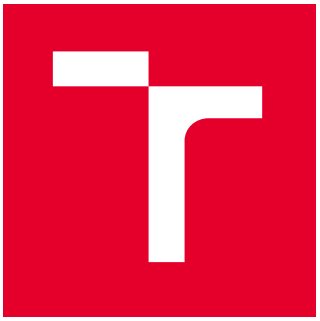


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

REALTIME SYSTÉM PRO WEBOVOU ANALYTIKU

REALTIME SYSTEM FOR WEB ANALYTICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Otto Hrubý**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radovan Holek, CSc.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Otto Hrubý ***ID:*** 211422

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Realtime systém pro webovou analytiku

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a implementujte systém pro webovou analytiku, který umožní efektivní sběr, zpracování a prezentaci dat z webových aplikací v reálném čase.

1. Proveďte rešerši existujících nástrojů pro webovou analytiku.
2. Nastudujte principy a technologie používané pro zpracování dat v reálném čase.
3. Analyzujte a stanovte požadavky na systém z hlediska funkcionality, rychlosti, dostupnosti.
4. Navrhněte způsob sběru a přenosu dat, datový model a informační systém pro správu nastavení a vizualizaci dat.
5. Realizujte vámi navržený systém.
6. Zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

KLEPPMANN M. Making Sense of Stream Processing, May 2016, O'Reilly Media, Inc., ISBN: 9781491937280, dostupne z:https://assets.confluent.io/m/ 2a60fabedb2dfbb1/ original/20190307-EB-Making_Sense_of_Stream_Processing_Confluent.pdf

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 15.5.2024

Vedoucí práce: Ing. Radovan Holek, CSc.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Tato práce se zabývá návrhem realtime systému pro webovou analytiku. Cílem práce bylo realizovat tento systém. Nejdříve byla provedena rešerše existujících nástrojů a vhodných technologií, na jejichž základě byl proveden návrh architektury systému. Data z webů se sbírají označkováním stránky jazykem JavaScript. Nasbíraná data se posílají do veřejného cloudu, kde je systém provozován. Systém data zpracuje, uloží je do databáze a prostřednictvím informačního systému data zpřístupňuje uživateli.

Klíčová slova

webová analytika, zpracování dat, veřejný cloud, databáze, informační systém

Abstract

This thesis deals with the design of a realtime system for web analytics. The aim of this work was to implement this system. At first, research was conducted on existing tools and suitable technologies, based on which the system architecture was designed. Data from websites are collected by tagging the pages with JavaScript. The collected data are sent to the public cloud, where the system is deployed. The system processes the data, stores them in a database and makes it accessible to users through an information system.

Keywords

web analytics, data processing, public cloud, databases, information system

Bibliografická citace

HRUBÝ, Otto. *Realtime systém pro webovou analytiku.* Brno, 2024. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/160072. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Radovan Holek.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Bc. Otto Hrubý
VUT ID studenta:	211422
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Realtime systém pro webovou analytiku

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 2. května 2024 ----------

podpis autora

Poděkování

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radovanovi Holkovi, CSc. za odborné vedení. Dále bych chtěl poděkovat firmě https://www.haltimo.com za možnost otestování systému na jejich webu.

V Brně dne: 2. května 2024 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

V dnešní době jsou téměř všichni lidé v moderní civilizaci připojeni k internetu a každý den konzumují jeho obsah. Úspěšné firmy poskytující služby prostřednictvím webu rozhodnutí nedělají na základě předpokladů, pocitů a domněnek, ale na základě faktů získaných z kvalitně nasbíraných, zpracovaných a interpretovaných dat.

Redaktory vydávající články na zpravodajské weby může zajímat počet prokliků a zobrazení článků z domovské stránky. Vydavatelé sledují, jak se článkům vede a na základě dat dostupných v rámci minut, pak mohou měnit nabízené články, či měnit jejich pořadí na domovské stránce. Každý klik zobrazí obsah článku, který zpravidla obsahuje reklamu generující zisk pro provozovatele stránek.

Majitelé e-shopů mohou mít zájem o sledování počtu zobrazení produktů a jestli jsou dostupné, průchodnosti nákupního procesu nebo o počtu uskutečněných nákupů. Může se stát, že byla vydána nová verze aplikace a uživatel nemůže projít konkrétním krokem nákupního procesu. Pokud nastane tento jev, tak výrazně klesne, v nejhorším případě úplně na nulu, počet uskutečněných nákupů. Každá minuta nefunkčnosti způsobuje ztrátu. Čím dříve se porucha opraví, tím menší jsou ztráty. Další problém nastává, pokud si uživatelé zobrazují nedostupné produkty, nebo pokud web uživatelům dokonce nedostupné produkty doporučuje.

Odvětví zabývající se vyhodnocováním dat z webů se nazývá webová analytika. Existuje mnoho nástrojů pro webovou analytiku, ale většina z nich obsahuje pouze základní statistiky v reálném čase jako je počítadlo uživatelů, návštěv a událostí. Neposkytují však možnost zobrazení detailních přehledů v reálném čase, protože neumožňují například filtrovat data nebo seskupovat události na základě vybraného atributu. Existují však specializované nástroje, které jsou zaměřeny například pro zpravodajské weby, ale jejich cena může být startovat od desítky tisíc korun za měsíc pro malý web. Většina nástrojů však umožňuje zobrazovat detailní denní statistiky, ale ty jsou zpřístupněny například s denním zpožděním.

Cílem této práce je vytvořit cenově dostupný systém, který je schopen spolehlivě sbírat, uložit a zpřístupnit data uživateli, aby byl schopen data použít v reálném čase.

1. WEBOVÁ ANALYTIKA

Webová analytika zahrnuje vše od získávání, předzpracování, uložení až po vizualizaci dat z webů pro získání přehledu o chování a akcích uživatelů. Nástroje webové analytiky slouží například pro sledování počtu a zdrojů návštěvníků, sledování jejich chování pro vylepšení uživatelské zkušenosti nebo k vyhodnocení výkonnostního marketingu. [1, 2]

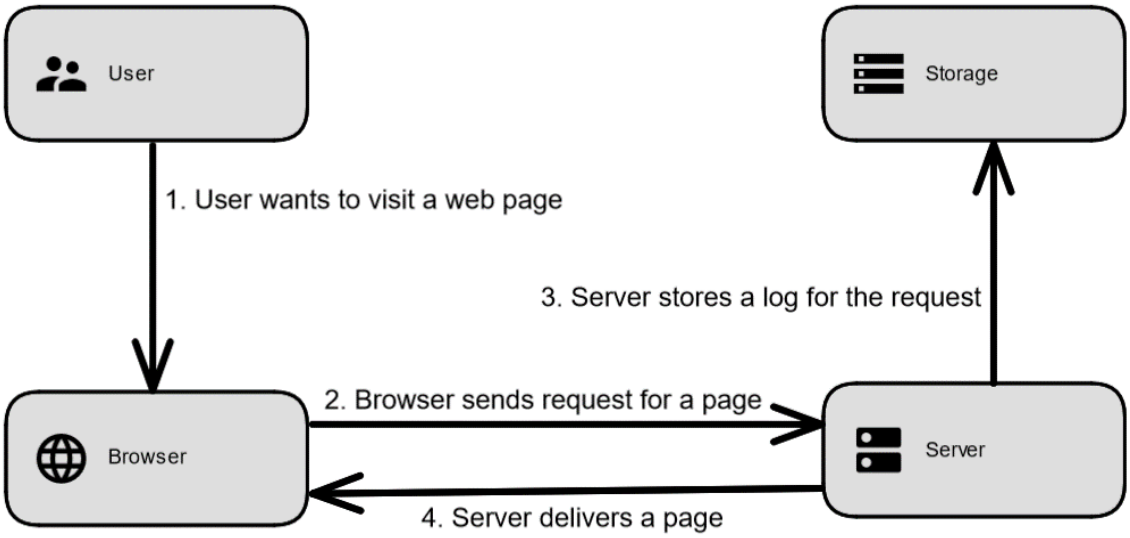
1.1 Techniky sběru dat z webů

Kvalita dat je nejdůležitější částí každého systému, který data využívá pro generování výstupu. Pokud jsou vstupní data nekvalitní, tak systém nemůže vygenerovat kvalitní výstup. Pro tento jev je využívaný pojem „smetí dovnitř, smetí ven“. [3]

1.2 Server logy

První způsob sběru dat, který existuje z raných dob internetu, vychází z logů, které ukládá webový server. Jejich původní účel byl pro zachycování chyb. Později byly logy rozšiřovány nejen pro provozní, ale pro analytické a marketingové potřeby. Proces sběru dat je znázorněn na obrázku 1.1 a skládá se ze čtyř kroků [1, 4, 5]:

1. Uživatel požádá server o obsah stránky, například přímým zadáním URL v prohlížeči nebo přesměrováním z jiné stránky.
2. Server obdrží požadavek od klienta.
3. Server vytvoří na základě požadavku log obsahující například název stránky, IP adresu uživatele, stavový kód a časovou značku.
4. Server pošle uživateli zpátky odpověď s obsahem.



Obrázek 1.1 Měření pomocí sběru server logů

1.2.1 Výhody

Většina serverů v sobě má zabudované mechanismy pro sběr a zaznamenávání logů. Servery buď logují automaticky, nebo se sběr logů dá zapnout změnou v nastavení serveru. Z tohoto plyne první výhoda použití server logů a to, že se jedná o snadno a rychle dostupný zdroj dat. [1, 5, 6]

Navíc je to jediná technika sběru dat, co se používá pro spolehlivé zachytávání a ukládání návštěv robotů sloužících k indexaci stránek pro webové vyhledávače. Většina robotů totiž nenačítá JavaScript, co se vkládá do stránek, takže je jiné techniky měření nemusí zachytit. Pokud se požaduje sledovat, jestli stránku navštěvují tito roboti, tak vzniká potřeba analyzovat serverové logy. Existují volně dostupné nástroje pro zpracování záznamů ze serveru. Je tedy poměrně jednoduché rychle obdržet základní statistiky. [1, 4, 5]

Dalším výhodou je to, že uživatelé webových stránek nemohou zabránit, aby si server ukládal záznamy. Nejsou také schopni určit, jestli je někdo sleduje a tím, že sběr dat provádí server, tak měření ani nezatěžuje prohlížeče. [5, 6]

Posledním benefitem je skutečnost, že data vlastní provozovatel stránek a data nemusí opouštět server. Provozovatel má plnou kontrolu nad svými daty. Pokud provozovatel pracuje s citlivými daty, pak je tahle metoda jedinou vhodnou volbou. Ostatní metody posílají data na další server. [4, 5, 6]

1.2.2 Nevýhody

Logy jsou primárně určeny pro technické informace jako je měření chyb stránek, odezvy a zatížení serveru. Záznamy nejsou bez rozšíření vhodné pro získávání přehledu třebas o marketingu. Pokud je potřeba mít přehled o netechnických informacích je potřeba úzká spolupráce s IT oddělením kvůli rozšíření měření. Další problém nastává, pokud klienty obsluhuje vícero serverů. [5, 6]

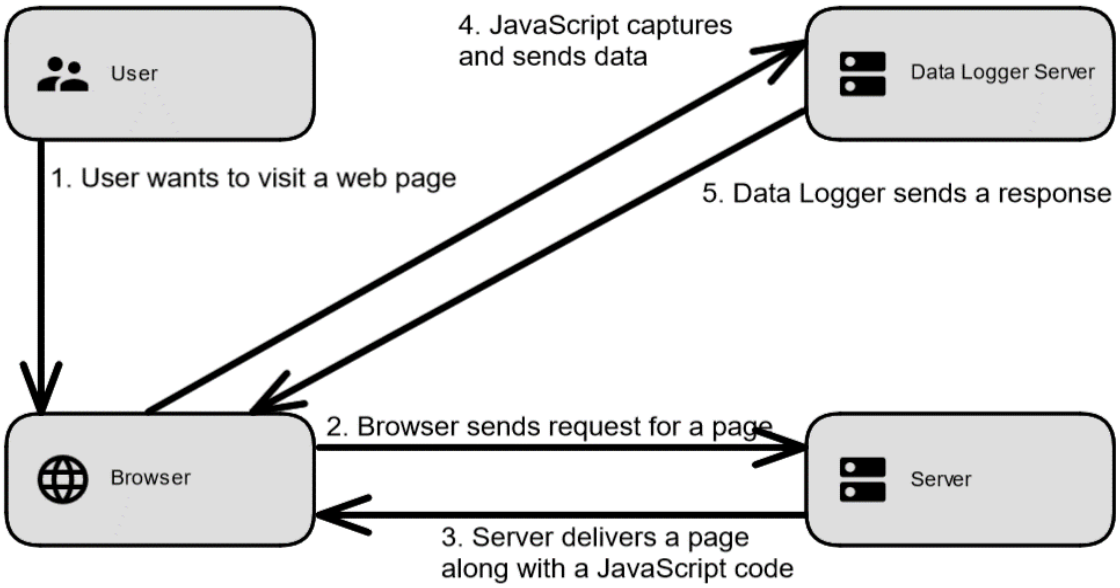
Technika sběru dat pomocí logů je používána pro záznam všech příchozích požadavků na server. Při požadavku na analýzu dat o chování uživatelů je nutné data vhodně vyfiltrovat od šumu jako je již zmíněná návštěvnosti robotů pro indexaci, požadavků pro stáhnutí obrázků nebo souborů obsahující kaskádové styly. [1, 5]

Webové prohlížeče a proxy servery navíc používají cachování, což vede k tomu, že server požadavek vůbec neobdrží. Tím pádem server danou návštěvnost vůbec nezaznamená. [5]

1.3 Sběr dat přidáním JavaScript značek v prohlížeči

Sběr dat pomocí přidání JavaScript značek v prohlížeči vznikl pro vyřešení nevýhod plynoucích z měření pomocí serverových logů. Tahle technika vychází z toho, že v dnešní době drtivá většina internetových prohlížečů dokáže interpretovat skriptovací jazyk JavaScript. Měření dat je znázorněno na obrázku 1.2 a skládá se z následujících kroků [1, 4, 5]:

1. Uživatel požádá server o obsah stránky, například přímým zadáním URL v prohlížeči nebo přesměrováním z jiné stránky.
2. Server obdrží požadavek od klienta.
3. Server pošle uživateli stránku, která v sobě zahrnuje měřicí značky v jazyku JavaScript. Jedná se o kód s logikou pro měření. Kód také vytváří soubory cookies, do nichž se ukládají informace o identitě uživatele, identifikace návštěvy či jejím zdroji.
4. Jakmile se stránka načte, tak se kód spustí a sbírá informace, co odesílá na server.
5. Server pro sběr dat vrátí odpověď, případně nastaví cookies.



Obrázek 1.2 Měření pomocí JavaScript značek v prohlížeči

1.3.1 Výhody

Existuje mnoho nástrojů, co poskytují vlastní JavaScript knihovny. Zprovoznění základního měření, tak může být velmi snadné, rychlé, a tedy i levné. Při vložení pár dalších značek do stránky je rozšiřování měření o další informace poměrně jednoduché. Tato technika navíc umožňuje oddělit logiku pro sběr dat od prezentace obsahu. Poskytuje tak velkou flexibilitu. Není potřeba zatěžovat do tak velké míry IT oddělení kvůli požadavkům týkajících se sběru dat. Je od něj však vyžadováno, aby byl kód pro měření všude tam, kde má být. Zejména větší přesnost a flexibilita vedla k obrovské popularitě této metody. [1, 5, 6]

Kód v sobě navíc obsahuje logiku pro předzpracování dat. Sbíraná data jsou tak strukturovaná a lépe čitelná. Je možné z nich snadno získat základní přehledy. Jak již bylo zmíněno u předchozí metody, tak roboti zpravidla nenačítají JavaScript na stránkách. Při sledování aktivit uživatelů není potřeba data do takové míry filtrovat a čistit. [5, 6]

Metoda také řeší problém s používáním cache. JavaScript kód se spustí a odesílá data nezávisle na tom, odkud byla stránka poskytnuta prohlížeči. Kód tak naměří větší část reálné návštěvnosti oproti měření pomocí logů. [5]

1.3.2 Nevýhody

Měření na straně klienta má však i své nevýhody. V první řadě je, že kód musí být umístněný na každé stránce, kde má probíhat měření, což vede ke zpomalení načítání stránek. S tímto souvisí skutečnost, že prohlížeč musí podporovat a musí mít povolené interpretování jazyka JavaScript. Pokud podmínka není splněna, pak se nenačte kód a měření neprobíhá. [1, 5, 6]

Situaci navíc komplikuje blokování JavaScript či cookies na straně klienta. Někteří uživatelé mohou mít kvůli soukromí, ale také kvůli rychlosti načítání nebo blokování rušivého obsahu, ve svém prohlížeči buď JavaScript přímo vypnutý, nebo častěji mají nainstalované různé doplňky, co blokují kód, který vyhodnotí, že slouží pro sběr dat. Nicméně přání skupiny uživatelů, co si nepřejí být sledováni by mělo být z hlediska soukromí respektováno i při použití jiných metod měření. [5, 7]

1.4 Nástroje webové analytiky

Na trhu figuruje mnoho nástrojů pro webovou analytiku. Většina z nich jsou poskytovány jako webová služba, takže se uživatel nemusí kromě nasazení JavaScript kódu na své stránky o nic víc starat. Existují však i open-source nástroje, pro jejichž provoz je potřeba mít vlastní infrastrukturu pro sběr a ukládání dat. [8]

Podle statistik z 22.10.2023 dle internetového poskytovatele průzkumů W^{3}Techs, který prohledává nástroje nasazené na webech, vyplývá, že stránky používají zpravidla více než 1 měřicí nástroj a že dominantní postavení zaujímá nástroj Google Analytics, jež je nasazen na 84,7 % stránek používajících alespoň 1 měřicí nástroj. [9, 10]

Statistiky zahrnují také nástroje jako jsou Facebook Pixel (17,4 %), Microsoft UET (2,6 %) a LinkedIn Insight Tag (1,3 %). Uvedené nástroje primárně slouží pro marketingové účely a nejsou pro porovnání čistě analytických nástrojů relevantní. [9, 11, 12, 13]

Dále existují úzce specializované nástroje pro optimalizaci uživatelské zkušenosti (UX) jako je Hotjar (4,9 %) nebo doplňky pro webové stránky postavené nad platformou WordPress jako je WordPress JetPack (9,4 %). [14, 15]

Na základě průzkumu bylo vybráno 5 nejvýznamnějších čistě analytických nástrojů. Porovnání je uvedeno v tabulce 1.1 a bylo provedeno na základě zdrojů [9, 14, 16, 17, 18, 19].

Tabulka 1.1 Porovnání nástrojů pro webovou analytiku

Vlastnosti	Google Analytics 4	Matomo	Cloudflare Web Analytics	Snowplow	Plausible
Podíl na trhu	84,7 %	2,5 %	1,8 %	1,4 %	0.2 %
Free verze	Ano	Ano	Ano	Ano	Ano
Typ sběru dat	JavaScript	JavaScript Server log	JavaScript	JavaScript	JavaScript
Provoz na vlastním HW	Ne	Ano	Ne	Ano	Ano
Škálovatelnost	Ano	Ne	Ano	Ano	Ne
Open-source	Ne	Ano	Ne	Ano	Ano
Přístup k datům	Ano (Cloud)	Ano (MySQL)	Ano (API)	Ano (Cloud)	Ano (PostgreSQL)
Uživatelské rozhraní	Ano	Ano	Ano	Ne	Ano
Realtime přehledy	Ano (Základní)	Ano (Základní)	Ne	Ne	Ano (Základní)

1.4.1 Google Analytics 4

Jedná se o nejpoužívanější nástroj pro webovou analytiku. Velká výhoda nástroje je možnost tvorby vlastních přehledů. Nicméně velká nevýhoda nástroje je, že dopočítávání dat v přehledech může trvat až 2 dny. Nástroj poskytuje také přehledy v reálném čase, ale ty obsahují pouze informaci o aktuálním počtu uživatelů, nových návštěv, navštívených stránek a počítadlo událostí. V přehledech však není možné aplikovat třeba filtrování nebo seskupovat data podle atributů. [14, 16]

Nástroj poskytuje také export dat v reálném čase do databáze Google BigQuery. Nicméně databáze není vhodná pro časté čtení záznamů, zejména kvůli rychlosti a ceně. Platí se totiž 6,25 $ za každý TB zkonzumovaný dotazem do databáze. Je důležité poznamenat, že nejmenší účtovaná jednotka pro dotaz je 10 MB. To znamená, že při čtení každou minutu by každý měsíc byl účtován minimálně 0,432 TB dat, což by stálo 2,7 $. Data však nejsou efektivně ukládána a cena navíc roste přímo úměrně s počtem uložených dat. Pokud by web navštívil větší počet uživatelů, tak by cena mohla dosáhnout stovek až tisíců dolarů měsíčně. [14, 16, 17]

1.4.2 Matomo

Matomo je monolitický open-source nástroj s možností provozu na vlastní infrastruktuře. Nástroj má možnost zobrazovat statistiky v reálném čase, ale jedná se pouze o počítadlo uživatelů, návštěv a událostí, bez možnosti rozpadu například dle názvu události nebo stránky, kde událost proběhla. [14, 18]

1.4.3 Cloudflare Web Analytics

Jedná se o software společnosti Cloudflare, který je nabízen jako webová služba. Slouží pouze pro analytiku webů používajících Cloudflare. Navíc nezobrazuje přehledy v reálném čase. [19]

1.4.4 Snowplow

Snowplow je open-source nástroj určený pro společnosti s velkým datovým oddělením, co potřebují získat maximum ze svých dat. Nástroj je velmi komplexní. Bohužel, obsahuje pouze logiku pro sběr a ukládání dat. Nenabízí ani uživatelské rozhraní. [14]

1.4.5 Plausible

Plausible je další z open-source nástrojů. Je velmi minimalistický a jednoduchý. Slouží pouze pro zpracování jednoduchých statistik týkajících se zobrazení stránek. Jedná se o monolitickou aplikaci vhodnou pro malé weby. [14]

2. ZPRACOVÁNÍ DAT V REÁLNÉM ČASE

Zpracování dat v reálném čase zahrnuje akvizici dat, přenos mezi systémy, jejich předzpracování a uložení do databáze ve vhodném formátu.

2.1 Komunikace

Před samotným zpracováním je potřeba data sbírat. To v našem případě může vyřešit JavaScript kód zmíněný v předchozí kapitole. Nasbíraná data potřebují být přenesena z webového prohlížeče na server. To znamená, že mezi stroji musí existovat komunikace. Systém posílající data se nazývá klient a systém, co odpovídá klientovi pak server. [20]

Application Programming Interface (API) slouží systémům ke vzájemnému sdílení dat a funkcí. Skládá se z technické specifikace popisující způsob komunikace a softwarového rozhraní, které specifikaci implementuje. V současnosti existuje mnoho stylů pro tvorbu API. [20, 21]

2.1.1 REST API

Zkratka REST znamená Representional State Transfer. V dnešní době se jedná o nejpopulárnější styl pro tvorbu webových API pro komunikaci prohlížečem a webovým serverem. REST pracuje s tzv. zdroji (resources). Každý zdroj má svůj jedinečný identifikátor, tzv. Uniform Resource Identificator (URI). Zdroj reprezentuje entitu aplikace a umožňuje její manipulaci prostřednictvím různých metod. [20, 22]

Důležitou vlastností REST API je, že server nesleduje stav klienta. Proto musí každý požadavek klienta vždy obsahovat všechny potřebné parametry. Tahle skutečnost znamená, že spojení je při každém požadavku znovu navázáno. Vyměňovaná data jsou v textovém tvaru a nejčastěji se používá flexibilního formátu JavaScript Object Notation (JSON). Z uvedených důvodů vyplývá, že se nejedná zrovna o nejrychlejší způsob komunikace, avšak server uvedeným způsobem dokáže obsloužit velký počet klientů. [20, 22, 23]

REST k přenosu dat mezi klientem a serverem nejčastěji používá protokol Hypertext Transfer Protocol (HTTP). HTTP definuje metody pro určení, jaký typ funkce chce klient provést při přístupu na server. Nejčastěji používané metody jsou [21, 22]:

- GET – slouží k získání informací o zdroji
- POST – uplatňuje se pro připojení nové informace ke zdroji
- PUT – používá se pro úpravu zdroje
- DELETE – poskytuje možnost odstranění zdroje

Při obdržení požadavku server provádí kontrolu, zdali je daný požadavek validní a klientovi vrátí odpovídající stavový kód. Hodnoty kódů jsou pro HTTP standardizované podle normy RFC 9110. Norma rozděluje stavové kódy do 5 skupin [21, 22]:

- (100 – 199) Informační
- (200 – 299) Úspěšné
- (300 – 399) Přesměrování
- (400 – 499) Chyba klienta
- (500 – 599) Chyba serveru

Server nemusí pracovat se všemi stavovými kódy. Minimálně by však měl používat následující [21, 22]:

- 200 (OK) – Standardní odpověď serveru, když vše proběhne správně.
- 301 (Moved Permanently) – Cílový zdroj byl přesunut na jinou URI.
- 400 (Bad Request) – Server nemůže zpracovat požadavek kvůli chybě na straně klienta. Jev může vzniknout, pokud klient pošle požadavek ve špatném formátu.
- 404 (Not Found) – Zdroj nebyl nalezen.
- 405 (Method Not Allowed) – Klient se pokusil využít metodu, kterou zdroj nepodporuje.
- 500 (Internal Server Error) – Signalizuje vznik chyby na straně serveru.

2.1.2 SOAP API

SOAP API je postavené na protokolu Simple Object Access Protocol (SOAP), kde systémy mezi sebou komunikují pomocí zpráv ve formátu Extensible Markup Language (XML). Velká nevýhoda SOAP je v tom, že XML zprávy jsou mnohem delší oproti formátu JSON, proto byl tento styl používán spíš v minulosti. Dnes se preferuje pro tvorbu API právě REST kvůli větší rychlosti a flexibilitě. [20, 24]

2.1.3 gRPC API

Zkratka gRPC označuje Google Remote Procedure Call. Jedná se o open-source framework navržen pro tvorbu vysokorychlostních a efektivních API využívaných pro komunikaci mezi distribuovanými systémy. Je založen na principu Remote Procedure Call (RPC), který klientovi umožňuje volat metody serveru tak, jako by byly lokální. [25]

Existují 4 typy služeb [25]:

- Unary RPC – Klient odešle 1 požadavek a server vrátí 1 odpověď.
- Server-streaming RPC – Klient odešle 1 požadavek a server odesílá klientovi proud dat.
- Client-streaming RPC – Klient odesílá serveru mnoho požadavků. Server pošle klientovi 1 odpověď, a to například po obdržení první zprávy, nebo po obdržení všech zpráv.
- Bidirectional-streaming RPC – Klient zahájí komunikaci. Se serverem si pak navzájem vyměňují proud dat.

Velkou nevýhodou je, že standardní specifikaci gRPC nelze přímo implementovat v prohlížeči. Problém řeší specifikace gRPC-Web, která umožňuje komunikaci

z webového prohlížeče. Do systému se však musí vložit proxy server pro překládání požadavků mezi specifikacemi. Framework gRPC tak není vhodnou volbou pro tvorbu webových API. [25, 26]

Pro definování služeb a struktury zpráv se používá především Protocol Buffers (Protobuf). Formát podporuje vytváření složitých struktur a vyžaduje silnou typovou kontrolu, což vede k větší spolehlivosti komunikace. Protobuf se navíc využívá při serializaci dat do binárního formátu, který je mnohem efektivnější než textový formát. [25]

2.2 Ukládání dat

Nasbíraná data je nutné vhodně ukládat. K tomuto účelu se používají databáze. Existují dva hlavní typy databází, z nichž každý má svůj způsob, jak s daty pracují a jak je uchovávají.

2.2.1 Relační databáze a SQL

Relační databáze mají přesně definované schéma. Základním blokem jsou tabulky. Každá tabulka obsahuje řádky (záznamy) se sloupci s jasně definovanými datovými typy, což zajišťuje kontrolu nad daty. [27]

Informace o relacích mezi tabulkami se uchovávají ve sloupcích obsahující tzv. cizí klíč. Cizí klíč odkazuje do jiné tabulky na primární klíč, který slouží jako unikátní identifikátor pro záznamy v tabulce. V relačních databázích se nejčastěji používají následující relace [27]:

- One-to-one (1:1) – Jeden záznam v jedné tabulce odpovídá jednomu záznamu v druhé tabulce.
- One-to-many (1:M) – Jeden záznam v jedné tabulce může být spojen s více záznamy v druhé tabulce.
- Many-to-many (M:N) – Více záznamů v jedné tabulce může být propojeno s více záznamy v druhé tabulce. K spojení se používá pomocná přechodová tabulka.

Pro práci s relační databází se používá Structured Query Language. SQL obsahuje příkazy rozdělené do 4 skupin [27, 28]:

- Data Manipulation Language (DML) – Obsahuje příkazy, jakou jsou vkládání (INSERT), odstraňování (DELETE) a aktualizace dat v databázi (UPDATE).
- Data Control Language (DCL) – Patří do ní příkazy zejména pro udělení (GRANT), zakázání (DENY) a odebrání (REVOKE) přístupových práv k datům.
- Data Definition Language (DDL) – Slouží pro vytváření (CREATE), odstraňování (DROP) a upravování (ALTER) objektů databáze.

- Data Query Language (DQL) – Umožňuje získávat data z databáze pomocí příkazu SELECT.

2.2.2 Nerelační databáze a NoSQL

Termín not only SQL (NoSQL) se používá, pokud databáze nepracuje pouze s SQL dotazy nebo neukládá data do tabulek. Nerelační databáze nabízí velkou flexibilitu a schopnost pracovat s daty, které nemusí být uloženy v tabulce s pevnou strukturou. Nicméně nedodržování pevné struktury a menší míra standardizace může vést k problémům při udržování integrity dat. [27, 28]

NoSQL databáze jsou specializované na práci s určitým druhem dat a existuje mnoho typů [27, 29]:

- Document based (dokumentové) – Data se ukládají ve formě dokumentů. Často se využívá formát JSON. Jednotlivé dokumenty tak mohou mít libovolnou strukturu.
- Graph (grafové) – Slouží pro ukládání dat ve formě grafů. Entity se ukládají jako uzly. Hrany reprezentují relace mezi entitami.
- Key-value (klíč-hodnota) – Každý záznam se ukládá jako pár klíč-hodnota. Každý klíč slouží jako identifikátor hodnoty.
- Wide column (sloupcové) – V zásadě se podobají relační databázi. Data se však ukládají denormalizovaně. Sloupcové databáze jsou optimalizovány pro čtení a zápis po sloupcích.
- Time series (časové řady) – Databáze jsou optimalizované pro práci s časově závislými daty.

2.3 Škálování

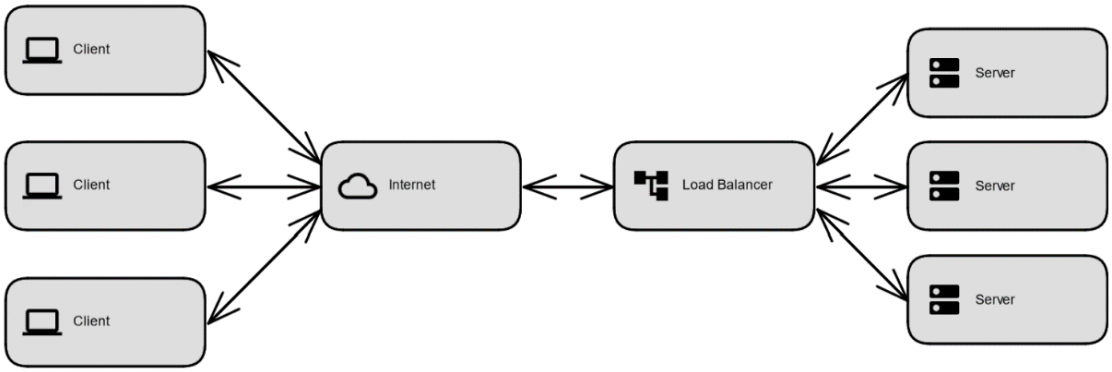
Objem přenášených dat se může rychle i postupně měnit a tím i požadavky na výpočetní výkon a úložiště systému. Škálování představuje schopnost systému přidávat nebo odebírat své prostředky tak, aby zvládal tyto požadavky plnit. Prostřednictvím veřejného cloudu lze dosáhnout také automatického škálování, kde si lze podle potřeby pronajímat prostředky, například na základě metrik, jakou jsou využití procesoru nebo využití paměti. Existují 2 hlavní techniky škálování: horizontální a vertikální. [30, 34]

2.3.1 Horizontální škálování

Při horizontálním škálování se zvyšují dostupné prostředky systému tím, že se do něj zapojí vícero instancí prostředků. Zvýší-li se požadavky na systém, pak se do něj zapojí více prostředků. Naopak, když zátěž klesne, prostředky lze postupně odebírat. Systém může dokonce postupně odebrat všechny prostředky. Prostředek může být například virtuální stroj. [30, 31, 32]

Tato technika je především užitečná v situacích, kde se potřebuje zvládat vysoký a proměnlivý tok požadavků. Horizontální škálování zvyšuje výkon serverů či distribuovaných systémů jako jsou NoSQL databáze. [30, 33]

V případě přidávání serverů je klíčové efektivně vyvažovat zátěž mezi tyto servery. K tomu slouží tzv. Load Balancer umístněný před servery. Jeho účelem je zajistit, že všechny servery pracují s maximální účinností. [31, 32]



Obrázek 2.1 Systém s vyvažováním zátěže

Když by byl 1 server přetížen a jiný nevyužit, klesá tak účinnost celého systému. Proto existuje mnoho algoritmů pro vyvažování zátěže. Každý se využívá pro splnění rozdílných požadavků [31, 32]:

- Round Robin – Požadavky se mezi servery distribuují sekvenčně.
- Least Connections – Požadavek obdrží server, co má navázán nejméně spojení.
- Least Time – Požadavek se pošle serveru podle algoritmu, který zajistí klientovi nejrychlejší odpověď.
- Hash – Klíč klienta určuje server, co obdrží jeho požadavek. Klíčem může být IP adresa klienta.

2.3.2 Vertikální škálování

Vertikální škálování je technika používaná ke zvyšování výkonu a kapacity systému tak, že se přidělí výkonnější prostředky dostupné stroji jako je větší úložiště na disku, větší operační paměť, výkonnější procesor s větší rychlostí a větším počtem jader. Nicméně nevýhodou oproti horizontálnímu škálování je, že prostředky nelze nekonečně zvětšovat a není tak flexibilní. Provoz výkonného stroje bývá zpravidla dražší než provoz vícero menších strojů. [30, 34]

Hlavní výhodou vertikálního škálování je jeho schopnost zvětšovat výpočetní kapacitu bez nutnosti provádět rozsáhlé změny a úpravy aplikace. Stroj také musí splňovat minimální hardwarové požadavky nezbytné pro provoz aplikace, v tomto

případě je vertikální škálování jedinou volbou. Metoda vertikálního škálování se používá zejména u monolitických aplikací a relačních databází. [30, 33, 34]

2.4 Události a zpracování proudu dat

Pro zpracování proudu dat se často používá technika, kdy se data strukturují jako události. Událost představuje neměnný fakt a mohou ji vyjadřovat v kontextu webové analytiky různé akce vygenerované stránkou, například zobrazení stránky, přihození do košíku a zakoupení zboží. Existují různé techniky, co se pro zpracování událostí používají. [35]

První technika spočívá v tom, že se ukládají všechny události bez dalšího zpracování, což poskytuje maximální flexibilitu při analýze. Avšak nevýhoda je ta, že při velkém počtu událostí, například při výpočtu ročních statistik, může docházet k pomalejšímu výpočtu, protože se prochází velký počet uložených událostí. [35]

Druhá technika využívá agregaci dat a je vhodná pro zpracování velkého množství dat. Ukládají se pouze výsledky operací provedené nad událostmi. Operací může být například součet událostí za hodinu. Techniku lze použít také pro monitorování překročení určeného limitu. [35]

2.4.1 Message broker

Message broker je software, který slouží k výměně informací mezi různými komponenty v systému. Message broker izoluje jednotlivé komponenty, což snižuje závislost mezi a umožňuje snazší rozšíření systému. Zpravidla poskytuje také možnost monitorování provozu. Je zvláště vhodný pro situace, kdy jsou jednotlivé zprávy generované asynchronně. Message broker většinou obsahuje frontu pro uchovávání událostí, dokud nejsou spolehlivě doručeny. Tímto se předchází ztrátě dat při výpadku nebo pomáhá zvládat nápor při špičkách. Message broker používá jeden ze dvou mechanismů posílání událostí. [36, 37]

Prvním mechanismem je Point-to-Point (P2P) messaging. Používá se pro komunikaci mezi 2 entitami – odesílatelem a příjemcem. Producent vytvoří a pošle zprávu do fronty zpráv. Fronta zajistí, aby data vydržela do jejich zkonzumování příjemcem nebo do vypršení časového limitu. Zprávu obdrží maximálně 1 příjemce. [36, 37]

Druhým mechanismem je Publish/Subscribe (Pub/Sub). Kdy producent (publisher) adresuje jednotlivé zprávy tématu (topic). Jeden či více odběratelů (subscriber) se přihlásí k tématu a odebírají z něj zprávy. Všechny zprávy adresované danému tématu jsou doručovány všem odběratelům. Některé typy systémů navíc obsahují frontu pro uchovávání zpráv, aby každý odběratel vždy spolehlivě obdržel zprávu. Pokud systém frontu nepoužívá, pak se zprávy posílají všem aktuálně přihlášeným odběratelům. Pokud daný odběratel přihlášen není, pak zprávu neobdrží. [36, 37]

3. POŽADAVKY NA SYSTÉM

3.1 Funkcionality

V následujících bodech jsou popsány klíčové funkcionality vytvářeného systému, jež jsou inspirovány z funkcionalit existujících nástrojů pro webovou analytiku.

3.1.1 Sledování návštěvníků

Pro efektivní sledování návštěvnosti je nezbytné monitorovat samotné návštěvníky. Definice a metodika identifikace návštěvníka se může v různých systémech lišit. Jeden z přístupů vychází ze sledování IP adresy, ze které posílá návštěvník požadavek. Problémem metody je vysoká nepřesnost, protože dochází k situacím, kdy vícero skutečných uživatelů sdílí stejnou IP adresu. Navíc je uživatel identifikován jako vícero různých návštěvníků, pokud změní svoji síť. [38]

Rozšířenější přístup využívá cookies, což zajišťuje jejich mnohem přesnější identifikaci. Je však důležité brát v úvahu, že návštěvník se v tomto případě identifikuje na základě aktuálně použitého prohlížeče. Pokud uživatel využívá vícero prohlížečů, tak je rozlišen jako vícero návštěvníků. [38]

Nejpřesnějším způsobem sledování využívá interního identifikátoru uživatele, který obvykle vznikne při registraci do systému. Uživatel pak lze identifikovat napříč různými zařízeními a prohlížeči. Je však nezbytné, aby aplikace umožňovala registraci. Jedná se tak spíš o doplněk k předchozím technikám. [39]

3.1.2 Sledování návštěv

Pro seskupování událostí návštěvníka v daném časovém okně se využívá návštěva (session). Začátek návštěvy nastane při příchodu návštěvníka na web. Ukončení návštěvy je však složitější, neboť se i definice návštěv v rámci nástrojů může lišit. Většinou se za konec návštěvy považuje situace, kdy návštěvník během 30 minut neprokázal žádnou aktivitu na webu. Informace o návštěvě se obvykle ukládají do cookies. [40, 41]

3.1.3 Sledovaní zdrojů návštěv

Uživatelé zahajují návštěvu prostřednictvím různých zdrojů a metod. Sledovat tyto údaje je důležité například pro vyhodnocování výkonu marketingových kampaní. Běžně se používají 3 parametry [42]:

- Médium (medium) – metoda, která přivedla uživatele
    - organic – příchod na web pomocí vyhledávače
    - cpc (cost per click) – příchod na web pomocí placené reklamy
    - referral – proklik z jiného webu
    - none – médium chybí, nastane například při přímém přístupu na web

    - email a další libovolné hodnoty
- Zdroj (source) – zdroj příchodu
    - direct – přímý přístup na web
    - google
    - seznam
    - newsletter a další libovolné hodnoty
- Kampaň (campaign) – obsahuje název marketingové kampaně

Pro identifikací zdroje návštěvy se používají 2 metody. První metoda používá manuální označení pomocí Urchin Tracking Module (UTM) parametrů v URL aktuální stránky. Druhý způsob se uplatní v případě absence UTM parametrů. V této situaci se používá kombinace aktuální URL stránky a referreru, což je odkaz, přes který uživatel přistoupil na stránku. Pro lepší pochopení je uvedeno pár příkladů v tabulce 3.1. [42]

Tabulka 3.1 Příklady zdrojů návštěvnosti

Aktuální URL stránky	Referrer	Zdroj	Médium	Kampaň
web.cz? utm_source=seznam &utm_medium=cpc &utm_campaign=sleva1	nemá vliv	seznam	cpc	sleva1
web.cz	google.com	google	organic	(organic)
web.cz	žádný	direct	(none)	(direct)
web.cz/a	web.cz/b	direct	(none)	(direct)
web.cz	jinyweb.cz/blog	referral	jinyweb.cz	(referral)

3.1.4 Rozpoznávání typů zařízení

Uživatelé na web přistupují z různých zařízení. Nejčastěji se využívají mobilní telefony, osobní počítače a tablety. Méně obvyklé typy jako jsou televize, elektronické čtečky a další nebudou uvažovány. Rozpoznávání typu zařízení může sloužit pro analýzy týkající se optimalizace obsahu a designu webu.

3.1.5 Sledování událostí

Akce uživatelů či stránky budou sbírány ve formě událostí. Každá akce bude představovat jednu neměnnou událost. Standardně nástroje webové analytiky sbírají některé základní události automaticky jako je zahájení návštěvy, zobrazení stránky, vertikální posouvání po stránce, kliknutí na odkaz a další. Automaticky lze s každou událostí měřit i určité atributy jako je stránka, kde se stala událost, zdroje návštěvy, typ zařízení atd. Specifické události a atributy si musí uživatel definovat sám a musí zajistit jejich měření přidáním logiky do webové aplikace, protože jsou běžně spojeny s konkrétním prvkem na stránce. [43]

3.1.6 Dimenze a metriky

Dimenze a metriky jsou běžně používané systémy pro webovou analytiku a společně slouží pro porozumění a analyzování dat. Dimenze slouží pro organizaci a agregaci dat. Dimenze je reprezentována textovým řetězcem. Metriky poskytují kvantitativní informaci jako je průměr, počet, hodnota, procento, cena, doba trvání. Každá metrika je vždy číslo spojené s jednotkou. [44]

3.1.7 Požadavky pro prezentaci dat

Systém bude také události efektivně ukládat, přičemž poskytne uživatelské rozhraní pro prezentaci výsledků a pro úpravy nastavení.

3.2 Rychlost

Při sběru dat je důležité, aby proces příliš neovlivňoval samotnou rychlost webové aplikace. Zpomalení by mohlo negativně ovlivnit uživatelskou zkušenost, což je nežádoucí jev.

Co se týká rychlosti zpřístupnění dat pro vizualizaci, tak je důležité, aby realtime data byla k dispozici do jedné minuty, v nejhorším případě pak do jednotek minut. Udržování vyšší rychlosti nemá z hlediska potřeb webové analytiky praktický význam a zvyšuje výpočetní nároky na systém.

3.3 Spolehlivost

Systém musí zajistit spolehlivost zejména při sběru a uložení dat do databáze. Jedním z aspektů spolehlivosti je dostupnost systému. Konkrétně část systému zabývající se příjmem nasbíraných událostí musí být co nejvíce funkční, protože při výpadku jsou data nevratně ztracena. Systém by si měl ze stejných důvodů poradit i v případě neúspěšného navázání spojení s databází.

3.4 Bezpečnost a soukromí

Pro zajištění bezpečnosti systému a poskytnutí soukromí uživatelům byly definovány následující opatření:

- Komunikace by měla používat šifrování pro bezpečný přenos dat.
- Podle evropského nařízení GDPR mohou být osobní údaje uživatele zpracovávána a ukládána pouze pokud k tomu dal jedinec souhlas. Souhlas může být udělen například zaškrtnutím políčka na webových stránkách. Podle nařízení GDPR platí: „Osobním údajem je jakákoli informace o identifikované či identifikovatelné osobě“. Osobní údaje tedy zahrnují také IP adresu a používání identifikátorů ze souborů cookies. [45]

- Všechny identifikátory uživatelů budou anonymizovány. Nikde nebudou uloženy osobní údaje o uživateli, které by samostatně vedly k jeho přesné identifikaci nebo lokalizaci.
- Data budou zpřístupněna pouze schváleným uživatelům s přiděleným heslem.

3.5 Náklady

Náklady systému souvisejí s infrastrukturou, na které systém běží, a s provozem této infrastruktury. Mělo by být usilováno o minimalizaci nákladů. Cílem je zajistit rovnováhu mezi náklady a výkonem.

4. ARCHITEKTURA SYSTÉMU

Pro splnění požadavků byla navržena architektura skládající se ze následujících modulů: měřicí skript, data logger, message broker, konzument zpráv, databáze a informační systém. Jednotlivé prvky budou v této kapitole popsány společně s výběrem konkrétních technologií a představení celkové architektury systému.

Pro minimalizaci nákladů bude systém provozován ve veřejném cloudu, čímž odpadá nutnost nákupu vlastního hardwaru a udržování vlastní infrastruktury. Náklady budou spojené pouze s využíváním jednotlivých služeb cloudu. V úvahu připadají 3 nejvýznamnější poskytovatelé těchto služeb, a to Amazon Web Services (AWS), Microsoft Azure (Azure) a Google Cloud Platform (GCP). Bude však vybrán pouze 1 poskytovatel.

4.1 Měřicí skript

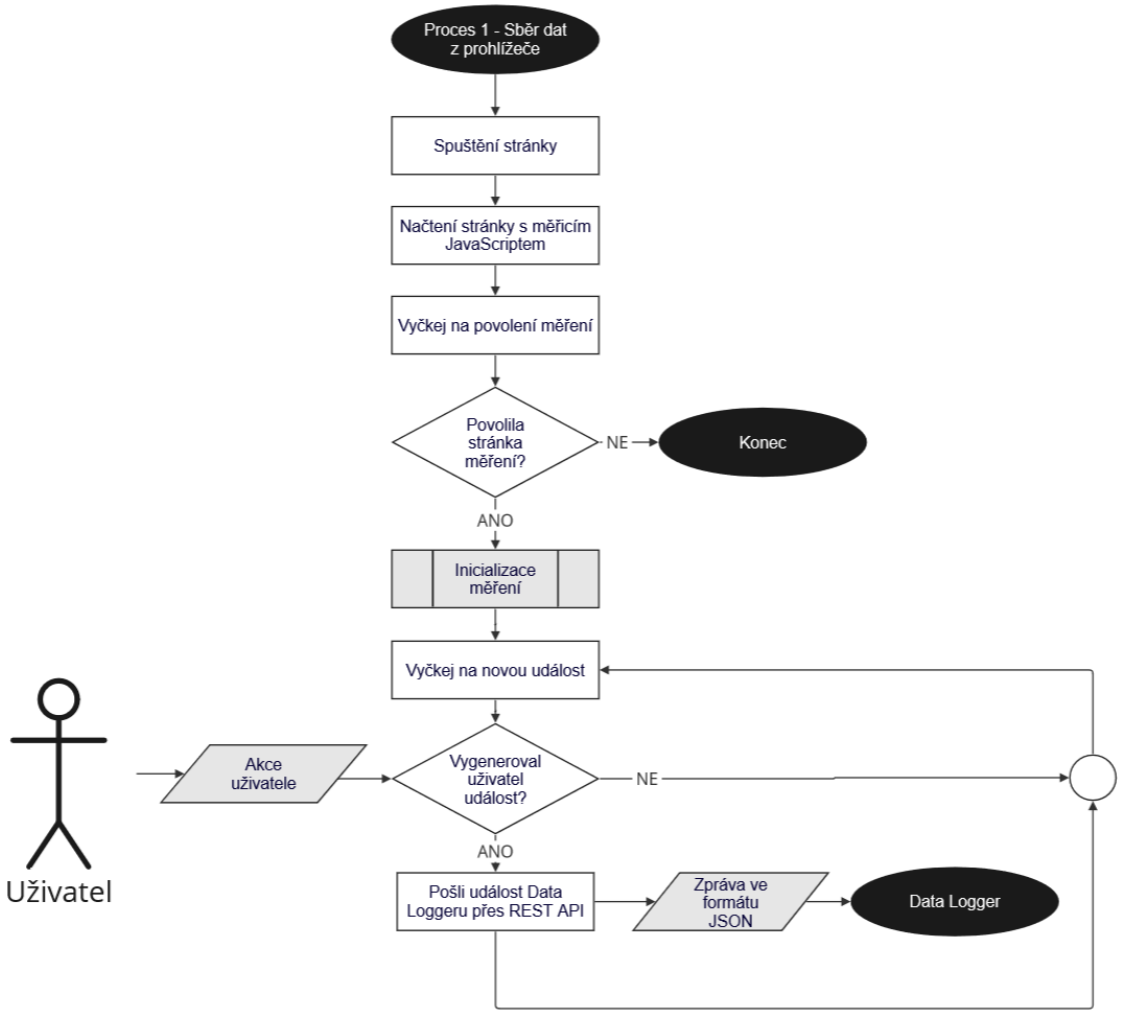
Měření bude prováděno pomocí JavaScript značek s cílem dosáhnout vyšší přesnosti a flexibility, zejména s ohledem na snadné a rychlé nasazení na webové stránky. Měřicí skript bude pracovat podle principů, které byly probrány v kapitole 1.3. Realizován bude jako JavaScript knihovna obsahující základní logiku pro sběr dat. Po aktivaci bude knihovna automaticky zachytávat některé běžné události a parametry, jak uvádí tabulka 4.1. Pro specifické případy bude možné využít připravenou metodu z knihovny.

Tabulka 4.1 Přehled automatických událostí a parametrů

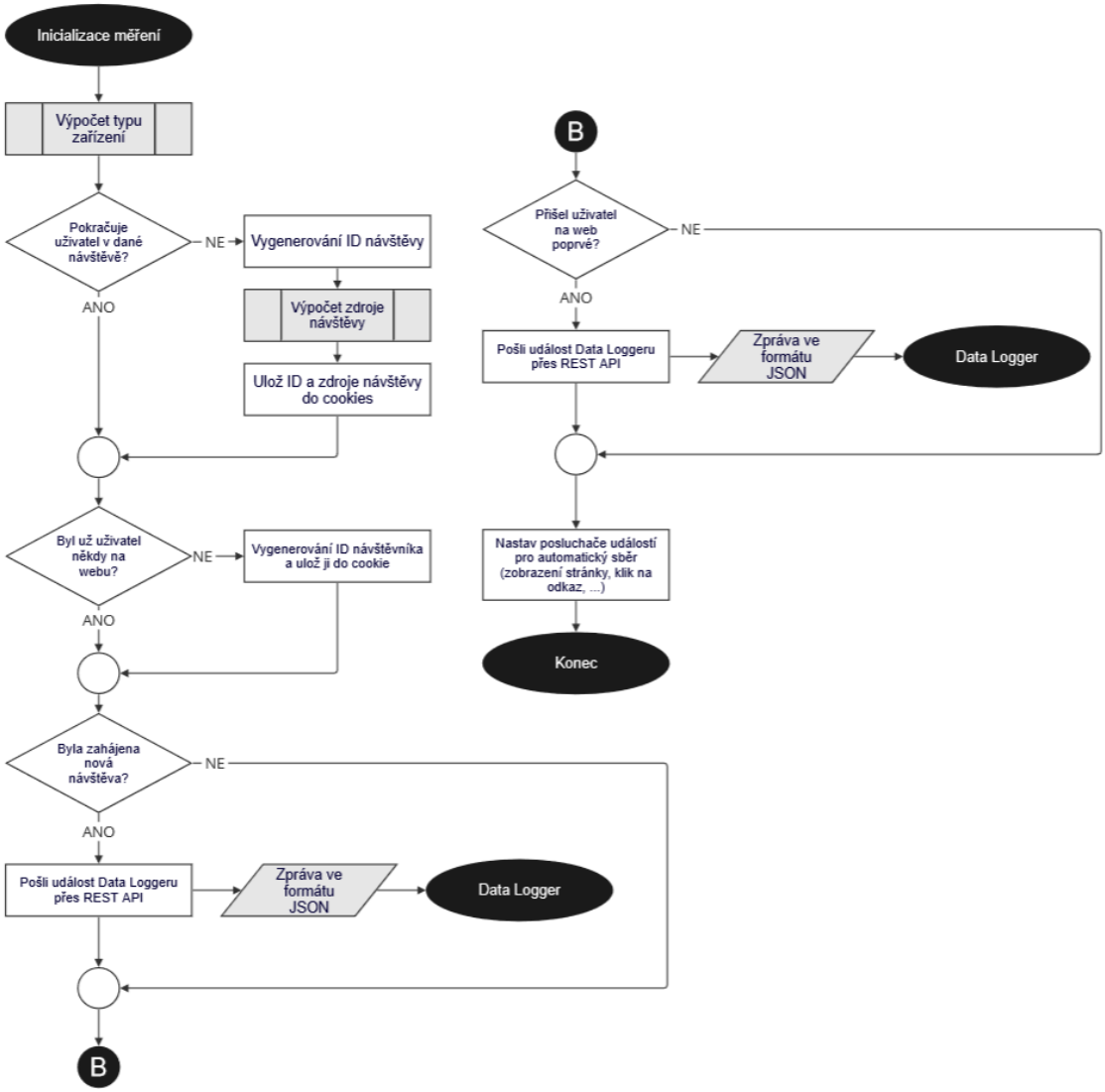
Název	Typ	Popis
new_visitor	událost	návštěvník zahájil první návštěvu na daném zařízení
new_session	událost	návštěvník zahájil novou návštěvu
page_view	událost	zobrazení stránky
scroll	událost	vertikální posun po stránce
link_click	událost	kliknutí na odkaz
visitor_id	parametr	identifikátor návštěvníka
device_type	parametr	typ zařízení
page_domain	parametr	doména webové stránky
page_path	parametr	cesta na webové stránce
page_title	parametr	název stránky
page_referrer	parametr	referrer
session_id	parametr	identifikátor návštěvy
session_source	parametr	zdroj návštěvy
session_medium	parametr	médium návštěvy
session_campaign	parametr	kampaň návštěvy

Proces sběru dat je znázorněn na obrázku 4.1. Zahájí se spuštěním stránky, což může být provedeno různými způsoby, například přímým zadáním adresy uživatelem, kliknutím na odkaz ve vyhledávači nebo kliknutím na odkaz z jiné stránky. Potom nastane načtení stránky s měřicím JavaScript kódem. Vzhledem k tomu, že měření bude ukládat identifikátory pro potřeby analytiky do souborů cookies, tak musí být respektována ochrana osobních údajů uživatele, jak bylo zmíněno kapitole 3.4.

Měření musí být spuštěno až poté, co uživatel udělí souhlas s analytickými cookies prostřednictvím cookie lišty. Integraci cookie lišty a spuštění měření až po odsouhlasení využívání analytických cookies musí zajistit provozovatel webových stránek. V knihovně bude obsažena metoda pro jednoduché spuštění měření. Po spuštění proběhne inicializace měření, která je zobrazena na obrázku 4.2.

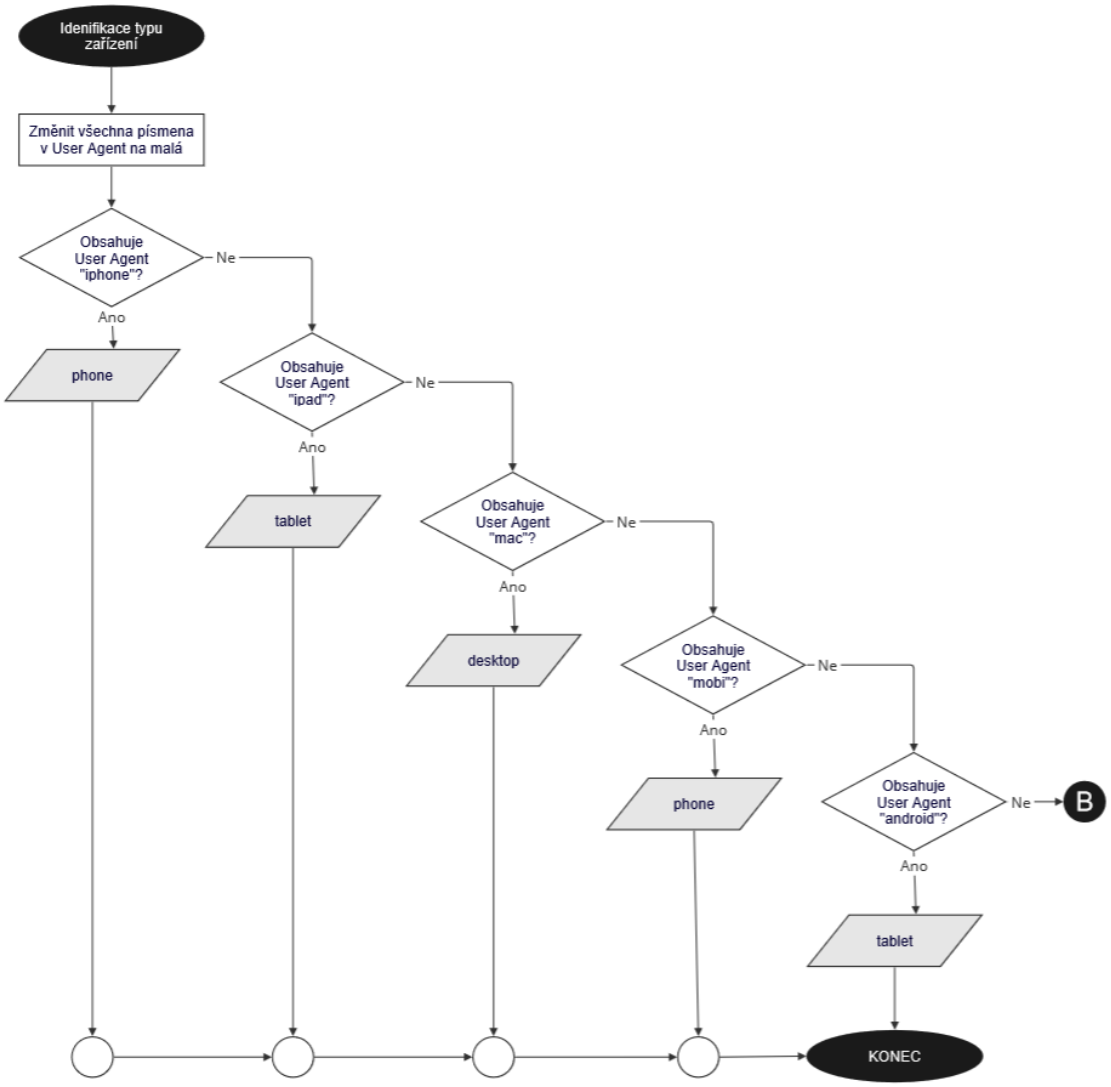


Obrázek 4.1 Proces sběru dat z prohlížeče



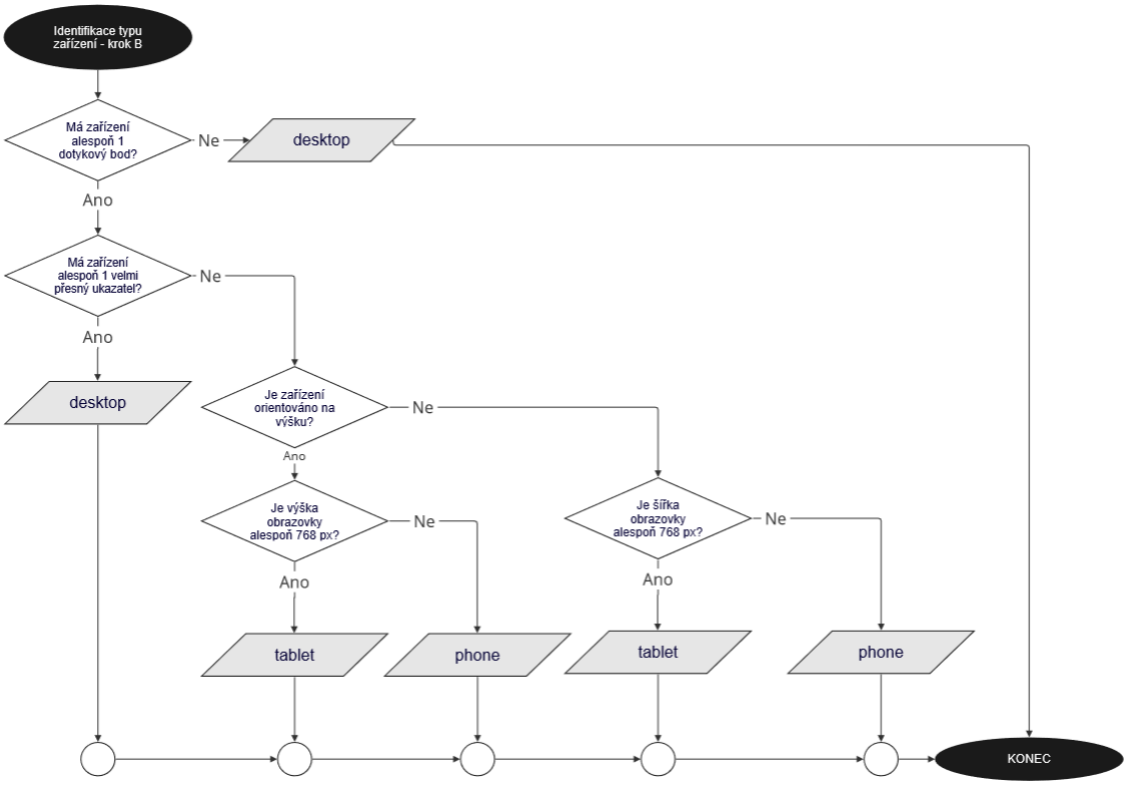
Obrázek 4.2 Inicializace měření

V první fázi inicializace se provádí identifikace typu zařízení. Na základě zdrojů [46, 47, 48] byl navrhnut algoritmus, co je znázorněn na obrázcích 4.3 a 4.4. V prvním kroku se algoritmus pokusí identifikovat zařízení na základě informací z User Agent, což ilustruje obrázek 4.3. User Agent je posílán v každém HTTP požadavku a je to textový řetězec obsahující informace jako je typ a verze prohlížeče nebo operační systém zařízení, které přistupuje na webovou stránku [47].



Obrázek 4.3 Identifikace typu zařízení pomocí User Agent

V dalším kroku naznačeném na obrázku 4.4, probíhá identifikace typu zařízení pomocí Media Query, což je technika používaná při aplikaci CSS stylů tak, aby byl vybrán styl přizpůsobený pro konkrétní zařízení, například s ohledem na jeho rozlišení [48]. Je nutné podotknout, že navržený algoritmus nemusí správně rozlišit všechny existující zařízení, neboť jich existuje obrovské množství. Nicméně nejpoužívanější zařízení by měl identifikovat s vysokou přesností.

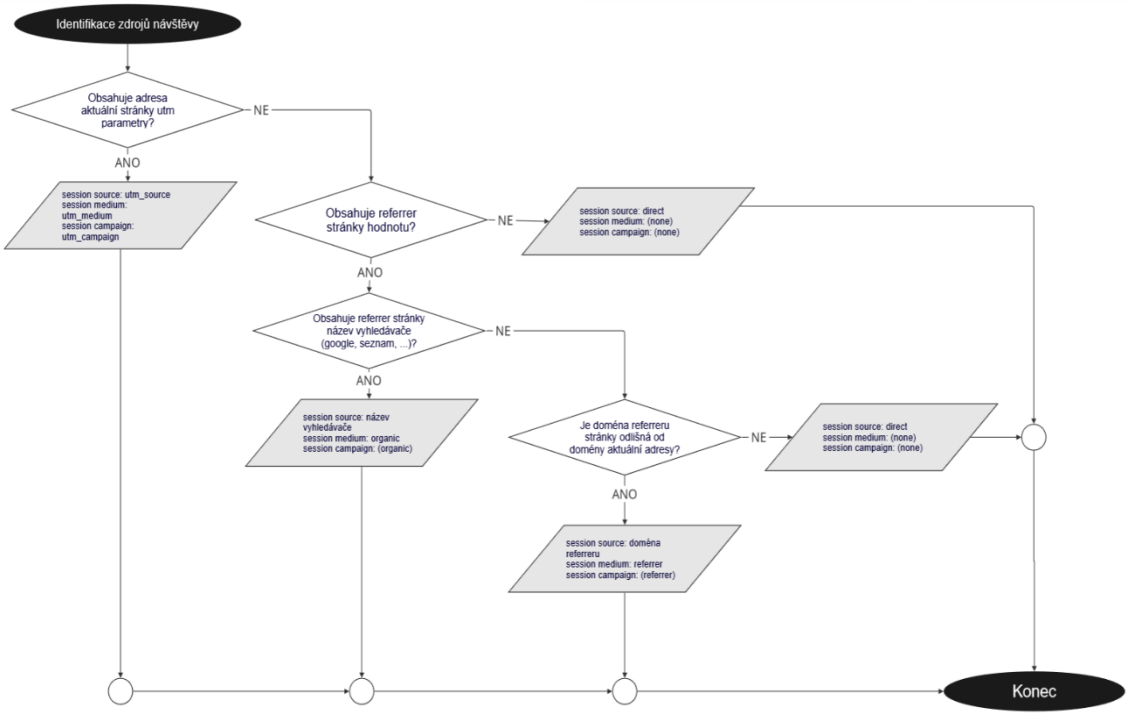


Obrázek 4.4 Identifikace typu zařízení pomocí Media Query

V další fázi inicializace se načte identifikátor o návštěvě ze souborů cookies. Daná cookie bude mít životnost 30 minut, tak jak bylo zmíněno v kapitole 3.1.2 a bude obsahovat aktuální ID návštěvy a její zdroje. Při interakcích bude cookies prodlužována. Tohle zajistí, aby návštěva vypršela až po 30 minutách neaktivity uživatele na stránce. Při vygenerování nové návštěvy proběhne identifikace zdrojů návštěvy podle poznatků z kapitoly 3.1.3. Celý proces je znázorněn na obrázku 4.5. Po zpracování údajů o aktuální návštěvě proběhne čtení identifikátoru návštěvníka z dalšího souboru cookies s životností nastavenou na 90 dní. Podobně jak tomu bylo v předchozím kroku, tak při interakcích bude životnost prodlužována.

Pokud se vygeneruje nový identifikátor uživatele, tak se odešle událost new_visitor. V případě vytvoření nové návštěvy nastane odeslání události new_session. V posledním kroku inicializace proběhne nastavení posluchačů událostí, které budou sloužit pro sběr zbývajících automatických událostí jako je page_view, scroll a dalších.

Po inicializaci skript čeká na nové události, jež jsou vyvolávány prostřednictvím interakcí uživatele se stránkou. Jednotlivá data budou odesílána data loggeru prostřednictvím zpráv ve formátu JSON pomocí REST API. Zvolený formát zpráv a rozhraní byl vybrán s ohledem na jednoduchost a flexibilitu. Většina dostupných prohlížečů podporuje snadné použití REST API s JSON zprávami.



Obrázek 4.5 Identifikace zdrojů návštěvy

4.2 Data Logger

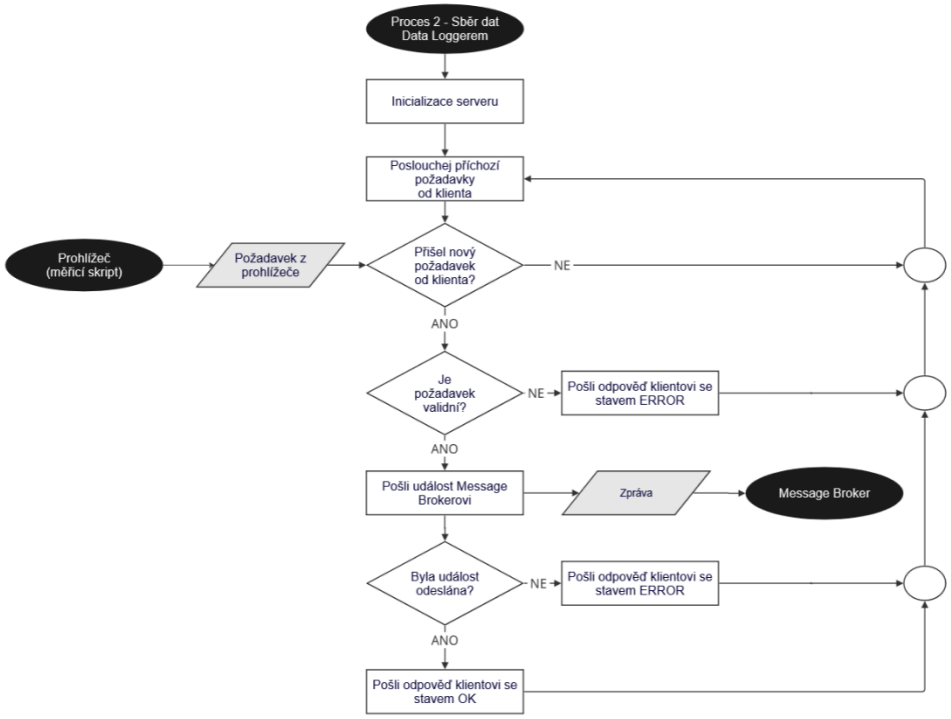
Události generované měřicím skriptem je nutné spolehlivě zachytávat, validovat a předzpracovat. K tomu bude sloužit data logger. Jedná se o webový server, jehož princip fungování ilustruje obrázek 4.6. Po počáteční inicializaci server poslouchá požadavky od klienta. V našem případě se jedná o prohlížeč s měřicím skriptem. Po obdržení JSON zprávy s událostí od měřicího skriptu server provede validaci požadavku.

Pokud má zpráva předpokládaný formát, který je uveden v tabulce 4.2, tak se požadavek odešle message brokerovi, který slouží primárně jako fronta a bude rozebrán v další kapitole.

Server vždy na každý požadavek odpoví klientovi zpátky. V případě, že vše proběhne správně, tak vrátí klientovi zpátky odpověď se stavem Ok. Když nastane v procesu chyba, v jejímž důsledku je zpráva buď vyhodnocena jako neplatná, nebo nebyla úspěšně odeslána do dalšího modulu, tak vrátí odpověď se stavem Error. Klient s odpovědí nijak dále nepracuje. Výhoda zavedení zpětné vazby od serveru má své uplatnění pro monitoring, protože je možné sledovat a vyhodnocovat stavové kódy odpovědí.

Tabulka 4.2 Formát JSON zprávy posílané měřicím skriptem

Parametr	Popis	Typ	Povinný	Příklad
ev_ts	Čas události, v ISO formátu „YYYY-MM-DDTHH:mm:ss.sssZ“	String	Ano	"2011-10-05T14:48:00.000Z"
en_id	Identifikátor názvu události	Integer	Ano	0
lg_id	Identifikátor loggeru	Integer	Ano	0
aw_id	Identifikátor agregačního okna	Integer	Ano	0 (reálný čas)
dims	Pole objektů dimenzí, každý obsahuje:	Array of objects	Ano	[{„id“: 0, „val“: „abc“}]
id	Identifikátor dimenze	Integer	Ano	5
val	Hodnota dimenze	String	Ano	„test.cz“
metrics	Pole objektů metrik, každý obsahuje:	Array of object	Ano	[{„id“: 0, „val“: „abc“, „unit“: 0}]
id	Identifikátor metriky	Integer	Ano	0
val	Hodnota metriky	Numeric	Ano	1
unit	Identifikátor jednotky, který je spojen s metrikou	Integer	Ano	0



Obrázek 4.6 Proces sběru dat pomocí data loggeru

4.2.1 Dostupné technologie

Pro vytvoření data loggeru v cloudu existuje několik typů služeb, které se od sebe odlišují úrovní kontroly uživatele nad prostředky služby. Možnost největší kontroly poskytují služby typu Infrastructure as a Service (IaaS). Tyto služby poskytují zákazníkovi možnost pronájmu prostředků nejčastěji prostřednictvím virtuálního stroje. Uživatel se tedy nemusí zabývat nákupem a zprovozněním IT infrastruktury jako je server, síťové prvky, úložiště a dalších. Výhodou je vysoká kontrola uživatele nad službou a lehká předvídatelnosti ceny, protože se zpravidla platí za dobu pronájmu prostředků. Tohle však může být zároveň i nevýhodou, protože si instalaci, aktualizaci a konfiguraci softwarových prostředků musí uživatel zajistit sám. Ačkoliv IaaS umožňuje vertikální a horizontální škálování, tak tento proces není tak dynamický, jak u dalších typů služeb, což je nevýhodou pro provoz našeho serveru. [49, 50, 51]

Dalším typem služeb je Platform as a Service (PaaS), která nabízí nižší kontrolu nad prostředky. Uživatel nemusí řešit chod hardware ani operačního systému, nad kterými navíc nemá ani kontrolu. PaaS poskytuje například platformu pro hostování aplikací, co si uživatel vytvoří. Nasazení a provoz vlastního serveru je tak značně jednodušší než při využití IaaS. [49, 50]

Speciálním typem služby je Function as a Service (FaaS), která nabízí maximální flexibilitu. Uživatel je zodpovědný pouze za naprogramování funkce. Může si zvolit maximálně programovací jazyk, jeho verzi, velikost paměti a výpočetního výkonu. Nejčastěji se platí za počet spuštění funkce a za dobu provedení funkce. Tohle vede k největší nevýhodě FaaS, a to je vyšší cena ve srovnání s IaaS a PaaS, v případě častého spouštění většího počtu funkcí, k čemuž u serveru pro sběr dat z prohlížečů běžně dochází. [49, 52]

Na základě uvedených vlastností bude server provozován pomocí PaaS. U každého provozovatele byla vybrána nejvhodnější služba pro provoz vlastní aplikace data logger. Srovnání služeb je uvedeno v tabulce 4.3 a bylo čerpáno ze zdrojů [53, 54, 55, 56].

Container Apps vychází jako nejhorší varianta. Přestože uplatňuje podobný cenový model jako Cloud Run, tak jsou všechny položky dražší. Fargate má nejnižší náklady za pronájem CPU a RAM, avšak jsou fixní a bez slevy při neaktivitě. Cloud Run alokuje prostředky pouze, pokud zpracovává požadavky a nic neúčtuje při neaktivitě. Navíc poskytuje měsíčně zdarma všechny položky. Při použití Fargate je nezbytné pro veřejné připojení přes internet využít službu Application Load Balancer (ALB). S tím vznikají další náklady. Provoz ALB se účtuje fixně za hodinu a za každou Load Balancer Capacity Unit (LCU), která je vypočítána na základě 4 metrik. Metriky, které zahrnuje 1 LCU je 25 nových připojení za sekundu, 3000 aktivních připojení za minutu, 1000 vyhodnocených pravidel za sekundu, 1 GB přenesených dat za hodinu. Pro velké aplikace může být výhodnější vybrat Fargate. U menších až středních aplikací se Cloud Run jeví jako nejjednodušší a nejlevnější volbou. [53, 54, 55, 56]

Tabulka 4.3 Srovnání služeb pro provoz data loggeru

	Fargate	Container Apps	Cloud Run
Poskytovatel	AWS	Azure	GCP
Typ služby	PaaS	PaaS	PaaS
Automatické škálování	Ano	Ano	Ano
Škálování na 0	Ano	Ano	Ano
Účtované položky	CPU, RAM, Application Load Balancer (ALB)	CPU, RAM, požadavky	CPU, RAM, požadavky
Vybraný region	Europe (London)	West Europe	europe-west1 (Belgium)
Cena za sekundu provozu 1 virtuálního procesoru (vCPU)	0,0000129 $	Při aktivním využití - 0,0000340 $ (zdarma prvních 180 000 za měsíc) Při neaktivitě – 0,0000040 $	Při aktivním využití - 0,0000240 $ (zdarma prvních 180 000 za měsíc) Při neaktivitě – 0
Cena za sekundu provozu 1 GB operační paměti	0,0000014 $	Při aktivním využití - 0,0000040 $ (zdarma prvních 360 000 za měsíc) Při neaktivitě – 0,0000040 $	Při aktivním využití - 0,0000025 $ (zdarma prvních 360 000 za měsíc) Při neaktivitě – 0
Cena za síť	0,02646 $ / h + 0,0084 $ / h za 1 LCU	0,56 $ za 1 M příchozích požadavků (zdarma 2 M požadavků za měsíc)	0,40 $ za 1 M příchozích požadavků (zdarma 2 M požadavků za měsíc)
Veřejná IP a možnost vlastní domény	Ano (Při použití ALB)	Ano	Ano

4.3 Message broker

Jak bylo zmíněno v předchozí kapitole, tak data logger bude posílat zprávy do dalšího modulu s názvem message broker. Tento modul byla do systému zařazen pro dosažení větší spolehlivosti systému. Z důvodu škálování data loggeru by se při přímém vkládání do databáze mohlo stát, že by se vyčerpal maximální počet možných připojení do databáze. Databáze by pak ve špičkách nemusela nápor zvládnout. Navíc by při neúspěšném navázání spojení s databází docházelo ke ztrátě dat. Proto byl do systému zařazen message broker, který zmíněné problémy vyřeší. Message broker v sobě musí obsahovat frontu zpráv. Do ní se budou vkládat události z data loggeru. Další modul si pak bude události z fronty vybírat a vkládat do databáze.

4.3.1 Dostupné technologie

Message broker bude provozován pomocí integrované platformy daným poskytovatelem z důvodů jako je snadné a rychlé zprovoznění společně se spolehlivým provozem. Nevýhoda je limitace vybranou službou a nemožnost dělat si vlastní úpravy. Uvažované služby se nachází v tabulce 4.4 a bylo čerpáno z [57, 58, 59].

Tabulka 4.4 Srovnání služeb pro provoz message brokera

	SQS	**Service Bus**	**Pub/Sub**
Poskytovatel	AWS	Azure	GCP
Typ služby	PaaS	PaaS	PaaS
Mechanismus zasílání zpráv	Point-to-Point	Point-to-Point Publish/Subscribe	Publish/Subscribe
Podporované API	REST	REST	REST, gRPC
Účtované položky	Počet požadavků, velikost zpracovaných dat	Počet požadavků (pouze při použití mechanismu Point-to-Point)	Velikost zpracovaných dat
Vybraný region	Europe (London)	West Europe	region europe-west1 (Belgium)
Cena za 1 GB příchozích dat do fronty	0,09 $ (zdarma 15 GB za měsíc)	0	0,04 $ (zdarma 10 GB za měsíc)
Cena za 1 GB odchozích dat z fronty	0,09 $ (zdarma 15 GB za měsíc)	0	0,04 $ (zdarma 10 GB za měsíc)
Cena za 1 M operací (vložení, přečtení, smazání)	0,40 $ (zdarma 1 M operací měsíčně)	0,05 $	0

V případě výběru této technologie vychází nejlépe Service Bus a Pub/Sub. Rozdíl v účtování je ten, že Service Bus pracuje s operací, kdežto Pub/Sub účtuje na základě velikosti zpracovaných dat. Pub/Sub opět poskytuje měsíčně zdarma zpracování 10 GB dat. [58, 59]

Pro provoz systému byl po tomto srovnání vybrán poskytovatel GCP hlavně kvůli přítomnosti měsíčních limitů pro bezplatný provoz u velkého počtu služeb, které pokryjí náklady pro prototyp a výrazně zlevní provoz menších aplikací. Všichni poskytovatelé nabízí širokou škálu služeb, z nichž některé jsou si navzájem, jak tomu je například u Service Bus a Pub/Sub či Container Apps a Cloud Run. Proto klíčovým kritériem pro výběr poskytovatele je dnes zejména cena poskytovaných služeb.

Pub/Sub navíc podporuje velmi rychlý a efektivní přenos dat přes rozhraní gRPC. Popis fungování message brokera je znázorněn na obrázku 4.7. Pub/Sub vyžaduje po konzumentovi potvrzení přijmutí zprávy. Potvrzení může být pozitivní, nebo negativní, které může být využito, pokud konzument nebyl schopný zprávu z jakéhokoliv důvodu vložit do databáze. Služba pak pošle zprávu znovu. [59]

Schéma zprávy se určuje prostřednictvím Protobuf verze 3 (proto3). Byla vytvořena následující definice zprávy:

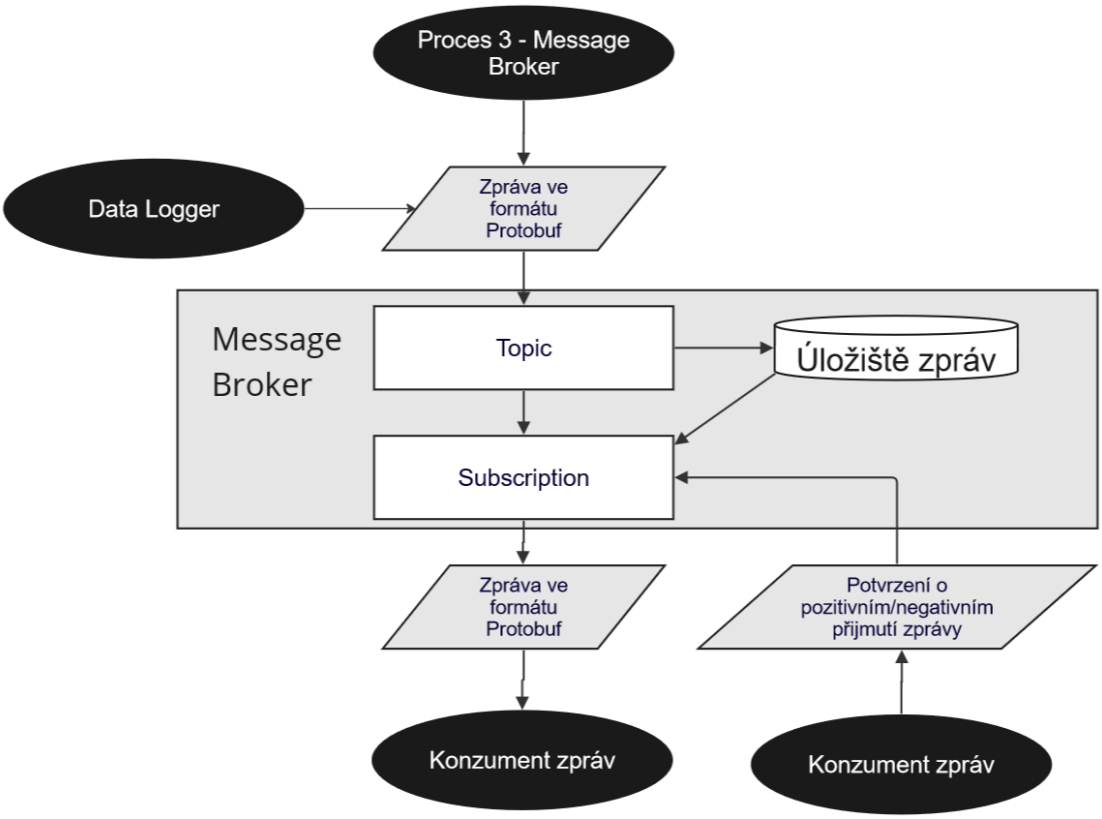
```
syntax = "proto3";

message Event {
	message Dimension {
		int64 id = 1;
		string val = 2;
	}

	message Metric {
		int64 id = 1;
		string val = 2;
		int64 unit = 3;
	}

	string ev_ts = 1;
	int64 ev_id = 2;
	int64 lg_id = 3;
	int64 aw_id = 4;
	repeated Dimension dimensions = 5;
	repeated Metric metrics = 6;
}
```

Schéma definuje zprávu Event, která slouží k přenosu naměřených dat z data loggeru. Schéma zahrnuje nejen jednoduché položky jako je čas události (ev_ts), identifikátoru názvu události (ev_id), identifikátoru loggeru (lg_id) a identifikátoru agregačního okna (aw_id), ale také objekty Dimension a Metric pro reprezentaci pole objektů dimenzí a metrik. Pro možnost přenosu pole objektů musí být použito klíčové slovo „repeated“.



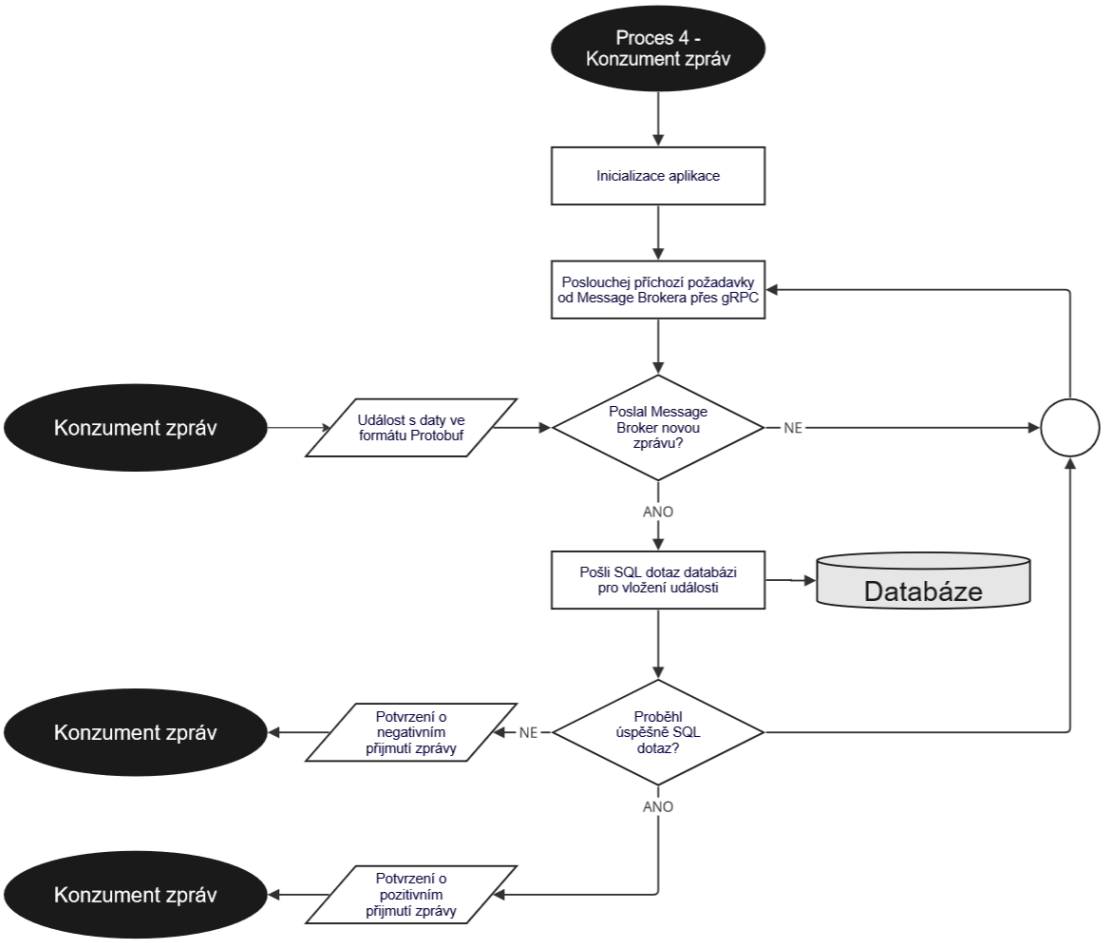
Obrázek 4.7 Proces message brokeru

4.4 Konzument zpráv

Konzument odebírá zprávy z message brokera. Obsah zpráv pak následně uloží do databáze ve formě událostí. Pro realizaci přichází v úvahu služby typu FaaS a IaaS. Nicméně spouštět funkci pro každou přijatou zpráv není příliš efektivní. Jak již bylo zmíněno, tak zpráv může být vygenerován velký počet. Při spuštění většího množství funkcí by mohlo dojít k podobnému problému, jak bylo zmíněno u data loggeru, a to zahlcení databáze velkým počtem připojení.

Proto byla vybrána služba typu IaaS. To znamená, že bude pronajat virtuální stroj prostřednictvím služby Compute Engine, na kterém bude provozována jednoduchá aplikace. Cena za pronájem nejmenšího virtuálního stroje e2-micro vychází fixně zhruba za 7 $ za měsíc, což je přijatelná cena a měl by rozhodně stačit pro malé aplikace. [60]

Aplikace bude odebírat zprávy z message brokera. Pokud obdrží zprávu, tak vezme její obsah a vloží jej do databáze. Pokud vše proběhne bez chyby, tak aplikace pošle potvrzení o pozitivním přijmutí zprávy. Při chybě pak pošle negativní potvrzení, takže se message broker pokusí zprávu odeslat v dalším pokusu. Chyba může vzniknout například při výpadku databáze nebo nemožnost navázání spojení. Celý proces aplikace je znázorněn na obrázku 4.8.



Obrázek 4.8 Proces konzumenta zpráv

4.5 Databáze

V dalším fázi návrhu byla vybrána služba a databáze pro ukládání dat. Byla rozhodnuto, že bude použita relační databáze hlavně kvůli možnosti ukládat data strukturovaně, a navíc efektivně při použití normalizace. Relační databáze také zajišťují integritu dat i v případě možných chyb. Vzhledem k tomu, že jsou sbírána data z prohlížeče, tak k chybám může pravidelně docházet, a proto je poslední vlastnost klíčová.

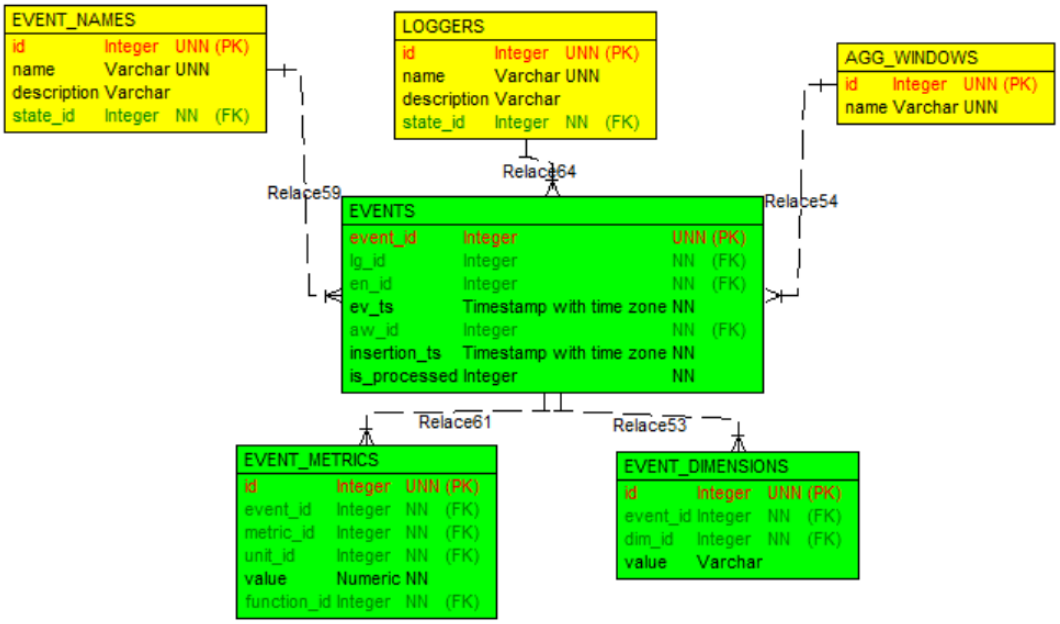
GCP nabízí službu typu PaaS s názvem Cloud SQL pro jednoduchý hosting databází MySQL, SQL Server a PostgreSQL. SQL Server nepřichází v úvahu kvůli tomu, že se nejedná o open-source databázi a je nutno vlastnit licenci od Microsoftu, což by způsobilo dražší provoz oproti ostatním možnostem. Zbylé databáze jsou open-source. PostgreSQL nabízí mnohem více funkcionalit, a proto byla upřednostněna před MySQL. Pronájem databáze začíná zhruba od 12 $ za měsíc. Výhodou služby je, že lze měnit parametry jako je výkon, paměť a úložiště bez ztráty dat, a to v rámci minut. Služba také zajišťuje automatické zálohování, zabezpečení a pravidelné aktualizace. [61, 62]

4.6 Datový model

Další fáze návrhu zahrnuje vytvoření datového modelu. Celý model je znázorněn v příloze A. Zeleně zvýrazněné entity slouží pro uchování informací o událostech. Žlutě jsou zvýrazněny číselníky. Oranžově jsou odlišeny pomocné tabulky. V rámci této kapitoly budou představeny klíčové prvky modelu.

4.6.1 Submodel událostí

Submodel událostí, znázorněný na obrázku 4.9, je klíčovou součástí modelu. Byl navrhnut tak, aby byly splněny požadavky pro efektivní ukládání událostí společně s metrikami a dimenzemi. První entita s názvem EVENTS obsahuje záznamy událostí. Primárním klíčem je event_id, což je unikátní ID záznamu. Tenhle atribut slouží pro spojení entit EVENT_METRICS a EVENT_DIMENSIONS, kde jsou uloženy metriky a dimenze. Cizí klíče entity EVENTS pak odkazují do číselníků pro standardizaci názvů události (viz tabulka 4.5), loggeru a agregačního okna. Číselník LOGGERS může sloužit pro odlišení různých webů nebo společností. Atribut is_processed slouží pro rozlišení surových a agregovaných záznamů. Atribut insertion_ts se plní časem, kdy byla událost vložena do databáze, ev_ts obsahuje čas vygenerování události v prohlížeči. Časové atributy berou v potaz navíc časové zóny.



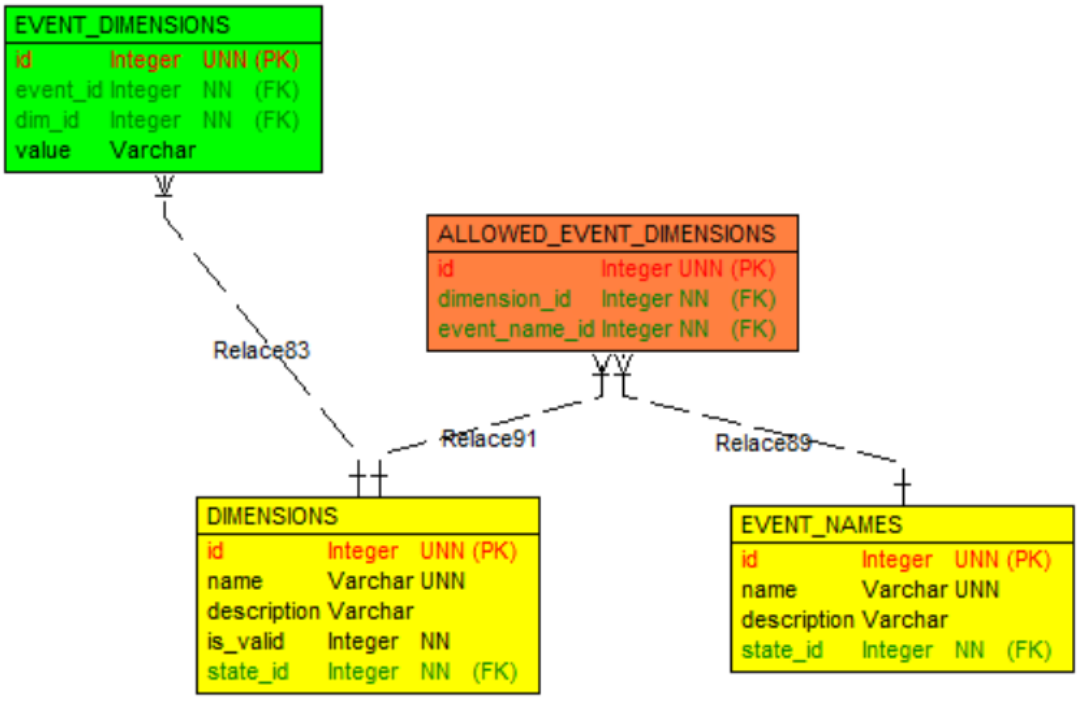
Obrázek 4.9 Submodel událostí

Tabulka 4.5 Číselník základních událostí

ID	Název	Automatický	Popis
0	new_visitor	ano	návštěvník zahájil první návštěvu
1	new_session	ano	návštěvník zahájil novou návštěvu
2	page_view	ano	zobrazení stránky
3	scroll	ano	vertikální posun po stránce
4	link_click	ano	kliknutí na odkaz
5	purchase_item	ne	nákup
6	view_item_detail	ne	zobrazení detailu položky
7	click_item	ne	klik na položku
8	add_item_to_cart	ne	přidat položku do košíku

4.6.2 Submodel dimenzí

Submodel dimenzí, znázorněný na obrázku 4.10, je navázán na submodel událostí a rozšiřuje specifikaci dimenzí. Číselník DIMENSIONS obsahuje seznam dimenzí. Základní dimenze se nachází v tabulce 4.6. Číselník EVENT_NAMES byl představen již v předchozím submodelu. Mezi číselníky existuje vazební tabulka ALLOWED_EVENT_DIMENSIONS obsahující vztahy mezi názvem události a dimenzemi. Například dimenze item_id nemá smysl pro událost scroll. Má smysl například pro události view_item_detail, purchase_item a add_item_to_cart.



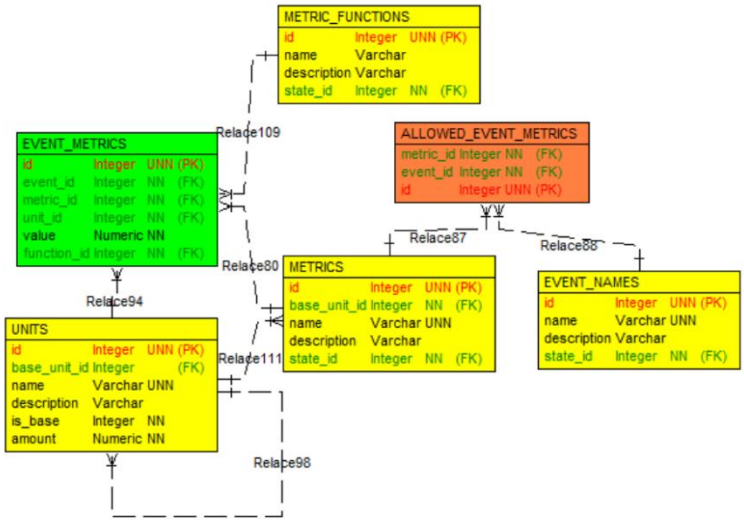
Obrázek 4.10 Submodel dimenzí

Tabulka 4.6 Číselník základních dimenzí

ID	Název dimenze	Automatický	Popis	Příklad
0	visitor_id	ano	identifikátor návštěvníka	12345.67890
1	device_type	ano	typ zařízení	desktop
2	page_domain	ano	doména webové stránky	example.com
3	page_path	ano	cesta na webové stránce	/blog/clanek
4	page_title	ano	název stránky	Blog
5	page_referrer	no	referrer	https://web.cz
6	session_id	ano	identifikátor návštěvy	67890213
7	session_source	ano	zdroj návštěvy	google
8	session_medium	ano	médium návštěvy	cpc
9	session_campaign	ano	kampaň návštěvy	letnivyprodej
10	link_url	ne	URL odkazu	https://web.cz
11	item_id	ne	identifikátor položky	0
12	item_name	ne	název položky	produktA

4.6.3 Submodel metrik

Submodel metrik, znázorněný na obrázku 4.11, je navázán na submodel událostí a rozšiřuje specifikaci metrik. Číselník METRICS obsahuje seznam metrik. Základní metriky se nachází v tabulce 4.7. Každá metrika je svázána s funkcí, která se využívá při agregaci dat. Aktuálně systém pracuje se sumou, maximem a minimem. Mezi číselníky existuje vazební tabulka ALLOWED_EVENT_METRICS obsahující vztahy mezi názvem události a metrikami. Například metrika quantity nemá smysl například pro událost page_view, ale má smysl třeba u událostí purchase, view_item_detail, add_item_to_cart a podobných.



Obrázek 4.11 Submodel metrik

Tabulka 4.7 Číselník základních metrik

ID	Název metriky	Povinný	Popis
0	count	ano	počet událostí
1	value	ne	hodnota/cena
2	quantity	ne	počet položek
3	position	ne	pozice
4	percent	ne	počet procent

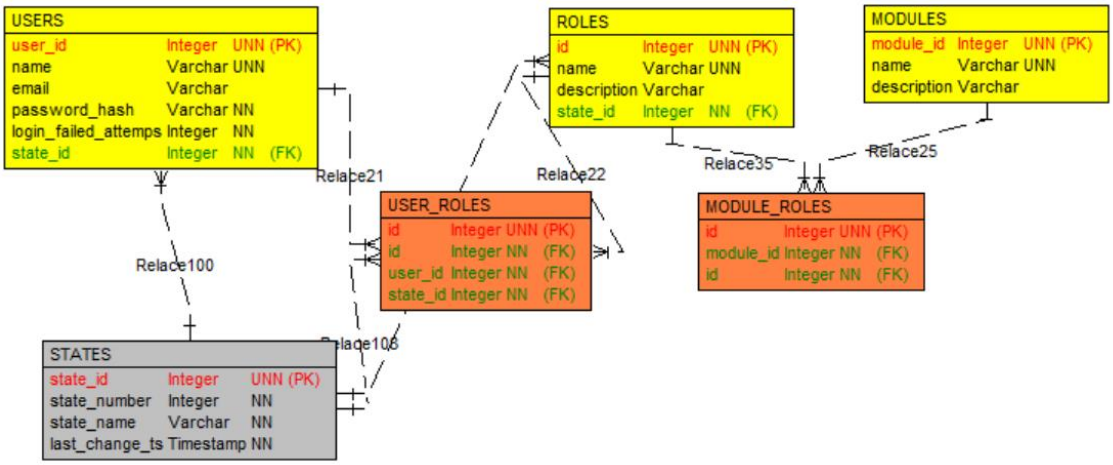
Jak již bylo zmíněno, tak metrika je spojena s jednotkou. Cizí klíč base_unit_id odkazuje v rámci stejné tabulky na ID základní jednotky. U základních jednotek nemá atribut base_unit_id žádnou hodnotu. Atribut amount obsahuje převodní vztah mezi odvozenou a základní jednotkou. Seznam základních jednotek je uveden v tabulce 4.8.

Tabulka 4.8 Číselník základních jednotek

ID	Kód jednotky	Je jednotka základní	Přepočet k základní jednotce	Popis
0	-	ano	1	bezrozměrná veličina
1	%	ano	1	procento
2	CZK	ano	1	česká koruna
3	EUR	ne	24	euro

4.6.4 Submodel uživatelů a rolí

Submodel dimenzí, znázorněný na obrázku 4.12, slouží pro správu uživatelů a jejich rolí. Tabulka STATES slouží pro flexibilní uchovávání stavů záznamů. Stavy lze navázat s jakoukoliv tabulkou datového modelu. Číselník ROLES obsahuje seznam rolí. V informačním systému budou použity 2 základní role – administrátor a uživatel, co si může zobrazit přehledy. Číselník MODULES zahrnuje seznam modulů. Budou použity 2 moduly, a to přehledy se statistikami a administrace, kde budou mít přístup administrátoři. Vazební tabulka USER_ROLES a obsahuje vztahy mezi uživateli a jejich rolemi. Vazební tabulka MODULE_ROLES obsahuje vztahy mezi rolemi a moduly.



Obrázek 4.12 Submodel uživatelů a rolí

4.7 Informační systém

Informační systém bude sloužit k zobrazení statistik uživateli a k administraci. Bude realizován jako webová aplikace, kde klientská část aplikace (frontend) slouží pro interakci uživatele a zobrazení výstupu v prohlížeči, na základě přijatých dat ze serverové části aplikace (backend), který obsahuje logiku aplikace a se stará mimo jiné o komunikaci s databází. Backend bude provozován prostřednictvím služby Compute Engine. Cena za pronájem nejmenšího virtuálního stroje e2-micro vychází fixně zhruba na 7 $ za měsíc, jak již bylo zmíněno. [60]

4.8 Infrastruktura systému

Na základě získaných poznatků byla navrhnuta infrastruktura systému, která bude provozována v GCP z důvodů uvedených v kapitole 4.3.1. Blokové schéma systému je uvedeno na obrázku 4.13. Vstup do systému bude generovat uživatel interagující se stránkou, co v sobě obsahuje měřicí skript, jenž posílá informace na data logger přes REST API. Data logger provede validaci a předzpracování událostí. Následně pak událost pošle message brokerovi přes gRPC API. Konzument zpráv je pak přihlášený k odběru zpráv také pomocí gRPC API. Při úspěšném vložení zprávy do databáze, pak odešle pozitivní potvrzení o přijmutí zprávy. Zpráva se tak odstraní zprávu z fronty message brokera. Při neúspěšném vložení zprávy do databáze, pak vrátí negativní potvrzení o přijmutí zprávy. Message broker tak pošle po uplynutí časového limitu zprávu znovu.

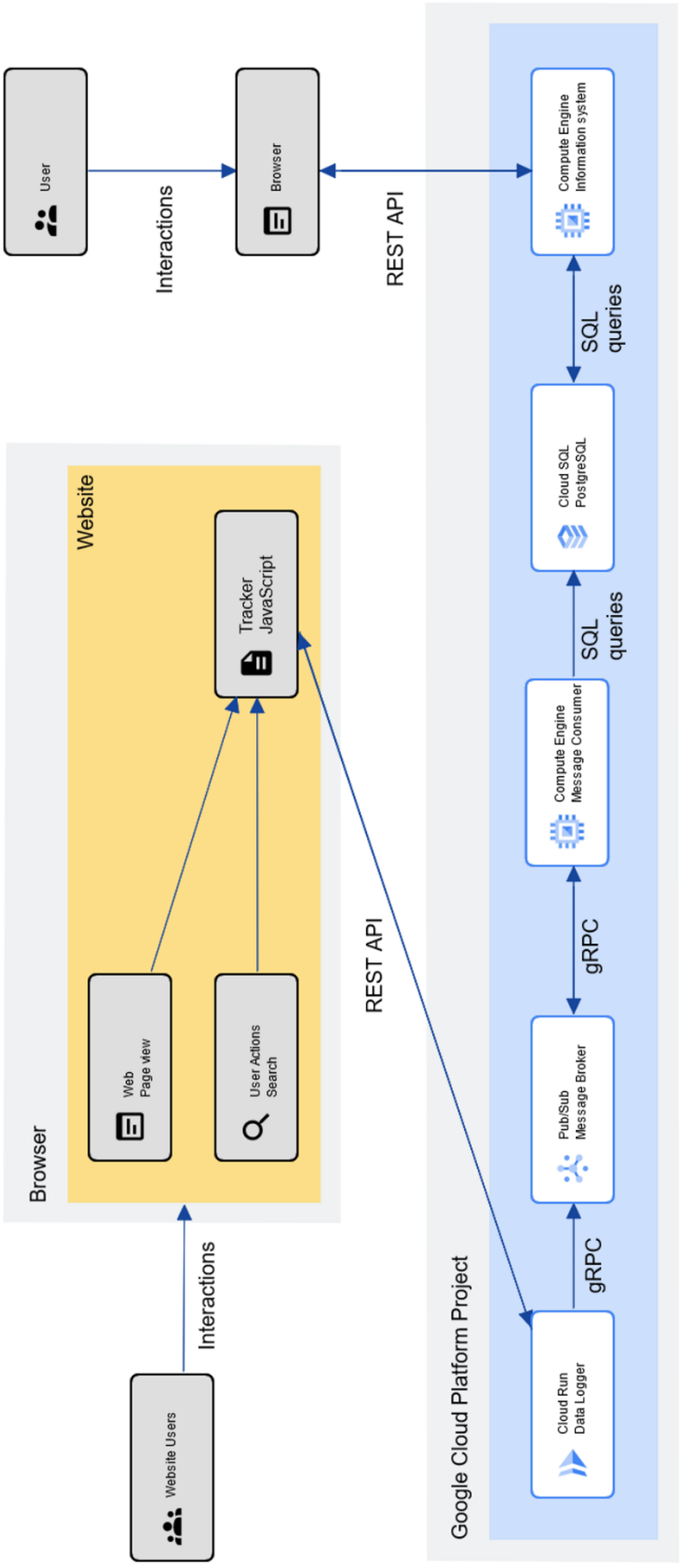
Uživatel systému pak bude mít data k dispozici prostřednictvím informačního systému, což bude webová aplikace. Informační systém bude ukládat a pracovat s daty uložených v relační databázi. V kapitole 4.6 byl navrhnut datový model, který umožňuje řídit přístup uživatelů pomocí přihlášení a přidělení rolí. Datový model také efektivně a flexibilně ukládá nasbírané události.

Tabulka 4.9 obsahuje přehled prvků systému společně s odhadovanými náklady za měsíc a vychází z předchozích podkapitol. Fixní náklady pro minimální nasazení byly stanoveny na necelých 26 $ měsíčně, z toho 12 $ za databázi, 7 $ za konzumenta zpráv a 7 $ za provoz backendu informačního systému.

Variabilní náklady, co se odvíjejí od počtu zpracovaných událostí, jsou pro Pub/Sub zhruba 0,8 $ za 1 M zpráv o velikosti 10 kB, avšak měsíční limit 1 M zpráv je pokryt bezplatně. Náklady za provoz data loggeru není možné určit přesně. Pokud by trvalo zpracování zprávy 100 ms, pak by cena za zpracování 1 M zpráv vycházela na 2,65 $, avšak každá instance je schopna zpracovávat vícero zpráv současně. Odhad tedy počítá s nejhorším scénářem. Navíc vychází, že bezplatný měsíční limit pokryje asi 1,8 M zpráv. Dále se pak některé náklady mohou zvednout při použití výkonnějších komponent.

Tabulka 4.9 Přehled prvků systému a odhadované náklady

	Služba	**Typ služby**	**Fixní náklady za měsíc**	**Variabilní náklady**
Měřicí skript	(prohlížeč)	-	-	-
Data logger	Cloud Run	PaaS	-	~ 2,65 $ za 1 M zpráv, pokud trvá zprávu zpracovat 100 ms (zdarma asi 1,8 M zpráv měsíčně)
Message broker	Pub/Sub	PaaS	-	~ 0,8 $ za 1 M zpráv o velikosti 10 kB (zdarma asi 1 M zpráv měsíčně)
Konzument zpráv	Compute Engine	IaaS	od 7 $	Vyšší náklady při použití výkonnějšího stroje
Databáze	Cloud SQL	PaaS	od 12 $	Vyšší náklady při použití výkonnějšího stroje
Informační systém (FE)	(prohlížeč)	-	-	-
Informační systém (BE)	Compute Engine	IaaS	od 7 $	Vyšší náklady při použití výkonnějšího stroje



Obrázek 4.13 Blokové schéma systému

5. REALIZACE SYSTÉMU

Po dokončení návrhu byl systém realizován. Pro zefektivnění vývoje a snadnou správu systému byly využity technologie Terraform, Docker a Git. Terraform slouží pro definování infrastruktury pomocí kódu, čímž lze infrastruktura systému verzovat. Technologie tedy usnadní instalaci a správu systému. [63, 64, 65]

Docker byl použit pro nasazení aplikací do GCP. Aplikace jsou pomocí Dockeru zabaleny do tzv. kontejnerů, které lze spustit na jakémkoliv systému s nainstalovaným Dockerem. Aplikace pak běží vždy v izolovaném prostředí, což umožní snadno a bezpečně provozovat aplikace nezávisle na zvolené platformě. Git byl použit pro verzování veškerého kódu, což umožní sledování historie změn. [63, 64, 65]

Pro snadnou údržbu a lepší přehlednost byla definována také struktura projektu. Každá část byla pojmenována jako modul a je umístněná ve vlastní složce. Ta obsahuje vlastní kód modulu, případně Dockerfile a soubory Terraformu s příponou .tf pro zajištění potřebné infrastruktury. Na základě návrhu z přechozí kapitoly byly implementovány následující moduly:

- Měřicí skript (Tracker)
- Data logger
- Message broker
- Konzument zpráv
- Databáze (PostgreSQL)
- Informační systém

5.1 Měřicí skript

Na základě návrhu z kapitoly 4.1 byla vytvořena JavaScript knihovna obsahující třídu Tracker, která slouží pro sběr a odesílání dat z webu na sběrný server. Knihovna pracuje pouze s čistým JavaScriptem a běží ve webovém prohlížeči na straně klienta. Pro použití Trackeru na webové stránce je nutné tuto stránku označkovat pomocí přiloženého kódu v příloze práce. Tímto se vytvoří instance TrackerDP s dvěma veřejnými metodami, a to:

- **enable**(tracking url, logger_id=1)
- **push**(en_id, dims, metrics)

Metoda enable aktivuje základní měření a používá parametr tracking_url, kam se vloží adresa data loggeru a parametr logger_id, který má výchozí hodnotu 1 a slouží pro případné rozlišení událostí, například pro rozlišení měření z 2 různých domén. Metoda enable by měla být spuštěna pouze, pokud dal uživatel souhlas se sběrem dat za účelem zpracování analytickými nástroji.

Metoda push slouží pro rozšíření měření o libovolné další události, které se neměří automaticky. Zavoláním metody se odešle příslušná událost na data logger. Parametr en_id obsahuje id události, dims pak objekt s dimenzemi a metrics objekt s metrikami, co

přísluší k dané události. Metoda může být navěšená například na událost přidání do košíku a volání by pro odeslání dimenzí item_id a item_name s metrikami value a quantity by vypadala následujícím způsobem:

```
TrackerDP.push(
      9,
      {"12": {"val": product. id}, "13": {"val": product.name}},
      {
            "2": {"val": product.value, "unit": 3},
            "3": {"val": product.quantity, "unit": 1}
      }
);
```

5.2 Data logger

Data logger byl implementován v programovacím jazyce Python s využitím frameworku Flask pro vývoj webových aplikací [66]. Tato kombinace byla zvolena s ohledem na rychlé nasazení první verze aplikace. Data logger přijímá události od Trackeru, který události posílá na endpoint ve tvaru https://doména_loggeru/events/collect/v1 pomocí POST metody. Použití protokolu HTTPS zajišťuje bezpečný přenos událostí.

Při obdržení události provede aplikace validaci požadavku, což zahrnuje kontrolu formátu těla zprávy, ověření existence alespoň jedné validní události v těle zprávy, kontrolu jedinečnosti dimenzí a metrik v rámci události. V případě nesplnění validace server odpoví chybovým kódem.

Pro zajištění deterministického chování a snadného nasazení nezávisle na zvolené platformě byla aplikace zapouzdřena pomocí Dockeru. Následující Dockerfile definuje prostředí pro běh data loggeru:

```
FROM python:3.10-slim-buster
COPY requirements.txt .
RUN pip install --trusted-host pypi.python.org -r ./requirements.txt
COPY . .
EXPOSE 8080
CMD ["gunicorn", "-b", "0.0.0.0:8080", "app:app"]
```

5.3 Message Broker

Realizace modulu message broker byla snadná díky výběru služby Pub/Sub [59]. Pro připravení infrastruktury bylo potřeba pouze napsat definice do souboru modules/message-broker/main.tf, který Terraform využije pro vytvoření infrastruktury. Tento soubor obsahuje společně se souborem modules/message-broker/variables.tf, ve kterém jsou proměnné, kód pro nasazení topicu, který přijímá zprávy podle definovaného schématu popsaného v kapitole 4.3.

```
resource "google_pubsub_topic" "data-logger-events" {
      project = var.project_id
      name = "data-logger-events"
      depends_on = [google_pubsub_schema.data-logger-events]
      schema_settings {
      schema =
"projects/${var.project_id}/schemas/${google_pubsub_schema.data-
```

```
logger-events.name}"
            encoding = "JSON"}}
```

Následně byla vytvořena definice pro subscription typu pull, která funguje jako fronta. Bylo nastaveno, aby se zprávy po 30 minutách smazaly, pokud nedošlo k jejich přečtení, což pomáhá předejít případnému zahlcení, například při výpadku databáze. Dále byl nastaven časový limit 20 s, po kterém, pokud odeslaná zpráva nebyla potvrzena příjemcem, dojde k jejímu opětovnému odeslání.

```
resource "google_pubsub_subscription"
 "data-logger-events--pull--realtime" {
      project = var.project_id
      name = "data-logger-events--pull--realtime"
      topic = google_pubsub_topic.data-logger-events.name
      message_retention_duration = "1800s" # 30 minutes
      ack_deadline_seconds = 20
      enable_message_ordering = false
}
```

5.4 Konzument zpráv

Konzument zpráv byl implementován pomocí jazyka Python s využitím knihovny google.cloud.pubsub_v1 pro odběr dat z pull subscription message brokera prostřednictvím gRPC. Přijímání a potvrzování jednotlivých zpráv probíhá asynchronně. Přijaté zprávy se po jedné vkládají do databáze SQL dotazy prostřednictvím knihovny psycopg2 [67].

Pro vkládání událostí do databáze byla implementována přímo v databázi funkce insert_event_data v rámci modulu databáze. Tuto funkci volá konzument zpráv, což umožňuje vkládání jednotlivých událostí do tabulek event_stats, event_dimensions a event_metrics. V případě výskytu chyby se data nezapíšou do žádné tabulky a událost se zahodí. Současně se chyba zaznamená do tabulky errors pro případnou diagnostiku.

Aby nedošlo k vytvoření nadměrného množství spojení, které by mohlo zahltit databázi, tak byl použit tzv. connection pool. Aplikace si tak vytvoří předem stanovený počet připojení s databází, který je udržován i pro budoucí požadavky. Pro tento účel byla využita třída ThreadedConnectionPool, co je zahrnuta v knihovně psycopg2 [67].

5.5 Databáze

Definice infrastruktury pro běh modulu databáze je pomocí Terraform jednoduchá, a to díky možnosti specifikace požadovaných parametrů databáze. Vzhledem k nízké ceně byl použit méně výkonný stroj db-f1-micro, který lze v případě potřeby změnit na výkonnější typ. Pro databázi byla zvolena verze PostgreSQL 15. Jako úložiště byl zvolen persistentní SSD disk s počáteční velikostí 10 GB. Pro zajištění spolehlivého provozu bylo zapnuto automatické navýšení velikosti disku až na 100 GB s krokem 10 GB pomocí parametrů disk_autoresize_limit a disk_autoresize. Důležitým krokem bylo také zapnutí příznaku cloudsql.enable_pg_cron na hodnotu „On“, což umožňuje využít plánovač úloh pg_cron, který byl použit pro automatické spouštění funkce process_event_data, jež agreguje data

po minutách, hodinách, dnech a měsících. Agregovaná data jsou ukládána do stejných tabulek jako neagregovaná data.

```
resource "google_sql_database_instance" "main" {
      project = var.project_id
      region = var.region
      name = "main-instance"
      database_version = "POSTGRES_15"

      settings {
            tier = "db-f1-micro" # 0.6 GB RAM, 1 CPU ~ 8 USD / mo
            edition = "ENTERPRISE"
            availability_type = "ZONAL"

            disk_autoresize = true
            disk_autoresize_limit = 100
            disk_size = 10
            disk_type = "PD_SSD"
            database_flags {
                  name = "cloudsql.enable_pg_cron"
                  value = "On"
            }
      }
}
```

Dále následuje vytvoření tabulek a nastavení počátečních hodnot. Pro ukázku je uvedena inicializace tabulky event_names. Nejdříve se tabulka vymaže, pokud již existovala, a pak se vytvoří. Následně se do ní pomocí cyklu přidávají výchozí události s počátečním stavem. Podobný postup se použije i pro vytvoření ostatních tabulek.

```
DROP TABLE IF EXISTS analytics.event_names CASCADE;
CREATE TABLE analytics.event_names (
    id serial PRIMARY KEY,
    name varchar not null,
    description varchar,
    state_id int REFERENCES analytics.states(id)
);

DO $$
DECLARE
    name text;
    state_id int;
BEGIN
    FOR name IN
      SELECT UNNEST(ARRAY[
              'new_visitor', 'new_session', 'page_view',
              'scroll_to_end', 'link_click', 'purchase',
              'view_item_detail', 'purchase_item', 'add_item_to_cart'])
    LOOP
      -- Insert state
      INSERT INTO analytics.states (state_number, state_name)
      VALUES (1, 'ACTIVE')
      RETURNING id INTO state_id;

            -- Insert event name
      INSERT INTO analytics.event_names (name, description, state_id)
      VALUES (name, '', state_id);
    END LOOP;
END $$;
```

5.6 Informační systém

Pro tvorbu backendu informačního systému byl opět využit programovací jazyk Python společně s frameworkem Flask [69]. Jelikož je Flask velmi odlehčený, tak poskytuje uživateli velkou flexibilitu pro výběr dalších rozšíření.

Pro ukládání dat byla využita databáze PostgreSQL, která již byla vytvořena v rámci modulu Database. V ní budou nejen nasbíraná a zpracovaná data, ale také stavy a konfigurace systému. K usnadnění práce s databází bylo využito rozšíření pro objektově relační mapování s názvem SQLAlchemy, které zjednodušuje interakci Flasku s databází, protože umožňuje v kódu aplikace pracovat s objekty místo SQL dotazů.

Dále byl využit Jinja2 [70], což je šablonovací systém umožňující vkládání dynamických dat z Flasku do HTML šablon. Vygenerované šablony jsou následně odeslány z backendu do frontendu. Frontend pak šablony zobrazuje uživatelům a umožňuje jim s nimi interagovat. Pro tento účel byla využita kombinace technologií HTML, CSS a JavaScript. Pro vytvoření responzivního designu stránek byl zvolen CSS framework Bootstrap. Frontend našeho systému je dostupný prostřednictvím webového prohlížeče.

5.6.1 Model-View-Controller architektura

Architektura Model-View-Controller je návrhový vzor pro efektivní a modulární návrh aplikací, které se pak dají snáze vyvíjet a udržovat. Aplikace se organizuje do tří komponent. [68]

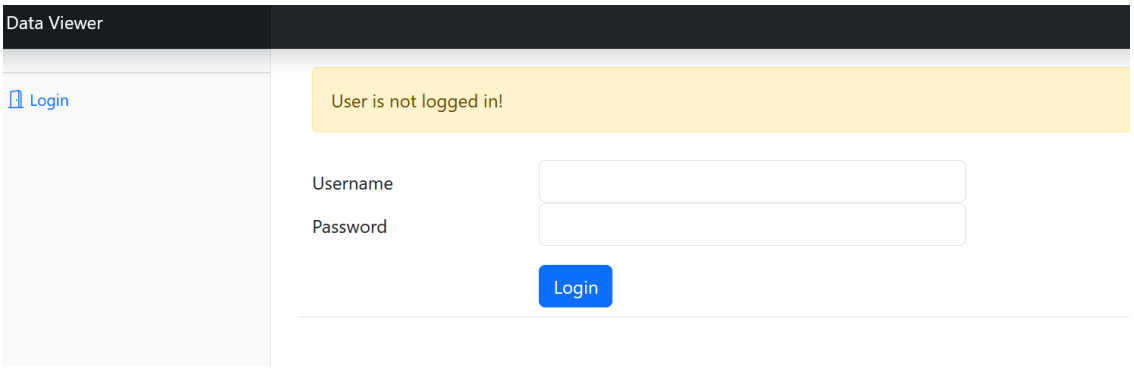
1. Model reprezentuje datový model aplikace, který byl popsán v kapitole 4.6. Tato komponenta komunikuje s databází. Zodpovídá za vytváření, čtení, aktualizaci a mazání záznamů z databáze.
2. View (pohled) zajišťuje zobrazení výstupu uživateli. V našem případě se skládá především z šablon.
3. Controller (ovladač) slouží jako prostředník mezi modelem a pohledem.

5.6.2 Moduly a role

I při návrhu frontendu byly využita myšlenka organizace do modulů. V tomto případě se modul skládá z jednoho či více formulářů, které společně slouží pro čtení a úpravu dat z datového modelu. Byly navrženy následující moduly:

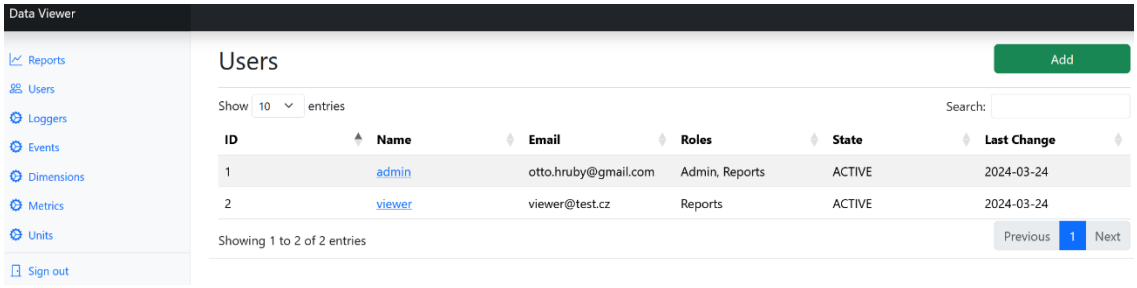
- Modul pro tvorbu a zobrazení reportů
- Modul pro správu uživatelů
- Modul pro správu loggerů
- Modul pro správu událostí
- Modul pro správu dimenzí
- Modul pro správu metrik
- Modul pro správu jednotek

Pro splnění stanovených požadavků ohledně bezpečnosti z kapitoly 3.4 systém pracuje také s uživateli, kteří mohou mít jednu či více rolí. Uživatel může přistupovat k jednotlivým modulům pomocí nabídky menu. Nabídka je generovaná dynamicky na základě rolí, kterými uživatel disponuje. Pokud se nepřihlášený uživatel pokusí přistoupit ke konkrétním datům, tak je přesměrován na přihlašovací obrazovku a je mu znemožněn přístup k dalším obrazovkám. Tento případ je znázorněn na obrázku 5.1.



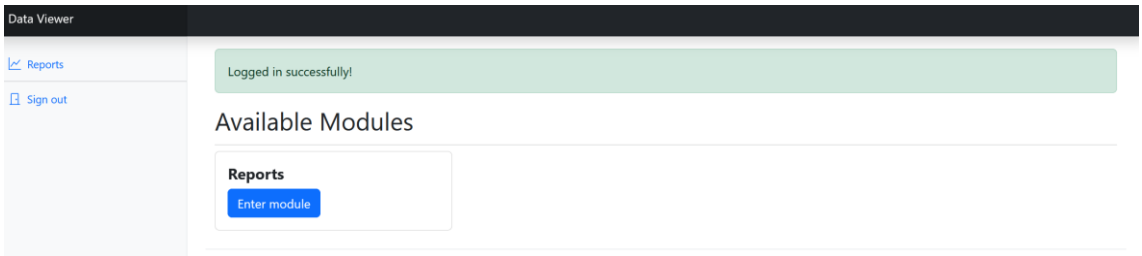
Obrázek 5.1 Přihlašovací obrazovka

Stejně systém reaguje, pokud se uživatel pokusí přistoupit k modulu, ke kterému nemá požadovanou roli. Role, se kterýma systém pracuje jsou Admin a Reports. Uživatel s rolí Admin má zpřístupněny všechny moduly pro správu dat. Může tedy vytvářet či zablokovat další uživatele a přidělovat či odebírat role. Obrazovka s výpisem uživatelů a jejich rolí je na obrázku 5.2.



Obrázek 5.2 Obrazovka s výpisem uživatelů a jejich rolí

Pokud má uživatel pouze roli Reports, pak je mu pouze zpřístupněn modul Reports. Ostatní moduly jsou nepřístupné. Uvedená situace je zaznamenána na obrázku 5.3.

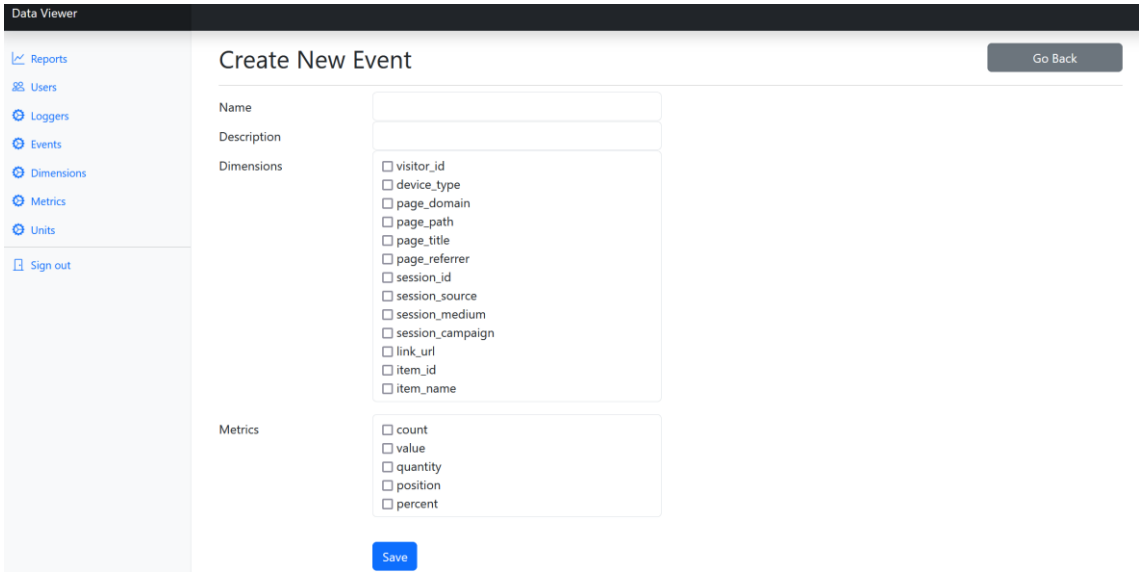


Obrázek 5.3 Obrazovka s výpisem reportů

5.6.3 Formuláře

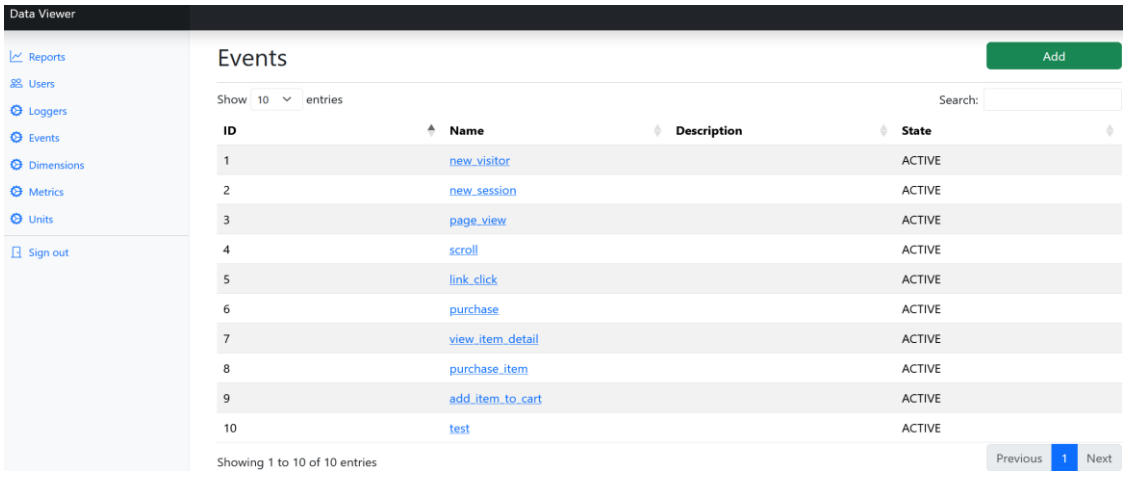
Informační systém pracuje také se zmiňovanými formuláři. Pro jejich flexibilní vytváření byla zvolena knihovna WTForms. WTForms zajišťuje také kontrolu přijatých dat a ochranu pomocí Cross-Site Request Forgery (CSFR). Tato ochrana je důležitá pro prevenci útoků, kdy se útočník pokouší zneužít indentitu přihlášeného uživatele. WTForms spolupracuje s libovolným šablonovacím systémem, takže je kompatibilní i s Jinja2. [69,70]

Na každé stránce se nachází maximálně jeden formulář. Aplikace využívá vícero typů formulářů. Prvním typem je formulář pro vytváření nových položek, který je znázorněný na obrázku 5.4. V tomto případě slouží pro přidání nové události a zvolení její dimenzí a metrik. Vytvořené položce se vždy přiřadí stav ACTIVE.



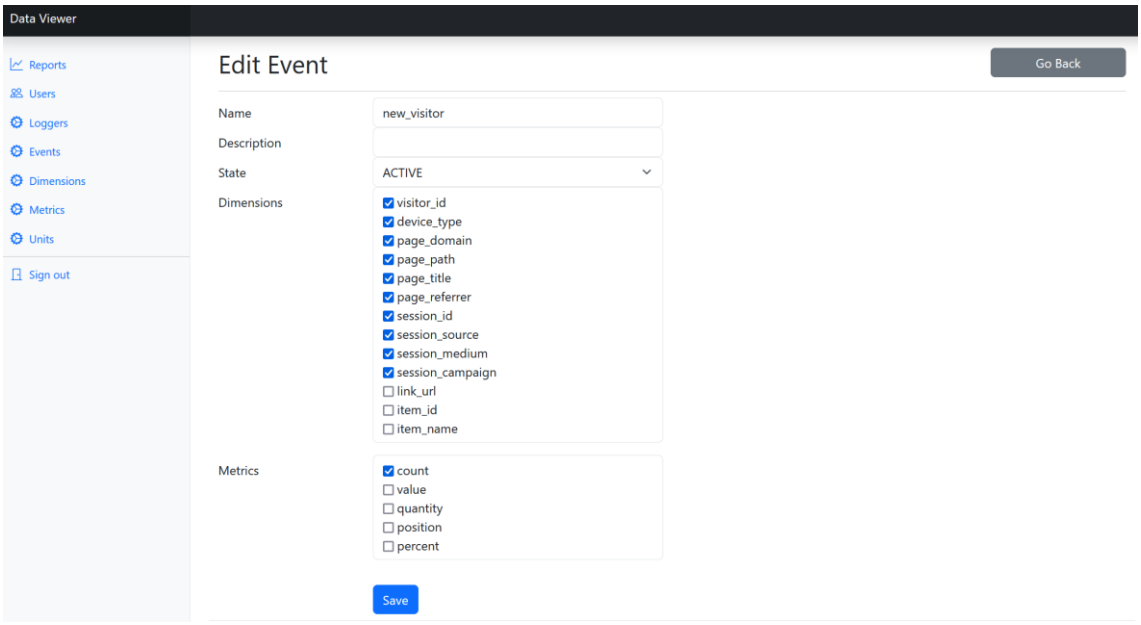
Obrázek 5.4 Blokové schéma systému

Druhým typem je formulář pro filtrování a zobrazení seznamu položek. Na obrázku 5.5 je zobrazen formulář s výpisem existujících událostí.



Obrázek 5.5 Blokové schéma systému

Posledním typem je formulář pro úpravu položky, který lze vidět na obrázku 5.6. Vypadá podobně jako formulář pro přidávání nových položek. Pracuje však s již existující položkou a umožňuje měnit také stav položky. Při nastavení položky na stav INACTIVE s ní pak nelze pracovat v dalších modulech. V případě nastavení události na stav INACTIVE pak nelze zvolit v modulu pro správu reportů.



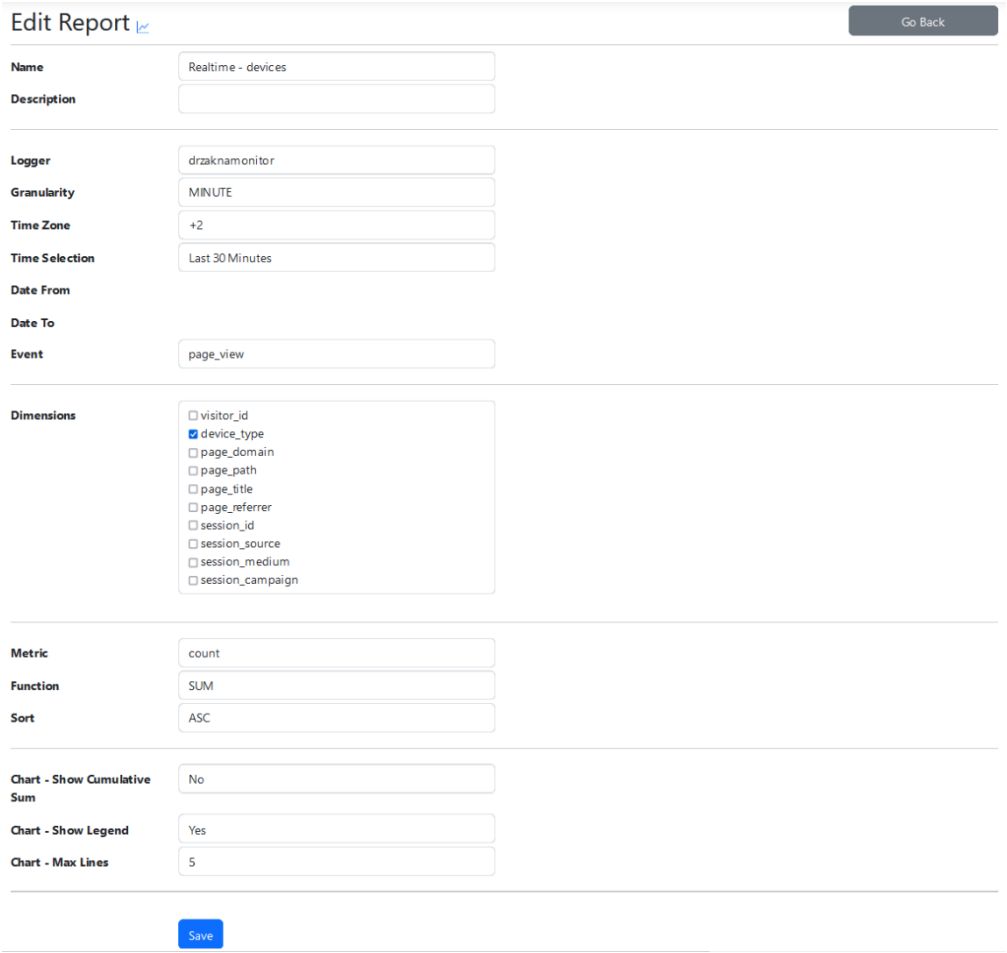
Obrázek 5.6 Formulář pro úpravu položky

5.6.4 Vizualizace dat

Vizualizace nasbíraných a zpracovaných dat uživateli je klíčovou součástí informačního systému. Je implementována v rámci modulu pro tvorbu a zobrazení reportů. Uživatel si nejdříve pomocí formuláře, který lze vidět na obrázku 5.7, specifikuje, jaká data chce zobrazit. Tento modul nabízí vysokou flexibilitu. Umožňuje kombinovat různé parametry a dynamicky zobrazuje pouze relevantní položky v souladu s nastavením, které lze měnit v rámci ostatních modulů pro správu parametrů.

Uživatelé mohou specifikovat název události, granularitu dat (minutové, hodinové, denní nebo měsíční statistiky) a časové okno, které je buď relativní, nebo absolutní. Dále si mohou vybrat, zda chtějí zobrazit počet událostí, maximální či minimální hodnotu zvolené metriky. Je také možné události rozpadnout dle různých dimenzí.

Uživatel má možnost zobrazit časový průběh zvolené metriky na čase, kde každá kombinace dimenzí je zobrazena jako samostatná křivka. Pro zachování přehlednosti je však zobrazeno maximálně 10 křivek. Navíc lze zvolit, že uživatel chce zobrazit kumulativní součet, což způsobí sumaci dat v grafu.

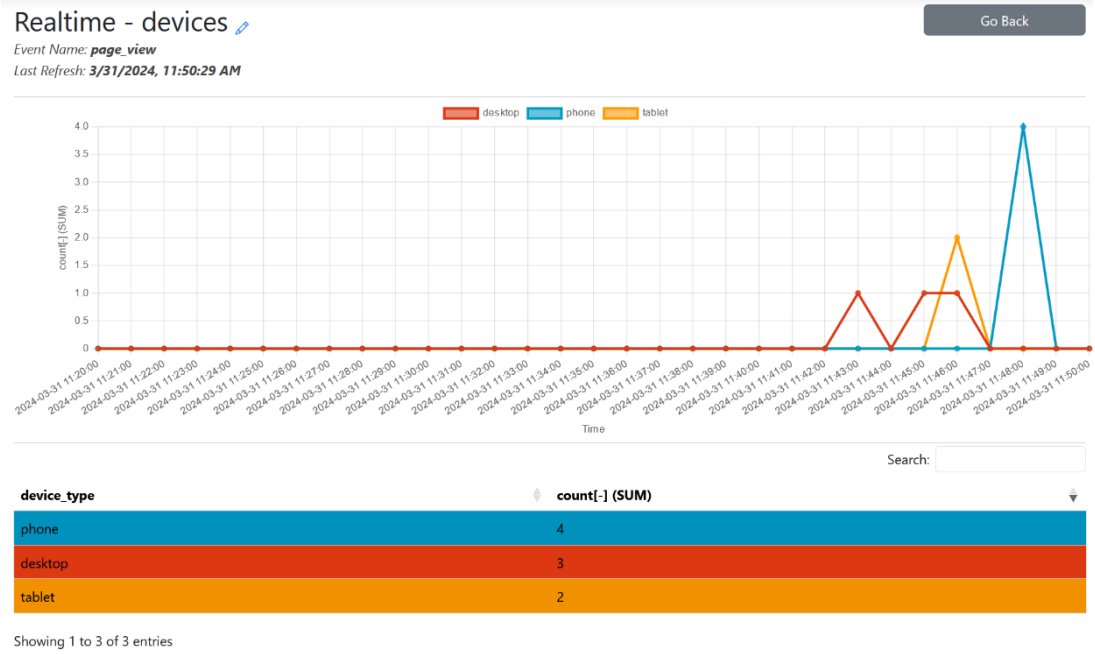


Obrázek 5.7 Formulář pro tvorbu reportu

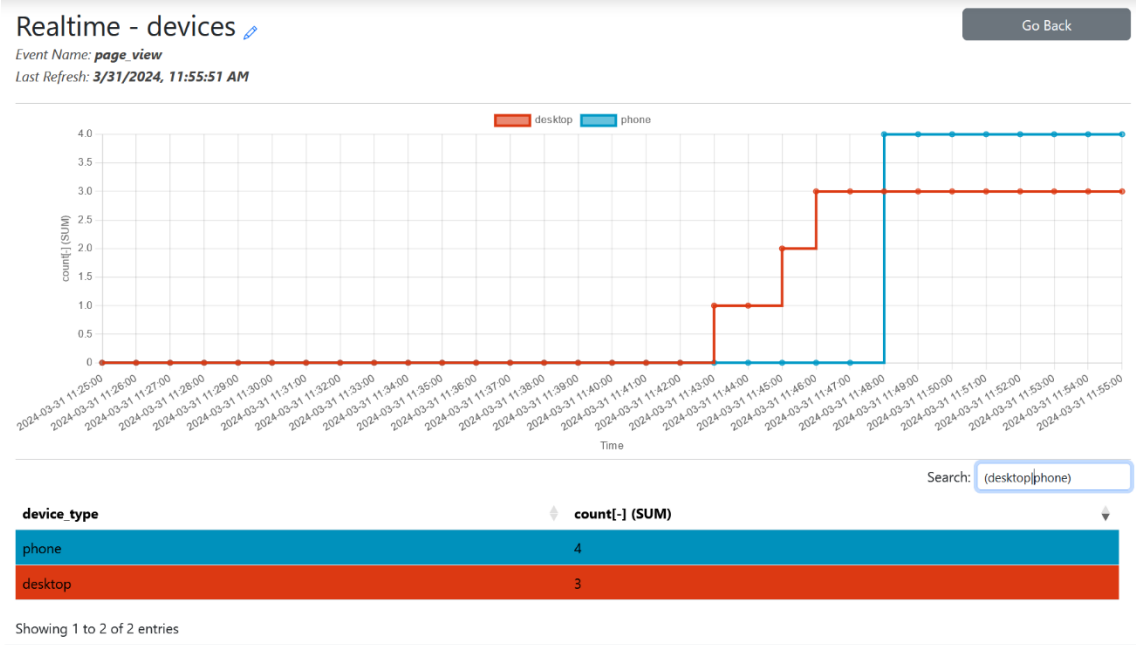
Na obrázku 5.8 je zobrazena obrazovka s vizualizací dat. V horní části se nachází hlavička, kde jsou název reportu, název události a poslední čas aktualizace. Pod hlavičkou se pak nachází graf zobrazující hodnotu metriky v závislosti na čase. Pro tvorbu tohoto grafu byla využita knihovna Chart.js [71], protože poskytuje široké možnosti nastavení a splnila obsahuje požadované vlastnosti.

Pod grafem je umístněná tabulka, pro kterou byla použita JavaScript knihovna s názvem DataTables [72]. V této tabulce jsou zobrazeny dimenze a vpravo pak agregovaná metrika pro danou skupinu dimenzí. Tabulka dále umožňuje řazení a filtrování dat pomocí regulárních výrazů. Položky v tabulce, které jsou zobrazeny v grafu, jsou obarveny stejnou barvou, což usnadňuje uživatelům orientaci.

Pro zvolenou konfiguraci z obrázku 5.7, pak odpovídá vizualizace, která je zobrazená právě na obrázku 5.8. Pokud uživatel nastaví požadavek na zobrazení kumulovaného součtu v grafu, může tuto volbu upravit přímo ve formuláři. Pokud ho zajímá například jen počet zobrazení stránek pomocí zařízení typu desktop a mobil, pak může v políčku pro vyhledávání zadat regulární výraz s hodnotou (desktop|mobile). Výslednou vizualizaci pak lze vidět na obrázku 5.9.



Obrázek 5.8 Vizualizace dat – časový průběh



Obrázek 5.9 Vizualizace dat – kumulativní součet a filtrování

5.7 Instalace a zprovoznění infrastruktury

Po přípravě potřebných kódů systému byl založen projekt v GCP. Před samotnou instalací je nezbytné zadat platební metoda pro úhradu nákladů. Bez tohoto kroku lze využívat pouze omezený počet bezplatných služeb.

Následným krokem je použití nástroje Cloud Shell. Jedná se o prostředí integrované přímo v cloudu pro správu služeb. Obsahuje příkazovou řádku a má v sobě předinstalované nástroje jako jsou právě Git, Docker i Terraform.

Pomocí následujících příkazů pak proběhne naklonování repozitáře z GitHubu do Cloud Shellu a instalace infrastruktury pomocí Bash skriptu, který vytvoří Docker kontejnery a spustí Terraform. První parametr v Bash skriptu specifikuje název projektu pro instalaci, další parametr pak nastaví heslo do databáze a poslední parametr není povinný a slouží pro specifikaci vlastní domény pro data logger. Bez specifikace tohoto parametry je serveru přidělena adresa Google Cloudem.

```
git clone https://github.com/ottohruby/web_analytics_dp.git
>
cd web_analytics_dp
>
bash install.sh -p otto-hruby-dp-final -d <HESLO> \
-u dp-logger.ottohruby.cz
>
```

Použití vlastní domény pro Data Logger vyžaduje ověření domény prostřednictvím DNS záznamu typu TXT. Pro mapování vlastní domény na službu Cloud Run v GCP pomocí DNS záznamu typu CNAME je potřeba nastavit hodnotu ghs.googlehosted.com pro subdoménu dp-logger.

6. OVĚŘENÍ FUNKČNOSTI

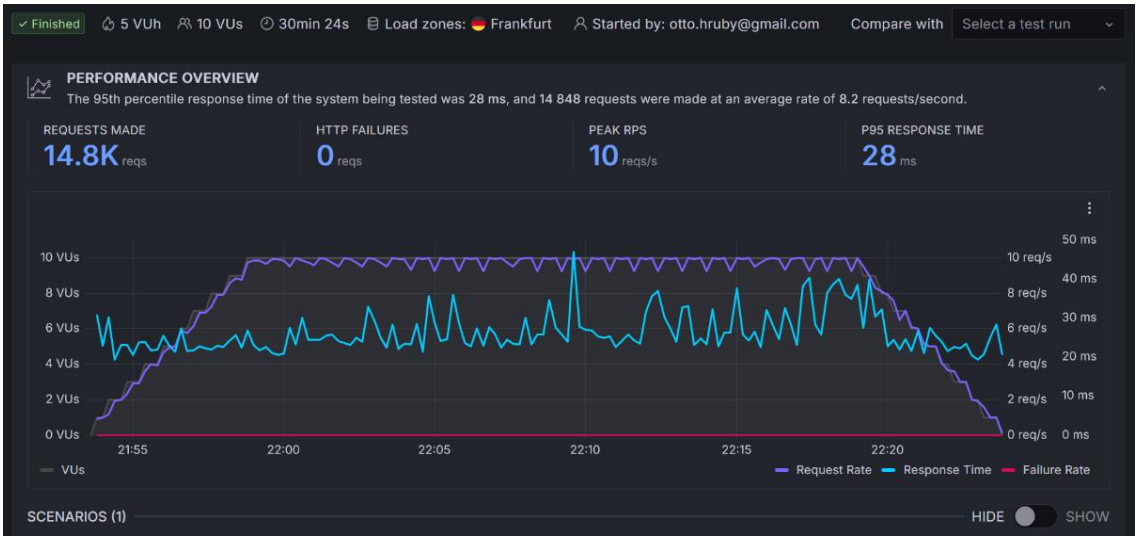
Před uvedením systému do reálného provozu byla provedena série zátěžových testů, aby byla ověřena stabilita systému a odhaleny případné chyby. Po jejich provedení bylo provedeno nasazení na reálném webu.

6.1 Zátěžový test

Pro ověření funkčnosti systému byla provedena série 3 zátěžových testů, které byly realizovány prostřednictvím nástroje Grafana Cloud K6 [73]. Jedná se o specializovaný nástroj přímo pro zátěžové testování systémů. Testování simulovalo požadavky generované měřicím skriptem pro data logger. Tímto způsobem byly otestovány veškeré moduly systému kromě samotného měřicího skriptu.

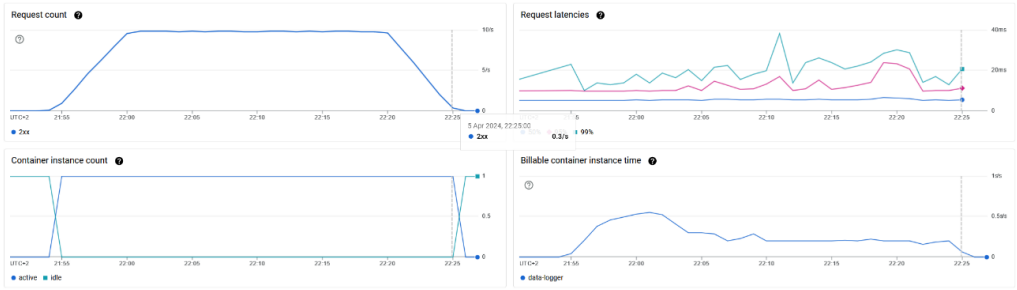
6.1.1 Testovací scénář 1

První scénář byl navržen tak, aby simuloval lehkou zátěž, konkrétně 10 událostí za sekundu celkem od 10 virtuálních uživatelů (VUs). Zátěž měla definovanou náběžnou a sestupnou hranu trvající 5 minut. Tento scénář trval celkem 30 minut. Celkový počet vygenerovaných událostí činil 14848, přičemž všechny byly úspěšně odeslány a zpracovány data loggerem. Hodnota 95. percentilu doby odezvy byla 28 ms. Výsledky z pohledu virtuálních uživatelů jsou zobrazeny na obrázku 6.1.



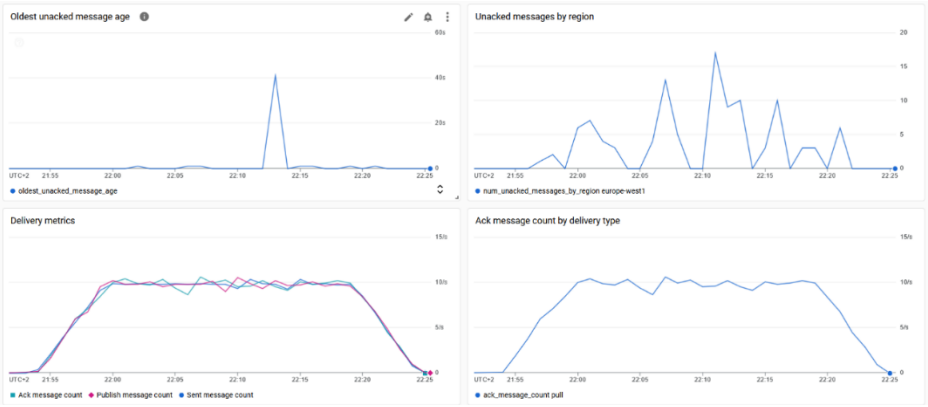
Obrázek 6.1 Testovací scénář 1 – výsledky z pohledu virtuálních uživatelů

Následně byl proveden monitoring z pohledu data loggeru. Využití veřejného cloudu přineslo výhodu možnosti používat monitorovací služby přímo integrované v rámci webového rozhraní. Průběhy počtu požadavků za sekundu a doby odezvy odpovídaly průběhům z nástroje Grafana Cloud K6 z obrázku 6.1. Dále lze vidět, že na zpracování zátěže stačila jedna instance data loggeru, což je znázorněno na obrázku 6.2.



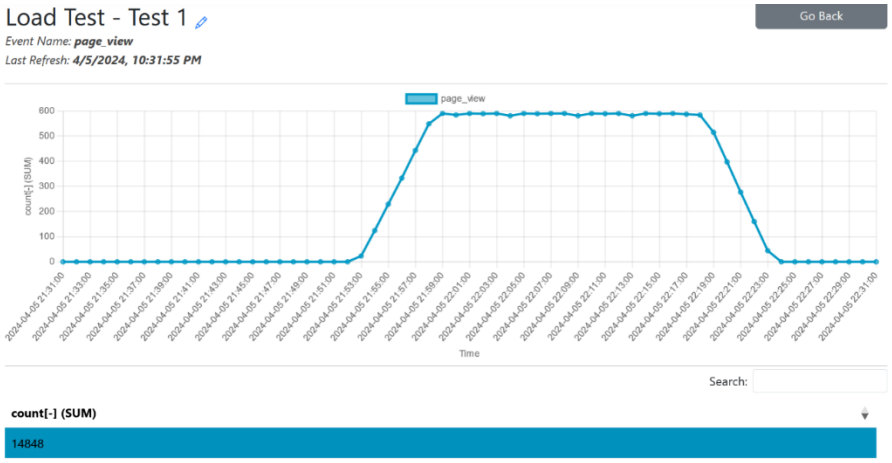
Obrázek 6.2 Testovací scénář 1 – výsledky z pohledu data loggeru

Předposlední krok zahrnoval monitorování stavu fronty v message brokerovi. Obrázek 6.3 ilustruje, že většinou byly zprávy byly zpracovány okamžitě. Všechny zprávy byly zpracovány do 1 minuty od jejich přijetí. Konzument zpráv úspěšně odebírá zprávy z fronty a vkládá je do databáze.



Obrázek 6.3 Testovací scénář 1 – výsledky z pohledu message brokera

V posledním kroku byla ověřena funkčnost agregování událostí v databázi a funkčnost celého informačního systému. Obrázek 6.4 zobrazuje graf znázorňující závislost počtu obdržených zpráv na čase. Celkový počet událostí za sledované období je 14848 a shoduje se tak s počtem událostí vygenerovaných v testovacím nástroji.



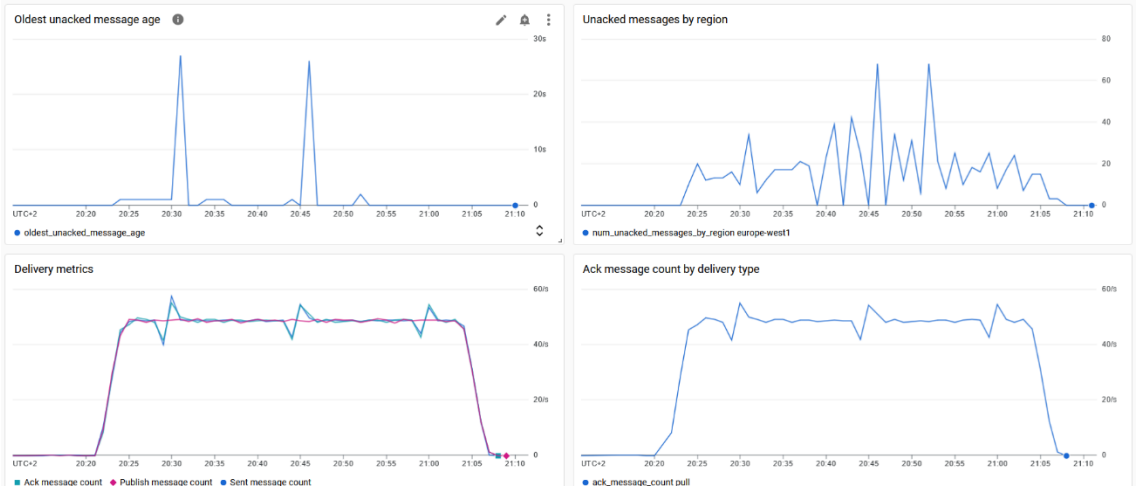
Obrázek 6.4 Testovací scénář 1 – výsledky v informačním systému

6.1.2 Testovací scénář 2

Druhý scénář simuloval střední zátěž, konkrétně 50 událostí za sekundu celkem od 50 virtuálních uživatelů (VUs). Zátěž měla definovanou náběžnou a sestupnou hranu trvající 2,5 minuty. Tento scénář trval celkem 45 minut. Celkový počet vygenerovaných událostí činil 124202, přičemž všechny byly úspěšně odeslány a zpracovány data loggerem. Hodnota 95. percentilu doby odezvy byla 34 ms. Výsledky z pohledu virtuálních uživatelů jsou zobrazeny na obrázku 6.1. Chování data loggeru bylo obdobné jako v prvním scénáři, a proto není jeho průběh znovu zobrazen.

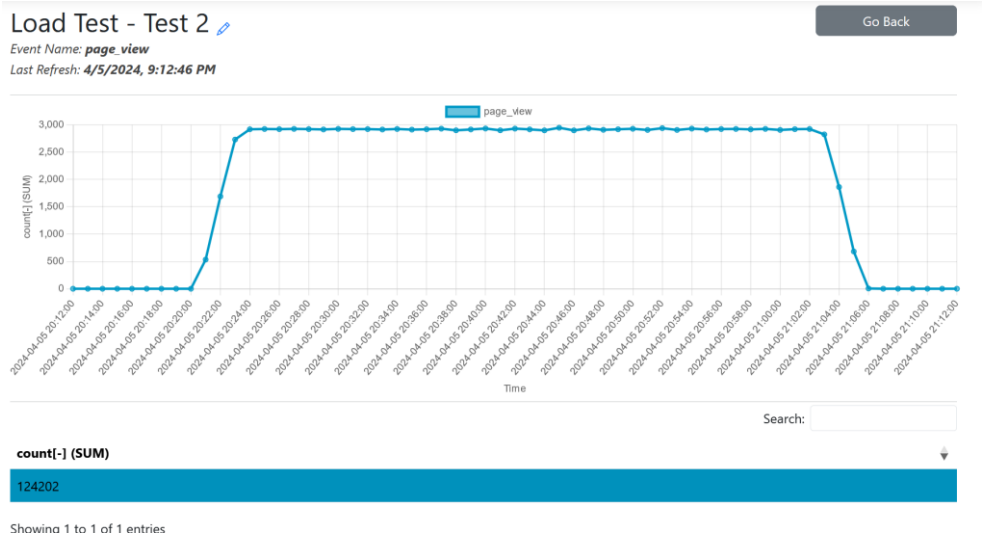
Obrázek 6.5 Testovací scénář 2 – výsledky z pohledu virtuálních uživatelů

Monitorování stavu fronty v message brokerovi, který je zobrazen na obrázku 6.6, ukazuje, že byla opět většina zpráv zpracována okamžitě. Všechny zprávy byly zpracovány do 1 minuty od jejich přijetí. Konzument zpráv úspěšně odebírá zprávy z fronty a vkládá je do databáze i v tomto scénáři.



Obrázek 6.6 Testovací scénář 2 – výsledky z pohledu message brokera

V posledním kroku byla opět ověřena funkčnost agregování událostí v databázi a funkčnost celého informačního systému. Obrázek 6.7 zobrazuje graf znázorňující závislost počtu obdržených zpráv na čase. Celkový počet událostí za sledované období je 124202 a shoduje se tak s počtem událostí vygenerovaných v testovacím nástroji.

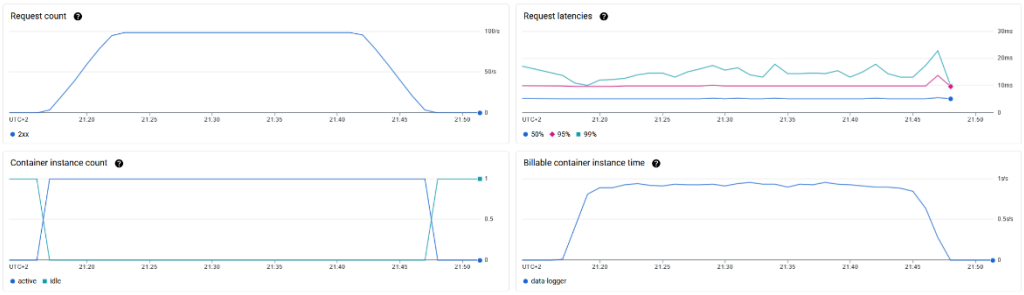


Obrázek 6.7 Testovací scénář 2 – výsledky v informačním systému

6.1.3 Testovací scénář 3

Třetí scénář simuloval větší zátěž, konkrétně 100 událostí za sekundu celkem od 100 virtuálních uživatelů (VUs). Zátěž měla definovanou náběžnou a sestupnou hranu trvající 5 minut. Tento scénář trval celkem 30 minut. Celkový počet vygenerovaných událostí činil 147192, přičemž všechny byly úspěšně odeslány a zpracovány data loggerem. Hodnota 95. percentilu doby odezvy byla 27 ms. Výsledky z pohledu virtuálních uživatelů jsou zobrazeny na obrázku 6.8. Dále lze vidět, že na zpracování zátěže opět stačila jedna instance data loggeru, což je znázorněno na obrázku 6.9.

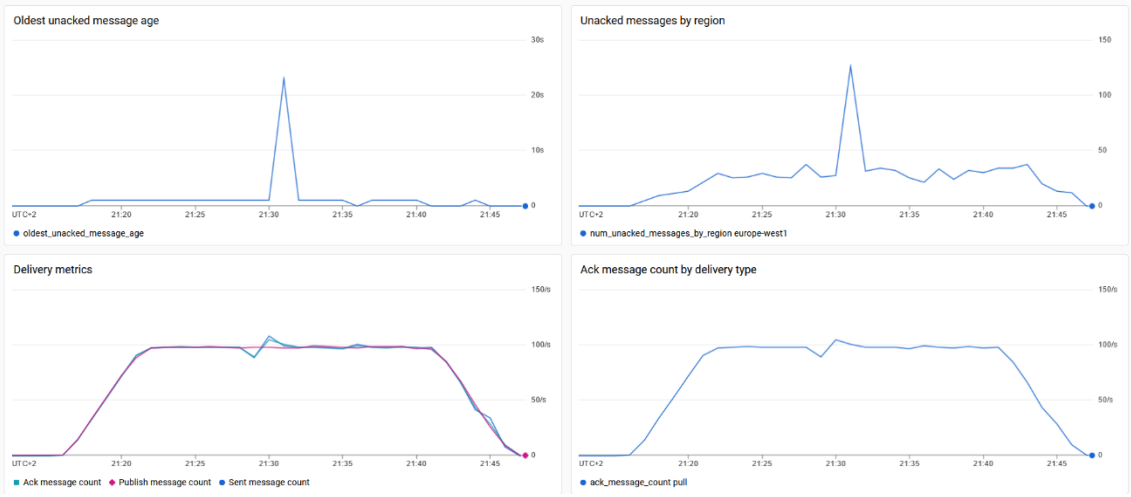
Obrázek 6.8 Testovací scénář 3 – výsledky z pohledu virtuálních uživatelů



Obrázek 6.9 Testovací scénář 3 – výsledky z pohledu data loggeru

Monitorování stavu fronty v message brokerovi, který je zobrazen na obrázku 6.10, ukazuje, že byla opět většina zpráv zpracována okamžitě. Všechny zprávy byly zpracovány do 1 minuty od jejich přijetí, což je dostačující. Konzument zpráv úspěšně odebírá zprávy z fronty a vkládá je do databáze i v tomto scénáři.

Bylo zaznamenáno, že ve všech scénářích nastávaly výkyvy v počtu zpracovaných zpráv za sekundu. Tento jev se vyskytoval nezávisle na objemu zpracovaných dat a je přisuzován službě Pub/Sub. Jedna z nevýhod použití této služby je, že se chová jako černá skříňka a není možné přesně pochopit interní fungování. Nicméně tyto výkyvy nemají vliv na funkčnost systému.



Obrázek 6.10 Testovací scénář 3 – výsledky z pohledu message brokera

V posledním kroku byla opět ověřena funkčnost agregování událostí v databázi a funkčnost celého informačního systému. Obrázek 6.11 zobrazuje graf znázorňující závislost počtu obdržených zpráv na čase. Celkový počet událostí za sledované období je 147192 a shoduje se s počtem událostí vygenerovaných v testovacím nástroji. Zátěžový test systému tak byl vyhodnocen jako úspěšný a mohlo dojít k ověření funkčnosti na reálném webu.

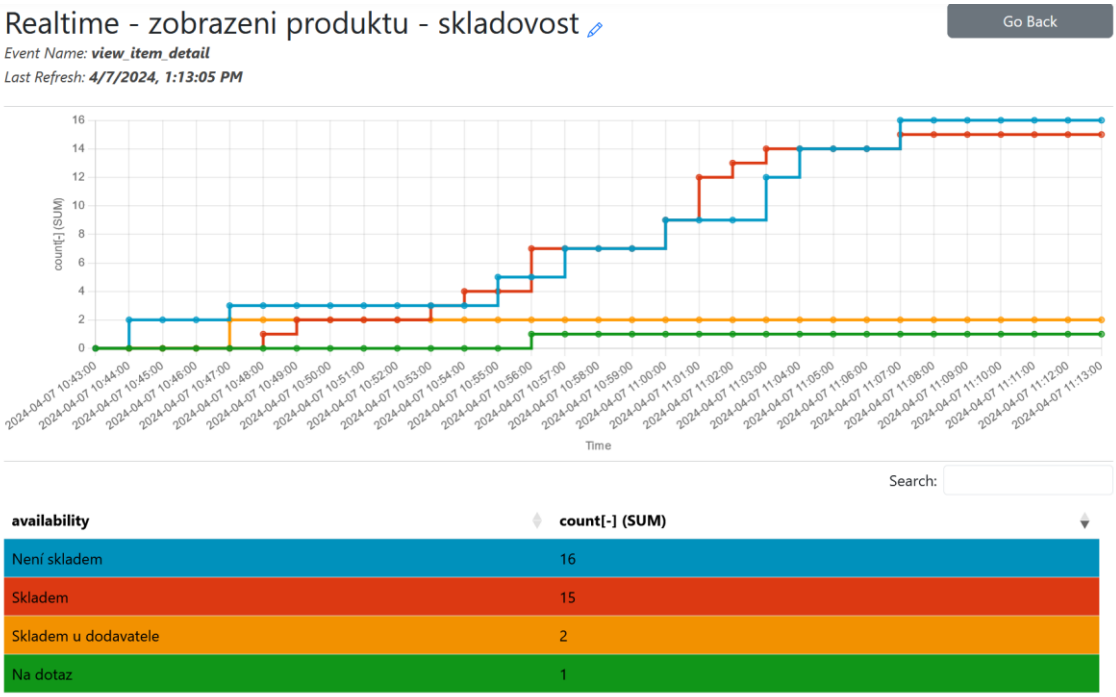
Obrázek 6.11 Testovací scénář 3 – výsledky v informačním systému

6.2 Ověření funkčnosti na reálném webu

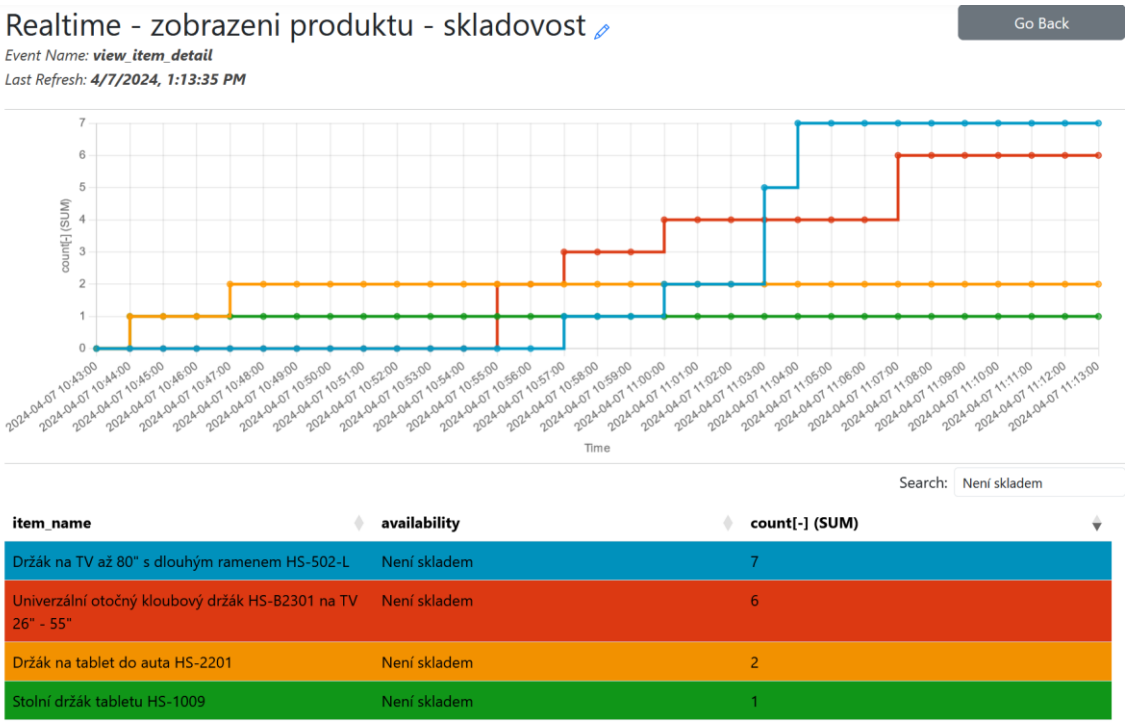
Po dokončení zátěžového testu byl nasazen měřicí skript na web drzaknamonitor.cz. Do modulu pro správu dimenzí byly přidány další položky, a to značka (brand), kategorie (category), dostupnost (availability). Následně byly tyto dimenze přiřazeny události zobrazení detailu produktu (view_item_detail) v modulu pro správu událostí.

Tato úprava umožnila nástroj efektivně využít v praxi. Díky ní je možné v systému sledovat počet zobrazení konkrétních produktů podle jejich značky, názvu, kategorie, dostupnosti a všech možných kombinací těchto dimenzí. Po nastaví lze report zobrazit. Na obrázku 6.12 je zobrazen report ukazující kumulativní součet počtu zobrazení produktů podle dostupnosti na čase. Vytvořený report může pomoci identifikovat situace, kdy uživatelé zobrazují produkty, které však nejsou na skladě. Často v takových případech uživatel produkt nezakoupí a opustí stránku, což vede k potenciální ztrátě objednávky, případně ztráty nespokojeného zákazníka. Díky použití tohoto systému je možné tuto situaci sledovat v reálném čase.

Pokud je zaznamenána situace, kdy uživatelé zobrazují produkty, které nejsou skladem, lze report jednoduše upravit, aby zobrazoval data detailněji. Toho lze dosáhnout zahrnutím dimenze názvu produktu (item_name) do statistik. Report lze navíc vyfiltrovat tak, aby zobrazoval pouze zobrazení produktů, které nejsou skladem. Tato situace je zobrazena na obrázku 6.13.



Obrázek 6.12 Report zobrazení produktů podle dostupnosti



Obrázek 6.13 Report zobrazení názvů produktů, které nejsou skladem

Závěr

V první kapitole byly rozebrány základní techniky sběru dat. Jedna z technik vychází z analýzy serverových logů. Druhá technika používá označkování stránek pomocí jazyka JavaScript a měření probíhá z prohlížeče. Také byly rozebrány výhody i nevýhody jednotlivých technik. Následně byla provedena rešerše nástrojů pro webovou analytiku. Dominantním nástrojem je Google Analytics 4, avšak má nevýhodu, že poskytuje pouze velmi omezené statistiky v reálném čase. Podobně na tom jsou i ostatní nástroje.

V další kapitole byly představeny techniky pro zpracování dat v reálném čase. Pro komunikaci mezi aplikacemi se používá API. Nejrozšířenějším stylem pro tvorbu webových API je REST. Existují však i styly umožňující rychlejší komunikaci, a to třeba gRPC, který je však není vhodný pro sběr dat z prohlížečů. Pro ukládání dat byla použita relační databáze, zejména protože budou dat mít pevnou strukturu a bude tak zajištěna integrita a konzistence dat. Proud dat lze strukturovat do zpráv. Pro spolehlivé předávání zpráv mezi aplikacemi se využívá specializovaný software, tzv. message broker.

Následně byly rozebrány požadavky na systém pro webovou analytiku. Systém umí zaznamenávat návštěvníky, návštěvy, zdroje návštěv a typy zařízení. Data jsou ukládána ve formě událostí a je kladen důraz na soukromí uživatelů.

V další kapitole byla navržena architektura systému. Skládá z měřicího skriptu, data loggeru, message brokera, konzumenta zpráv, databáze a informačního systému. Přidáním message brokera a konzumenta zpráv bylo dosáhnuto větší stability a spolehlivosti systému. Součástí návrhu je také datový model, co je uveden v příloze A. Při návrhu byl kladen důraz na funkčnost, flexibilitu a splnění dalších stanovených požadavků z kapitoly 3.

V následujícím kroku byla provedena implementace systému na základě návrhu. Pro provoz systémy byl zvolen Google Cloud Platform s ohledem na optimalizaci nákladů pro malé až střední aplikace. Při implementaci byly využity technologie jako Terraform, Docker a Git, což umožnilo efektivní vývoj a správu systému. Pro snadnou instalaci systému je k dispozici skript. Veškerý kód je k dispozici v příloze B.

V závěrečné fázi práce proběhlo ověření funkčnosti systému. Byl úspěšně proveden zátěžový test simulující zátěž až 100 událostí za sekundu, což je dostačující pro malé až střední aplikace.

V posledním kroku byl systém otestován na reálném webu drzaknamonitor.cz. Pro ukázku praktického využití v praxi byl vytvořen report umožňující sledovat v reálném čase sledované produkty podle dostupnosti a názvu.

Vytvořený systém je velmi flexibilní. Při naplnění databázových tabulek jinými hodnotami lze systém využít nejen pro potřeby webové analytiky, ale například i pro zpracování a vyhodnocování dat z televizí nebo různých zařízení v rámci internetu věcí.

LITERATURA

[1] Popular, but hardly used: Has Google Analytics been to the detriment of Web Analytics? Online. In: *WebSci '23: 15th ACM Web Science Conference 2023*. Austin, TX, USA: Association for Computing Machinery, 2023, s. 304–311. ISBN 979-8-4007-0089-7. Dostupné z: https://doi.org/10.1145/3578503.3583601. [cit. 2023-10-30].

[2] Performing web analytics with Google Analytics 4: a platform review. Online. *Journal of Marketing Analytics*. 2023. ISSN 2050-3326. Dostupné z: https://link.springer.com/article/10.1057/s41270-023-00244-4. [cit. 2023-10-30].

[3] Towards efficient Big Data and data analytics: A review. Online. In: *2014 Conference on IT in Business, Industry and Government (CSIBIG)*. Indore, India: IEEE, 2014, s. 1-6. ISBN 978-1-4799-3064-7. Dostupné z: https://ieeexplore.ieee.org/abstract/document/7056933. [cit. 2023-10-30].

[4] Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity. Sybex, 2009. ISBN 978-0470529393.

[5] Web Analytics: An Hour a Day. Sybex, 2007. ISBN 978-0470130650.

[6] *Comparison of JavaScript Tracking and Web Log Analytics*. Online. Piwik PRO Analytics Suite | Modern analytics made simple. C2023. Dostupné z: https://piwik.pro/blog/javascript-tracking-web-log-analytics/. [cit. 2023-10-30].

[7] Evaluation of Adblock Software Usage. Online. In: *Complex Systems Informatics and Modeling Quarterly*. Katowice: RTU Press, 2019, s. 51-63. Dostupné z: https://www.researchgate.net/publication/338525395_Evaluation_of_Adblock_Software_Usage. [cit. 2023-10-30].

[8] Tools for Academic Business Intelligence & Analytics Teaching – Results of an Evaluation. Online. In: *Conference: 4th Business Analytics Congress, BAC 2015*. Texas, USA: Christoph Kollwitz, 2015. Dostupné z: https://www.researchgate.net/publication/289819551_Tools_for_Academic_Business_Intelligence_Analytics_Teaching_-_Results_of_an_Evaluation. [cit. 2023-10-30].

[9] *Usage Statistics and Market Share of Traffic Analysis Tools for Websites, October 2023*. Online. W3Techs - extensive and reliable web technology surveys. C2009-2023. Dostupné z: https://w3techs.com/technologies/overview/traffic_analysis. [cit. 2023-10-30].

[10] Google Analytics Alternatives: A Guide to Navigating the World of Options Beyond Google. Quantable, 2022. ISBN 979-8987124505.

[11] *Meta pixel: Měření, optimalizace a retargetování reklam na Facebooku a Instagramu | Meta for Business*. Online. Meta for Business: marketing na Facebooku. C2023. Dostupné z: https://www.facebook.com/business/tools/meta-pixel. [cit. 2023-10-30].

[12] *Universal Event Tracking - Microsoft Advertising*. Online. Microsoft Advertising. C2023. Dostupné z: https://about.ads.microsoft.com/en-us/solutions/tools/universal-event-tracking. [cit. 2023-10-30].
[13] *LinkedIn Insight Tag | LinkedIn Marketing Solutions*. Online. Marketing & Advertising on LinkedIn | LinkedIn Ads. C2023. Dostupné z: https://business.linkedin.com/marketing-solutions/insight-tag. [cit. 2023-10-30].
[14] *Alternativy ke Google analytics 4 Marek Lecián Webový analytik*. Online. Marek Lecián Webový analytik. C2023. Dostupné z: https://mareklecian.cz/alternativy-ke-google-analytics-4/. [cit. 2023-10-30].
[15] *Jetpack – WP Security, Backup, Speed, & Growth – WordPress plugin | WordPress.org*. Online. C2023. Dostupné z: https://wordpress.org/plugins/jetpack/. [cit. 2023-10-30].
[16] *[GA4] Realtime report - Analytics Help*. Online. Analytics Help. C2023. Dostupné z: https://support.google.com/analytics/answer/9271392?hl=en. [cit. 2023-10-30].
[17] *Pricing | BigQuery: Cloud Data Warehouse | Google Cloud*. Online. Cloud Computing Services | Google Cloud. C2023. Dostupné z: https://cloud.google.com/bigquery/pricing. [cit. 2023-10-30].
[18] *The visits in real-time report FAQ - Reports - Matomo Analytics Platform*. Online. Matomo Analytics - The Google Analytics alternative that protects your data. C2023. Dostupné z: https://matomo.org/faq/reports/the-visits-in-real-time-widget/. [cit. 2023-10-30].
[19] *Enabling Cloudflare Web Analytics · Cloudflare Analytics docs*. Online. Home · Cloudflare Docs. C2023. Dostupné z: https://developers.cloudflare.com/analytics/web-analytics/getting-started/. [cit. 2023-10-30].
[20] *What is an API? - Application Programming Interface Explained - AWS*. Online. Cloud Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/what-is/api/. [cit. 2023-10-30].
[21] Hands-On RESTful API Design Patterns and Best Practices. Birmingham: Packt Publishing, 2019. ISBN 1788992660.
[22] RESTful Web APIs: Services for a Changing World. O'Reilly Media, 2013. ISBN 978-1449358068.
[23] *GRPC vs REST - Difference Between Application Designs - AWS*. Online. Cloud Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/compare/the-difference-between-grpc-and-rest/. [cit. 2023-10-30].
[24] *Programming Web Services With SOAP*. O'Reilly Media, 2002. ISBN 978-0596000950.
[25] GRPC: Up and Running: Building Cloud Native Applications with Go and Java for Docker and Kubernetes. O'Reilly Media, 2020. ISBN 978-1492058335.

[26] *The state of gRPC in the browser | gRPC*. Online. GRPC. C2023. Dostupné z: https://grpc.io/blog/state-of-grpc-web/. [cit. 2023-10-30].
[27] SQL and NoSQL Databases: Modeling, Languages, Security and Architectures for Big Data Management. 2nd ed. Springer, 2023. ISBN 978-3-031-27907-2.
[28] *What is SQL? - Structured Query Language (SQL) Explained - AWS*. Online. Cloud Services - Amazon Web Services (AWS). Dostupné z: https://aws.amazon.com/what-is/sql/. [cit. 2023-10-30].
[29] *Nerelační data a NoSQL - Azure Architecture Center | Microsoft Learn*. Online. Microsoft Learn: Build skills that open doors in your career. C2023. Dostupné z: https://learn.microsoft.com/cs-cz/azure/architecture/data-guide/big-data/non-relational-data. [cit. 2023-10-30].
[30] *Design applications for scaling - Microsoft Azure Well-Architected Framework | Microsoft Learn*. Online. Microsoft Learn: Build skills that open doors in your career. C2023. Dostupné z: https://learn.microsoft.com/en-us/azure/well-architected/scalability/design-scale. [cit. 2023-10-30].
[31] *What is Load Balancing? - Load Balancing Algorithm Explained - AWS*. Online. Cloud Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/what-is/load-balancing/. [cit. 2023-10-30].
[32] *What Is Load Balancing? How Load Balancers Work*. Online. Advanced Load Balancer, Web Server, & Reverse Proxy - NGINX. C2023. Dostupné z: https://www.nginx.com/resources/glossary/load-balancing/. [cit. 2023-10-30].
[33] *NoSQL Vs SQL Databases | MongoDB*. Online. MongoDB: The Developer Data Platform | MongoDB. C2023. Dostupné z: https://www.mongodb.com/nosql-explained/nosql-vs-sql. [cit. 2023-10-30].
[34] *Automatické škálování ve službě Azure Monitor - Azure Monitor | Microsoft Learn*. Online. Microsoft Learn: Build skills that open doors in your career. C2023. Dostupné z: https://learn.microsoft.com/cs-cz/azure/azure-monitor/autoscale/autoscale-overview. [cit. 2023-10-30].
[35] *Making Sense of Stream Processing*. O'Reilly Media, 2016. ISBN 9781491937280.
[36] *What are Message Brokers? | IBM*. Online. IBM - United States. Dostupné z: https://www.ibm.com/topics/message-brokers. [cit. 2023-10-31].
[37] *Understanding Message Brokers*. O'Reilly Media, 2017. ISBN 978-1-491-98153-5.
[38] *Web Analytics Demystified*. Celilo Group Media, 2004. ISBN 0974358428.
[39] *[GA4] Measure activity across platforms with User-ID - Analytics Help*. Online. Analytics Help. C2023. Dostupné z: https://support.google.com/analytics/answer/9213390?hl=en. [cit. 2023-11-13].
[40] *What are sessions and how are they counted? | Piwik PRO help center*. Online. Help center | Piwik PRO. C2023. Dostupné z: https://help.piwik.pro/support/questions/what-are-sessions-and-how-are-they-counted/. [cit. 2023-11-13].

[41] *[GA4] About Analytics sessions - Analytics Help*. Online. Analytics Help. C2023. Dostupné z: https://support.google.com/analytics/answer/9191807?hl=en. [cit. 2023-11-13].
[42] *[GA4] Campaigns and traffic sources - Analytics Help*. Online. Analytics Help. C2023. Dostupné z: https://support.google.com/analytics/answer/11242841?hl=en#zippy=%2Cin-this-article. [cit. 2023-11-13].
[43] *[GA4] About events - Analytics Help*. Online. Analytics Help. C2023. Dostupné z: https://support.google.com/analytics/answer/9322688?hl=en#zippy=%2Crealtime-report%2Cdebugview-report. [cit. 2023-11-13].
[44] *What are dimensions and metrics? | Piwik PRO help center*. Online. Help center | Piwik PRO. C2023. Dostupné z: https://help.piwik.pro/support/reports/what-are-dimensions-and-metrics/. [cit. 2023-11-13].
[45] *Ochrana osobních údajů podle nařízení GDPR - Your Europe*. Online. Chcete se dozvědět více o svých právech v EU? Nebo o svých povinnostech? - Your Europe. Dostupné z: https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_cs.htm. [cit. 2023-11-13].
[46] *Browser detection using the user agent - HTTP | MDN*. Online. C1998-2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent#considerations_before_using_browser_detection. [cit. 2023-11-17].
[47] *Firefox user agent string reference - HTTP | MDN*. Online. MDN Web Docs. C1998-2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent/Firefox. [cit. 2023-11-17].
[48] *Using media queries - CSS: Cascading Style Sheets | MDN*. Online. C1998-2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries. [cit. 2023-11-17].
[49] *PaaS vs IaaS vs SaaS: What's the difference? | Google Cloud*. Online. Cloud Computing Services | Google Cloud. Dostupné z: https://cloud.google.com/learn/paas-vs-iaas-vs-saas#section-3. [cit. 2023-11-22].
[50] *Typy cloud computingu – definice | Microsoft Azure*. Online. Microsoft – cloud, počítače, aplikace a hry. C2023. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/types-of-cloud-computing. [cit. 2023-11-22].
[51] Sufficient Comparison Among Cloud Computing Services: IaaS, PaaS, and SaaS: A Review. Online. *International Journal of Science and Business*. 2021, roč. 2021, č. 5, s. 17-30. Dostupné z: https://ijsab.com/wp-content/uploads/667.pdf. [cit. 2023-11-22].
[52] Using a Microbenchmark to Compare Function as a Service Solutions. Online. *European Conference on Service-Oriented and Cloud Computing*. 2018, roč. 2018, s. 146-160. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-319-99819-0_11. [cit. 2023-11-22].

[53] *Azure Container Apps - Pricing | Microsoft Azure*. Online. Microsoft – cloud, počítače, aplikace a hry. C2023. Dostupné z: https://azure.microsoft.com/en-us/pricing/details/container-apps/. [cit. 2023-11-25].
[54] *Pricing | Cloud Run | Google Cloud*. Online. Cloud Computing Services | Google Cloud. Dostupné z: https://cloud.google.com/run/pricing. [cit. 2023-11-25].
[55] *Serverless Compute Engine–AWS Fargate Pricing–Amazon Web Services*. Online. Cloud Computing Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/fargate/pricing/. [cit. 2023-11-25].
[56] *Application and Network Traffic Distribution - Elastic Load Balancing Pricing - AWS*. Online. Cloud Computing Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/elasticloadbalancing/pricing/. [cit. 2023-11-25].
[57] *Amazon SQS Pricing | Message Queuing Service | AWS*. Online. Cloud Computing Services - Amazon Web Services (AWS). C2023. Dostupné z: https://aws.amazon.com/sqs/pricing/. [cit. 2023-12-09].
[58] *Pricing - Service Bus | Microsoft Azure*. Online. Microsoft – cloud, počítače, aplikace a hry. C2023. Dostupné z: https://azure.microsoft.com/en-us/pricing/details/service-bus/. [cit. 2023-12-09].
[59] *Pub/Sub for Application & Data Integration | Google Cloud*. Online. Cloud Computing Services | Google Cloud. Dostupné z: https://cloud.google.com/pubsub. [cit. 2023-12-09].
[60] *VM instance pricing | Compute Engine: Virtual Machines (VMs) | Google Cloud*. Online. Cloud Computing Services | Google Cloud. Dostupné z: https://cloud.google.com/compute/vm-instance-pricing. [cit. 2023-12-09].
[61] *Cloud SQL for MySQL, PostgreSQL, and SQL Server*. Online. Cloud Computing Services | Google Cloud. Dostupné z: https://cloud.google.com/sql?hl=en. [cit. 2023-12-09].
[62] *SQL Server vs MySQL vs Postgresql: Which One Is the Best*. Online. Enterprise Software Development Company - Jelvix. C2023. Dostupné z: https://jelvix.com/blog/mysql-postgresql-sql-server. [cit. 2023-12-09]
[63] *Terraform | HashiCorp Developer*. Online. HashiCorp Developer. Dostupné z: https://developer.hashicorp.com/terraform. [cit. 2024-04-02].
[64] *Docker overview | Docker Docs*. Online. Docker Docs. C2013-2024. Dostupné z: https://docs.docker.com/get-started/overview/. [cit. 2024-04-02].
[65] *Get started with GitHub documentation - GitHub Docs*. Online. GitHub Docs. C2024. Dostupné z: https://docs.github.com/en/get-started. [cit. 2024-04-02].
[66] *Welcome to Flask — Flask Documentation (3.0.x)*. Online. Welcome to Flask — Flask Documentation (3.0.x). C2010. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/. [cit. 2024-04-02].
[67] *Psycopg – PostgreSQL database adapter for Python — Psycopg 2.9.9 documentation*. Online. Psycopg – PostgreSQL database adapter for Python —

Psycopg 2.9.9 documentation. C2001-2021. Dostupné z: https://www.psycopg.org/docs/. [cit. 2024-04-02].
[68] *Learning Python Application Development*. Packt Publishing, 2016. ISBN 1785889192.
[69] *Flask-WTF — Flask-WTF Documentation (1.2.x)*. Online. Flask-WTF — Flask-WTF Documentation (1.2.x). C2010. Dostupné z: https://flask-wtf.readthedocs.io/en/1.2.x/. [cit. 2024-04-02].
[70] *Jinja — Jinja Documentation (3.1.x)*. Online. Jinja — Jinja Documentation (3.1.x). C2007. Dostupné z: https://jinja.palletsprojects.com/en/3.1.x/. [cit. 2024-04-02].
[71] *Chart.js | Chart.js*. Online. Chart.js | Chart.js. Dostupné z: https://www.chartjs.org/docs/latest/. [cit. 2024-04-02].
[72] *Manual*. Online. DataTables | Javascript table library. C2007-2024. Dostupné z: https://datatables.net/manual/. [cit. 2024-04-02].
[73] *Grafana k6 documentation | Grafana k6 documentation*. Online. Grafana: The open observability platform | Grafana Labs. C2024. Dostupné z: https://grafana.com/docs/k6/latest/. [cit. 2024-04-07].

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

URL	Uniform Resource Locator
IP	Internet Protocol
IT	Informační technologie
CSS	Cascading Style Sheets
API	Application Programming Interface
REST	Representational State Transfer
URI	Uniform Resource Identifier
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
RFC	Request for Comments
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
gRPC	Google Remote Procedure Call
RPC	Remote Procedure Call
Protobuf	Protocol Buffers
SQL	Structured Query Language
DML	Data Manipulation Language
DCL	Data Control Language
DDL	Data Definition Language
DQL	Data Query Language
NoSQL	Not Only SQL
VM	Virtual Machine
P2P	Point-to-Point
Pub/Sub	Publish/Subscribe
CPC	Cost per Click
UTM	Urchin Tracking Module
GDPR	General Data Protection Regulation
AWS	Amazon Web Services
GCP	Google Cloud Platform
ID	Identificator
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
FaaS	Function as a Service
CPU	Central Processing Unit
vCPU	Virtual Central Processing Unit
RAM	Random-access memory
ALB	Application Load Balancer

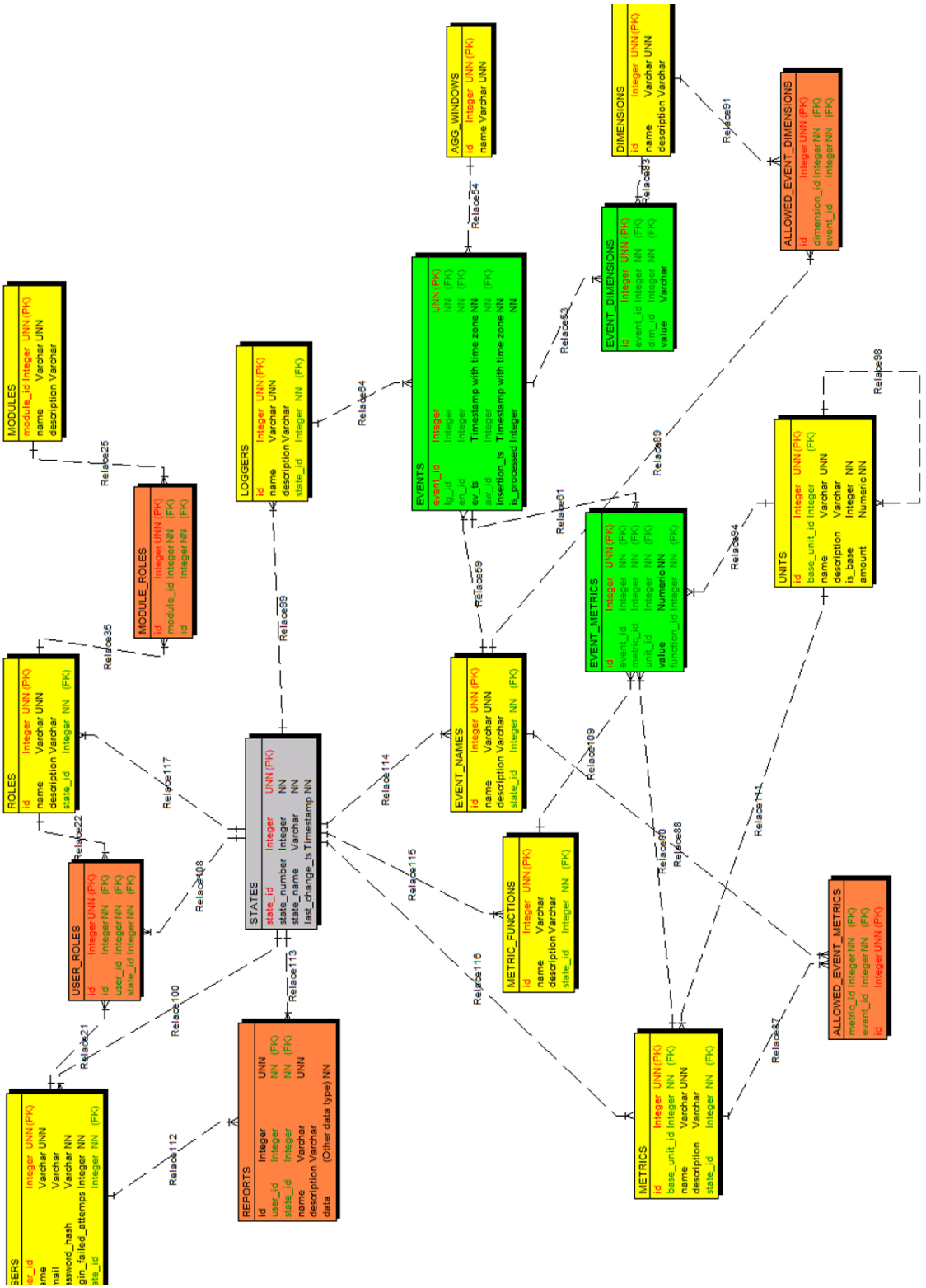
LCU	Load Balancer Capacity Units
HTTPS	Hypertext Transfer Protocol Secure
HTML	Hypertext Markup Language
CSFR	Cross-Site Request Forgery
DNS	Domain Name System
TXT	Text Record
CNAME	Canonical Name Record
VUs	Virtual Users
kB	Kilobyte
MB	Megabyte
GB	Gigabyte
TB	Terabyte
M	Milión
ms	Milisekunda

Symboly:

%	Procento
$	Dolar

SEZNAM PŘÍLOH

Příloha A - Datový model



Příloha B - Obsah elektronické přílohy

- Text diplomové práce ve formátu PDF
- Zdrojové kódy systému