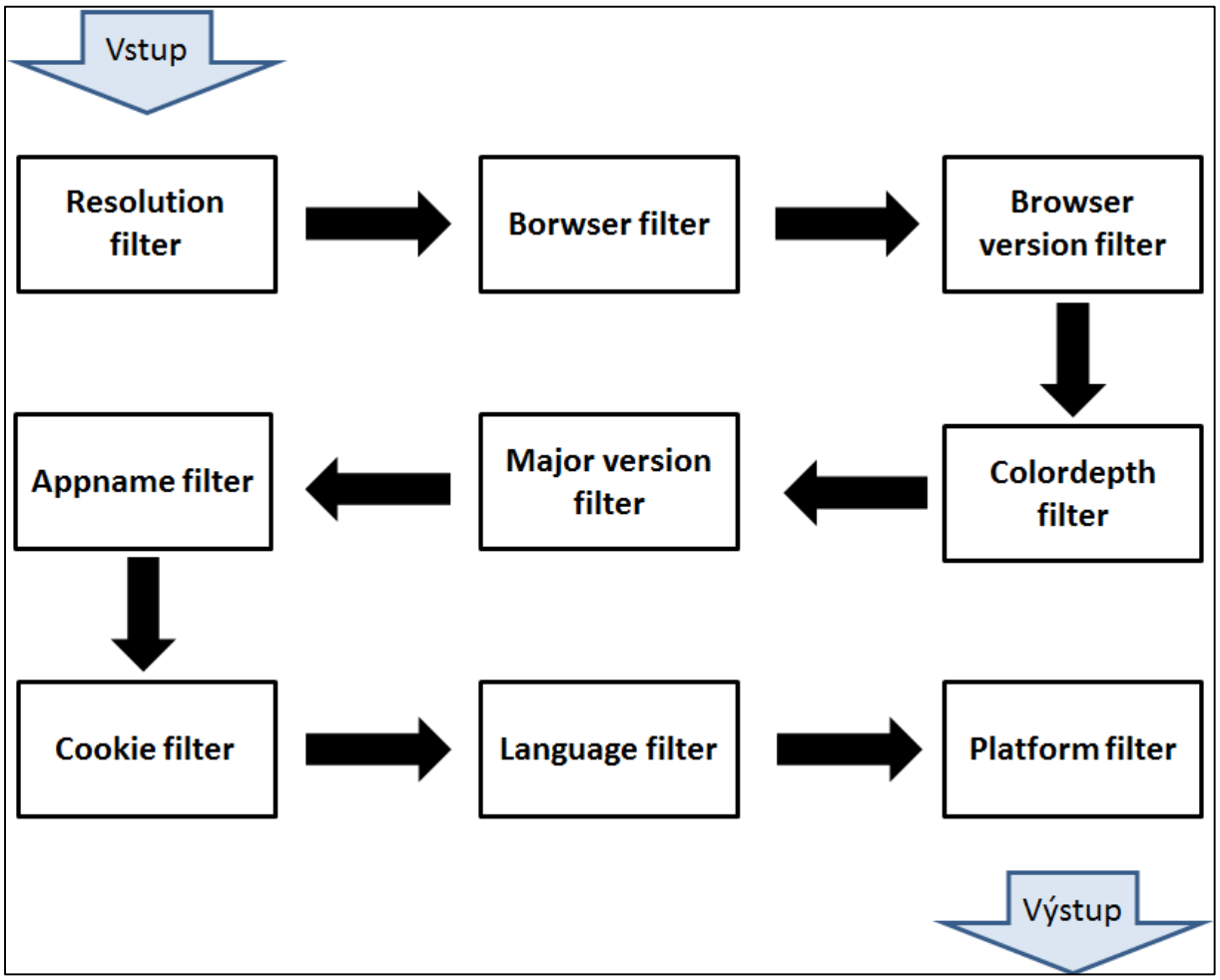
Analýza
Vstupom do komponentu *Tech Filter* sú všetky záznamy o návštevníkoch, ktoré sa nachádzajú v našej aplikácii, ktorých vybrané atribúty sú porovnávané s novým návštevníkom. Pre potreby vymedzenia užšej množiny možných vstupov pre komponent *Histogram,* boli identifikované nasledujúce atribúty ako filtrovacie kritéria:

- rozlíšenie (resolution)
- prehliadač (browser)
- plná verzia prehliadača (borwser_version)
- farebná hĺbka (colordepth)
- hlavná verzia prehliadača(major_version)
- cookie(cookie)
- jazyk (language)
- platforma (platform)

Návrh
Pri návrhu toho komponentu bolo dôležité zodpovedať otázky v akom poradí majú jednotlivé filtre za sebou nasledovať a čo ma komponent spraviť ak nejaký filter vráti prázdnu množinu.
Na prvú otázku sa bez otestovania komponentu v praxi dá len ťažko odpovedať a preto sme sa pokúsili zvoliť stratégiu, kedy filtre, ktoré dokážu najviac zmenšiť vstupnú množinu budú aplikované ako prvé. Následne sme dostali postupnosť:

1. rozlíšenie (resolution)
2. prehliadač (browser)
3. plná verzia prehliadača (borwser_version)
4. farebná hĺbka (colordepth)
5. hlavná verzia prehliadača (major_version)
6. cookie (cookie)
7. jazyk (language)
8. platforma (platform)

Meniť poradie filtrov je vďaka zvolenému architektonickému vzoru popísanému nižšie len triviálna záležitosť a preto sa bude počas behu aplikácie s poradím experimentovať v snahe dosiahnuť čo najlepšie výsledky.
Na druhú otázku sme ako odpoveď aplikovali architektonický vzor „*Chains of Resposibility*“. Architektonický návrh komponentu *Tech Filter* založenom na vzore „*Chains of Resposibility*“ je možné vidieť na*Obrázok 13 - Štýl pre implementovanie komponentu TechFitra*.



Obrázok 13 - Štýl pre implementovanie komponentu TechFitra

Ako je možné vidieť, tak postupne sú aplikované všetky filtre, ktoré sú dostupné. V prípade, že nejaký filter vráti nulový výstup, tzn. žiadny evidovaný návštevník nemá rovnaký požadovaný atribút ako nový (porovnávaný), tak sa jednoducho zoberie posledný nenulový výstup a ten poslúži ako vstup do nasledujúceho filtra.

Realizácia

Komponent bol podľa návrhu realizovaný ako séria za sebou nasledujúcich filtrov. Každý z týchto filtrov je pre krátkosť zdrojového kódu implementovaný ako samostatná metóda, pričom vo výsledku sú všetky metódy filtrov zavolané zoskupujúcou metódou nazvanou *filter_all*, ktorá určuje poradie vykonávania jednotlivých filtrov. Táto metóda taktiež vyhodnocuje podmienky, tzn. určuje, ktorý vstup je poslaný do ďalšieho filtra (tu sa implementácia líši od vzoru „Chains of Responsibility“, keďže v tomto vzore sú podmienky vyhodnocované vo filtroch a nie nad nimi).

Použitie metódy filter_all:

result = TechFilter.filter_all(visitors, new_visitor)
visitors– množina návštevníkov, z ktorej sa má filtrovať, resp. v ktorej sa majú hľadať zhody.

new_visitor – nový návštevník, resp. filtrovacie kritérium. Atribúty objektu *new_visitor* sa budú vyhľadávať v množine *visitors*.

Testovanie

Funkcionalita bola testovaná nad umelo vytvorenou množinou záznamov o počte 1000. Túto množinu automaticky vygeneroval pripravený script, spolu so správnymi výstupmi, ktoré má komponent poskytnúť.

3.13 Webpagesmanger

Naša aplikácia musí vedieť rozpoznať, ktorý používateľ monitoruje, ktoré webové stránky. Tieto informácie sú dôležité ako pri prijímaní dát, kedy je potrebné dáta priradiť práve jedenému registrovanému používateľovi, tak pri ich zobrazovaní, aby nedochádzalo k zámenám a miešaniu informácií. Práve túto funkcionalitu zabezpečuje komponent *Webpages manager*.

Analýza
Realizácii tohto komponentu nepredchádzala žiadna výrazná analýza, keďže problém, resp. funkcionalita, ktorú komponent zabezpečuje je jednoduchá a pomerne jasná.

Návrh
Aby bolo možné registrovanú stránku jednoznačne určiť, je potrené jej vygenerovať unikátny API kľúč (jednoznačný identifikátor). S cieľom dosiahnuť čo možno najväčšiu variabilnosť bol pre generovanie API kľúča použitý vzorec:

ABSOLUTNÁ_HODNOTA(HASH(„používateľov e-mail“ + STRING(„aktuálny timestamp“)))

Aj keď by tento vzorec mal zabezpečovať vysokú pravdepodobnosť unikátnosti je pre istotu každý novo vygenerovaný API kľúč porovnávaný so zoznamom všetkých API kľúčov.
Okrem vygenerovania API kľúča sa pri registrácii automaticky vytvorí väzba medzi používateľom a webovou stránkou, na základe čoho bude možné následne pri prijatí requestu jednoznačne určiť z akého webu sú prichádzajúce dáta a ktorému používateľovi sú určené.

Realizácia
Komponent bol realizovaný ako riešenie, ktoré okrem backendovej funkcionality, generovanie API kľúčov a vytváranie nových záznamom pre registrované webové stránky, poskytuje aj elektronické formuláre na zadávanie informácii a registrovanej stránke (bližšie popísané v kapitole 0*E-form*). Pre generovanie API kľúča boli použité metódy dostupných knižníc Ruby on Rails, kde následne bolo možné zostrojiť vlastnú metódu pre vytvorenie API kľúča, ktorá vyzerá nasledovne:

```
defapiHashUniq
        webs = WebRegistration.pluck(:api_key)
        api_hash = "#{current_user.email} + #{Time.now}".hash.abs.to_s
        api_hash = "#{current_user.email} + #{Time.now}".hash.abs.to_swhile (webs.include?(api_hash))
        api_hash
end
```

Ako je možné vidieť, tak metóda zabezpečujeaj kontrolu na prípadné vygenerovanie už existujúceho API kľúča. Aj keď tento interný while cyklus spôsobí zvýšenie časovej náročnosti, nepredpokladá sa príliš vysoká frekvencia spúšťania tejto metódy. V porovnaní s problémami a chybami, ktorý by spôsobili dva rovnaké API kľúče je takéto zvýšenie časovej náročnosti prijateľné.

Testovanie

Testovanie prebiehalo v dvoch etapách:

1. Testovanie registrácie a editácie webovej stránky – boli realizované základné prípady použitia, kedy používateľ pridával a editoval stránky. Pridávanie stránok bolo realizované pod viacerými používateľmi s cieľom zistiť, či nedôjde k zlému zobrazeniu.
2. Testovanie unikátneho generovania API kľúča –boli realizované automatizované testy, kedy sa metóda vložila do cyklu s 10 000 opakovaniami pri použití toho istého prihlasovacieho e-mailu. Následne sa cyklus spustil a pri každej zhode v API kľúčoch hlásil chybovú správu. Pretože cyklus bežal v prvých iteráciách príliš rýchlo bolo v prvej stovke zopár správ o vygenerovaní rovnakých API kľúčov, tie ale následne ošetrila samotná metóda svojím interným while cyklom.

4 Záver

Naša aplikácia po zimnom semestri a realizovaných piatich šprintoch už dokáže plniť svoju základnú funkciu. Okrem prijímania a ukladania prijatých dát z webových stránok dokáže s týmito dátami pracovať, vytvoriť model návštevníka. S aktuálnymi dátami dokážemes presnosťou 100% rozlíšiť nového a vracajúceho sa návštevníka. Nad týmito dátami realizujeme štatistické výstupy, ktorých časť vykresľujeme do grafov a časť do tabuliek, kde môžeme spomenúť napríklad počet a taktiež čas strávený na sedení na používateľa, počet prezretých stránok na sedenie a zobraziť návštevníkov z akej stránky prišli.Dôležitá je najmä ich presnosť zobrazovania získaných dát vďaka odstraňovaniu duplicít.Taktiež dokážeme rozlíšiť či stránku navštívil používateľ alebo bot respektíve crawler.
V najbližších týždňoch budeme pracovať na vizualizovaní získaných dát v podobe štatistík, ktoré budeme naďalej pridávať. Čaká nás implementácia heat mapy a aj úprava grafického rozhrania aplikácie.Úlohou je nasadiť našu aplikáciu na rôzne webové stránky z ktorých získame ďalšie dáta. Toto nám pomôže naďalej zlepšovať naše riešenie.

5 Inštalačná príručka

Pre spojazdnenie funkčnej verzie našej aplikácie je ako prvé potrebné mať webový server, či už virtuálny alebo fyzický s minimálnymi parametrami 16 GB priestoru na disku, 2GB RAM a dvojjadrový procesor s frekvenciou aspoň 2 GHz. Vhodným jadrom je Apache webový server (najnovšia verzia) bežiaca pod operačným systémom Ubuntu 64-bit.

Náš projekt beží na platforme Ruby on Rails, čiže je potrebné si nainštalovať Ruby interpreter dostupný príkazom:

```
$ sudo apt-get install ruby-full
```

Náš systém beží nad databázou PostgreSQL, ktorú je potrebné nainštalovať:

```
$ sudo apt-get install postgresql
```

Ďalším krokom je nainštalovanie Bundler-a a Rails-ov a Git-u:

```
$ gem install bundler
```

```
$ gem install rails
```

Po nakopírovaní zdrojových súborov na server (buď priamo alebo naklonovaním z repozitára na GitHub-e) je potrebné spustiť doinštalovanie potrebných gemov.

```
$ bundle install
```

Na nastavenie databázy do prvotného stavu:

```
$ rake db:schema
```

Na nastavenie do finálneho stavu:

```
$ rake db:migrate
```

Pre správne fungovanie komponentu Heatmap je potrebné doinštalovať dve binárne súbory podľa verzie systému dostupné na stránke:

```
http://www.imagemagick.org/script/binary-releases.php
```

```
http://download.gna.org/wkhtmltopdf/0.12/0.12.3/
```

V pípade využitia serveru postaveného na inej ako Unix-like architektúre je potrebné nainštalovať ešte:

```
https://sourceforge.net/projects/gnuwin32/files/sed/4.2.1/sed-4.2.1-setup.exe/download
```

V prípade akýchkoľvek problémov sa obráťte na náš vývojový tím na adrese visitortrack.team@gmail.com.