VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

GRAFICKÁ VIZUALIZÁCIA SPRÁVY UDALOSTÍ

GRAPHICAL VISUALIZATION OF INCIDENT HANDLING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VERONIKA MARKOVIČOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Studentka: **Markovičová Veronika**
Program: Informační technologie
Název: **Grafická vizualizácia správy incidentov**
Graphical Visualization of Incident Handling
Kategorie: Uživatelská rozhraní

Zadání:

1. Preštudujte nástroj firmy Flowmon pre detekciu anomálií.
2. Urobte prieskum existujúcich spôsobov vizualizácií vhodných pre návrh prehľadových obrazoviek typu dashboard so zameraním na vizualizáciu udalostí a incidentov. Preštudujte technológie pre tvorbu webových užívateľských rozhraní a vizualizáciu dát (React, D3.js, apod.).
3. Analyzujte požiadavky firmy Flowmon na vizualizáciu stavu incidentov.
4. Navrhnite grafický prvok splňujúci požiadavky z bodu 3.
5. Navrhnutý grafický prvok implementujte formou webovej parametrizovateľnej komponenty použiteľnej v rôznych situáciách.
6. Otestujte riešenie v spolupráci s firmou Flowmon.

Literatura:

- Few, S.: *Information Dashboard Design: The Effective Visual Communication of Data.* O'Reilly, 2006, ISBN: 978-059-6100-162.
- Harris, R. L.: *Information Graphics: A Comprehensive Illustrated Reference.* Oxford University Press, 2000. ISBN: 978-0-1951-3532-9.
- Cichonski, et al.. *Computer security incident handling guide*. 2012, NIST Special Publication.
- D3.js: *D3.js - GitHub Wiki* [online]. 2019 [cit. 2020-10-15]. Dostupné z: https://github.com/d3/d3/wiki
- Interní dokumentace firmy Flowmon.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Táto bakalárska práca popisuje návrh a implementáciu komponentov, ktoré vizualizujú dáta o incidentoch a ich stavoch. Cieľom týchto komponentov bolo vytvoriť prehľadnú vizualizáciu incidentov a ich stavov, ktorá by ich užívateľom uľahčila hľadanie informácií o incidentoch a zefektívnila tak prácu pri ich riešení. Jeden z vytvorených komponentov zobrazuje detailný popis jednotlivých stavov incidentu. Druhý vytvorený komponent zobrazuje buď jeden alebo viac incidentov ako postupnosť ich stavov na časovej ose. Ako vhodný nástroj pre vizualizáciu a demonštráciu vytvorených komponentov bol zvolený dashboard. Komponenty boli konzultované a testované s firmou Flowmon.

Abstract

This bachelor thesis describes the proposal and implementation of components that visualize data on incidents and their states. These components aims to create a clear visualization of incidents and their states that would facilitate the search for information on incidents and thus streamlining their work. One of the components created shows a detailed description of individual states of incident. The second created component shows either one or more incidents as the sequence of their states on the timeline. As a suitable tool for visualization and demonstration of created components, dashboard has been chosen. Components were consulted and tested with Flowmon.

Kľúčové slová

vizualizácia, udalosti, incidenty, stavy, správa incidentov, detekcia anomálií, ReactJS, D3.js, nivo, dashboard, grafy, komponenty, časová osa

Keywords

visualization, events, incidents, states, incident handling, anomaly detection, ReactJS, D3.js, nivo, dashboard, graphs, components, timeline

Citácia

MARKOVIČOVÁ, Veronika. *Grafická vizualizácia správy udalostí.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek, Ph.D.

Grafická vizualizácia správy udalostí

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Jiřího Hynka, Ph.D. a uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Veronika Markovičová
10. mája 2021

Poďakovanie

Týmto by som chcela poďakovať vedúcemu práce, Jiřímu Hynkovi, za vedenie práce a za cenné rady a pripomienky. Taktiež by som chcela poďakovať svojim rodičom a priateľovi za veľkú psychickú podporu.

Obsah

Kapitola 1

Úvod

S nárastom aktivity na sieti, Internete, pribúda tiež množstvo udalostí, ktoré môžu byť potenciálne nebezpečné a je potreba ich riešiť. Preto je dôležitá ich detekcia a vyriešenie predtým, než postihnú nejaké dôležité dáta užívateľa. Firmy, ktoré sa zaoberajú touto detekciou anomálií, často využívajú rôzne nástroje pre zobrazovanie udalostí, ktoré im sprehľadňujú tieto incidenty, a tak uľahčujú prácu s nimi a ich vyriešenie.

V tejto práci som sa preto zamerala na vytvorenie niekoľkých parametrizovateľných komponentov, ktoré v rámci vizualizácie udalostí neboli nikde implementované a bolo potreba ich vytvoriť. Tieto komponenty zobrazujú incidenty a ich stavy z niekoľkých pohľadov: komponent 1 zobrazuje jeden incident ako postupnosť stavov v čase, komponent 2 zobrazuje viacero incidentov ako postupnosť ich stavov v čase nad sebou, či komponent 3 s detailným popisom k jednotlivým stavom incidentu.

Komponenty 1 a 2 boli vytvorené s cieľom čo najlepšie oboznámiť užívateľov o danej udalosti, tak aby najdôležitejšie informácie boli viditeľné hneď na prvý pohľad a ich detaily skryté hlbšie. Komponent 3 je zameraný na detailný popis jednotlivých stavov incidentov, ktorý zobrazuje podrobnejšie informácie pre užívateľov, ktorí s komponentami budú pracovať. Cieľom týchto komponent tiež bolo, aby mohli byť použiteľné nielen pre prácu s incidentami a ich stavmi, ale i v rôznych ďalších podobných situáciách.

Práca obsahuje teoretický úvod do problematiky detekcie anomálií popísaný v kapitole 2 a teoretický úvod do problematiky vizualizácie incidentov v kapitole 3. V kapitole 4 je popísaná analýza požiadavkov na vizualizáciu incidentov a ich stavov od firmy Flowmon a následne je uvedený návrh riešenia týchto požiadavkov v kapitole 5. V ďalšej kapitole 6 je popísaný spôsob implementácie jednotlivých komponentov a použité technológie pre ich tvorbu. Kapitola 7 popisuje testovanie vytvorených komponentov v spolupráci s firmou Flowmon.

Kapitola 2

Detekcia anomálií

Skôr než sa začneme venovať pojmom ako anomália a detekcia anomálií je vhodné uviesť, čo je to udalosť a kde vzniká. Podľa Cichonského [15] *udalosť* obecne vzniká na sieti alebo v systéme. Udalosťou je napríklad pripojenie užívateľov k zdieľanému súboru, server dostávajúci požiadavok na webovú stránku, užívateľ posielajúci email, firewall blokujúci pokus o pripojenie atď. Ďalej potrebujeme zaviesť *koncept normality*, ktorý definuje napríklad Zhang [30] takto:

> The notion of 'normal' is usually provided by a formal model that expresses relations between the fundamental variables involved in the system dynamics.

Môžeme teda povedať, že udalosť je zaradená ako anomália, pokiaľ má dostatočne veľkú odchýlku od definovaného modelu normálneho správania systému [30]. Alebo jednoducho povedané, *anomália* je odchýlka od normálneho správania. Thottan [27] i Bhuyan [13] rozdeľujú anomálie do dvoch kategórií podľa ich dôsledkov na *performace related* (dôsledky spojené s výkonom) a *security related* (dôsledky spojené s bezpečnosťou).

2.1 Systém pre detekciu anomálií

Ako uvádza Zhang v [30]:

> Formally, an anomaly detection system S can be defined as a pair $S = (M, D)$, where M is the model of normal behavior of the system and D is a similarity measure that allows obtaining, given an activity record, the degree of deviation that such activities have with regard to the model M.

Jadro systému je teda rozdelené do 2 modulov: *modelovacieho* a *detekčného* [30]. Modelovací modul pracuje počas tréningovej fázy a vykonáva spracovávanie udalostí, s cieľom získať model M normálneho správania systému. Čím presnejšie budeme modelovať toto správanie, tým lepšiu detekciu anomálií získame [27]. Tento získaný model je následne použitý v detekčnom module na vyhodnotenie nových udalostí. Toto vyhodnotenie je určovanie stupňa odchýlky od modelu M. Oba moduly pracujú samostatne, avšak systémy sa stále vyvíjajú a je potreba ich periodicky upravovať, aby boli stále aktuálne.

2.2 Metódy detekcie anomálií

K detekcii anomálií je možné využiť rôzne techniky ako napríklad umelú inteligenciu, strojové učenie, stavové automaty, štatistickú analýzu, porovnávanie vzorov, atď. Nie je ľahké

presne určiť triedy metód, pretože každá literatúra uvádza iné triedenie týchto metód. Thottan v [27] rozdeľuje metódy podľa *prístupov založených na pravidlách, modely konečných stavových automatov, porovnávanie vzorov a využitie štatistickej analýzy.* Zhang v [30] popisuje rozdelenie metód pre detekciu anomálií pomocou *klasifikátora, strojového učenia, štatistík a konečných stavových automatov.* Avšak v literatúrach často nachádzame i spoločné rozdelenie detekcie anomálií napríklad podľa štatistík, stavových automatov či klasifikátora.

Detekcia anomálií pomocou štatistík

Pre zisťovanie anomálií systém sleduje aktivitu subjektov a generuje ich profily [30]. Spravidla sa uchovávajú o každom subjekte dva profily: *aktuálny* a *uložený* profil. Po spracovaní udalostí v systéme (na sieti) sa aktualizuje aktuálny profil a pravidelne počíta skóre anomálií porovnaním aktuálneho profilu s uloženým profilom. Pokiaľ je skóre vyššie než určitý prah, systém vygeneruje výstrahu.

Výhody štatistických metód:

- nevyžadujú predchádzajúcu znalosť bezpečnostných chýb či samotných útokov,
- môžu poskytnúť presné oznámenie o škodlivých činnostiach, ktoré sa zvyčajne vyskytujú počas dlhšieho časového obdobia.

Nevýhody štatistických metód:

- môže byť ťažké určiť prahové hodnoty, ktoré vyvážia pravdepodobnosť falošne pozitívnych s pravdepodobnosťou falošne negatívnych,
- potrebujú presné štatistické rozdelenie,
- nie všetky spôsoby správania je možné modelovať pomocou čisto štatistických metód.

Detekcia anomálií pomocou klasifikátora

Závisí na myšlienke, že normálne charakteristické vlastnosti sa dajú odlíšiť od abnormálneho správania [30]. Klasifikátor sa môže použiť na predikciu prichádzajúcej normálnej udalosti vzhľadom na aktuálnu udalosť. Ak počas fázy monitorovania nasledujúca udalosť nie je tá, ktorú predpovedal klasifikátor, je označená ako anomália. Patria sem napríklad podľa Zhanga [30] *techniky generovania indukčného pravidla, techniky založené na fuzzy logike, genetické algoritmy* alebo v [12] sú uvedené: *Support Vector Machines, Bayesiánske siete či neurónové siete.*

Detekcia anomálií pomocou strojového učenia

Cieľom strojového učenia je odpovedať na rovnaké otázky ako štatistika [30]. Avšak na rozdiel od štatistických prístupov, ktoré sa zameriavajú na porozumenie procesu, ktorý vytvoril dáta, techniky strojového učenia sa sústredia na budovanie systému, ktorý zlepšuje svoj výkon na základe predchádzajúcich výsledkov. Teda systémy založené na strojovom učení majú schopnosť meniť stratégiu vykonávania na základe novo získaných informácií.

Detekcia anomálií pomocou konečných stavových automatov

Model normálneho správania je zložený zo stavov, prechodov a akcií [30]. V tomto modeli *stav* uchováva informácie o minulosti, *prechod* označuje zmenu stavu a je popísaný podmienkou, ktorá musí byť splnená, aby mohol byť umožnený. *Akcia* popisuje činnosť, ktorá sa má vykonať v danom okamihu.

Pomocou konečných automatov je možná nielen detekcia anomálií, ale je tiež možné identifikovať a diagnostikovať problém. Nevýhodou je, že pred tým ako použijeme konečný stavový automat, musíme ho dopredu zostaviť na základe už zaznamenaných dátach a chybách a musíme ho udržiavať stále aktuálny [27]. Konečný stavový automat bol použitý napríklad pre detekciu útokov na DSR (Dynamic Source Routing) protokol v [24].

Detekcia anomálií pomocou zhlukovania

Zhlukovanie je metóda hľadania súvislostí v celku a následné utriedenie objektov na základe podobných vlastností do zhlukov. Zhlukovanie pri detekcii anomálií sa týka algoritmov učenia bez dozoru, ktoré na získanie pravidiel na zoskupovanie podobných inštancií údajov nevyžadujú vopred označené údaje [12].

V práci [12] sa uvádzajú tri kľúčové predpoklady, ktoré sa určujú pri detekcii anomálii pomocou zhlukovania:

1. Môžeme vytvoriť zhluky iba normálnych dát. Potom nové údaje, ktoré sa nehodia už k existujúcim zhlukom sa považujú za anomálie.

2. Zhluk obsahuje normálne i anomálne dáta. Normálne dáta v zhluku ležia blízko pri jeho strede a anomálie sú ďaleko od stredu. Podľa tohto predpokladu sa teda anomálie určujú pomocou veľkosti vzdialenosti od stredu zhluku.

3. Zhluky môžu byť rôznych veľkostí a môžu mať rôznu hustotu. Potom menšie a redšie zhluky sú považované za anomálie a hrubšie zhluky sú normálne. Teda dáta, ktoré patria do zhluku s veľkosťou a hustotou menšou než určitý prah sú považované za anomálie.

2.3 Incident response

Ako popisuje Cichonski vo svojej práci [15], anomália môže byť detegovaná rôznymi spôsobmi. Môže ju nahlásiť napr. administrátor alebo nejaký zamestnanec firmy či priamo koncový užívateľ. Taktiež môže byť zistená automaticky napríklad pomocou skriptov či antivírusových programov.

Pri nahlásení udalosti vzniká incident, ktorý je treba riešiť. Vzniká tzv. *incident response*, odozva na udalosť, ktorá by mala byť rýchla a efektívna. Mala by odpovedať na incidenty systematicky podľa *incident handling methodology*, ktorá má niekoľko fáz a podľa AT&T [22] rozoznávame 2 druhy postupov pre riešenie incidentov: od NIST (National Institute of Standards and Technology) a SANS (SysAdmin, Audit, Network, and Security). Proces riešenia incidentu od NIST obsahuje 4 kroky:

1. Príprava

2. Detekcia a analýza

3. Obmedzenie, likvidácia a obnova

4. Činnosť po incidente

Proces riešenia incidentu od SANS obsahuje 6 krokov:

1. Príprava
2. Identifikácia
3. Obmedzenie
4. Likvidácia
5. Obnova
6. Ponaučenie

2.3.1 Postup riešenia incidentov

Ako popisuje Chichonski v [15], incident vzniká jeho detekciou v systéme alebo na sieti a je dôležité ho začať riešiť čo najrýchlejšie. V prvom rade by malo byť nahlásenie incidentov prístupné každému v organizácii, kto zistí alebo má podozrenie na anomáliu, aby ju mohol nahlásiť. Incident potom prevezme tím, ktorý sa ho bude snažiť vyriešiť. Rieši ho buď jednotlivec alebo skupina ľudí z tímu. Na začiatku sa incident analyzuje a priraďuje sa mu priorita alebo závažnosť, určí sa jeho dopad a konajú sa adekvátne kroky k jeho obmedzeniu a zničeniu, tak aby sa čo najviac zmiernili škody a pokiaľ je to možné, obnoveniu služby do normálneho stavu.

Existuje niekoľko typov incidentov, a preto je potreba vytvoriť postup pre riešenie každého typu incidentu. Organizácia by mala byť schopná riešiť hocijaký incident, no veľakrát sa opakujú alebo spadajú do podobnej kategórie, a tak je dobré mať pripravené inštrukcie ako pri každom zatiaľ známom type incidentu postupovať. Incident môže vyvolať napríklad:

- **externé médium** pripojené do systému, na ktorom môže byť nahratý škodlivý softvér,
- **vyčerpanie** – DDoS útok, ktorý napáda systém s cieľom vyradiť jeho funkčnosť,
- **web** – presmerovanie na stránku, ktorá nie je pravá a inštalácia malwaru,
- **email** – link na škodlivú webovú stránku v tele správy,
- **zosobnenie** – Man In The Middle útok – odpočúvanie komunikácie medzi účastníkmi, spoofing – napodobňovanie osoby alebo programu sfalšovaním dát,
- **nesprávne používanie** – inštalácia softvéru, ktorý nie je overený a tak zanesienie škodlivého softvéru do systému organizácie,
- **strata alebo krádež vybavenia** - laptop, mobil, ...

Najdôležitejšou súčasťou analýzy incidentu je určenie jeho priority, podľa ktorej sa bude ďalej riešiť. Priorita sa udáva podľa rôznych kritérií, ako je napríklad dopad incidentu na systém či užívateľa, porušenie bezpečnosti, funkčnosť systému apod. Cichonski v svojej práci [15] udáva tieto tri dopady incidentu na systém:

- **funkčný dopad incidentu** – dopad na biznis funkcionalitu systému, ktorého výsledkom je negatívny dopad na jeho užívateľov (možnosť poskytovať služby klientom),
- **informačný dopad incidentu** – incidenty môžu ovplyvniť dôvernosť, integritu a dostupnosť informácií organizácie (exfiltrácia citlivých informácií),
- **návratnosť z incidentu do normálneho stavu** – veľkosť incidentu a typ zdrojov, ktoré ovplyvňuje, určí množstvo času a zdrojov, ktoré je potrebné vynaložiť na zotavenie sa z tohto incidentu, v niektorých prípadoch však už nie je možné zotaviť sa späť do normálneho stavu (napr. pri narušení dôvernosti citlivých informácií).

2.4 Anomaly Detection System od firmy Flowmon

Text tejto podkapitoly bol čerpaný z užívateľskej príručky k ADS od firmy Flowmon [5], kde je tiež možné získať ďalšie informácie o tomto systéme.

Anomaly Detection System od firmy Flowmon je moderný systém pre detekciu anomálií a nežiadúcich vzorov správaní na sieti. Je založený na analýze dátových tokov na sieti a jeho hlavným cieľom je zvýšenie bezpečnosti počítačovej siete. Orientuje sa hlavne na celkové správanie zariadení na sieti, čo umožňuje reagovať na doposiaľ neznáme a špecifické hrozby.

K odhaľovaniu rôznych potenciálne nežiaducich aktivít na sieti ADS využíva detekčné metódy, ktoré sú preddefinované výrobcom, časom sa aktualizujú podľa potreby a sú roztriedené do nasledujúcich skupín:

- **bežné vzory správania siete** – generujú udalosti vždy na základe spracovania dávky flow dát,
- **bežné vzory správania pre SIP prevádzku** – fungujú len so zdrojmi dát, ktoré majú zapnuté spracovanie SIP prevádzky,
- **pokročilé vzory správania siete** – detegujú dlhodobé trendy v správaní siete na základe priebežného spracovania dátových tokov,
- **systém detekcie anomálií** – metódy obecného systému detekcie anomálií na základe zmeny správania zariadení v sieti.

Detekčné metódy dokážu odhaliť **anomálie v sieti** (prenos veľkých objemov dát na sieti, nedostupnosť služieb, priamy prístup do Internetu, potenciálne cudzie zariadenia v sieti), **vzdialenú správu** (detekcia protokolu Telnet), **útoky** (detekcia slovníkových útokov na služby FTP, HTTP, IMAP, SSH, útoky typu DoS, detekcia TCP scanov, odchádzajúceho spamu) či **používanie nežiadúcich služieb** (detekcia peer-to-peer sietí BitTorrent, zdieľaní pracovnej plochy pomocou programu TeamViewer).

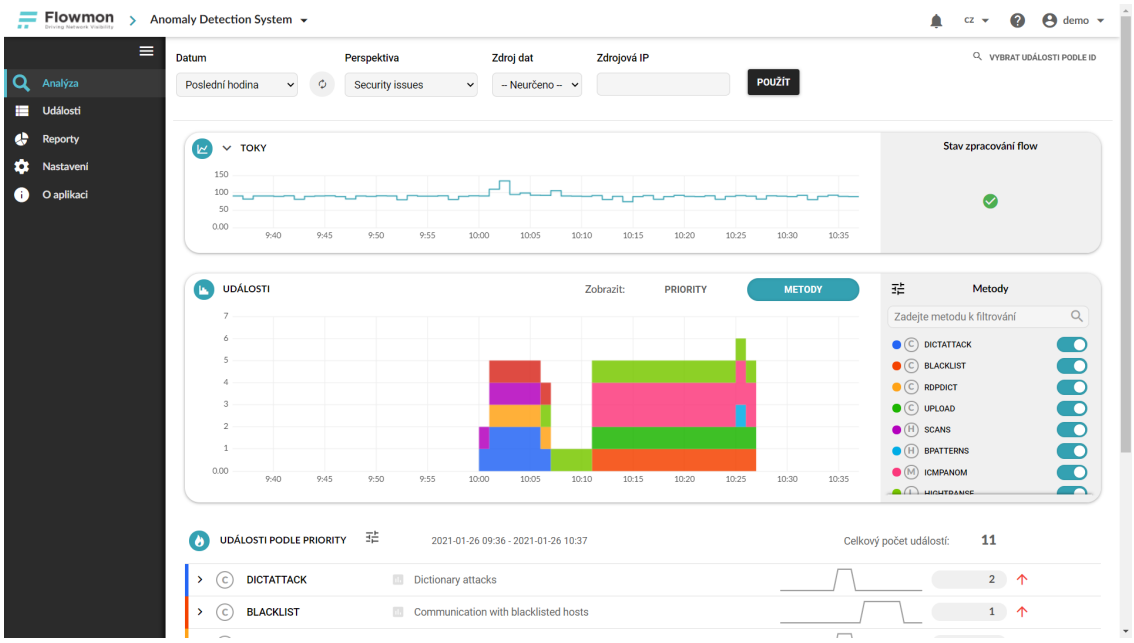
Všetky detekčné metódy majú 3 spoločné vlastnosti:

1. **generujú udalosti**, ktoré obsahujú päť kľúčových atribútov: *zdroj* (IP adresa, ktorá túto anomáliu spôsobila), *typ udalosti, podtyp, zdroj dát* a *inštancia metódy*, ktorá túto anomáliu detegovala, ďalej môže obsahovať časovú známku výskytu udalosti podľa dátových tokov, odkaz na zdroj dát, detail udalosti, zoznam cieľov udalosti (IP adresy) ai.,
2. **niektoré sa delia na podmetódy** – sú vždy uvedené u detailu udalosti,

3. **periodicky mažú udalosti** – detekčné metódy, ktoré generujú udalosti ich môžu aj zmazať nastavením konfiguračnej voľby, ktorá udáva po aký čas (počet dní) zostáva udalosť v pamäti uložená.

Užívatelia si tiež môžu sami nadefinovať vlastné vzory správania, ktoré chcú sledovať pomocou systému, môžu si tiež vytvárať blacklisty, nastavovať notifikácie, vkladať vlastné skripty apod.

Na nasledujúcom obrázku 2.1 je možné vidieť úvodnú obrazovku ADS, ktorá zobrazuje tok dát a zaznamenané udalosti v čase, ktoré sú farebne rozlíšené podľa typu metódy, ktorá ich zaznamenala.



Obr. 2.1: Úvodná obrazovka ADS od Flowmon obsahuje niekoľko komponentov, ktoré zobrazujú informácie o udalostiach na sieti. Prvý komponent *toky* zobrazuje flow za sekundu v danom časovom intervale. Graf *události* je skladaný stĺpcový graf, ktorý zobrazuje počty udalostí zoskupených podľa ich typu či priority. Taktiež je možné medzi týmito typmi prepínať v pravom paneli. V sekcii *Události podle priority* sú zobrazené všetky generované udalosti a ich počet pre každý typ udalosti [5].

Kapitola 3

Vizualizácia dát

Dáta je výraz pre údaje, ktoré popisujú jav alebo vlastnosti pozorovaného javu. Sú to teda fakty, údaje či hodnoty rôznych dátových typov, ktoré reprezentujú nejakú skutočnosť. V [26] sa uvádza, že sú to informácie v digitálnej podobe, ktoré môžu byť prenášané alebo spracovávané. Získavajú sa zápisom, pozorovaním či meraním a môžu byť reprezentované ľubovoľnými reťazcami znakov (čísiel, príkazov, viet). Spravidla samé o sebe nemajú žiadny význam, ale až sú pochopené, interpretované, komunikované a využité človekom alebo počítačom, tak sa stávajú zmysluplnými informáciami [3]. Jedno zo základných členení rozdeľuje dáta na *kvantitatívne* a *kvalitatívne* [7].

Kvalitatívne dáta

Kvalitatívne dáta [8] sú niekedy označované ako „mäkké“ dáta. Sú to nečíselné dáta, ktoré popisujú kvalitu alebo vlastnosti nejakých javov. Sú opakom kvantitatívnych dát. Je to napríklad farba, krása, spokojnosť zákazníka, atď. Kvalitatívne dáta sa tiež dajú rozdeliť do dvoch skupín [7]:

- **nominálne (klasifikačné) dáta**
  - hodnoty týchto dát sú rôzne a slúžia len k triedeniu,
  - nie sú porovnateľné a nedajú sa s nimi vykonávať matematické operácie,
  - príklady: farby, pohlavie, značky áut, krvné skupiny, atď.
- **ordinálne dáta**
  - sú to dáta, ktoré vyjadrujú poradie,
  - podobne ako nominálne predstavujú výber z nejakého počtu možností,
  - môžeme ich usporiadať, určiť poradie, určiť ktorá hodnota je lepšia/horšia alebo menšia/väčšia,
  - neposkytujú informácie o rozdiele medzi hodnotami (teda o koľko je jedna hodnota väčšia ako druhá),
  - využívajú sa pre triedenie dát a označovanie kritérií pre triedenie (napríklad triedenie pracovníkov podľa stupňa vzdelania),
  - nedajú sa s nimi vykonávať matematické operácie,
  - príklady: začiatočník/pokročilý/expert, základné/stredné/vysoké vzdelanie, atď.

Kvantitatívne dáta

Kvantitatívne dáta [9] sú tiež niekedy označované ako „tvrdé“ dáta. Sú to merateľné dáta, ktoré môžeme vyjadriť číslami. Je to napríklad vek, počet obyvateľov, teplota, rozmery, atď. Kvantitatívne dáta sa dajú rozdeliť ešte na dve skupiny [7]:

- **intervalové dáta** – majú vlastnosti ako ordinálne dáta, ale na rozdiel od ordinálnych sa dá určiť o koľko je jedna hodnota väčšia ako druhá,
- **pomerové dáta** – majú vlastnosti ako intervalové dáta, ale naviac vieme vypočítať koľkokrát je jedna hodnota väčšia ako druhá.

Dáta môžu mať tiež niekoľko dimenzií, podľa ktorých je treba vhodne vybrať vizualizačný nástroj pre ich zobrazenie. *Dimenzia* (v súvislosti s databázami) je množina hodnôt, ktorá poskytuje kontext pre hodnoty z množiny objektov analýzy [14] alebo zoznam hodnôt, ktoré poskytujú kategórie pre dáta [25]. Teda jednoducho povedané, je to množina hodnôt nejakého typu, ktorá popisuje kvantitatívne alebo kvalitatívne dáta. Dáta, ktoré majú viac ako jednu dimenziu, sa nazývajú *multidimenzionálne*. Príkladom multidimenzionálnych dát je napríklad počet predaných áut na určitých pobočkách, za nejaký čas a konkrétny typ auta 3.1.

Dátum	Miesto	Produkt	Počet
01.01.	Brno	Škoda	12
01.01.	Praha	Audi	56
02.01.	Olomouc	Audi	42
03.01.	Brno	Škoda	65

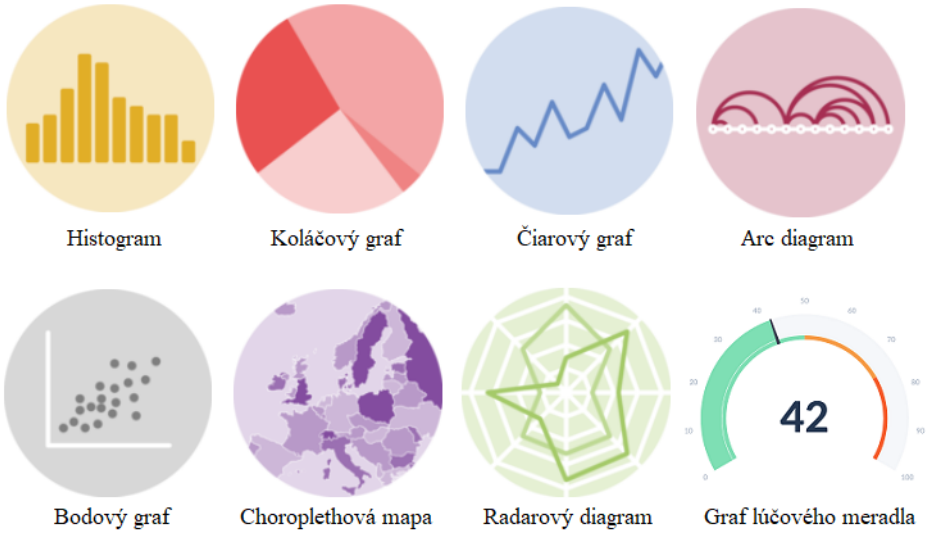
Tabuľka 3.1: Počet predaných áut

3.1 Vizualizačné nástroje

Vizualizačný nástroj je vizualizačný prvok, prostredníctvom ktorého sú reprezentované dáta buď grafickou alebo textovou formou či ich kombináciou. Jeho cieľom je prezentovať význam a účel dát. Každý vizualizačný nástroj je obmedzený počtom dimenzií, ktorý dokáže zobraziť. Harris v svojej práci [19] rozdeľuje vizualizačné nástroje do niekoľkých skupín podľa ich použitia a účelu: distribúcia hodnôt, korelácia, časť celku, porovnávanie a mapy, ktoré ďalej rozdeľuje na štatistické, popisné, flow a topografické. Existujú ešte ďalšie delenia, napríklad podľa typu dát a dimenzií [2]: numerické dáta (kvantitatívne), kategorické dáta (kvalitatívne), numerické a kategorické dáta, mapy, siete, časové rady. Alebo ich môžeme rozdeliť podľa účelu [1]:

- distribúcia hodnôt – histogram, graf hustoty, krabicový graf,
- korelácia – bodový, bublinový graf, korelogram,
- porovnanie (ranking) – stĺpcový, pavučinový graf,
- časť celku – výsekový graf, skladaný stĺpcový graf, stromová mapa, Venov diagram,
- vývoj v čase – spojnicový, plošný, špirálový graf,

- mapa – mapa, kartogram, mapa spojenia,
- tok (flow) – sieť, chord, Sankey diagram, Arc diagram,
- hodnota v rozsahu – gauge, bullet graf.

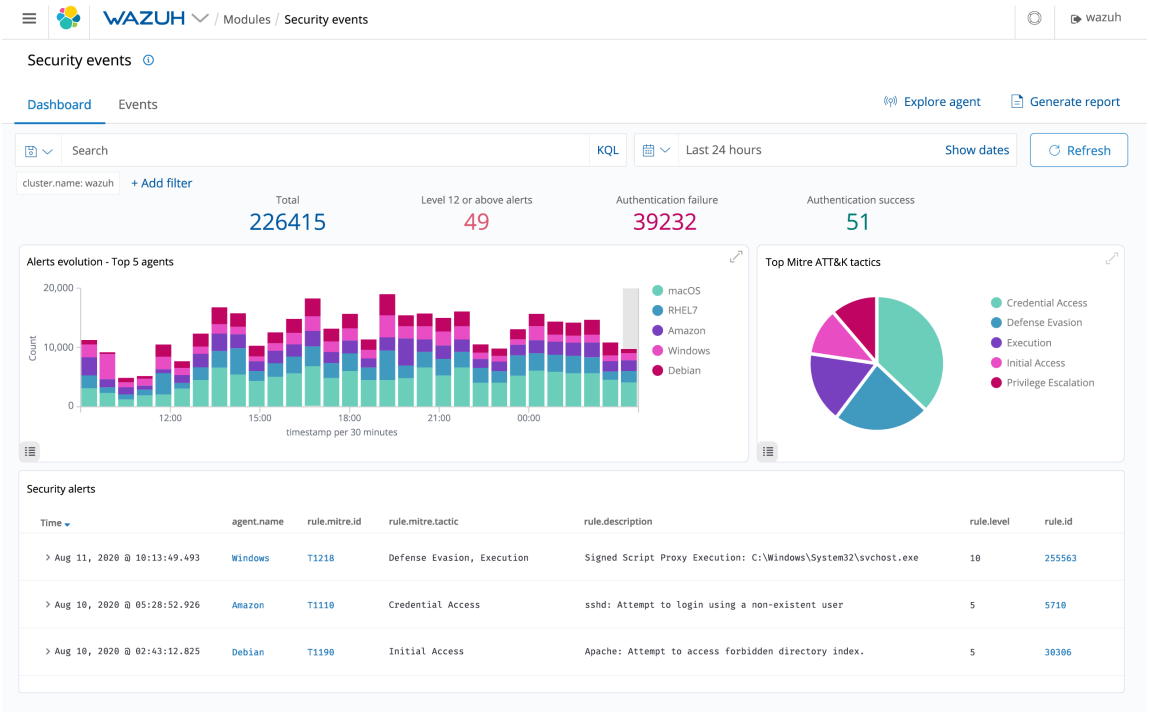


Obr. 3.1: Prehľad grafov podľa účelu [1].

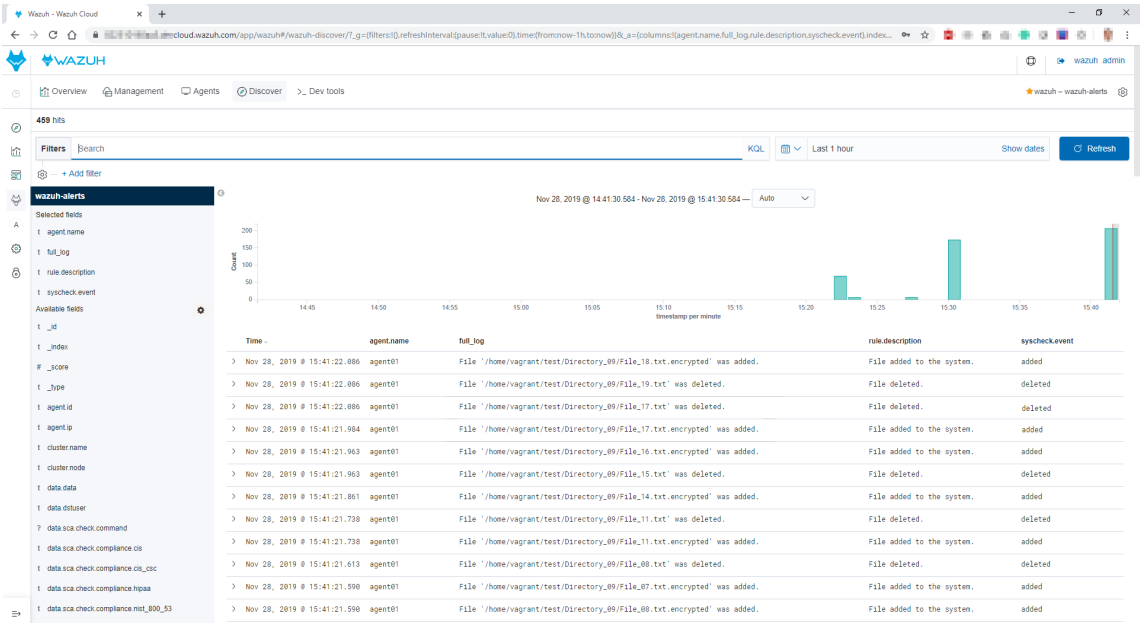
3.2 Vizualizačné nástroje pre zobrazovanie incidentov

Existuje niekoľko systémov pre detekciu anomálií a vniknutí do systému, ktoré pracujú s incidentami a prehľadne ich zobrazujú pomocou rôznych vizualizačných nástrojov ako napríklad voľne použiteľné [23] Wazuh (obrázok 3.2), MISP, The Hive či systémy určené pre podniky [20] IBM QRadar, ALienVault, Sumo Logic a ďalšie.

Tieto systémy využívajú rôzne vizualizačné nástroje podľa typu dát, ktoré chcú užívateľovi predstaviť. Dashboardy sú najpoužívanejšími prostriedkami pre zobrazovanie týchto incidentov na jednej obrazovke, kde užívateľ vidí všetky podstatné informácie hneď na prvý pohľad. Tieto dashboardy väčšinou obsahujú výsekový graf, ktorý zobrazuje napríklad pomer medzi typmi incidentov, ktoré v systéme nastali. Ďalej skladaný stĺpcový alebo iba jednoduchý stĺpcový graf, ktorý zobrazuje napríklad množstvo incidentov za určitý čas podľa ich typu. Tiež sa často stretávame s tabuľkou, ktorá obsahuje prehľadné zhrnutie všetkých typov incidentov na obrázku 3.2. U detailného zobrazenia udalosti sa väčšinou stretávame len s textovým popisom tejto udalosti, no niekedy je možné nájsť i graf, ako je napríklad na obrázku 3.3, ktorý zobrazuje množstvo zaznamenaných incidentov za určitý čas.



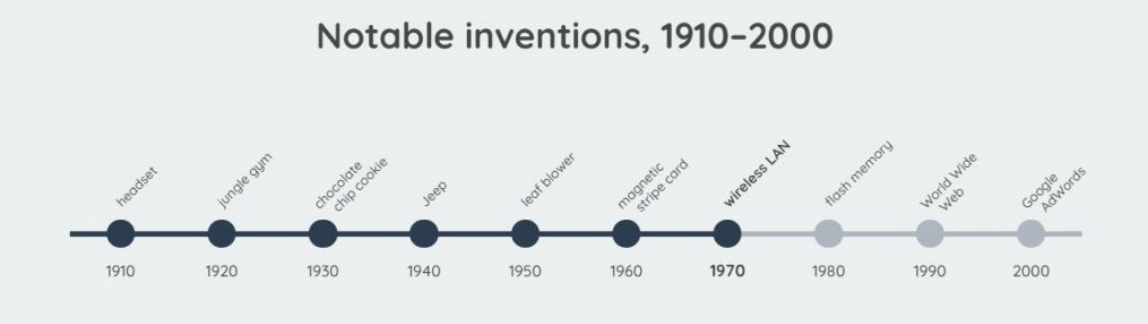
Obr. 3.2: Zobrazenie incidentov v systéme Wazuh prostredníctvom dashboardu: toto zobrazenie odpovedá väčšine nástrojov pre zobrazovanie incidentov, kde nájdeme celkové prehľadné zhrnutie zaznamenaných udalostí v systéme. [6]



Obr. 3.3: Zobrazenie detailu udalosti v systéme Wazuh: detail je zobrazený pomocou časového grafu a tabuľky, ktorá zobrazuje detaily o jednotlivých stavoch zobrazeného incidentu. [4]

Pretože som si ďalej v práci zvolila *graf časovej osi (timeline graf)* pre vizualizáciu komponentov 1 a 2, je vhodné ho v krátkosti popísať. Zväčša sa jedná o jednoosový graf, ktorý zobrazuje minulé a/alebo budúce udalosti, aktivity, požiadavky v takom poradí, v akom prebehli alebo v akom sú očakávané [19]. Hlavnou funkciou timeline grafov je zlúčenie a grafické zobrazenie časových informácií na účely analýzy a komunikácie. Os tohto grafu môže byť horizontálna (obrázok 3.4) alebo vertikálna, pričom po horizontálnej ose čas postupuje zľava doprava. U vertikálnej osi môže čas postupovať zdola hore alebo naopak. Tento čas môže byť logaritmický alebo lineárny.

Časové grafy dodržiavajú striktné usporiadanie dát, a preto je dôležité vybrať vhodný začiatok časovej osi ako i celkovú dĺžku periódy tejto osi [29]. Príliš malý alebo naopak príliš veľký graf môže odradiť čitateľa od nášho cieľa, prezentovať informácie, preto je dôležité vybrať správny pomer strán a veľkosť grafu.



Obr. 3.4: Príklad horizontálnej časovej osi. [10]

3.3 Charakteristiky dobrej vizualizácie

Prieskumom charakteristík dobrej vizualizácie dát sa tiež zaoberá i práca [18], ktorá skúma niekoľko rád odborníkov z rôznych profesných oblastí. Vizualizácia dát je široko používaná v rôznych oblastiach, vo vedeckých článkoch, v marketingu a obchode, novinách alebo médiách obecne, ai. Avšak mnoho týchto vizualizácií neprezentuje dáta tak, ako to ich tvorcovia zamýšľali. Preto sa rôzny odborníci [21], [28], [29], snažia skúmať vizualizáciu a nájsť návod, ako sa vyhnúť nesprávnym technikám pre ich tvorbu. Niekoľko týchto odborníkov sa však zhodlo na tom, že vizualizácia dát by mala pomôcť zlepšiť pochopenie väčšieho množstva numerických alebo textových dát. To však môže byť ťažké, pokiaľ dáta nie sú dobre vizualizované a čitateľ z nich nevie odvodiť to, čo autor zamýšľal.

Antony Unwin v článku [29] tvrdí, že existujú dva hlavné dôvody, prečo sa dáta zobrazujú v grafickej podobe a to je ich prezentácia alebo ich ďalšie skúmanie a analýza. Tvrdí, že grafika má v prvom rade prezentovať dáta čitateľom a pomocou rôznych štýlov, grafických prvkov a ich rozmiestnením túto prezentáciu len zdokonaliť.

Pokiaľ už vieme aké dáta chceme prezentovať divákom, mali by sme vybrať správny vizualizačný nástroj (histogram, graf, mapa, časová os, ...). Tento krok je najdôležitejšou súčasťou tvorby vizualizácie, pretože sa musíme rozhodnúť, ktorý nástroj nám poskytne najlepšie zobrazenie našich dát. Avšak každý vizualizačný prvok je nejako obmedzený, napríklad počtom dimenzií, ktoré dokáže zobraziť, veľkosťou priestoru, kde bude zobrazený (aby bola grafika dobre čitateľná), atď. Preto, keď už vieme, ktorý nástroj použijeme, musíme správne určiť jeho veľkosť, pomer strán, škály osí, poprípade utriediť alebo zoradiť

zobrazované dáta (napríklad u stĺpcových grafov), vyhnúť sa orámovaniu grafiky pokiaľ ju nechceme odlíšiť od inej grafiky či textu, vhodne zvoliť šrafovanie či kombináciu farieb apod.

Ďalej napríklad Tufte zaviedol vo svojej práci [28] niekoľko obecných doporučení, ktoré by sa mali dodržiavať pre vytvorenie skvelej prezentácie dát:

- Predovšetkým zobrazovať dáta.
- Maximalizovať data-ink ratio.
- Redukovať farby, ktoré nesúvisia s dátami.
- Redukovať farby, ktoré predstavujú redundantné dáta.

Podľa neho by vizualizácia mala pritiahnuť čitateľovu pozornosť k podstate a zmyslu veci a nemala by ho nijako rozptyľovať. Teda príliš veľa grafických prvkov vo vizualizácii, ktoré neslúžia pre zobrazovanie dát môže odpútať pozornosť čitateľa, a preto by sa mali obmedziť na čo najmenšiu mieru. Z tohto dôvodu vytvoril tzv. *data-ink ratio*, ktorý porovnáva plochu, ktorá je určená pre prezentovanie dát a plochu využitú pre všetky prvky vizualizácie. Z toho vyplýva, že čím viac plochy je zameranej na zobrazenie dát, tým lepšie. Podľa Tufteho [28] každý kúsok farby v grafike vyžaduje určitý dôvod a skoro vždy je to reprezentácia nejakej informácie.

Andy Kirk v [21] zasa udáva 4 kľúčové štádiá pri návrhu vizualizácie:

1. Naplánovanie, definovanie a zahájenie projektu.
2. Práca s dátami – zber, spracovanie a príprava dát.
3. Definícia toho, čo chceme prezentovať publiku.
4. Zváženie všetkých možností návrhu vizualizácie a zahájenie výrobného cyklu (iterácia medzi myšlienkou, prototypom a samotnou vizualizáciou).

3.4 Dashboard

Keďže mnoho systémov pre detekciu anomálií a vniknutí do systému používa veľmi často pre zobrazovanie incidentov dashboard, ďalej si povieme, čo to je dashboard, aké druhy dashboardov existujú a akých postupov je dobré sa držať pri ich vytváraní.

Vhodnú definíciu dashboardu poskytuje napríklad Few [17]:

> A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.

Podľa tejto definície by mal dobrý dashboard spĺňať niekoľko pravidiel:

- Dashboard kladie dôraz najmä na vizuálne zobrazenie dát, preto jeho najväčšiu časť musia tvoriť grafické prvky.
- Toto zobrazenie dát sa musí zmestiť na jednu obrazovku tak, aby nebolo nutné v dashboarde rolovať či sa preklikávať na ďalšie podstránky s cieľom nájsť dôležité informácie.
- Pri neočakávanej zmene či výkyve by mal užívateľa nejakým spôsobom upozorniť na túto skutočnosť.

3.4.1 Klasifikácia dashboardov

Dashboardy sa líšia v dátach, ktoré zobrazujú i v spôsobe ich používania. Preto sa dajú zaradiť do niekoľkých kategórií podľa rôznych kritérií. Existuje zoznam premenných, ktorý prezentoval Few v [17], podľa ktorých je možné dashboardy klasifikovať ako je uvedené v tabuľke 3.2. Aj keď existuje množstvo kategórií, do ktorých môžeme dashboardy radiť, najpoužívanejšie triedenie je podľa ich rolí na *strategické, analytické* a *operačné*.

Premenné	**Hodnoty**
Role	strategické, analytické, operačné
Typy dát	kvantitatívne, kvalitatívne
Dátové domény	predaj, financie, marketing, výroba, ľudské zdroje
Typy meraní	Balanced Scorecard, Six sigma, nevykonané
Rozpätie dát	celý podnik, oddelenie, individuálne
Aktualizácia dát	mesačná, týždenná, denná, hodinová, real-time
Interaktivita	statické alebo interaktívne zobrazenie (filtre, drill-down)
Spôsoby zobrazenia	prevažne grafické, prevažne textové, oboje
Portálové funkcie	funkcie získania dodatočných dát, bez funkcie

Tabuľka 3.2: Možnosti kategorizácie dashboardov [17].

Dashboardy pre strategické účely

Strategické dashboardy sa často využívajú pri rozhodovaní vo firme, a preto obsahujú prehľady o súčasnej situácii, ale môžu obsahovať aj predpovede do budúcnosti, porovnávanie dát či krátku históriu. Vzhľad by mal byť jednoduchý a zameraný na prehľadné zobrazenie dát. Nepotrebujú real-time dáta ani interaktívne zobrazenie. Pracujú s dátami, ktoré sú zbierané mesačne, týždenne či denne.

Dashboardy pre analytické účely

Musia byť komplexnejšie ako strategické. Tiež nepotrebujú sledovať real-time dáta, ale je potrebná interakcia. Mali by obsahovať sofistikovanejšie nástroje zobrazenia ako napríklad filtre, grafy časovej osi, možnosti iného zobrazenia dát či možnosti zobrazenia detailu k danej problematike, pretože analytik potrebuje pre svoju prácu poznať širší kontext než mu dashboard dokáže na prvý pohľad poskytnúť. Niektoré poskytujú aj možnosť predpovedí, vytvárania vlastných pohľadov a výpočtov nad dátami.

Dashboardy pre operačné účely

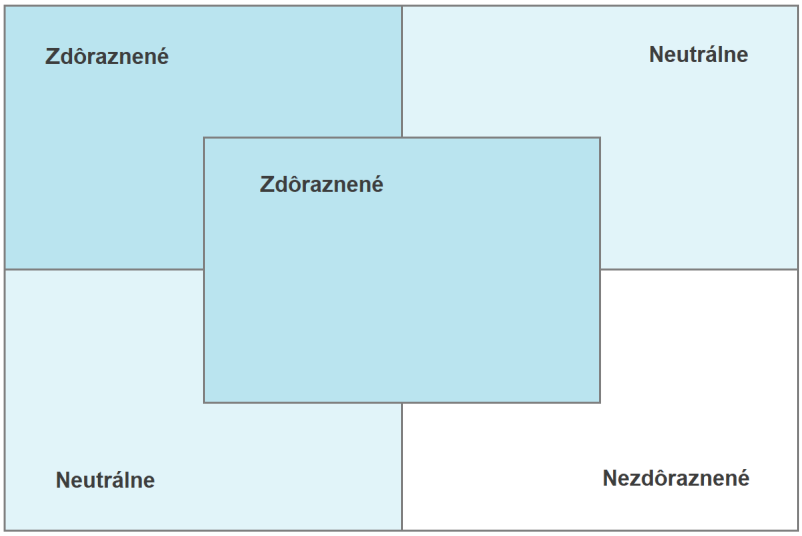
Tieto dashboardy slúžia na monitorovanie aktuálnych dát. Musí brať do úvahy dynamičnosť a vysokú frekvenciu aktualizácie dát. Vyznačuje sa najmä jednoduchým zobrazením dát a interaktivitou. Mal by obsahovať nástroje, ktoré upozornia užívateľa, ak hodnota padne do intervalu kritických hodnôt, teda prekročí určitý prah a je nutné upozornenie. Tieto upozornenia často obsahujú aj detailný popis problému a užívateľ by na ne mal čo najrýchlejšie reagovať.

Príklad použitia týchto dashboardov v rámci firmy pre detekciu anomálií: Admin pracuje s operačným dashboardom, ktorý ho upozorní pri výskyte novej udalosti, ktorá sa má začať riešiť. Admin následne zvolí analytika, ktorý bude daný incident riešiť. Analytik pre svoju prácu využíva analytický dashboard, v ktorom skúma udalosť a zisťuje príčiny jej vzniku. V strategickom dashboarde potom užívatelia môžu vidieť sumarizované dáta o týchto incidentoch.

3.4.2 Charakteristiky dobre navrhnutého dashboardu

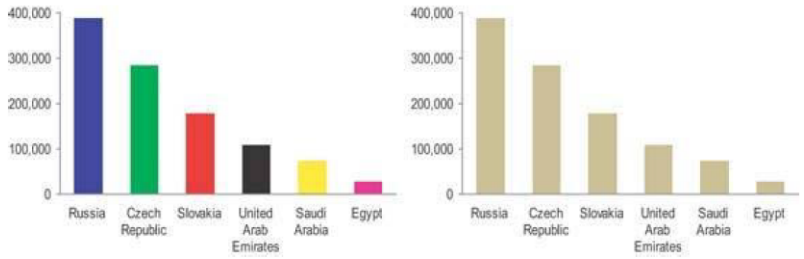
Pre tvorbu dashboardov bolo navrhnutých niekoľko pravidiel a rád, ktoré by tvorcovia dashboardov mali dodržiavať. Avšak v praxi sa stretávame aj s takými dashboardami, ktoré nespĺňajú vyššie uvedenú definíciu dashboardu či nasledujúce pravidlá. Príklad takéhoto dashboardu je napríklad na obrázku 3.11. Je teda dobré držať sa určitých pravidiel a rád, ktoré popisuje napríklad Few [17] či Eckerson [16]. Tieto rady nám môžu pomôcť vylepšiť náš dashboard a je možné ich zhrnúť do nasledujúceho postupu:

1. *Stanovenie účelu dashboardu a identifikácia užívateľov, ktorý s ním budú pracovať.*
2. *Výber dát a stanovenie metrík* – výber dát je dôležitým kritériom pri tvorbe dashboardu, pretože dashboard má zobrazovať len potrebné dáta. S tým súvisí i výber metrík, ktorý je podmienený identifikáciou užívateľov a ich požiadavkov.
3. *Voľba zobrazovacích médií* – po stanovení metrík je vhodné ich ilustrovať pomocou vhodne zvolených grafických vizualizačných prvkov (stĺpcové, čiarové, timeline grafy, atď.).
4. *Rozmiestnenie na obrazovke* – grafy, ktoré sme vytvorili tvoria výsledné komponenty dashboardu, ktoré je treba vhodne umiestniť na obrazovku podľa ich účelu, pretože výsledná pozícia grafov v dashboarde má vplyv na zdôraznenie informácií, ktoré prezentujú 3.5.
5. *Opakované zjednodušovanie a vylepšovanie.*

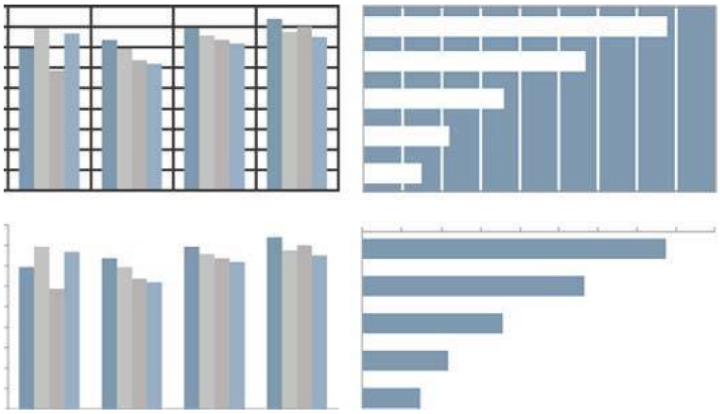


Obr. 3.5: Obrázok popisuje, ktoré oblasti dashboardu vníma pozorovateľ najviac a kde je lepšie umiestniť informácie, na ktoré sa kladie najväčší dôraz. Založené na obrázku z [17].

Few [17] taktiež pri návrhu dashboardu radí sledovať *data-ink ratio*, ktoré definoval Tufte v [28]. Teda snažiť sa odstrániť čo najviac plochy, ktorá nesúvisí so zobrazovanými dátami (*non-data*), bez straty niečoho užitočného a zdôrazniť tú plochu, ktorá reprezentuje dáta. Na nasledujúcich obrázkoch 3.6 až 3.9 sú demonštrované niektoré nevhodné i dobré praktiky pri vizualizácii dashboardov.



Obr. 3.6: Využitie farieb pre každý stĺpik zvlášť nemá v tomto prípade žiaden zmysel [17].

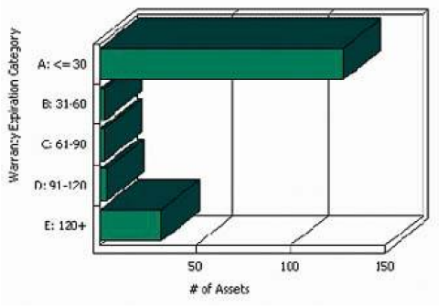


Obr. 3.7: Mriežka v grafoch je len veľmi málo užitočná a často vedie len k odvedeniu pozornosti čitateľa [17].

Salesperson	Jan	Feb	Mar
Robert Jones	2,834	4,838	6,131
Mandy Rodriguez	5,890	6,482	8,002
Terri Moore	7,398	9,374	11,748
John Donnelly	9,375	12,387	13,024
Jennifer Taylor	10,393	12,383	14,197
Total	$35,890	$45,464	$53,102

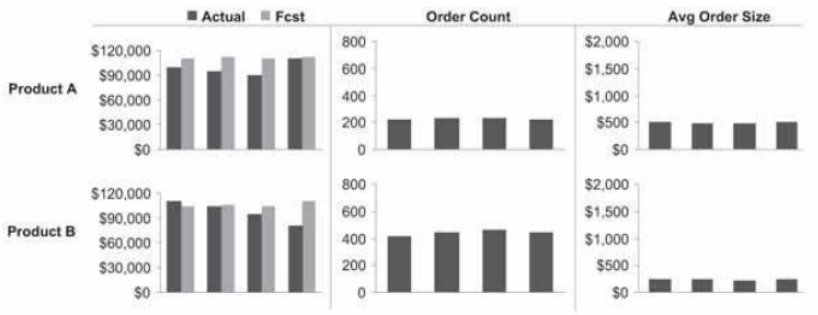
Salesperson	Jan	Feb	Mar
Robert Jones	2,834	4,838	6,131
Mandy Rodriguez	5,890	6,482	8,002
Terri Moore	7,398	9,374	11,748
John Donnelly	9,375	12,387	13,024
Jennifer Taylor	10,393	12,383	14,197
Total	$35,890	$45,464	$53,102

Obr. 3.8: Takisto i mriežka tabuľky je príliš rušivá pri čítaní dát [17].

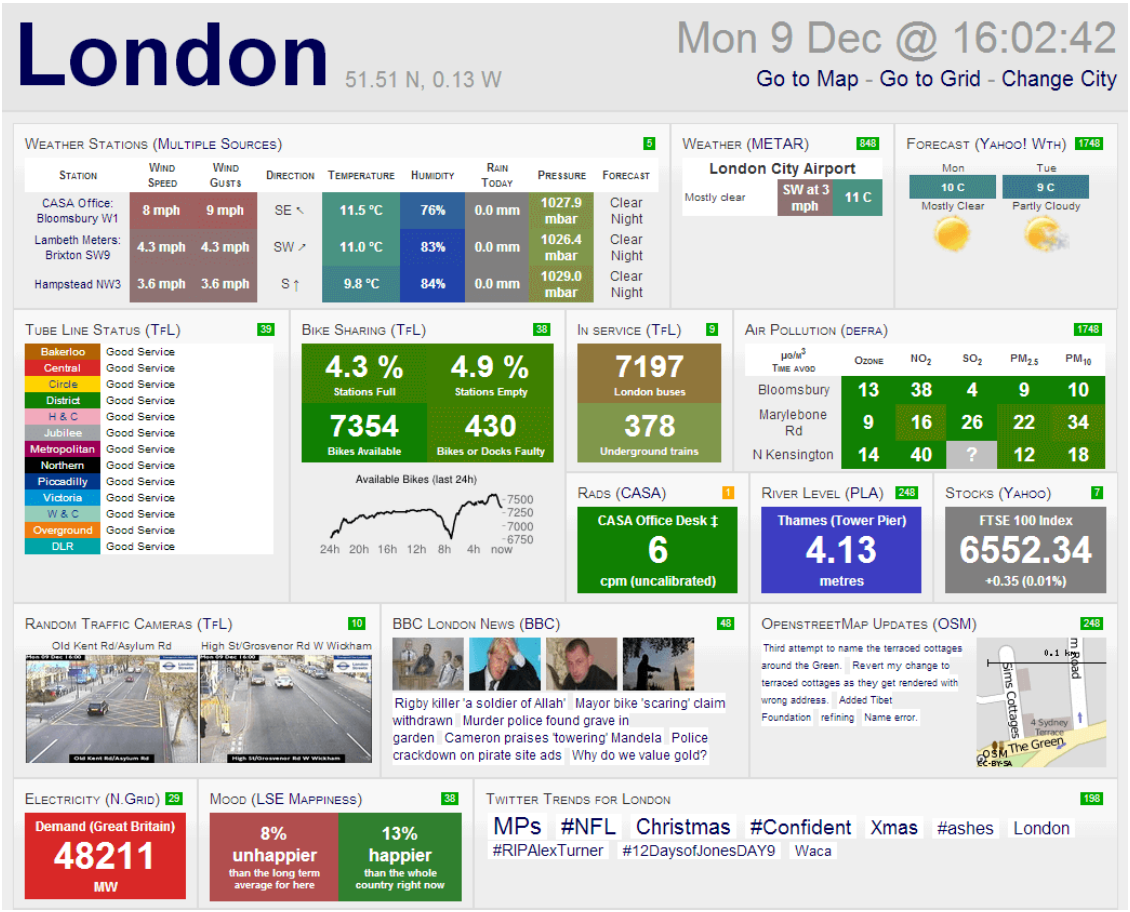


Obr. 3.9: Tretia dimenzia v tomto grafe nereprezentuje žiadne dáta, preto by sa tu nemala vôbec využívať [17].

Few [17] tiež tvrdí, že niektoré nedátové plochy je však dôležité zachovať, pretože v niektorých situáciách nám môžu pomôcť lepšie sprehľadniť výsledný dashboard, napríklad pri oddelení grafických prvkov (obrázok 3.10). Pri zachovaní non-data je však potrebné ich upraviť tak, aby neboli príliš rušivé pre čitateľa, tzn. slabšie odtiene farieb, tenké čiary atď. Taktiež hovorí o tom, že v dashboarde by malo byť čo najmenej nepodstatného textu ako napríklad inštrukcie pre používanie prvkov dashboardu apod.



Obr. 3.10: Oddelenie grafických prvkov pomocou jemného orámovania [17].



Obr. 3.11: Príklad dashboardu, ktorý nevyužíva všetky doporučenia uvedené vyššie. Snaží sa zachytiť príliš veľa informácií na malom priestore a taktiež využíva príliš veľa farebných odtieňov, ktoré môžu rušiť pohľad na dáta [11].

3.5 Webová vizualizácia dát

Pre vizualizáciu dát na webovej stránke je možné použiť HTML *Canvas* alebo *SVG* elementy. *Canvas*[1] sa využíva pre vykresľovanie rastrovej grafiky na webovú stránku. Spôsob, čo sa má vykresliť je implementovaný pomocou jazyka Javascript. Akonáhle je však grafika vykreslená a my potrebujeme niečo prekresliť, musí sa prekresliť celý vykresľovaný objekt. Využíva sa pre vytváranie animácií, v hernej grafike, spracovávaní fotiek apod.

SVG[2] slúži pre vykresľovanie vektorovej grafiky. Je založené na XML jazyku. Grafika je zložená z niekoľkých elementov, ktoré sú po vytvorení dostupné z SVG DOM, cez ktorý je možné meniť ich atribúty. Narozdiel od Canvas je možné ku každému SVG elementu priradiť EventHandler, ktorý sa stará o funkčnosť pri udalostiach ako je napr. kliknutie na element apod. Pokiaľ chceme zmeniť časť grafiky, nie je potreba prekresľovať celý obrázok, ale prekreslí sa len zmenená časť. Využíva sa napríklad pre tvorbu máp, diagramov, grafov, či animácií.

D3.js[3] je knižnica jazyka JavaScript, ktorá umožňuje jednoduchú prácu s dokumentami (napr. SVG elementy) a dátami, ktoré chceme v dokumentoch prezentovať. Pomocou predpripravených metód je možné vytvárať SVG elementy, upravovať ich atribúty, štýlovať ich pomocou CSS, vytvárať animácie, prechody, atď. Po vytvorení napríklad rôznych diagramov pomocou D3.js je častokrát potreba znovu ich v budúcnosti využiť. Preto je vhodné vytvoriť z nich znovu-použiteľné parametrizovateľné komponenty pomocou existujúcich frameworkov pre vytváranie komponentov ako je napríklad *React*[4], *Vue.js*[5], *AngularJS*[6] apod.

React je knižnica jazyka JavaScript, ktorá umožňuje vytvárať komponenty pre ich ďalšie použitie. Každá vytváraná komponenta má svoj životný cyklus riadený niekoľkými metódami ako je napríklad `render`, ktorá vykresľuje komponent do vybranej oblasti. Táto metóda je povinná a volá sa vždy, keď sa zmení stav komponentu. Príkladom diagramov vytvorených pomocou knižnice React a D3.js je knižnica Nivo[7], v ktorej môžeme nájsť niekoľko komponentov, ktoré je možné znovu používať. Každý komponent obsahuje dokumentáciu, ktorá popisuje správne nastavenie atribútov a dát, s ktorými daný komponent pracuje. Vybrané diagramy je možné si priamo vyskúšať v nástroji Storybook, v ktorom je možné dynamicky meniť atribúty a výsledok je vidieť okamžite.

[1] https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
[2] https://developer.mozilla.org/en-US/docs/Web/SVG/Element/svg
[3] https://d3js.org/
[4] https://reactjs.org/docs/getting-started.html
[5] https://vuejs.org/v2/guide/
[6] https://angularjs.org/
[7] https://nivo.rocks/

Kapitola 4

Analýza požiadavkov

V tejto kapitole sa budeme zaoberať analýzou požiadavkov užívateľov firmy Flowmon, ktorí budú s našimi komponentami pracovať. Uvedieme si spoločné požiadavky ako i požiadavky na jednotlivé komponenty. A taktiež si predstavíme aj krátky popis aktuálneho stavu riešenia incidentov vo firme a popis niektorých vlastností incidentu, ktoré sú uvedené v tabuľke v sekcii 4.1.

4.1 Súčasný stav

V rámci firmy Flowmon existuje projekt pre detekciu a riešenie incidentov a následnú obnovu z bezpečnostných a operačných problémov, s cieľom zabrániť poškodeniu organizácie a jej biznisu. Tento projekt sa sústreďuje najmä na automatizáciu detekcie incidentu i odpovede naň.

Incidenty sú generované automaticky pomocou skriptov či detekčných metód alebo ich môže nahlásiť manuálne admin či iný užívateľ. Po vytvorení sa incidenty logujú a kategorizujú na základe pôvodu problému. Tiež sa im určuje priorita na základe ich dopadu a nutnosti vyriešenia. Dopad incidentu môže byť rôzny, môže sa jednať o dopad na užívateľa, jeho organizáciu a biznis či na systém celkovo, napríklad bezpečnosť či funkčnosť systému. Cieľom určovania priorít incidentov je pomôcť užívateľom Flowmonu vyriešiť najskôr tie incidenty, ktoré majú najväčší dopad na užívateľa alebo systém či je požadované ich rýchle vyriešenie.

Na základe kategórie incidentu sa potom vyberajú tzv. *incident handlers*, ľudia v rámci firmy, ktorí riešia incident, ktorým je po jeho vytvorení odoslané upozornenie, že majú začať riešiť nový incident. Spolu s detailným popisom sú riešiteľovi incidentu poskytované aj návrhy na jeho vyriešenie. Každý riešiteľ môže upravovať incident na základe jeho práv a rolí, napríklad môže pridať výsledky dodatočného vyšetrovania, pridať komentár, zmeniť prioritu, priradiť incident inému riešiteľovi, vyriešiť ho, uzavrieť, apod.

Títo riešitelia incidentov budú teda pre svoju prácu potrebovať určité vizualizačné nástroje, ktoré im umožnia celkový pohľad na incidenty, ktoré je potreba riešiť alebo už sú v štádiu riešenia. Tieto vizualizačné nástroje by preto mali byť prehľadné a jednoduché na orientovanie sa v nich i prácu s nimi. Bude preto potreba vytvoriť niekoľko komponentov, ktoré budú tieto incidenty zobrazovať z niekoľkých pohľadov, najmä celkový pohľad nad všetkými incidentmi a detailný pohľad na zvolený incident.

Položka	Možné hodnoty
Rada incidentov	Aplikačná/Bezpečnostná/Sieťová/Iná...
ID	ID incidentu v rade incidentov
Kategória	Z užívateľom definovaných kategórií.
Status	Nový ...->... Uzavretý
Vytvoril	Udalosť/Užívateľ/Pravidlo
Dopad	Vysoký/Stredný/Nízky
Urgencia	Vysoká/Stredná/Nízka
Priorita	Vypočítaná (vysoký dopad + naliehavosť = kritická, ...)
Priradené užívateľovi	Užívateľ
Priradené skupine	Skupina
Názov incidentu	Zadané manuálne alebo automaticky
Popis incidentu	Zadané manuálne alebo automaticky
Vytvorený	Čas, kedy bol incident vytvorený.
Otvorený	Čas, kedy bola zahájená práca na incidente.
Vyriešený	Čas, kedy bol incident naposledy vyriešený.
Uzavretý	Čas, kedy bol incident naposledy uzavretý.
Pozastavený	Vypočítaný čas, po ktorý bol incident pozastavený.
Trvanie	Vypočítaný čas, po ktorý bol incident otvorený.
Súvisiace udalosti	Zoznam udalostí.
Súvisiace incidenty	Zoznam incidentov.
Zdrojová IP adresa	1 alebo viac (ak boli detekované)
Cieľová IP adresa	1 alebo viac (ak boli detekované)
Zdrojová sieť	1 alebo viac (ak boli detekované)
Cieľová sieť	1 alebo viac (ak boli detekované)
Zdrojový užívateľ	1 alebo viac (ak boli detekovaní)
Prvý raz videné	Čas, kedy začal prvý stav udalosti.
Posledný raz videné	Čas, kedy skončil posledný stav.
Komentár	Text
Kategória komentáru	Z užívateľom definovaných kategórií.
Čas komentáru	Čas, kedy bol komentár uložený.

Tabuľka 4.1: Formát incidentu, ktorý poskytuje informácie o incidente užívateľom, ktorý ho budú riešiť. V tabuľke je uvedených len pár kľúčových položiek. Každý incident môže mať ešte niekoľko ďalších položiek ako napr. nejaký graf, obrázok či rôzne prílohy, top kategórie súvisiace s incidentom apod.

4.2 Užívateľské požiadavky

Zo súčasného stavu riešenia incidentov vyplýva, že je potreba vytvoriť grafickú vizualizáciu incidentov, ktorá užívateľom umožní ľahšiu a prehľadnú prácu s riešenými incidentmi a môže im tak pomôcť k efektívnejšiemu vyriešeniu týchto incidentov. Mojím konkrétnym cieľom je vytvoriť 3 parametrizovateľné komponenty, ktoré budú slúžiť pre zobrazovanie incidentov a ich stavov. Každý komponent má svoje špecifické požiadavky podľa jeho použitia vo výslednom projekte.

4.2.1 Požiadavky na komponent 1

Komponent 1 má zobrazovať len jeden incident ako postupnosť jeho stavov v čase. Mal by byť prehľadný a responzívny, teda prispôsobovať sa veľkosti plochy, na ktorú je vykresľovaný. Práca s ním by mala byť intuitívna a jednoduchá. Užívateľ by mal mať možnosť zvoliť si farbu stavov incidentov i ich tvar. Mal by mať tiež možnosť pridať k incidentom užívateľom zvolené ikonky, ktoré by indikovali napríklad výstrahu pre okamžité riešenie incidentu. Ďalej by mal podporovať funkciu zoomovania, teda aby sa dalo približovať a oddaľovať na stavy incidentu, ktoré užívateľa zaujímajú bližšie. Pri prechádzaní myšou nad stavom incidentu by sa malo ukazovať popup okno s krátkym popisom incidentu, ktorý si môže užívateľ nadefinovať sám. Pri kliknutí na určitý stav by sa mal v nasledujúcom komponente 3 ukázať detail daného stavu.

4.2.2 Požiadavky na komponent 2

Komponent 2 by mal zobrazovať viacero incidentov ako postupnosť ich stavov v čase nad sebou. Je založený na komponente 1, a preto sa jeho požiadavky zhodujú s tými, ktoré sú uvedené vyššie. Jediný rozdiel v požiadavkoch je pri kliknutí na stav incidentu, kedy sa má ukázať detail daného incidentu.

Zhrnutie požiadavkov:

- prehľadnosť a responzívnosť,
- intuitívnosť a jednoduché ovládanie.

Užívateľom definované vlastnosti:

- farba a tvar stavov,
- ikony,
- vzhľad popup okna.

Funkcionalita:

- zoom,
- filtrovanie,
- pri hover – popup okno s popisom,
- pri kliknutí – zobrazenie detailu.

4.2.3 Požiadavky na komponent 3

Komponent 3 má zobrazovať detailný popis jednotlivých stavov jedného incidentu. Mal by zobrazovať stavy incidentu od najaktuálnejšieho po najstarší. Pokiaľ by výška komponentu presiahla určitú definovanú výšku, tak by sa mal dať rolovať. Ak by sa incident skladal z príliš veľa stavov, mal by komponent umožniť stránkovanie pomocou tlačítiek *Show more* a *Show previous*. Ak užívateľ definoval vlastnú farbu stavov u komponentu 1, tak by sa táto farba mala zobraziť i u každého daného stavu v komponente 3. Tiež by mal byť responzívny a práca s ním jednoduchá a intuitívna ako s predchádzajúcimi komponentami. Vlastnosti pre kliknutie či hover nad komponentom neboli definované.

Zhrnutie požiadavkov:

- intuitívnosť a jednoduché ovládanie,
- responzívnosť.

Užívateľom definované vlastnosti:

- farba komponentu,
- výška komponentu,
- počet stavov na stránku.

Funkcionalita:

- zobrazenie stavov od najaktuálnejších po najstaršie,
- zobrazenie farby stavu podľa komponentu 1,
- pri prekročení výšky komponentu – rolovať cez stavy,
- stránkovanie – preklikávanie medzi stránkami s určitým počtom stavov na jednu stranu.

Kapitola 5

Návrh komponentov

Táto kapitola sa venuje výsledným návrhom vytváraných komponentov na základe užívateľských požiadavkov uvedených v predchádzajúcej kapitole 4.2. Popíšeme si v nej, s akými dátami bude každý z komponentov pracovať, ukážeme si tiež ich drôtené modely (anglicky *wireframe*) pre lepšiu predstavu ako i modely výsledného dashboardu.

5.1 Návrh komponentu 1

Komponent 1 bude pre vykresľovanie pracovať s tromi parametrami:

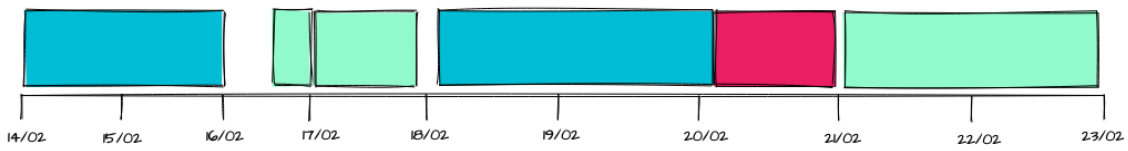
- Údaje o samotnom incidente (*incidents*) – obsahuje informácie o zobrazovaných incidentoch, ako je napr. názov incidentu, jeho stavy a ich popisy.
- Údaje o vzhľade komponentu 1 (*property*) – definuje ich užívateľ, sú to napríklad farba či výška stavov, formát popup okna, definovanie ikon apod. Tieto údaje nie sú povinné a užívateľ ich nemusí zadávať.
- Údaje o vzhľade komponentu 3 (*description*) – definuje ich tiež užívateľ, môže to byť napr. výška celej komponenty, počet stavov na stránku či farba komponentu. Takisto nie sú tieto údaje povinné a netreba ich uvádzať.

Tento komponent bude zobrazovať incident a jeho stavy na časovej ose s popismi dátumov. Jednotlivé stavy budú predstavené formou obdĺžnikov, kde ich šírka predstavuje ich celkové trvanie. Výšku týchto obdĺžnikov si môže užívateľ zvoliť ľubovoľnú a nemá vplyv na funkčnosť tohto komponentu. Každý incident bude mať svoje označenie podľa jeho identifikátora (*id*) na ľavej strane komponentu. Môže si tiež zvoliť farbu buď všetkých incidentov i farbu incidentov podľa určitej domény (obrázok 5.1) ako je napríklad status či priorita ai. Užívateľ si môže vyberať z nasledujúcich hodnôt statusu: „new“, „active“, „onhold“, „closed“, „reopened“, či hodnôt priority: 1, 2, 3, 4, poprípade vyšších. Užívateľ si tiež môže vybrať formát ikony, ktorú chce používať. Môže si zvoliť vlastnú ikonu vo formáte FontAwesome[1], url alebo base64 či si môže vybrať preddefinované ikony.

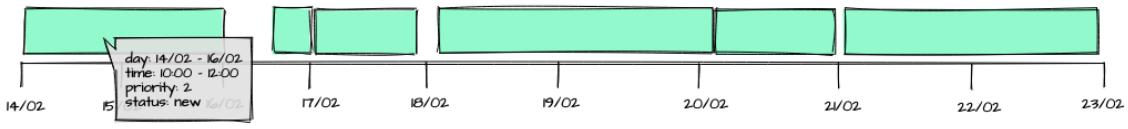
Ak užívateľ prejde myšou nad stavom incidentu, tento stav by sa mal nejako zvýrazniť a tiež by sa malo zobraziť popup okno s krátkymi informáciami o tomto stave (obrázok 5.2). Formát týchto informácií je preddefinovaný na dátum a čas začiatku a konca tohto stavu. Ďalšie informácie o tomto stave si môže užívateľ vybrať zvolením ľubovoľných informácií, ktoré chce zobraziť. Po kliknutí na stav incidentu by sa mali v komponente 3 zobraziť

[1] https://fontawesome.com/icons?d=gallery&p=2

detailné informácie o stave incidentu, na ktorý bolo kliknuté a tiež i informácie o ostatných stavoch incidentu. Komponent by mal umožňovať i zoomovanie (približovanie a odďaľovanie) na stavy incidentov. Tento zoom by mal byť realizovaný pomocou niekoľkých tlačítiek, ktoré užívateľovi umožnia približovanie, odďaľovanie ako i posúvanie sa po stavoch incidentu či zrušenie zoomu.



Obr. 5.1: Náčrt komponentu 1 s vyfarbenými stavmi (napr. podľa priority).



Obr. 5.2: Náčrt komponentu 1 s ukážkou popup okna.

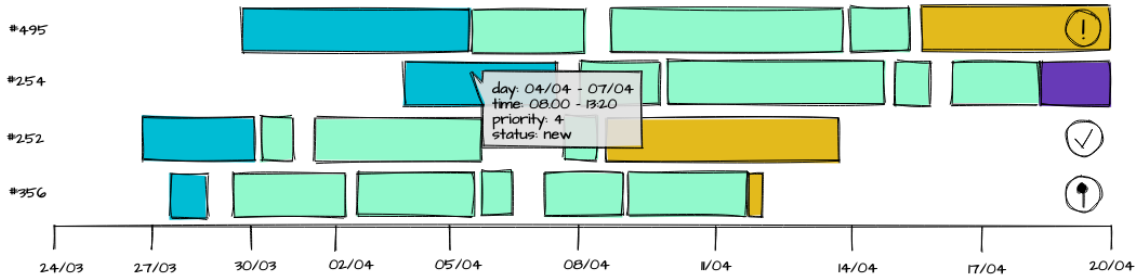
Príklad použitia definície všetkých možných vlastností pre vzhľad komponentu, ktoré sú detailne popísané nižšie, je uvedený vo výpise 5.1.

- **svg__height** – určenie výšky celého komponentu, (predvolená hodnota: 500),
- **state__height** – určenie výšky stavov v komponente, (predvolená hodnota: 20),
- **incidentColor** – definovanie vlastnej farby všetkých incidentov, (predvolená hodnota: `"#68a9c0"`),
- **popup** – definovanie vlastností pre popup okno grafu, ktoré sa v ňom majú zobraziť. Ak nie sú definované, zobrazuje sa len dátum a čas začiatku a konca daného stavu incidentu,
- **domain** – určenie domény, podľa ktorej sa majú vyfarbiť stavy,
- **colors** – pole, ktoré obsahuje atribúty pre určenie farieb stavov:
  - **domain** – hodnota vyššie zvolenej domény, na ktorú sa má aplikovať ďalej definovaná farba,
  - **color** – farba stavov,
- **icons** – slúži pre definovanie ikon incidentov:
  - **id** – je id incidentu, ktorému chceme priradiť ikonu,
  - **iconType** – typ ikony, ktorý si užívateľ môže zvoliť z nasledovných hodnôt: „own“, „hurry“, „important“, „check“, „label“, „mark“, „pin“. Pokiaľ si užívateľ zvolí „own“, musí ďalej definovať **iconFormat** a **icon**. Zvyšné typy sú predom definované FontAwesome ikony, ktoré môže užívateľ tiež použiť,
  - **iconFormat** – formát ikony („url“, „base64“, „FontAwesome“),
  - **icon** – zdroj ikony, ktorý môže byť formou url, base64 alebo FontAwesome ikon.

5.2 Návrh komponentu 2

Komponent 2 využíva pre vykresľovanie tie isté parametre ako komponent 1. Dáta o incidentoch sú však omnoho objemnejšie, keďže už pracuje s viacerými rôznymi incidentmi. Tento komponent je založený na komponente 1, a preto majú niekoľko spoločných vlastností.

Na rozdiel od komponentu 1 bude zobrazovať incidenty a ich stavy na časovej ose nad sebou. Taktiež môže užívateľ definovať vlastnú výšku stavov, farbu stavov, ikony apod. Rozdiel je pri klikaní myšou na stav, kedy sa nezobrazí detailný popis tohto stavu, ale zobrazí sa nové okno s komponentami 1 a 3, s ktorými bude možné ďalej pracovať. Ďalej by mal komponent umožniť užívateľovi filtrovať zobrazené incidenty napríklad podľa časového okna (za posledných 24 hodín, týždeň, mesiac apod.) či statusu posledných stavov incidentov. Na nasledujúcej skici 5.3 je zobrazený drôtený model tohto komponentu. A ďalej je udaný príklad definovania vlastností komponentu v 5.1 a príklad informácií o incidente v 5.2.



Obr. 5.3: Náčrt komponentu 2 s definovanými ikonami, označením každého incidentu, ukážkou popup okna a definovanou farbou stavov podľa statusu.

```
{
    svg_height:300,
    state_height:25,
    incidentColor:["#65b9c2"],
    popup: ["status", "priority", "owner"],
    domain:"priority",
    colors:[
        {
            domain:4,
            color:"#ff4133"
        }],
    icons:[
        {
            id:3,
            iconType:"label"
        },
        {
            id:7,
            iconType:"own",
            iconFormat:"url",
            icon:"https://github.com/favicon.ico"
        }]
  }
```

Výpis 5.1: Príklad použitia všetkých atribútov, ktoré užívateľ môže definovať vo formáte JSON.

```
[{
    ID:"2",
    name:"Incident2",
    description:"Something about incident 2.",
    created:"10 Feb 2021 09:30:14",
    opened:"10 Feb 2021 10:50:25",
    closed: "1 Mar 2021 12:54:00",
    resolved:"7 Mar 2021 16:54:32",
    states: [
        {
            status:"new",
            priority:4,
            owner:"Abby",
            starting_time:"10 Feb 2021 10:50:25",
            ending_time:"10 Feb 2021 17:36:04"
        },
        {
            status:"active",
            priority:2,
            owner:"John",
            starting_time:"10 Feb 2021 17:50:14",
            ending_time:"11 Feb 2021 18:20:31"
        },
        {
            status:"onhold",
            priority:2,
            owner:"John",
            starting_time:"11 Feb 2021 19:00:04",
            ending_time: "23 Feb 2021 16:50:02"
        },
        {
            status:"active",
            priority:1,
            owner:"Igor",
            starting_time:"23 Feb 2021 17:50:14",
            ending_time:"1 Mar 2021 12:54:00"
        },
        {
            status:"closed",
            priority:1,
            owner:"Admin",
            starting_time:"1 Mar 2021 13:15:02",
            ending_time:"7 Mar 2021 16:50:32",
            comment:"Something said about resolving this incident.",
            commentImage:"https://webpage.commentimage.com",
            commentTime:"7 Mar 2021 16:55:02"
        }
    ]
}]
```

Výpis 5.2: Jednoduchý príklad informácií o incidente vo formáte JSON.

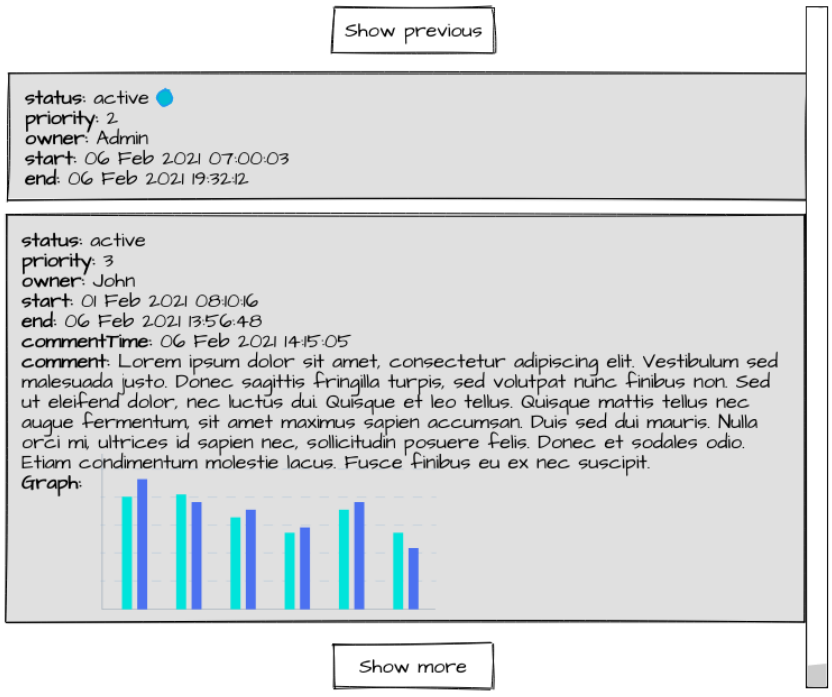
5.3 Návrh komponentu 3

Komponent 3 pre vykresľovanie pracuje so štyrmi parametrami:

- údaje o samotnom incidente (*incident*),
- údaje o vzhľade komponentu 1 (*property*),
- údaje o vzhľade komponentu 3 (*description*) – definuje ich užívateľ a sú to napríklad výška celého komponentu, počet stavov na stránku alebo farba komponentu,
- index stavu, na ktorý bolo kliknuté.

Tento komponent bude zobrazovať detailné informácie o jednotlivých stavoch incidentu od najaktuálnejšieho stavu po najstarší. Informácie o každom stave incidentu by mali byť nejako oddelené od ostatných stavov, aby bolo jasné, ktoré informácie patria, ku ktorému stavu. Komponent okrem zobrazovania textu musí byť schopný zobrazovať informácie o stavoch incidentu aj vo forme obrázkov, zoznamov, odkazov a príloh. Ak sú stavy incidentu v komponente 1 vyfarbené podľa určitej domény, mala by sa táto farba zobraziť aj v tomto komponente.

Pokiaľ je prekročený definovaný počet stavov na stranu, na spodnej časti stránky sa zobrazí tlačítko *Show more*, ktoré umožní zobraziť ďalšie stavy incidentu. Po kliknutí na toto tlačítko sa zobrazí na vrchnej časti stránky *Show previous*. Tieto tlačítka budú umožňovať pohyb medzi stránkami, a teda aj stavmi incidentu. Ak je presiahnutá predvolená alebo užívateľom definovaná výška komponentu, bude sa dať rolovať cez všetky stavy na stránke. Tieto vlastnosti sú ukázané na nasledujúcej skice 5.4 a popis vlastností, ktoré užívateľ môže definovať pre vzhľad komponentu 3 je predvedený v 5.3.



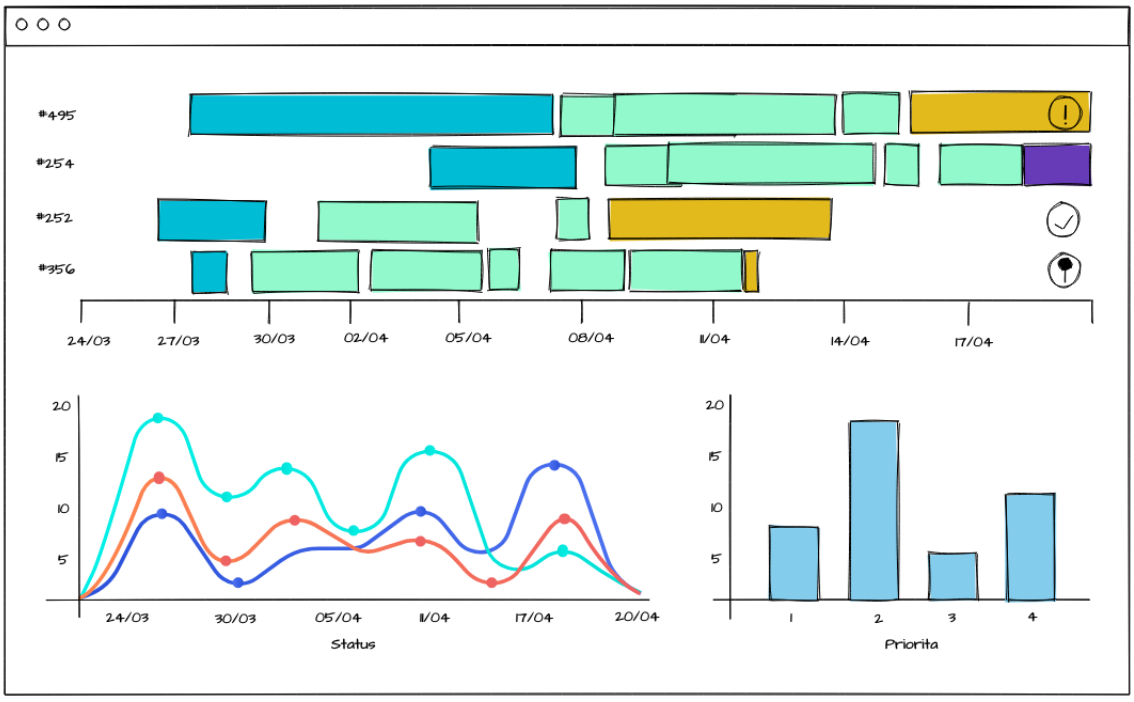
Obr. 5.4: Náčrt komponentu 3 s ukážkou tlačítiek pre posúvanie medzi stránkami a ukážka značky farby stavu incidentu pri statuse podľa jeho farby v komponente 1 (lepšia demonštrácia tejto značky je uvedená pri návrhu dashboardu na obrázku 5.6).

```
{
    height:600,
    color:"#b2ebed",
    statesOnPage: 5
}
```

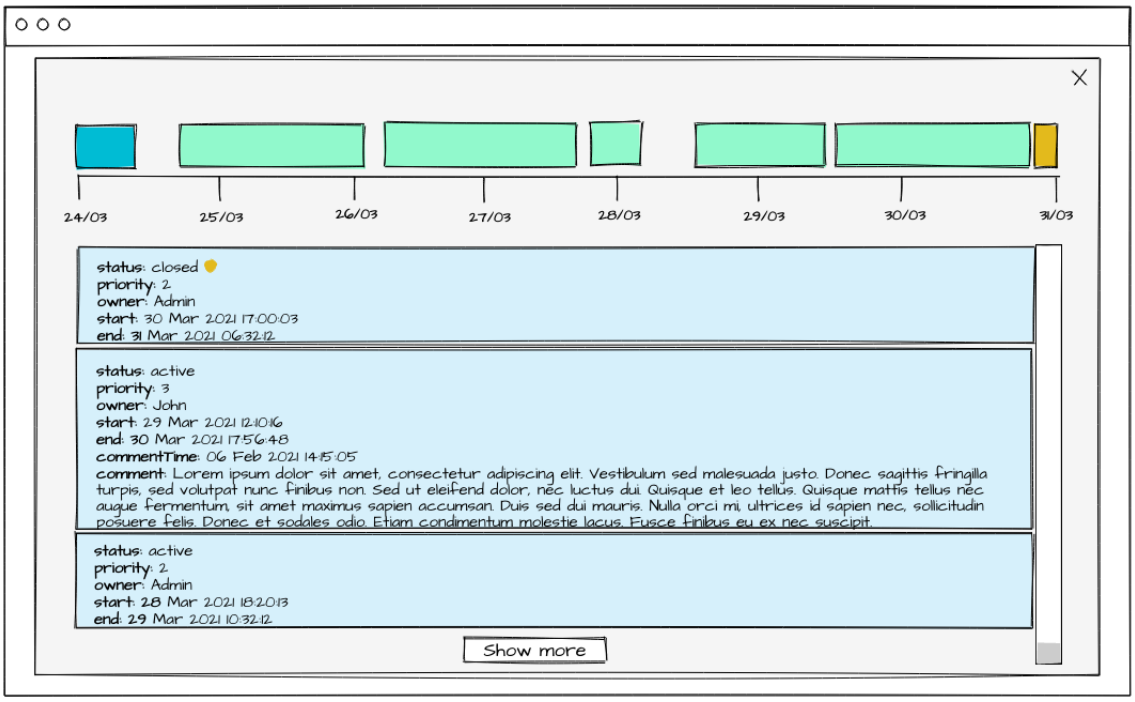
Výpis 5.3: Príklad všetkých možných atribútov *description*, kde **height** udáva výšku celého komponnetu, (predvolená hodnota: 500), **color** definuje farbu komponentu, (predvolená hodnota: "rgb(204, 204, 204)") a **statesOnPage** definuje počet stavov na stránku, (predvolená hodnota: 10).

5.4 Návrh dashboardu

Výsledný dashboard bude vytvorený najmä pre demonštráciu funkčnosti jednotlivých komponentov, ale môže byť tiež využitý v praxi pre analýzu riešených incidentov. Dashboard sa skladá z dvoch častí: úvodnej obrazovky a obrazovky po kliknutí na stav incidentu v komponente 2. Úvodná obrazovka dashboardu (obrázok 5.5) obsahuje 3 komponenty. V hornej polovici sa nachádza komponent 2, ktorý má byť najviac zdôraznený pri prvom pohľade na obrazovku. Ďalej sa v spodnej polovici dashboardu nachádzajú dva grafy: čiarový, ktorý zobrazuje postupný vývoj incidentov podľa ich statusu a stĺpcový, ktorý zobrazuje počet incidentov podľa priority. Obrazovka po kliknutí na stav incidentu (obrázok 5.6) obsahuje len komponent 1, ktorý zobrazuje celý incident v čase a komponent 3, ktorý zobrazuje detailný popis stavov tohto incidentu.



Obr. 5.5: Úvodná obrazovka dashboardu s komponentom 2, čiarovým a stĺpcovým grafom pre demonštráciu funkčnosti vytvorených komponentov.



Obr. 5.6: Obrazovka dashboardu po kliknutí na stav incidentu s komponentami 1 a 3, ktoré zobrazujú detail tohto incidentu, na ktorý bolo kliknuté.

Kapitola 6

Implementácia

Táto kapitola pojednáva o výslednej implementácii komponentov, o dátach, s ktorými komponenty pracujú i technológiách využitých k ich vytváraniu. Ďalej na obrázkoch 6.8 a 6.9 je uvedený výsledný demonštračný dashboard s popisom jeho prvkov a funkčnosti.

6.1 Použité technológie

Všetky komponenty sú vytvárané ako komponenty knižnice React, ktorá umožňuje parametrizáciu týchto komponentov i ich ľahkú znovu-použiteľnosť. Pre zapuzdrenie prvkov týchto komponentov bol vybraný ako vhodný nástroj práve React, s ktorým má firma Flowmon už skúsenosti. React je knižnica jazyka JavaScript pre vytváranie užívateľských rozhraní. Výsledný kód je zložený z komponentov, ktoré je možné parametrizovať pomocou tzv. *props*. Každý komponent má tiež svoj životný cyklus, ktorý môže obsahovať niekoľko metód, ako napríklad `shouldComponentUpdate`, `componentDidUpdate`, `componentDidMount`, `componentWillUnmount`, `render`, atď.

Ďalšie knižnice, ktoré sú použité pri tvorbe komponentov sú napríklad D3.js[1], ktorá je využívaná pre vykresľovanie komponentov vo formáte SVG a ich štýlovanie a prácu s nimi. Knižnica d3-timelines[2] je využívaná pre tvorbu timeline grafov v komponentoch 1 a 2. Z ďalšej knižnice Nivo[3] využívam stĺpcový a čiarový graf, ktoré sú umiestnené na úvodnej obrazovke dashboardu.

6.2 Komponenty 1 a 2

Keďže majú tieto komponenty tie isté požiadavky okrem funkcie klikania, boli vytvárané ako jeden React komponent. Životný cyklus tohto komponentu je okrem funkcie `render` riadený týmito funkciami:

- `componentDidMount` – nachádza sa tu EventListener[4] pre zaistenie responzívnosti komponentu a tiež sa tu volá funkcia `createTimelineChart`.
- `componentWillUnmount` – odstraňuje vytvorený EventListener.
- `componentDidUpdate(prevProps)` – spúšťa nové rendrovanie, ak sa zmenili props.

[1] https://d3js.org/
[2] https://github.com/denisemauldin/d3-timelines
[3] https://nivo.rocks/about
[4] https://developer.mozilla.org/en-US/docs/Web/API/EventListener

6.2.1 Props

Komponent pracuje s nasledujúcimi props: *incidents* (výpis 5.2), ktoré obsahujú informácie o incidentoch a ich stavoch, *property* (výpis 5.1), kde užívateľ môže definovať vlastnosti komponentu 1 alebo 2 a *description* (výpis 5.3), pre definovanie vlastností komponentu 3. Okrem props incidents sú ostatné props nepovinné a ak nie sú uvedené, sú pre dané vlastnosti nastavené preddefinované hodnoty. Povinnými atribútami props incidents sú `ID`, `created` a `states`. Prehľad ďalších možných atribútov týchto incidentov je možné nájsť v tabuľke 4.1.

6.2.2 Implementované funkcie

Celé vytváranie komponentu je zabalené vo funkcii `createTimelineChart`. V nej je implementované spracovanie dát z props a príprava HTML elementov, ktoré sú ďalej potreba, ako napríklad HTML div elementy pre popup okno (obrázok 6.1), filtrovanie (obrázky 6.2 a 6.3), zoom (obrázky 6.4 a 6.5), ikony apod. Je tu implementovaných i niekoľko ďalších pomocných funkcií, ktoré slúžia pre správnu funkčnosť a vykresľovanie diagramu ako napríklad `createChart`, `zoom`, `update`, atď.

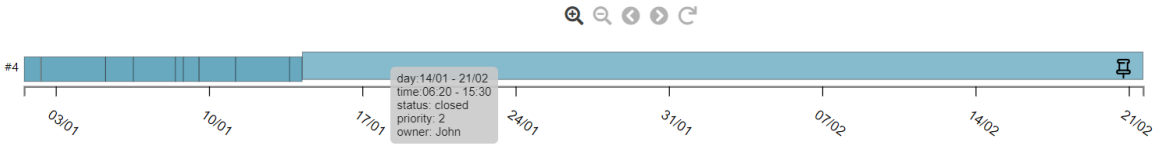
Tvorba diagramu

Nastavenie vlastností vykresľovaného timeline diagramu prebieha vo funkcii `createChart`, ktorá ako parametre prijíma začiatok a koniec diagramu. V tejto funkcii využívam diagram z knižnice d3-timelines, ktorému nastavujem vlastnosti ako napríklad okraje, výšku stavov, definovanú farbu stavov, začiatok a koniec grafu, formát zobrazenia dátumu či vlastnú definíciu funkcií pri *hover*, kliknutí alebo pri *mouseout*. Pre správne zobrazenie formátu dátumu je tu použitých niekoľko podmienok, ktoré zisťujú v akom časovom intervale sa graf nachádza a podľa toho vyberú vhodný formát. Okraje diagramu sa tiež prepočítajú podľa dĺžky identifikátora jednotlivých incidentov, aby ich označenie nepresahovalo až do výsledného diagramu.

V callback funkcii reagujúcej na udalosť `hover` je definované správanie pri prechádzaní myši nad stavmi incidentu. Pri umiestnení myši nad stav incidentu sa zistí, na ktorom stave je umiestnená myš (jeho id) a do pomocných premenných sa uloží pôvodná výška a farba stavu. Následne sa tieto vlastnosti zmenia pre zvýraznenie tohto stavu. Ďalej do predpripraveného HTML div elementu pre popup okno vložím údaje o danom stave, ktoré sa majú zobraziť a nastavím mu polohu na obrazovke v blízkosti kurzora myši. Callback funkcia reagujúca na udalosť `mouseout` nadväzuje na túto funkciu a po umiestení myši mimo stav nastavuje stavu uložené pôvodné hodnoty a popup oknu je nastavený transition pre hladké zmiznutie.

Callback funkcia reagujúca na udalosť `click` zaisťuje správnu funkčnosť po kliknutí na stav incidentu. Je rozdelená na dve časti: jedna časť je určená pre správanie po kliknutí definované pre komponent 1 a druhá pre správanie po kliknutí definované pre komponent 2. Po kliknutí na incident je zistené, o ktorý incident sa jedná a na ktorý stav bolo kliknuté. Pri každom kliknutí sa pripočítava hodnota `onClick1Incident`, ktorá pomáha určiť, či sa jedná o správanie pre komponent 1 alebo 2. Táto hodnota sa vždy vynuluje pri kliknutí na tlačítko pre vrátenie sa na úvodnú obrazovku. Ak sa táto hodnota rovná práve 1, vieme, že bolo prvýkrát kliknuté na komponent, a preto správanie odpovedá komponentu 2, teda vykresleniu obrazovky s komponentmi 1 a 3. Toto vykreslenie prebieha do vopred definovaného HTML div elementu, do ktorého sa zavolá rendrovanie komponentu 1 a 3

s príslušnými props. Pokiaľ je hodnota `onClick1Incident` väčšia ako jeden a je zobrazený len jeden incident, jedná sa o správanie definované pre komponent 1, teda v komponente 3 sa má vyrolovať daný stav na vrch stránky. Preto sa teda volá rendrovanie komponentu 3 s props `clickedState`, v ktorom sa nachádza zistený index stavu.

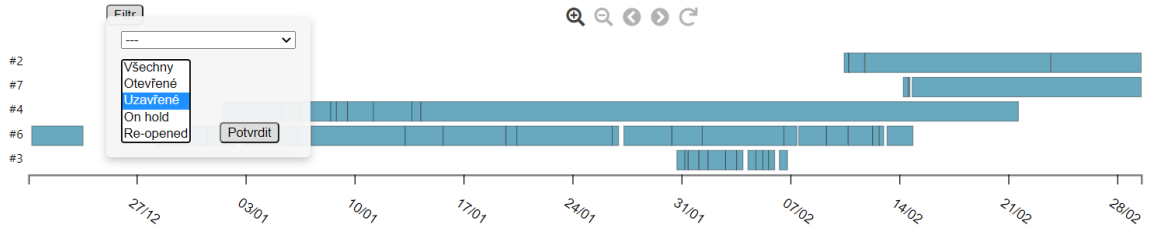


Obr. 6.1: Ukážka funkčnosti popup okna, ktorý má definované vlastnosti pre zobrazenie: status, owner a priority a zobrazuje i predvolené hodnoty, teda začiatok a koniec stavu.

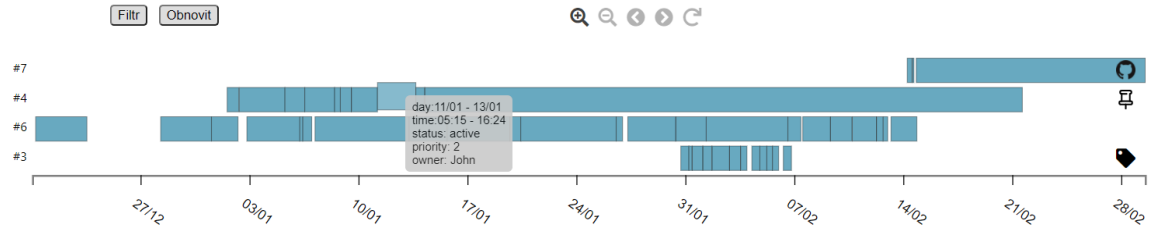
Filtrovanie

Filtrovanie je implementované pomocou niekoľkých HTML elementov, tlačítka *Filtr*, ktoré po kliknutí naň zobrazí vopred definovaný HTML div element s dvomi select elementmi a tlačítkom *Potvrdit* pre potvrdenie výberu možností. Select pre výber časového okna umožňuje užívateľovi vybrať len jednu zo štyroch hodnôt: *za posledných 24 hodín, za posledný týždeň, mesiac* alebo *nezmeneý čas*. Select pre výber filtrovania podľa statusu najaktuálnejších stavov umožňuje užívateľovi vybrať i viac ako jednu možnosť: *všetky stavy, otvorené, uzavreté, Onhold* alebo *Reopened*. Po stlačení tlačidla *Potvrdit* sa do premennej `ftime` uloží výber časového rozpätia a do poľa `fstate` sa uložia vybrané statusy.

Následne sa zavolá funkcia `update`, ktorá ako parametre prijíma aktuálne dáta (incidenty), `ftime` a `fstate`. V tejto funkcii sa pomocou niekoľkých podmienok zistí, či je možné aplikovať zvolený filter na incidenty. Teda či koniec grafu nie je starší ako zvolený interval alebo či najaktuálnejšie stavy incidentov obsahujú zvolené statusy. Pokiaľ je filtrovanie možné, uložia sa aktuálne dáta (incidenty) do pomocnej premennej a nové dáta sa vyberú z aktuálnych pomocou zvoleného filtra. Následne sa zvolí nový začiatok a koniec grafu a graf sa prekreslí. Ak nie je možné aplikovať zvolený filter, určí sa príznak, podľa ktorého sa následne zobrazí upozornenie, kde filtrovanie zlyhalo. Teda ak je aspoň jeden z príznakov `stateAlert` alebo `timeAlert` označený ako „true“, potom sa filtrovanie nepodarilo.



Obr. 6.2: Na snímke obrazovky je možno vidieť voľbu filtrovania.

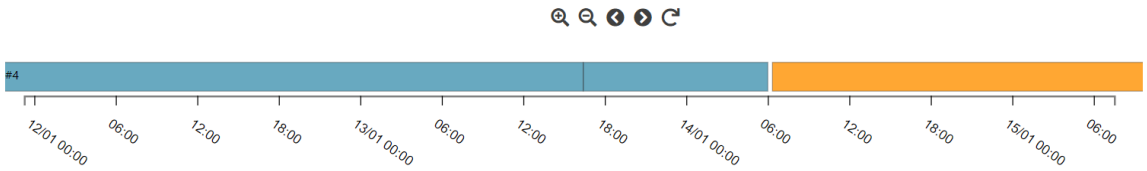


Obr. 6.3: Ukážka funkčnosti výberu filtrovania z obrázku 6.2 s tlačítkom pre obnovu do pôvodného stavu, definovanými ikonami a so zobrazeným popup oknom.

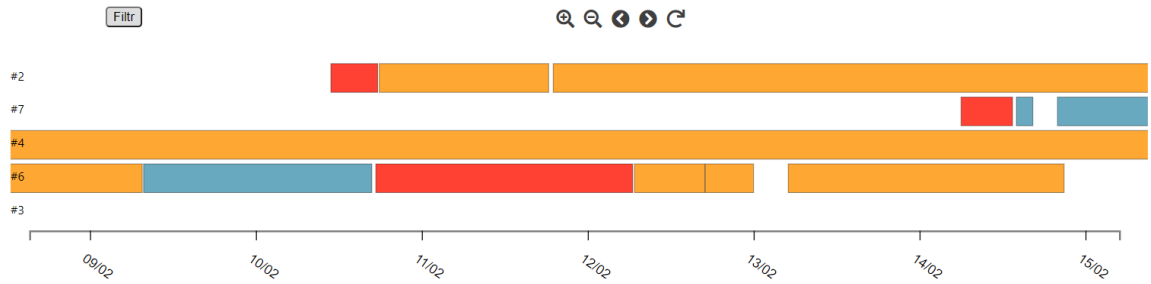
Zoom

Funkcia približovania a oddaľovania (ďalej len *zoom*) je implementovaná sadou piatich tlačítiek. Každé tlačítko má definovanú funkciu pre kliknutie naň. Všetky tlačítka okrem plus majú na začiatku zvolenú menšiu priehľadnosť, pretože ich nie je možné použiť. Až po kliknutí na tlačítko plus sa zvýraznia všetky tlačítka okrem šípky doprava a je uložený pôvodný časový interval do pomocnej premennej. Tlačítka plus a mínus využívajú definovanej funkcie `zoom`, ktorej predávajú príznak či sa jedná o priblíženie alebo oddialenie. Následne sa v tejto funkcii pomocou podmienok zisťuje či zväčšený začiatok a koniec nie sú mimo pôvodného intervalu. Teda či začiatok nie je starší ako pôvodný začiatok a koniec nie je väčší ako pôvodný koniec. Potom sa prepočíta nový začiatok a koniec podľa priblíženia a graf sa prekreslí. Taktiež sa tu mení i priehľadnosť tlačítiek podľa toho či je možné na ne ďalej kliknúť.

Tlačítka pre posúvanie v grafe fungujú na podobnom princípe ako zoom. Tiež pri kliknutí zisťujú či sa nepresiahol pôvodný časový interval. Potom sa posúvajú o tú istú hodnotu aká sa používa pre približovanie. Táto hodnota je vypočítaná z počtu dní a zvolenej ľubovolnej konštanty v milisekundách pre optimálne posúvanie a približovanie grafu. Po kliknutí na tlačítko pre obnovu sa zvolí uložený pôvodný začiatok a koniec a graf sa prekreslí.



Obr. 6.4: Snímka obrazovky komponentu 1, na ktorej je zobrazený incident po priblížení.



Obr. 6.5: Snímka obrazovky komponentu 2, ktorá tiež zachytáva pohľad na priblížené incidenty.

6.3 Komponent 3

Komponent 3 je taktiež implementovaný ako React komponent, ktorého životný cyklus je riadený funkciami: `componentDidMount` – volá sa tu funkcia `createRect`, v ktorej je implementované vytváranie celého komponentu, `componentDidUpdate(prevProps)` a `render`.

6.3.1 Props

Komponent pracuje s nasledujúcimi props: *property* (výpis 5.1), *incident*, ktorý obsahuje informácie o incidente a jeho stavoch. Jeho atribúty sa z väčšiny zhodujú s tými v 5.2, ID a states sú však nahradené názvami label a times, pretože táto komponenta získava dáta z komponentu 1 a 2 a túto úpravu bolo potreba urobiť kvôli grafu z knižnice d3-timelines. Ďalej props *description* (výpis 5.3), kde užívateľ môže definovať vlastnosti tohto komponentu a *clickedState*, kde je uvedený index stavu, na ktorý bolo kliknuté. Všetky props okrem incident sú nepovinné a ak nie sú zvolené, tak sú pre dané vlastnosti nastavené predvolené hodnoty.

6.3.2 Implementované funkcie

Tento komponent slúži pre zobrazovanie detailných informácií o jednotlivých stavoch incidentu. Vytváranie tohto komponentu je tiež umiestnené v samostatnej funkcii `createRect`. V nej je na začiatku implementované spracovanie dát z props a nastavenie vlastností komponentu. Taktiež sa tu nachádzajú i dve funkcie `fillRect` a `showMoreButtons`, ktoré zaisťujú vypĺňanie a správne fungovanie prepínania medzi stavmi incidentu.

Funkcionalitu rolovania stavu na vrch stránky pokiaľ bolo naň kliknuté zaisťuje jediná podmienka, ktorá zistí, či bol v props `clickedState` predaný index stavu. Ak taký index existuje, v cykle nájde, na ktorej stránke sa stav nachádza, daná stránka sa vykreslí a pomocou `scrollIntoView` sa vyroluje na daný stav.

Vypĺňanie elementov s informáciami o stavoch incidentu

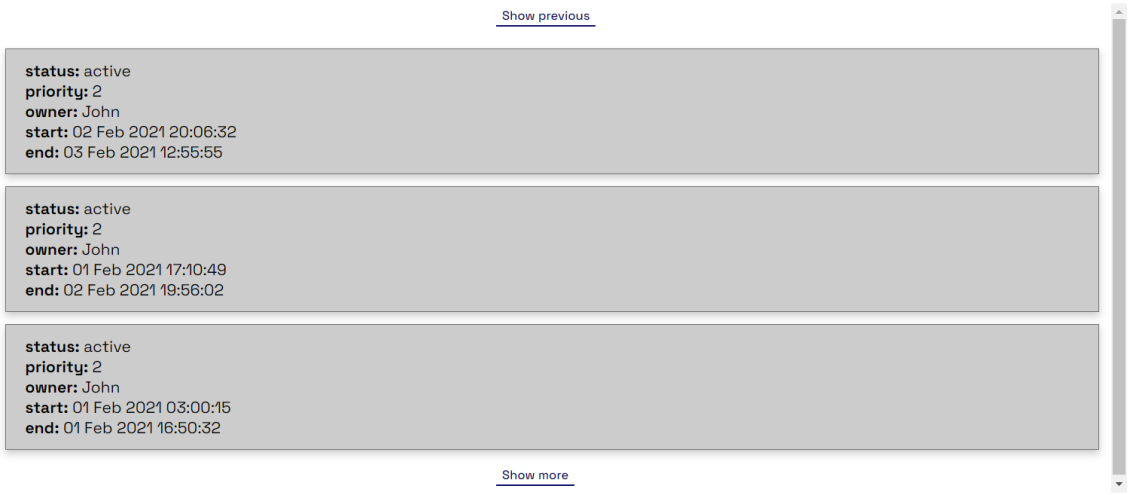
Funkcia `fillRect` slúži pre vypĺňanie rámčekov stavov s ich detailnými informáciami. Ako parameter prijíma stav, ktorý bude vykreslený do rámčeku. Na začiatku vytvorí do premennej `state` HTML div element, ktorý bude slúžiť pre zobrazovanie informácií o jednom stave. Následne mu priradí unikátne id. Ďalej sa zistí či bol stav vyfarbený podľa nejakej domény v komponente 1. Ak áno, táto farba sa uloží do pomocnej premennej `incidentColor` pre ďalšie použitie.

Na začiatku cyklu sa potom do pomocných premenných vytvoria potrebné HTML elementy, do ktorých sa následne vkladajú informácie o danom stave. Podľa niekoľkých podmienok funkcia zisťuje, aké elementy má využiť pre vykreslenie informácií o incidente, napríklad ak sa nájde v nejakom názve atribútu kľúčové slovo „Image“, tak sa vykreslí obrázok. Funkcia tiež obsahuje kontrolu pre správne zobrazovanie formátu času. Ak nájde atribút s kľúčovým slovom „Time“, upraví formát dátumu, aby bolo zobrazenie dátumov jednotné. Následne sa zistí či premenná `incidentColor` obsahuje farbu pre element, ktorý odráža farbu stavu podľa farby v komponente 1 a priradí ju prislúchajúcemu HTML elementu. Nakoniec sa všetky informácie o stave incidentu vložia do predpripraveného div elementu (obrázok 6.7).

Prepínanie medzi stavmi incidentu

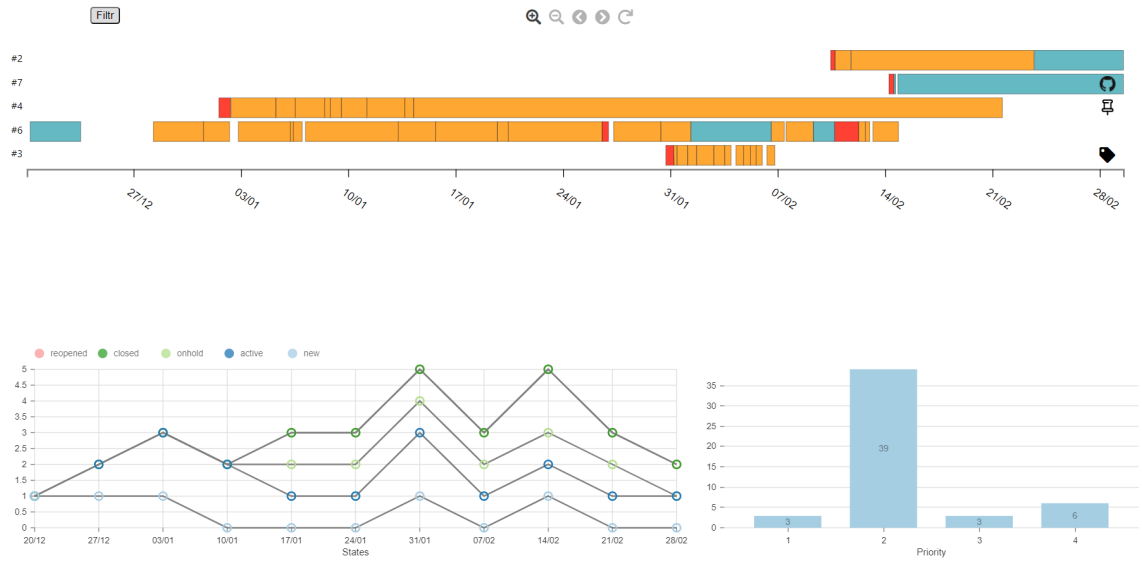
Funkcia `showMoreButtons` obstaráva správne fungovanie dynamického načítavania informácií o stavoch incidentu. Na začiatok stránky pridá tlačítko *Show previous* a nastaví mu viditeľnosť na „hidden“. Ďalej má definovanú funkciu po kliknutí naň, ktorá zisťuje či je predchádzajúca stránka prvá alebo len ďalšia a nakoniec zavolá funkciu `showMoreButtons`.

Následne po vytvorení tlačítka *Show previous* sa zisťuje či už bola vykreslená posledná stránka a ďalej sa v cykle volá funkcia `fillrect`. Potom sa pridá tlačítko *Show more* s nastavenou viditeľnosťou taktiež na „hidden“. Po kliknutí naň sa iba zavolá funkcia `showMoreButtons`. Ďalej sa len nastavuje viditeľnosť tlačítiek podľa toho, na ktorej stránke sa nachádzame. Ak sme na prvej a existujú ešte ďalšie stránky za ňou, tak nastavíme viditeľnosť len pre tlačítko *Show more*. Ak je stránka posledná, nastavíme viditeľnosť iba *Show previous*, inak nastavíme viditeľnosť obom tlačítkam (obrázok 6.6).



Obr. 6.6: Na snímke obrazovky je možno vidieť predvolenú farbu komponentu i s tlačítkami pre pohyb medzi stránkami s definovaným maximálnym počtom stavov na stránku: 3.

Obr. 6.7: Táto snímka obrazovky zobrazuje definovanú farbu komponentu užívateľom a prílohu ku komentáru vo forme obrázku.



Obr. 6.8: Na tejto snímke je možné vidieť úvodnú obrazovku dashboardu s vyrendrovaným komponentom 2, stĺpcovým a čiarovým grafom, ktoré boli použité z knižnice Nivo.

Obr. 6.9: Táto snímka obrazovky zobrazuje obrazovku dashboardu po kliknutí na stav incidentu v komponente 2, kde sa následne vykreslí okno s komponentami 1 a 3, s ktorými je možné ďalej pracovať. Toto okno môže užívateľ kedykoľvek zatvoriť pomocou tlačítka na obnovu v pravom hornom rohu. Taktiež tu môžeme vidieť, že po kliknutí na červený stav s popup oknom sa zobrazil na samotnom vrchu komponentu 3. Takisto je tu zobrazený aj indikátor farieb stavov v komponente 3 podľa farieb v komponente 1.

Kapitola 7

Testovanie

Výsledné komponenty boli testované pomocou niekoľkých rôznych vstupných dát, incidentov i vlastností grafov. Komponenty boli taktiež otestované s pracovníkmi firmy Flowmon, ktorý svoj názor k funkčnosti komponentov vyjadrili v predpripravenom dotazníku.

7.1 Testovanie vstupných dát a vlastností komponentov

Pre jednoduchšie testovanie vstupných dát a vlastností komponentov bol využitý nástroj Storybook[1]. Tento nástroj umožňuje jednoduché a ľahké testovanie izolovaných komponentov. Po vytvorení niekoľkých tzv. *stories* je možné demonštrovať vytvorené komponenty s rôznymi zvolenými parametrami. Tieto parametre je následne možné meniť a upravovať a výsledok je možné vidieť okamžite.

7.1.1 Testovanie komponentov 1 a 2

Testovanie najprv prebiehalo na malom množstve incidentov (5) bez definovania vlastností komponentu. Najmä bolo potreba zistiť či sa správne zobrazujú zadané stavy. Ďalej bolo treba otestovať správne filtrovanie týchto incidentov i približovanie a odďaľovanie. Filtrovanie bolo testované na prípady, keď si užívateľ zvolí len jednu z dvoch možností filtrovania alebo obe. Takisto bolo testované i pre chybové hlášky, ak užívateľ zvolil také parametre filtrovania, ktoré nevyhovovali výberu. Napríklad užívateľ vybral filtrovanie podľa statusu „reopened“, ale žiadny incident tomuto filtrovaniu nevyhovoval a bola zobrazená chybová hláška. Taktiež bola otestovaná funkcia zobrazovania popup okna po hover nad stavmi incidentu. Testované boli i krajné incidenty, na ktorých by mohlo byť popup okno nesprávne zobrazené. Funkcia zoomovania bola testovaná pre správne približovanie i odďaľovanie, ale i pre správne posúvanie sa po stavoch v komponente.

Po otestovaní funkčnosti komponentov pri malom množstve incidentov sa komponent testoval na väčšom množstve (100, 200). Toto množstvo incidentov však na celkovú funkčnosť komponentu nemalo nijaký vplyv a komponent bol stále funkčný, len o niečo pomalší.

Okrem testovania samotných incidentov boli tiež overené i definované vlastnosti komponentu užívateľom. Teda definovanie farieb všetkých stavov, farieb stavov podľa určitej domény, definovanie vlastných ikon vo formátoch url, base64 i FontAwesome i využitie preddefinovaných ikon. Taktiež i definovanie výšky stavov incidentov, výšky celého inci-

[1] https://storybook.js.org/docs/react/get-started/introduction

dentu alebo formátu popup okna. Všetky vyššie uvedené vlastnosti sa pri testovaní prejavili tak ako to bolo navrhnuté.

7.1.2 Testovanie komponentu 3

Testovanie tohto komponentu nebolo až tak zložité ako u komponentu 1 a 2, pretože komponent 3 má oveľa menej funkcií, ktoré užívateľ môže použiť. Bol tiež testovaný s malým množstvom (3) aj veľkým množstvom stavov (30). Testované boli rôzne možnosti zobrazovania detailov ako sú obrázky, odkazy, prílohy apod. Testovanie funkčnosti stránkovania bolo overené najskôr na pár incidentoch, ktoré vytvorili len dve stránky. Neskôr bola táto funkcia testovaná na stavoch, ktoré vytvorili len jednu stránku, a teda by sa nemalo zobraziť žiadne tlačítko. Taktiež bola testovaná i s veľkým počtom stavov. Overená bola tiež funkcionalita rolovania kliknutého incidentu na vrch stránky i užívateľom zadané vlastnosti komponentu ako je jeho výška, farba alebo počet stavov na stránku.

7.2 Testovanie použiteľnosti

Výsledné komponenty boli testované ďalšími 5 ľuďmi, pracovníkmi firmy Flowmon, ktorí na výslednom dashboarde testovali funkčnosť vytvorených komponentov a svoj názor vyjadrili pomocou predpripraveného dotazníka s otázkami k funkčnosti jednotlivých komponentov. Mali tiež možnosť vyjadriť sa ku každému komponentu nielen stupnicou od 1 (dobrý) až 5 (zlý), ale i slovne.

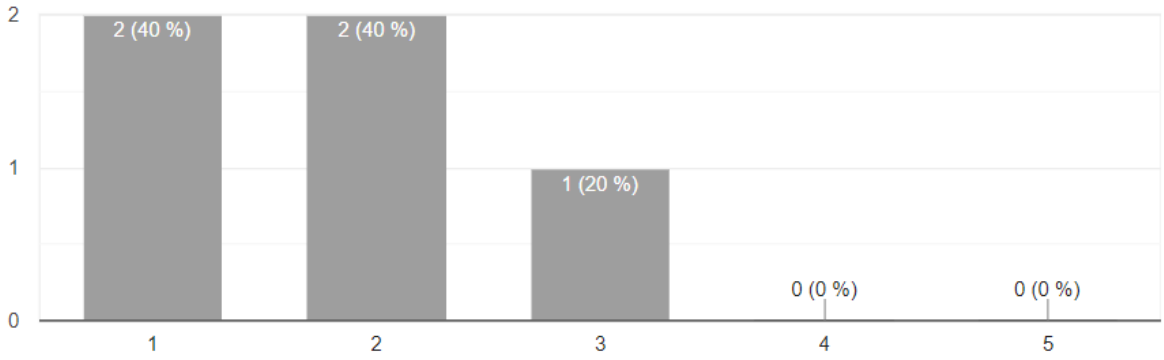
Užívatelia mali najväčšie problémy i výhrady k funkcii zoomovania, ktorá sa niektorým zdala mätúca pre používanie. Funkcia zoomovania fungovala tak, že približovala na najaktuálnejšie incidenty, no používatelia očakávali, že bude približovať na stred grafu. Niektorým tiež nevyhovoval spôsob ovládania pomocou tlačítiek a radšej by uvítali možnosť zoomovať pomocou tlačítiek myši. Užívatelia by tiež ocenili širšie možnosti filtrovania. Nielen podľa statusu a časového okna, ale chceli by mať možnosť filtrovať aj napríklad podľa priority či vlastníka. Podľa pripomienok užívateľov bolo filtrovanie rozšírené o možnosť filtrovať podľa priority a zoomovanie bolo upravené pre približovanie na stred diagramu.

V komponente 3 sa niektorým užívateľom nepáčili názvy tlačítiek či by radšej ocenili dynamické načítanie stavov než preklikávanie sa medzi ich stránkami. Ďalej tiež navrhli nejaké označenie stavov, aby vedeli na ktorej stránke, s ktorými stavmi sa aktuálne nachádzajú. Podľa pripomienok užívateľov k názvom tlačítkam boli následne ich názvy zmenené na *Next page*, *Previous page* a *Zrušit filtr*.

Celkovo však funkčnosť a orientáciu vo všetkých komponentoch zhodnotili kladne, v priemere známkou 1 a 2 (obrázok 7.1). Komponenty sa väčšine užívateľov páčili, len mali výhrady k UX (*User Experience*). Niektoré z výsledných hodnotení užívateľov prevzaté z dotazníku:

> „Vyzerá to celkom pekne, trochu by to chcelo zlepšiť UX, aby sa to lepšie používalo a bolo by to super.“

> „Treba doladiť praktické aspekty používania, celkový koncept je dobrý.“



Obr. 7.1: Graf zachytáva celkové hodnotenie komponentov užívateľmi Flowmon v predpripravenom dotazníku.

7.3 Zhodnotenie testov

Z výsledkov testov a pripomienok testovaných užívateľov je možné vyvodiť nasledujúce závery. Funkcionalita všetkých komponentov je dostačujúca avšak s pripomienkami k UX. Komponent 1 a 2 z celkového hodnotenia funkcionality a orientácie v nich dostali priemernú známku 2, ktorá odpovedá hodnote „celkom dobrý“. Najviac pripomienok mali užívatelia na funkciu zoomovania, ktorá sa im zdala mätúca a radšej by zvolili iný spôsob implementácie riešenia tejto funkcie. Orientáciu a používanie filtrovania hodnotili užívatelia kladne, len by ocenili rozšírené možnosti filtrovania. Vcelku boli s týmito komponentami spokojní, vedeli sa v nich orientovať i ich používať, mali však určité výhrady.

Komponent 3 z celkového hodnotenia funkcionality a orientácie dostal priemernú známku 2 a 1. Užívatelia sa vedeli jednoducho orientovať v komponente i pohybovať medzi stránkami. Výhrady mali k spôsobu zobrazovania stavov. Niektorí by chceli mať možnosť vidieť, kde sa aktuálne nachádzajú, na ktorých stavoch. Iní by viacej uvítali iný spôsob zobrazovania stavov, napr. dynamické načítavanie.

Kapitola 8

Záver

Cieľom tejto práce bolo vytvoriť prehľadnú vizualizáciu incidentov a ich stavov, ktorá by ich užívateľom uľahčila hľadanie informácií o incidentoch, a tak zefektívnila prácu pri ich riešení. Boli navrhnuté a vytvorené tri komponenty, ktoré zobrazujú incident a jeho stavy z rôznych pohľadov. Pre demonštráciu týchto komponentov a ich funkčnosti bol zvolený nástroj dashboard.

Najprv bol urobený prieskum v oblasti detekcie anomálií a vizualizácie dát. V rámci vizualizácie dát bolo popísaných niekoľko dobrých praktík pri návrhu a tvorbe vizualizácií i rady, ktorých je dobré sa držať pri tvorbe dashboardu. Po analýze požiadavkov užívateľov a zistení, s akými dátami budú komponenty pracovať, sa mohol začať návrh jednotlivých komponentov.

Po vytvorení návrhu boli komponenty následne implementované pomocou knižnice React. Ďalej bol tiež implementovaný výsledný dashboard, ktorý demonštruje vytvorené komponenty. Dôležitou súčasťou vývoja bolo tiež testovanie, ktoré v priebehu vývoja komponentov zaručovalo ich správnu funkcionalitu a nakoniec sa testovaním na užívateľoch zistilo, do akej miery sú výsledné komponenty použiteľné.

Výsledné komponenty budú využívané najmä vo firme Flowmon pre vizualizáciu incidentov a ich stavov. Je však možné tieto komponenty využiť i pre iné účely ako je zobrazovanie incidentov. Môže sa jednať o obecne nejaké udalosti zložené zo stavov, ktoré je potreba vizualizovať na časovej ose či vidieť detailný popis týchto stavov. Do budúcna je ešte treba urobiť pár úprav, aby sa užívateľom s komponentami lepšie pracovalo. Napríklad je potreba zapracovať na niektorých pripomienkach užívateľov z testovacieho dotazníku ako zoomovanie, filtrovanie, označenie aktuálnych stavov apod.

Literatúra

[1] *The D3 Graph Gallery – Simple charts made with d3.js.* [online]. Naposledy navštívené 27.01.2021. Dostupné z: https://www.d3-graph-gallery.com.

[2] *From data to Viz | Find the graphic you need.* [online]. Naposledy navštívené 27.01.2021. Dostupné z: https://www.data-to-viz.com.

[3] *KTD - Úplné zobrazení záznamu.* Databáze Národní knihovny ČR. [online]. Naposledy navštívené 26.01.2021. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000000442&local_base=KTD.

[4] *Preventing and detecting ransomware with Wazuh · Wazuh · The Open Source Security Platform.* [online]. Naposledy navštívené 25.04.2021. Dostupné z: https://wazuh.com/blog/preventing-and-detecting-ransomware-with-wazuh/.

[5] *Uživatelská příručka 11 - Anomaly Detection System.* Flowmon Networks. [online]. Naposledy navštívené 23.01.2021. Dostupné z: https://demo.flowmon.com/doc/adsplug/locale/cz/index.html?file=12845674.html.

[6] *Wazuh · The Open Source Security Platform.* [online]. Naposledy navštívené 25.04.2021. Dostupné z: https://wazuh.com/.

[7] *Data.* ManagementMania.com, 2018. [online]. Naposledy navštívené 26.01.2021. Dostupné z: https://managementmania.com/cs/data.

[8] *Kvalitativní data (Qualitative data).* ManagementMania.com, 2018. [online]. Naposledy navštívené 26.01.2021. Dostupné z: https://managementmania.com/cs/kvalitativni-data.

[9] *Kvantitativní data (Quantitative data).* ManagementMania.com, 2018. [online]. Naposledy navštívené 26.01.2021. Dostupné z: https://managementmania.com/cs/kvantitativni-data.

[10] *15 Customize Horizontal Timeline With CSS and JavaScript.* November 2020. [online]. Naposledy navštívené 25.04.2021. Dostupné z: https://jquerypluginsfree.com/15-customize-horizontal-timeline-with-css-and-javascript/.

[11] *Dashboard examples: The good, the bad and the ugly.* Január 2020. [online]. Naposledy navštívené 23.04.2021. Dostupné z: https://www.matillion.com/resources/blog/dashboard-examples-the-good-the-bad-and-the-ugly.

[12] Ahmed, M., Naser Mahmood, A. a Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications.* 2016, zv. 60, s. 19 – 31.

DOI: https://doi.org/10.1016/j.jnca.2015.11.016. ISSN 1084-8045. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1084804515002891.

[13] Bhuyan, M. H., Bhattacharyya, D. K. a Kalita, J. K. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys Tutorials*. 2014, zv. 16, č. 1, s. 303–336. DOI: 10.1109/SURV.2013.052213.00046.

[14] Chaudhuri, S. a Dayal, U. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.* New York, NY, USA: Association for Computing Machinery. marec 1997, zv. 26, č. 1, s. 65–74. DOI: 10.1145/248603.248616. ISSN 0163-5808. Dostupné z: https://doi.org/10.1145/248603.248616.

[15] Cichonski, P., Millar, T., Grance, T. a Scarfone, K. Computer security incident handling guide. *NIST Special Publication*. 2012, zv. 800, č. 61, s. 1–147.

[16] Eckerson, W. W. *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons, 2010.

[17] Few, S. *Information dashboard design: The effective visual communication of data*. O'Reilly Media, Inc., 2006.

[18] Gatto, M. A. Making research useful: Current challenges and good practices in data visualisation. Reuters Institute for the Study of Journalism. 2015.

[19] Harris, R. L. *Information graphics: A comprehensive illustrated reference*. Oxford University Press, USA, 1999.

[20] Kingatua, A. *9 Best Security Incident Response Tools for Small to Enterprise*. Máj 2020. [online]. Naposledy navštívené 02.02.2021. Dostupné z: https://geekflare.com/security-incident-response-tools/.

[21] Kirk, A. *Data visualisation: A handbook for data driven design*. Sage, 2016.

[22] Lawrence, J. *Incident Response Steps for NIST and SANS Framework*. ATT Cybersecurity. [online]. Naposledy navštívené 23.01.2021. Dostupné z: https://cybersecurity.att.com/blogs/security-essentials/incident-response-steps-comparison-guide.

[23] Maayan, G. D. *9 Free Tools to Automate Your Incident Response Process | AltexSoft*. Február 2020. [online]. Naposledy navštívené 02.02.2021. Dostupné z: https://www.altexsoft.com/blog/incident-response-tools/.

[24] Ping, Y., Xinghao, J., Yue, W. a Ning, L. Distributed intrusion detection for mobile ad hoc networks. *Journal of Systems Engineering and Electronics*. 2008, zv. 19, č. 4, s. 851–859. DOI: 10.1016/S1004-4132(08)60163-2.

[25] Soylemez, E., Welton, C., Dorman, G., Dombroski, M., Hopeman IV, A. A. et al. *Efficient SQL access to multidimensional data*. Google Patents, jún 12 2012. US Patent 8,200,612.

[26] Stevenson, A. *Oxford dictionary of English*. Oxford University Press, USA, 2010.

[27] Thottan, M. a Chuanyi Ji. Anomaly detection in IP networks. *IEEE Transactions on Signal Processing*. 2003, zv. 51, č. 8, s. 2191–2204. DOI: 10.1109/TSP.2003.814797.

[28] Tufte Edward, R. *The visual display of quantitative information.* Cheshire (Conn.), 2001.

[29] Unwin, A. Good graphics? In: *Handbook of data visualization.* Springer, 2008, s. 57–78.

[30] Zhang, W., Yang, Q. a Geng, Y. A Survey of Anomaly Detection Methods in Networks. In: *2009 International Symposium on Computer Network and Multimedia Technology.* 2009, s. 1–3. DOI: 10.1109/CNMT.2009.5374676.

Príloha A

Obsah CD

- **src/** – zdrojové súbory praktickej časti
- **doc/** – zdrojové súbory k technickej správe
- **other/** – priečinky so snímkami obrazovky a obrázkami návrhu, vytvorených komponentov a dotazníka