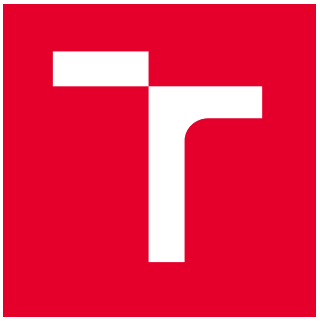


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ

FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY

INSTITUTE OF INFORMATICS

NÁVRH, TVORBA A IMPLEMENTACE SOFTWAROVÉ APLIKACE VE FIREMNÍM PROSTŘEDÍ

DESIGN, CREATION AND IMPLEMENTATION OF SOFTWARE APPLICATIONS IN THE CORPORATE ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Václav Pacal**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Lukáš Novák, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Václav Pacal**
Vedoucí práce:	**Ing. Lukáš Novák, Ph.D.**
Akademický rok:	2022/23
Studijní program:	Informační management

Garant studijního programu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh, tvorba a implementace softwarové aplikace ve firemním prostředí

Charakteristika problematiky úkolu:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je analyzovat, navrhnout a implementovat webovou aplikaci do firemního prostředí.

Základní literární prameny:

GÁLA, L., J. POUR a Z. ŠEDIVÁ. Podniková informatika. 2. přeprac. a aktualiz. vyd. Praha: Grada Publishing, 2009. ISBN 978-80-247-2615-1.

HARDCASTLE, E. Business Information Systems. Ventus Publishing ApS, 2008. ISBN 978-87-7681-463-2.

PRETTYMAN, S. Learn PHP 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. Apress, 2015. ISBN 978-1-4842-1730-6.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. 2. vyd. Brno: Computer

Press, 2000. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2022/23

V Brně dne 5.2.2023

L. S.

doc. Ing. Miloš Koch, CSc. garant	doc. Ing. Vojtěch Bartoš, Ph.D. děkan

Abstrakt

Diplomová práce se zabývá analýzou současného stavu, návrhem a implementací webové aplikace ve firemním prostředí pro zpracování a ukládání dat z aplikace, která nevyužívá relační databáze, ale pracuje s daty založenými na událostech. Aplikace navíc umožňuje export a transformaci těchto dat do externí databáze pro analytickou aplikaci. Součástí diplomové práce je analýza sledování změn dat v databázi a message brokerů. Kromě návrhu a implementace se závěrečná kapitola věnuje i ekonomickému zhodnocení práce.

Klíčová slova

Data založené na událostech, Analytická aplikace, Webová aplikace, Message brokers, Měření výkonu, Apache Kafka, RabbitMQ, Zpracování dat, Škálovatelnost, Transformace dat

Abstract

The thesis deals with the analysis of the current state, design and implementation of a web application in a corporate environment for processing and storing data from an application that does not use relational databases, but works with event-based data. In addition, the application allows the export and transformation of this data to an external database for an analytical application. The thesis includes an analysis of the tracking of data changes in the database and message brokers. In addition to the design and implementation, the final chapter discusses the economic evaluation of the thesis.

Key words

Event-based data, Analytics applications, Web application, Message brokers, Performance measurement, Apache Kafka, RabbitMQ, Data processing, Scalability, Data transformation

Bibliografická citace

PACAL, Václav. Návrh, tvorba a implementace softwarové aplikace ve firemním prostředí. Brno, 2023. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/149764. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Lukáš Novák.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a vypracoval jsem ji samostatně. Prohlašuji, že citace použitých zdrojů je úplná, a že jsem v práci neporušil autorská práva (ve smyslu Zákona č 121/2000 Sb., o právě autorském a o právech souvisejících s právem autorským).

V Brně dne 10. května 2023

……..

Podpis autora

Poděkování

Rád bych poděkoval vedoucímu mé práce Ing. Lukáši Novákovi, Ph.D., za odborné vedení a cenné rady, které mi poskytl. Zároveň bych rád poděkoval své rodině, která mě podporovala během celého studia.

OBSAH

ÚVOD

Vzhledem k rychlosti dnešní doby a snaze optimalizovat ve společnostech veškeré procesy které to umožňují, je obrovský zájem o informační systémy a aplikace které toto umožňují. Nejde však už pouze o co nejhladší běžný provoz společnosti, který je u informačního systému již považován tak nějak za samozřejmost, ale jde i o doplňkové služby, jako je například analýza provozních dat. Nejedná se již pouze o jednorázovou záležitost, kdy je například vytvořen datový sklad z historických dat a zákazník se dozví výstupy z analýzy jednorázově, ale o možnost analyzovat data periodicky, nejlépe v reálném čase, a z těchto dat vytvářet grafy, tabulky – obecně reporty pro optimalizaci procesů a rozhodování.

Společnosti si takové informační systémy a aplikace mohou vyvíjet sami, to však s sebou nese riziko velkých nákladů a výsledek je mnohdy nejistý. Mnohem rozšířenější praxí je nákup licencí některých již existujících systémů a jejich provoz – někdy je dokonce zajištěn provoz v ceně licencí u poskytovatele systému. Jen velmi zřídka jsou tyto systémy využívány samostatně tak, že pokryjí veškeré potřeby společnosti a jen zřídka daný systém přesně odpovídá všem požadavkům. V takových případech je logické, že portfolia nabízených produktů musí být neustále rozšiřována o produkty a služby které jsou žádány.

První kapitola této diplomové práce se bude věnovat teoretickým pojmům, které budou v této práci používány při analýze, návrhu a implementaci.

Druhá kapitola se bude věnovat představení a analýze současného vnitřního a vnějšího prostředí společnosti. Bude obsahovat popis současné situace práce s daty a analýzu dostupných message brokerů.

Třetí kapitola bude obsahovat vlastní návrh a popis implementace samotné aplikace včetně implementace všech sekundárních aplikací a jejich konfiguraci. Kapitola bude také obsahovat analýzu rizik, která s sebou vývoj nese i ve společnosti zaměřující se na vývoj firemních aplikací. Nakonec bude provedeno i ekonomické zhodnocení aplikace.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Cílem této diplomové práce je analyzovat současný stav, navrhnout a implementovat webovou aplikaci, která bude umožňovat několik funkcionalit.

Bude se jednat o zpracování, transformaci a export dat založených na událostech do relační databáze tak, aby bylo možné data používat v analytickém nástroji, který společnost nabízí jako jeden ze svých produktů. K docílení těchto cílů je také nutné určit způsob ukládání a typ uložiště těchto dat založených na událostech. Dílčím cílem je poté změřit časovou náročnost exportu většího množství záznamů dat založených na událostech.

Pro dosažení cílů uvedených výše je tedy nutné splnit následující body:

- Provést analýzu vnitřního a vnějšího prostředí společnosti
- Provést analýzu dostupných message brokerů
- Provést analýzu způsobů sledování změn dat
- Navrhnout a implementovat aplikaci, která bude umožňovat
    - Zpracovat data z message brokeru
    - Strukturovaně ukládat data založená na událostech
    - Export těchto dat do relační databáze
- Provést měření rychlosti exportu dat

1 TEORETICKÁ VÝCHODISKA PRÁCE

Tato kapitola obsahuje teoretický základ pro pochopení této diplomové práce. Je rozdělena do dvou podkapitol.

První podkapitola nese název analytické nástroje, a obsahuje popis nástrojů, které jsou využity ve druhé kapitole k analýze vnitřního prostředí společnosti a také k analýze vnějšího prostředí společnosti.

Druhá podkapitola obsahuje popis technologií, a přístupů využívaných při vývoji a provozu webových aplikací, které jsou využívány v této diplomové práci.

1.1 Analytické nástroje

1.1.1 PESTE analýza

Tato analýza se zaměřuje na strategickou analýzu faktorů vnějšího prostředí, které na společnost mohou působit v podobě příležitostí či hrozeb. Používá se ke zkoumání vnějších vlivů, které na společnost působí. Analýza se skládá z pěti dílčích částí, které mohou mít na společnost největší vliv v budoucnosti.[1]

Zaměřuje se na tyto oblasti:

- P – Political – Politickou
- E – Economical – Ekonomickou
- S – Social – Sociální
- T – Technological – Technologickou
- E – Ecological – Enviromentální

1.1.2 McKinsey 7S

Tato analýza strategického řízení podniku je zaměřena na 7 základních faktorů, které se navzájem ovlivňují a vedou k úspěchu. [2]

- **Strategie** – je v tomto ohledu myšleno jako forma, jak dosáhnout dílčích cílů stanovených společností.
- **Struktura** – lze ji chápat jako pohled na to, jak je společnost členěna a jak spolu jednotlivé elementy komunikují, stanovuje nadřízenost, podřízenost, formu kontroly a sdílení informací napříč společností.
- **Spolupracovníci** – jsou lidé co se podílí na rozhodování nebo na výkonu úkolů ve společnosti
- **Systémy řízení** – lze popsat jako metody, postupy, technologie a techniky řízení společnosti.
- **Sdílené hodnoty** – ukazují základní sociální, hospodářské a kulturní poslání jednotlivých složek společnosti a vytvářejí motivující prostředí.
- **Styl** – je způsob jakým se společnost prezentuje a jakým způsobem komunikuje navenek, ale i uvnitř společnosti, především jak komunikuje vrcholné vedení s ostatními ve společnosti
- **Schopnosti** – je soubor znalostí a dovedností kterými daný kolektiv disponuje

Jako primární faktory podílející se na dosažení úspěchu zde považujeme strategii, strukturu, systémy řízení a spolupracovníci, kdežto sdílené hodnoty, styl a schopnosti jsou považovány za pomocné faktory pro získání úspěchu. Nakonec je však důležitá vyváženost a harmonie mezi všemi faktory.[2]

1.1.3 Porterova analýza

Tato analýza se používá při formulaci strategie společnosti. Jejím cílem je pomocí pěti faktorů zhodnotit postavení a vyjednávací sílu společnosti na trhu.[3]

Zkoumané faktory:

- Síla zákazníků
- Síla dodavatelů
- Hrozba nových konkurentů
- Hrozba substitutů
- Stávající konkurenti

Vyjednávací silou zákazníků je myšleno jejich vliv na nás. Může se jednat o významného zákazníka jehož odchod by způsobil velký odliv prostředků. Nejde přitom jenom o vliv na nás, ale i o vliv na ostatní subjekty v odvětví trhu, pokud má zákazník vliv na jejich rozhodování.[3]

Velkou vyjednávací silou dodavatele lze považovat situaci, pokud má dodavatel na trhu majoritní zastoupení. Také to platí v případech, kdy jeho produkt je velmi specifický a velmi obtížně substituován nebo v případě kdy nejsme pro dodavatele klíčovým odběratelem. [3]

Hrozba vstupu nových konkurentů je vždy vysoká, dá se však za určitých podmínek minimalizovat. Za největší překážku lze považovat vysoké fixní náklady pro vstup do odvětví. Dalším z faktorů, které mohou ovlivnit vstup nového subjektu do odvětví může být existence přirozeného monopolu nebo nutnosti mít vybudovanou distribuční síť či vlastnit nějaké know-how. [3]

Hrozba substitutů se nedá nikdy úplně eliminovat, nicméně je možné ji minimalizovat optimalizací výroby pro snížení nákladů a cenovou politikou a tím udělat substituční produkty pro zákazníka neatraktivní. [3]

Rivalita mezi stávajícími konkurenty v daném odvětví se zvyšuje na základě různých faktorů, mezi ty nejdůležitější patří samotná velikost daného odvětví a také to, jak bude v budoucnu lukrativní. [3]

1.1.4 SWOT analýza

Asi nejznámější a nejpoužívanější analytická metoda zabývající se strategickou analýzou společnosti. Vznikla už koncem 60. a začátkem 70. let 20. století za účelem analyzovat 500 největších společností ve Spojených státech a odhalit tak nedostatky v jejich řízení. Analýza se skládá ze čtyř oblastí, kdy dvě jsou z interního prostředí a dvě externího. Mezi vnitřní faktory patří silné stránky a slabé stránky. Mezi vnější faktory patří příležitosti a hrozby.[4]

- S – Strenghts – Silné stránky
- W – Weaknesses – Slabé stránky
- O – Opportunities – Příležitosti
- T – Threats – Hrozby

Obrázek 1: SWOT analýza

(Zdroj: Vlastní zpracování)

Je důležité říct, že tuto analýzu nelze dělat vždy podle stejného klíče, její tvorba vždy úzce souvisí s účelem, proč vzniká, například v souvislosti zavádění nové služby či produktu na trh. [4]

Další částí SWOT analýzy je vytvoření IFE a EFE matice z jednotlivých faktorů, ohodnocení jejich důležitosti a míru jejich dopadu. [4]

Celková důležitost u jedné kategorie je vždy rovna jedné.
Váha = důležitost * hodnocení.
Celková váha je sumou dílčích faktorů v jedné kategorii. [4]

	Položka	Důležitost	Hodnocení	Váha	Celková váha
S					
W					
O					
T					

Obrázek 2: SWOT analýza - IFE EFE matice

(Zdroj: Vlastní zpracování)

1.2 Technologie

1.2.1 Programovací jazyky

Java

Jedná se o objektově orientovaný programovací jazyk vyšší úrovně, který byl navržen tak, aby byl všestranný. Jednou z předních vlastností javy je standard WORA (Write once, run anywhere), což znamená, že kód je zkompilován do byte kódu a může být spuštěn na jakémkoliv zařízení, které podporuje běh java aplikací a není závislý na operačním systému ani architektuře zařízení.[7]

Java je jedním z nejrozšířenějších programovacích jazyků ve světě a používá se pro mnoho druhů aplikací. Původně byla vyvíjena pouze pro vývoj desktopových aplikací, ale v průběhu let se její cílová skupina rozrostla i na vývoj webových, cloudových i mobilních aplikací. Další z velkých předností javy jsou nástroje, které poskytuje pro správu paměti, které umožňují vyvíjet velmi rychlé a spolehlivé aplikace. Díky architektuře, na které je java postavena, se jedná o velmi bezpečný programovací jazyk. [7]

Java nabízí i velké množství open-source frameworků a knihoven, které usnadňují vývoj v tomto jazyce. Mezi ty nejznámější patří Spring, z kterého vychází Spring Boot, nebo také Hibernate.[8]

Kotlin

Jedná se o staticky typovaný programovací jazyk vyšší úrovně, který podporuje objektově orientovaný přístup, i funkcionální programování. Stejně jako java je koltin kompilován, avšak nabízí možnost kompilace do JavaScriptu nebo JVM. [9]

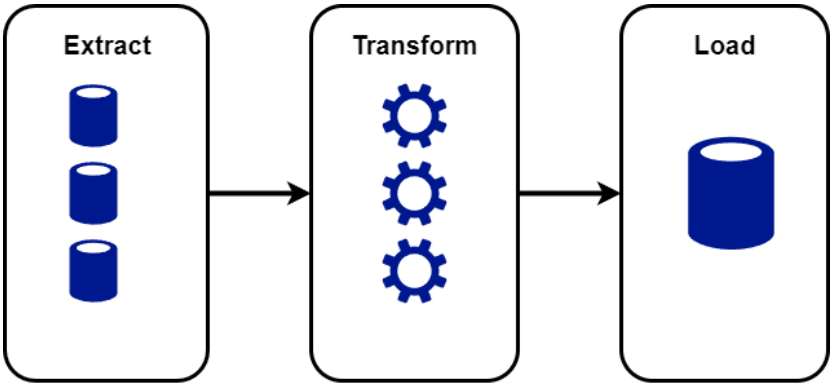
Mezi přednosti, které kotlin například nabízí je podpora lambda funkcí. Samotný jazyk je velmi flexibilní a umožňuje psát čitelný a snadno udržovatelný kód. [9]

Kotlin byl vyvinut a stále je vyvíjen jako alternativa a možná náhrada javy. Je to především tím, že nabízí řešení některých problémů, které jazyk java má. [9]

Kotlin se svojí rychlostí vyrovná jazyku java a jeho architektura rovněž poskytuje vysokou bezpečnost a efektivitu při práci s pamětí. [9]

1.2.2 ETL – Extrakce, transformace a načtení

Jedná se o proces, ke kterému se přistupuje obvykle v případech, kdy společnost nebo jiný subjekt má data uložena na několika místech anebo v různých formátech a struktuře.[6]



Obrázek 3: Metoda ETL

(Zdroj: Vlastní zpracování, podle[5])

Extrakce – V tomto kroku jsou data extrahována ze zdrojového systému. Data jsou poté uchovávána obvykle v podobě relační databáze, avšak mohou být uložena i v NoSQL, tedy nerelační databázi. Mohou být uložena i ve formátu JSON nebo XML. Dochází v tomto kroku také k validaci dat.[5]

Transformace – V tomto kroku jsou data transformována do podoby a struktury, která je podporována cílovým systémem. Data jsou také očištěna. Při transformaci dochází například k unifikaci znakové sady, výpočtu nových hodnot, agregaci dat nebo prostému výběru sloupců, které chceme v cílovém systému. [5]

Load - Nahrání dat do cílového systému. Nahrání může probíhat jako jednoduché vložení nových dat, ale může se také jednat o přepis již stávajících dat. [5]

1.2.3 Spring Boot

Jedná se o open-source framework určený k vytváření převážně webových aplikací, ale umožňuje vývoj i jiných aplikací. Je postaven na frameworkem Spring a poskytuje vývojářům jednoduchou a rychlou cestu ke spuštění vývoje, aniž by museli mnoho času strávit konfigurací. [10]

Spring Boot nabízí nástroj Spring Initializr, který je k dispozici buď jako webová aplikace, nebo ho lze integrovat do různých vývojových prostředí, jako například IntelliJ IDEA. Tento nástroj umožňuje relativně snadnou počáteční konfiguraci projektu a přidání potřebných závislostí. [10]

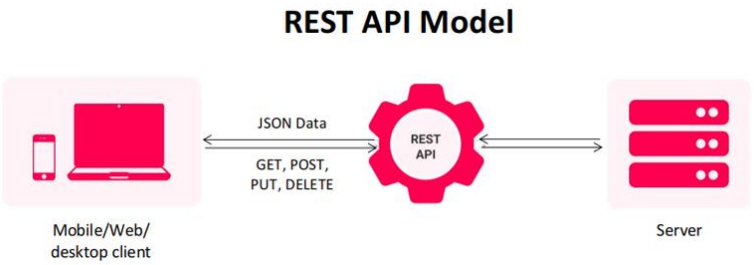
Spring Boot také nabízí širokou škálu sofistikovaných knihoven, které velice zefektivňují vývoj aplikací. Tyto knihovny jsou určeny například pro webový vývoj, práci s databázemi, testování aplikací a mnoho dalších věcí. Použití těchto knihoven minimalizuje množství kódu, který je nutné napsat, a tím umožňuje vývojářům se soustředit na samotnou funkcionalitu aplikace. [10]

1.2.4 REST API

Jedná se o architekturu pro rozhraní aplikace. Dalo by se popsat i jako soubor pravidel, kterými by se komunikace měla řídit. Rest API rozhraní se používá ke komunikaci a výměně dat mezi aplikacemi nebo službami.[11]

Jednou z výhod rest api je jeho flexibilita, jelikož ho je možné vytvořit v jakémkoliv programovacím jazyce a podporuje mnoho datových formátů. Další výhodou je relativně vysoká rychlost komunikace. [11]

Komunikace probíhá tak, že klient odešle request na server, který obsahuje data a informace o tom co za akci s nimi má server vykonat. Server poté ověří, zda má uživatel na takovouto akci oprávnění a pokud ano, akci provede a poté vrátí odpověď, že byla akce provedena. [11]



Obrázek 4: Rest api model

(Zdroj: [11])

1.2.5 JSON

JSON nebo také Javascript Object Notation je zkratka pro formát ukládání a výměnu strukturovaných dat. I přesto, že název obsahuje slovo Javascript, není s tímto jazykem nijak svázán. Umožňuje strukturovat data v lidsky čitelné podobě.[12]

Obrázek 5: Příklad jednoduché JSON struktury

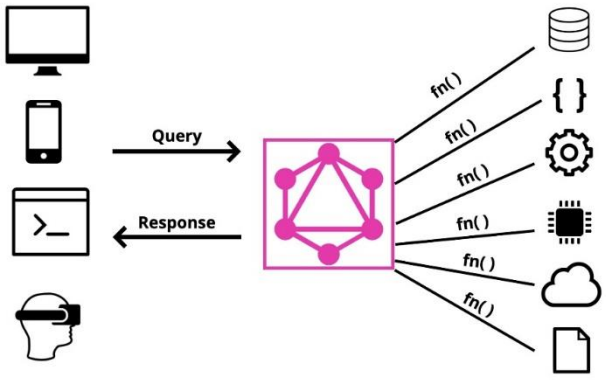
(Zdroj: Vlastní zpracování)

Jak je vidět na obrázku výše, data jsou ve formátu JSON vždy reprezentována v páru „identifikátor:hodnota“ a jednotlivé hodnoty jsou od sebe odděleny čárkou. [12]

Samotná data nemusí být pouze v textovém formátu, ale je možné přenášet i data ve formátu integer a boolean. Obrovskou výhodou formátu JSON je to, že všechny moderní jazyky tento formát podporují a nabízí nástroje, jak snadno tento formát strojově zpracovat. [12]

1.2.6 GraphQL

Jedná se o architekturu pro rozhraní aplikace. Na rozdíl od Rest api se však jedná o dotazovací jazyk, kdy v aplikaci existuje pouze jeden přístupový bod, přes který je možné přistupovat a specifikovat pouze data, která opravdu potřebujeme.[13]



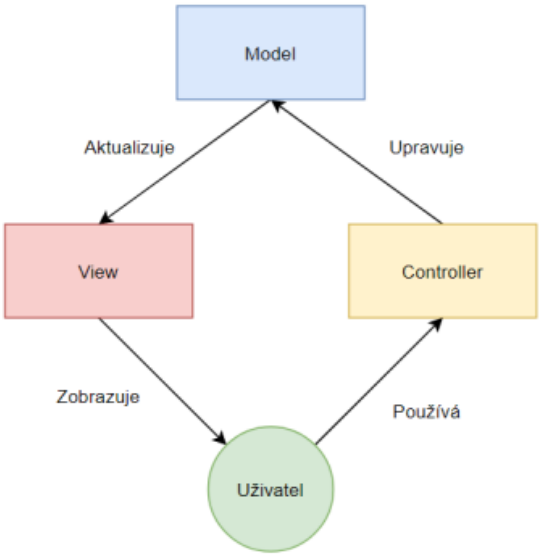
Obrázek 6: GraphQL model

(Zdroj: [13])

1.2.7 MVC

Model-View-Controller, jedná se o architekturu webových aplikací, kdy aplikace je rozdělena do tří částí, které jsou od sebe vzájemně odděleny. To umožňuje upravovat jednotlivé částí aplikace samostatně bez nutnosti větších zásahů do kódu.

Model – Je logická část aplikace, zajišťuje práci s daty, přičemž každá entita má svůj model.
View – Zajišťuje zobrazení dat poskytnutých modelem a umožňuje uživateli interagovat s Controllerem
Controller – Reaguje na vstup od uživatele a umožňuje na základě parametrů od uživatele pracovat s datovým modelem.



Obrázek 7: Model MVC

(Zdroj: Vlastní zpracování, podle [32])

1.2.8 Message Broker

Jedná se o nástroj, který se používá při vývoji aplikací, a který umožňuje komunikaci mezi aplikacemi, systémy a službami. Umožňuje přímou komunikaci mezi navzájem nezávislými systémy a není ovlivňován ani jazykem ve kterém je systém vyvíjen ani platformou na které je provozován. Mezi základní funkcionality každého message brokeru je schopnost validovat, ukládat, směrovat a doručovat zprávy. [14]

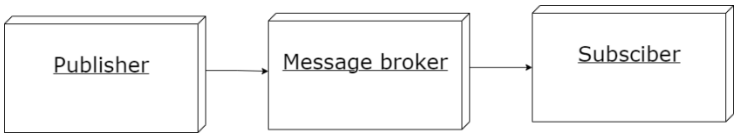
Všechny message brokery umožňují kromě dalších dva základní modely směrování zpráv. Směrování od jednoho zdroje – Publishera k jednomu odběrateli – Subscriberovi a také směrování od jednoho zdroje k několika odběratelům. Toho je docíleno pomocí queues – front. Jelikož zpráva je při čtený z fronty odebrána, pokud není provedeno nějaké nestandartní nastavení, je při více odběratelích nutné směrovat zprávy do několika front. [14]

Publisher

Publisher je v tomto případě označován ten, kdo odesílá zprávy do front v message brokeru. [14]

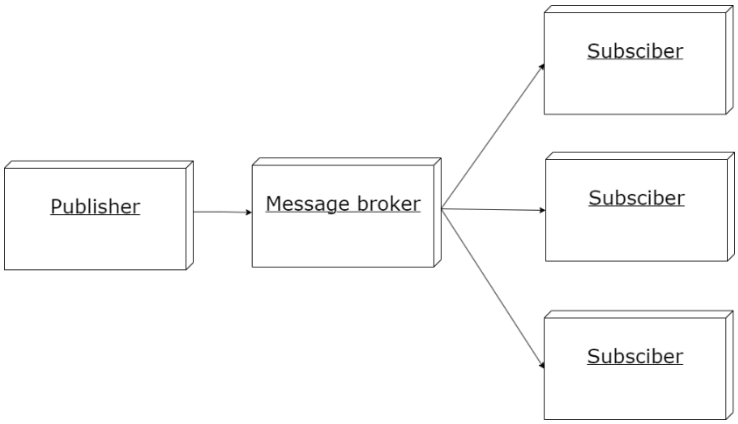
Subscriber

Subsciber je v tomto případě označován ten, kdo zprávy z front čte – konzumuje. [14]



Obrázek 8: Message broker s jedním odběratelem

(Zdroj: Vlastní zpracování, podle [14])



Obrázek 9: Message broker s několika odběrateli

(Zdroj: Vlastní zpracování, podle [14])

1.2.9 RabbitMQ Event listener

Jedná se o konstrukci, která vyčkává na specifickou událost a na tu poté zareaguje. Obvykle se jedná o interakci uživatele, ale může se jednat i o jinou událost. [15]

RabbitMQ event listener je vytvořen pro konzumaci zpráv z fronty. Event listener je postaven na protokolu AMQP, pomocí kterého vytváří spojení s RabbitMQ. [15]

1.2.10 Websocket

Jedná se o komunikační protokol, který umožňuje plně duplexní komunikaci prostřednictvím TCP připojení. Protokol umožňuje udržovat komunikaci mezi klientem a serverem mnohem snadněji a efektivněji než polo duplexní komunikace přes HTTP připojení. [16]

Komunikaci pomocí protokolu websocket je nutné vždy nejprve inicializovat odesláním HTTP požadavku na server na změnu komunikačního protokolu. Po vytvoření připojení je poté připojení udržování již po celou dobu, co je aplikace otevřena a umožňuje oboustrannou komunikaci mezi klientem a serverem v reálném čase. [16]

Pro obsluhování komunikace přes websocket je na straně klienta využíván obvykle jazyk Javascript. Pro obsluhu komunikace na straně serveru je poté možné použít širokou škálu jazyků jako Kotlin, C++, C#, a mnoho dalších. [16]

1.2.11 Docker

Jedná se o open-source kontejnerizační nástroj, který umožňuje zabalit aplikaci do kontejneru. To umožňuje aplikaci v kontejneru spustit a provozovat na jakémkoliv zařízení či v cloudu nehledě o jaké zařízení se jedná či jaký operační systém je na zařízení nainstalován.[17]

Mezi hlavní výhody patří snadná přenositelnost aplikací a konzistence. Dalšími výhodami je vysoká efektivita, škálovatelnost a jednodušší nasazení a správa aplikací.[17]

Docker Desktop

Jedná se o nástroj, který umožňuje vytvářet a spravovat docker kontejnery na lokálním počítači. Mezi hlavní přednosti toho nástroje patří integrace s repositářem oficiálních docker obrazů Docker Hub, kde je k dispozici většina běžně používaných aplikací. Podporuje také Kubernetes a orchestraci kontejnerů.[18]

Docker-compose

Jedná se o nástroj, který umožňuje definovat a spouštět sadu docker kontejnerů pomocí jednoho YAML souboru, což velice usnadňuje vývoj a správu komplexních aplikací. Docker-compose obsahuje definici jednotlivých kontejnerů a konfiguraci jejich enviromentálních proměných, volumes, atd.[19]

Kubernetes

Jedná se o open-source platformu pro orchestraci kontejnerů, která usnadňuje automatizaci nasazení, správu a škálovatelnost kontejnerů. K tomu nabízí širokou paletu dalších nástrojů. Mezi hlavní výhody Kubernetes patří možnost automatického vyrovnávání zátěže a sledování stavu jednotlivých kontejnerů. V případě problémů je schopen kontejnery automaticky restartovat či provést jiné nakonfigurované akce. [17]

1.2.12 Grafana

Jedná se o open-source nástroj pro monitorování a vizualizaci dat a metrik. Umožňuje integraci s mnoha platformami, a tak se jedná o velice flexibilní nástroj. Umožňuje vytvářet komplexní dashboardy pro sledování metrik a jelikož se těší velké oblibě v komunitě vývojářů, je k dispozici i velké množství již předpřipravených dashboardů.[20]

1.2.13 Prometheus

Jedná se o open-source nástroj pro periodický sběr dat. Byl speciálně vyvinutý pro monitorování cloudových prostředí a pro toto monitorování poskytuje širokou paletu nástrojů.[21]

1.2.14 Monolitická architektura

Jedná se o jeden ze způsobů návrhu aplikací, při kterém je aplikace vytvářena jako jeden velký celek. Při této architektuře jsou veškerá rozhraní pro přístup aplikace k databázi uvnitř kódu a obvykle je aplikace exportována do jediného spustitelného souboru. [30]

Taková to architektura s sebou nese mnoho výhod, jako je relativně snadné nasazení aplikace a testování. Většina aplikací vzniká s touto architekturou. V průběhu vývoje však s komplexností aplikace přichází problémy v podobě velikosti a složitosti celé aplikace, kdy i malá změna vyžaduje znalost celého systému, aby nedošlo ke kaskádovým chybám. Další nevýhodou je poté špatná škálovatelnost aplikace, která je díky dnešní majoritně využívané cloudové platformy, velkým problémem. V případě monolitické aplikace musí být škálovatelná celá tato aplikace, a nejen nejvíce vytížené části. [30]

1.2.15 Microservices architektura

Někdy také architektura mikroslužeb, je způsob návrhu aplikací, při kterém je aplikace rozdělena do menších částí, které jsou navzájem propojeny. Toto propojení je prováděno přes běžné komunikační protokoly.[31]

Tato architektura má své přednosti v tom, že umožňuje, aby každá služba, nebo soubor několika služeb, byli vyvíjeny odděleně. To má pozitivní vliv na jejich případnou následující škálovatelnost. Umožňuje také lepší přehlednost kódu pro jednotlivé služby a rychlejší rozhodování o implementaci nových funkcionalit. Mikroslužby však s sebou nesou i jisté nevýhody, a to především v podobě nárůstu složitosti systému, kdy musí být navíc řešena komunikace mezi jednotlivými službami. To má samozřejmě také vliv na zatížení síťové komunikace.[31]

1.2.16 Event-driven architektura

Jedná se o architekturu aplikací, při které jsou stanoveny jisté události zvané eventy. Ty jsou vyvolány prostřednictvím interakce uživatele a obsahují kromě typu eventu také data příslušné změny. Tato architektura je obvykle používána v kombinaci s mikroslužbami, které mezi sebou na základě těchto eventů komunikují. V běžné praxi se poté mezi tyto mikroslužby přidává mezičlánek, takzvaný message broker. Díky němu se dosáhne toho, že komunikace je oddělena, je možné ji směrovat a je asynchronní.[33]

1.2.17 Komunikační protokoly

HTTP a HTTPS

Jedná se o nejrozšířenější komunikační protokol pro přenos dat na internetu, při používání webových aplikací. Protokol funguje na principu navázání komunikace ze strany klienta (například webový prohlížeč) se serverem. Komunikace probíhá vždy tak, že klient odešle požadavek a server vrátí odpověď. [22]

Protokol podporuje několik metod komunikace se serverem.

- GET – získávání dat ze serveru
- POST – odesílání dat na server
- PUT – aktualizace již existujících dat
- DELETE – odstranění dat ze serveru

HTTPS poté rozšiřuje funkcionalitu protokolu HTTP o bezpečnou a šifrovanou komunikaci. [22]

AMQP

Jedná se o open-source komunikační protokol zaměřený na přenos zpráv. Je navržený tak, aby podporoval širokou škálu aplikací. Vyniká především rychlostí a efektivitou. [23]

Jedná se o binární protokol aplikační vrstvy. Kromě přenosu zpráv také umožňuje tyto zprávy směrovat a je určen primárně pro zprostředkovatele zpráv – message brokery.[23]

STOMP

Jedná se o komunikační protokol určený pro asynchronní přenos textových zpráv. Jedná se o jednodušší protokol, nežli je AMQP a umožňuje zasílání jen méně komplexních zpráv než ostatní protokoly. Jeho hlavní předností je však jeho jednoduchost a široká škála možných použití.[24]

MQTT

Jedná se o komunikační protokol pro přenos textových zpráv. Je převážně rozšířen v oblasti komunikace mezi IoT zařízeními. Protokol umožňuje obousměrnou komunikaci a je využíván převážně v situacích, kdy je klíčovým parametrem přenosu zpráv to, aby zprávy byli co možná nejmenší a neobsahovali žádné nadbytečné informace. Mezi přednosti protokolu patří například to, že umožňuje komunikovat i mezi zařízeními, které navážou komunikaci v rozdílném čase.[25]

gRPC

Jedná se o open-source komunikační protokol, který se vyznačuje svojí rychlostí a efektivitou. Jedná se o binární protokol, který umožňuje obousměrnou komunikaci. Tato komunikace může dosáhnout až několika násobně vyšší rychlosti nežli komunikace přes jiné protokoly. Jednotlivé zprávy jsou navíc až o polovinu menší nežli velikosti zpráv u jiných protokolů. [26]

Další výhodou tohoto protokolu je rozsáhlá dokumentace a podpora drtivou většinou programovacích jazyků a frameworků. [26]

2 ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

Tato kapitola se zabývá analýzou společnosti, pro kterou je aplikace vyvíjena. V úvodu této kapitoly je společnost podrobena analýzám vnějšího a vnitřního prostředí. V druhé části kapitoly je popsána současná situace sledování změn v datech. V poslední části jsou poté analyzovány message brokery.

2.1 Analýza společnosti

Tato podkapitola se bude zabývat analýzou společnosti pomocí analýzy vnějšího prostředí PESTE, analýzy vnitřního prostředí 7S, Porterovou analýzou oborového okolí a souhrnnou analýzou SWOT.

2.1.1 Představení společnosti

Společnost se pohybuje v oboru vývoje a poskytování firemních informačních systémů a poskytování konzultačních služeb pro customizaci těchto systémů na míru. Společnost nabízí svoje systémy a platformu, které je možné použít pro téměř všechny nutné úkony pro běh společnosti, jako je například správa zakázek, incident managment, vykazování času atd.

K těmto hlavní produktům nabízí další doplňkové služby, jako je úprava systému přesně na míru či analytický nástroj pro analýzu, včetně konzultantských služeb pro jeho chod, který umožňuje analyzovat data, a tak optimalizovat procesy ve společnosti.

Společnost si nepřála, aby byl uváděn její název, proto odkazy na ni budou uvedeny v této práci pouze jako „Společnost“.

2.1.2 PESTE analýza

Politické faktory

Stabilita politického prostředí: Společnost působí v politicky stabilním prostředí, což zajišťuje předvídatelnost a jistotu pro podnikání. Politická stabilita umožňuje společnosti dlouhodobě plánovat a minimalizovat politická rizika.

Regulace a zákony: Společnost dodržuje pravidla a zákony týkající se oblasti softwarového průmyslu, ochrany spotřebitele, ochrany soukromí a duševního vlastnictví. Dodržování těchto zákonů a předpisů zajišťuje, že je společnost schopna si udržovat svou dobrou pověst na trhu.

Daňová politika: Společnost zohledňuje místní daňové zákony a předpisy, které mohou ovlivnit její finanční výkonnost a náklady na provoz.

Mezinárodní vztahy: Jakožto společnost působící na mezinárodním trhu společnost sleduje politické vztahy mezi zeměmi, ve kterých působí, a ty, které mohou ovlivnit její obchodní činnost. Vztahy mezi zeměmi totiž mohou ovlivnit obchodní příležitosti.

Lokální politické zájmy: Společnost je také obeznámena s místními politickými zájmy a aktivitami, které mohou mít vliv na její obchodní činnost. To zahrnuje strategie hospodářského, rozvoje nebo infrastrukturní projekty, které mohou ovlivnit její konkurenceschopnost, růst ale také příležitosti na zajímavé projekty.

Ekonomické faktory

Lokální ekonomický růst: Rychlost ekonomického růstu v zemi ovlivňuje poptávku po softwarových produktech a službách, které společnost nabízí. Příznivý ekonomický růst v oblasti IT v posledních letech vedl k vyšší poptávce po jejich produktech a službách a umožnil společnosti konstantně růst.

Globální ekonomické trendy: Společnost je ovlivňována i globálními ekonomickými trendy, jako jsou hospodářské cykly nebo fluktuace měn, které mohou ovlivnit její činnost na jiných světových trzích. Společnost sleduje globální ekonomický vývoj a přizpůsobuje mu svoji strategii, aby minimalizovala rizika a využila možných příležitostí na trhu.

Inflace: Inflace a úrokové sazby ovlivňují náklady na kapitál a provoz společnosti. Vysoká inflace nutí společnost například ke zvyšování mezd, a má také vliv na růst nákladů.

Nezaměstnanost a trh práce: Nezaměstnanost a stav na trhu práce ovlivňují schopnost společnosti najít kvalifikované pracovníky a udržet si konkurenční výhodu. Nízká nezaměstnanost tento tlak ještě zvyšuje, jelikož je vyšší zájem o talentované lidi z čehož plynou zvýšené náklady na mzdy.

Hospodářská politika: Společnost sleduje hospodářskou politiku vlády a centrální banky, která může ovlivnit ekonomický růst, inflaci a úrokové sazby.

Sociální faktory

Demografické trendy: Společnost sleduje demografické trendy, jako jsou změny ve věkové struktuře, populační růst a migrační toky. Tyto faktory ovlivňují poptávku po produktech a službách společnosti, stejně jako její schopnost najít a udržet si kvalifikované pracovníky.

Kulturní hodnoty a preference: Jelikož Společnost působí na mezinárodním trhu musí brát v úvahu kulturní hodnoty a preference zákazníků v různých zemích a regionech, ve kterých působí. To zahrnuje respektování místních norem a tradic, přizpůsobení marketingových kampaní a produktových nabídek.

Úroveň vzdělání: Úroveň vzdělání populace ovlivňuje schopnost společnosti najít kvalifikované pracovníky a rozvíjet jejich dovednosti. Vyšší úroveň vzdělání také vede k vyšší poptávce po sofistikovanějších produktech a službách. Společnost podporuje vzdělávání a odbornou přípravu svých zaměstnanců a spolupracuje se vzdělávacími institucemi. Příkladem toho jsou například téměř každý rok vypsaná témata bakalářských a diplomových pracích.

Technologické faktory

Inovace a vývoj: Společnost investuje do inovací a vývoje, aby udržela konkurenční výhodu a přizpůsobila se rychle se měnícím technologickým trendům. Společnost sleduje a analyzuje nové technologie, které by mohly ovlivnit její obchodní model a produkty.
Digitalizace a automatizace: Společnost pečlivě sleduje tyto trendy a implementuje nové technologie, které mohou zlepšit efektivitu, snížit náklady a zlepšit zákaznický zážitek.
Cloud computing a softwarové služby: Cloud computing a softwarové služby nabízejí nové možnosti pro ukládání a zpracování dat, snižování nákladů na IT infrastrukturu a zlepšení flexibility a škálovatelnosti aplikací. Společnost proto provozuje téměř všechny své interní aplikace i produkty jako cloudové služby.
Kybernetická bezpečnost: S narůstajícím významem digitálních technologií a datových systémů se stává kybernetická bezpečnost jednou z klíčových priorit pro Společnost. Společnost investuje do zabezpečení svých systémů a dat a dělá vše co je možné, aby zajistila, že je chráněna před hrozbami, jako jsou hackeři, malware a ransomware.
Umělá inteligence a strojové učení: Umělá inteligence a strojové učení nabízejí nové možnosti pro zpracování a analýzu dat, predikci chování zákazníků a optimalizaci procesů. Společnost sleduje vývoj těchto technologií a zvažuje jejich integraci do svých produktů a služeb, aby získala konkurenční výhodu.

Environmentální faktory

Regulace a zákony týkající se životního prostředí: Společnost se snaží být obeznámena s environmentálními zákony a regulacemi, které ovlivňují její obchodní činnost. To zahrnuje dodržování předpisů týkajících se nakládání s odpady, recyklace a energetické účinnosti. Společnost dále monitoruje změny v environmentálních zákonech a předpisech a zajišťuje, že její aktivity jsou v souladu s těmito požadavky.
Společenský tlak: Společnosti v posledních letech čelí stále většímu tlaku ze strany veřejnosti, zákazníků a investiční komunity, aby byly více environmentálně odpovědné. Společnost zohledňuje tyto očekávání a komunikuje své environmentální cíle a úspěchy prostřednictvím transparentních zpráv o udržitelnosti.

2.1.3 Porterova analýza

Stávající konkurenti

Rivalita mezi konkurenty na trhu vývoje softwaru je vysoká. Existuje mnoho společností, které nabízejí podobné produkty a služby, a silná konkurence je jak v cenách, tak v inovacích. Společnost musí neustále zlepšovat svoji nabídku, zvyšovat efektivitu, snižovat náklady a posilovat svou značku, aby se odlišila od konkurence a zajistila si udržitelný růst a úspěch. Mezi její hlavní konkurenty v oblasti vývoje firemních informačních systémů patří společnosti SAP a ServiceNow.

Síla dodavatelů

Dodavatelé ve vývoji softwaru zahrnují poskytovatele technologických platforem, hardwaru, cloudových služeb a dalších nástrojů a služeb. Síla dodavatelů je střední až nízká, jelikož existuje mnoho alternativních dodavatelů a společnost může snadno přejít k jinému dodavateli, pokud by to bylo nezbytné. Společnost věnuje pozornost dlouhodobým partnerstvím a smlouvám s dodavateli, aby zajistila stabilitu a konkurenceschopné ceny.

Síla zákazníků

Zákazníci v oblasti vývoje softwaru mají vysokou sílu, protože mají širokou škálu možností a mohou relativně snadno přejít ke konkurenci, pokud nejsou spokojeni s produkty nebo službami. Společnost zaměřuje své úsilí na poskytování vysoké kvality, inovativních řešení a vynikající zákaznické podpory, aby si udržela loajalitu svých zákazníků a získala nové klienty.

Hrozba nových konkurentů

Na trhu vývoje softwaru existuje relativně nízká bariéra pro vstup, což zvyšuje hrozbu nových konkurentů. Technologický pokrok, přístup ke zdrojům a relativně snadná dostupnost talentů mohou vést k rychlému nástupu nových hráčů na trh. Společnost neustále investuje do inovací, zlepšuje své produkty a služby a buduje silné vztahy se zákazníky, aby minimalizovala tuto hrozbu.

Hrozba substitutů

Hrozba substitutů nebo služeb je v oblasti vývoje softwaru vysoká. Rychlý technologický pokrok a inovace znamenají, že se neustále objevují nové produkty a služby, které mohou nahradit stávající nabídku společnosti. Společnost se soustředí na sledování technologických trendů, rozvoje inovativních řešení a udržení rychlého tempa vývoje, aby zůstala konkurenceschopná a minimalizovala hrozbu substitučních produktů nebo služeb.

2.1.4 McKinsey 7S

Strategie

Důležitou součástí strategie společnosti je diverzifikace produktů a služeb, aby byla schopna uspokojit širokou škálu zákaznických potřeb a přizpůsobit se měnícím tržním podmínkám. To zahrnuje rozvoj nových softwarových řešení, které řeší aktuální problémy a výzvy zákazníků, a vstup do nových oborů a odvětví, kde mohou být její zkušenosti využity.

Kromě toho strategie společnosti zahrnuje silné partnerství a spolupráci s klíčovými hráči v oboru, jako jsou dodavatelé, technologičtí partneři a zákazníci. Tímto způsobem může společnost získat přístup k novým tržním příležitostem, sdílet znalosti, zdroje a spolupracovat na vývoji nových produktů a řešení.

Strategie společnosti také zohledňuje ekologické a sociální aspekty podnikání, například tím, že se soustředí na snižování svého ekologického dopadu, podporu sociálně odpovědných praktik. Tímto způsobem si společnost buduje silný a udržitelný obchodní model, který přispívá k celkovému růstu společnosti a zajišťuje dlouhodobý úspěch na trhu.

V rámci své strategie společnost také pečlivě sleduje výkonnost a úspěšnost svých projektů, aby zajistila, že její zdroje jsou efektivně využívány k dosažení stanovených cílů. To zahrnuje pravidelné hodnocení a zlepšování interních procesů.

Struktura

Jedním z klíčových aspektů struktury společnosti je členění do týmů a oddělení, které se specializují na různé oblasti podnikání, jako je vývoj software, prodej, marketing, zákaznická podpora, financování a lidské zdroje. Tyto týmy jsou schopny úzce spolupracovat a koordinovat svoji činnost, aby dosáhly společných cílů a zajistily hladký průběh projektů.

Kromě toho společnost investuje do rozvoje firemní kultury, spolupráce a otevřené komunikace mezi svými zaměstnanci. Tímto způsobem mohou být názory a nápady sdíleny napříč organizací. Tato firemní kultura podporuje inovace a kreativitu, což je nezbytné pro udržení konkurenceschopnosti společnosti v rychle se měnícím tržním prostředí.

Dalším aspektem struktury společnosti je řízení a vedení. Společnost má silné a schopné vedení, které je schopno inspirovat a motivovat zaměstnance a vést společnost k dosažení firemních cílů. Vedení má velkou důvěru v management jednotlivých poboček, a proto jsou jednotlivé kanceláře poměrně individuální.

Styl

Jedním z klíčových aspektů stylu řízení, který společnost provozuje, je participativní přístup. To znamená, že vedení společnosti umožňuje zapojovat zaměstnance do rozhodovacích procesů, umožňovat jim vyjadřovat své názory a nápady a aktivně se podílet na vytváření a realizaci strategií a projektů. Tento přístup podporuje pocit vlastnictví a zodpovědnosti mezi zaměstnanci a pomáhá vytvářet prostředí, ve kterém se cítí cenní a zapojení.

Kromě toho je styl řízení ve společnosti pružný a adaptabilní, aby se mohl přizpůsobit měnícím se podmínkám trhu, různým potřebám a očekáváním zaměstnanců. To zahrnuje schopnost přijmout nové přístupy a techniky, experimentovat s různými metodami řízení a učit se z chyb a úspěchů.

Dalším důležitým aspektem stylu řízení ve společnosti je transparentnost. Vedení je otevřené a upřímné při sdílení informací o finančních výsledcích, strategických plánech, úspěších a neúspěších společnosti. Transparentnost pomáhá vytvářet důvěru a soudržnost mezi zaměstnanci a umožňuje jim lépe porozumět směřování a cílům společnosti.

Styl řízení také podporuje profesní rozvoj a růst zaměstnanců. To zahrnuje poskytování pravidelné zpětné vazby o výkonnosti a poskytování příležitostí pro vzdělávání a rozvoj dovedností.

Systémy

Jelikož společnost se věnuje vývoji firemních informačních systémů, tak i její vlastní systémy jsou velmi kvalitní a robustní. Mezi klíčové vlastnosti interních systémů patří jejich efektivita a schopnost distribuovat informace napříč celou organizací.

Kromě interních systémů společnost využívá i některé systémy třetích stran jako je kancelářský balíček Office 365 a další systémy třetích stran používané primárně při vývoji a správě produktů. Mezi tyto systémy patří například vývojová prostředí Visual Studio Code nebo IntelliJ IDEA.

Sdílené hodnoty

Jednou z klíčových hodnot, které společnost propaguje, je zákaznická orientace. To znamená, že společnost vždy usiluje o to, aby byly potřeby a očekávání zákazníků na prvním místě, a to jak při vývoji produktů a služeb, tak při poskytování podpory a péče o zákazníka po prodeji. Zaměstnanci jsou neustále povzbuzováni, aby se zaměřovali na zlepšování zákaznických zkušeností a usilovali o vytváření dlouhodobých vztahů se zákazníky.

Další důležitou hodnotou, kterou společnost podporuje, je inovace. Ve světě technologií a softwarových řešení je neustálý vývoj a zlepšování produktů a služeb klíčovým faktorem úspěchu. Zaměstnanci jsou povzbuzováni k myšlení mimo zaběhnuté koleje, zkoumání nových nápadů a přístupů a sdílení svých nápadů a zkušeností s ostatními v organizaci.

Týmová práce a spolupráce jsou dalšími základními hodnotami, které jsou ve společnosti. To znamená, že zaměstnanci jsou povzbuzováni k úzké spolupráci a vzájemné podpoře, aby řešili problémy a dosáhli společných cílů. Tato hodnota podporuje otevřenou komunikaci, sdílení znalostí a zdrojů a vzájemné učení.

Spolupracovníci

Spolupracovníci představují klíčový zdroj společnosti, protože jejich dovednosti, odborné znalosti a zkušenosti jsou nezbytné pro úspěšný vývoj a implementaci softwarových řešení. Pro společnost je důležité, aby si udržovala vysokou úroveň kvalifikace a motivace svých zaměstnanců, aby zajistila konkurenceschopnost a růst na trhu.

Společnost má efektivní proces náboru, který zajišťuje rychlé přizpůsobení se měnícím technologickým trendům a získání talentovaných odborníků. Společnost hledá zaměstnance nejen s technickými dovednostmi, ale také s komunikačními a týmovými schopnostmi, které jsou důležité pro spolupráci a koordinaci projektů.

Společnost investuje také do školení a rozvoje svých zaměstnanců, aby zajistila, že budou mít znalosti a dovednosti potřebné pro úspěšné zvládnutí svých pracovních úkolů. Tím se navýší efektivita práce a spokojenost zaměstnanců, což může vést k lepším výsledkům a loajalitě zaměstnanců.

Schopnosti

Společnost disponuje širokou škálou technických dovedností a odborných znalostí v oblasti vývoje softwaru, což jí umožňuje poskytovat špičkové softwarové řešení a služby svým zákazníkům. Společnost neustále aktualizuje a rozšiřovat své technické dovednosti, aby se přizpůsobila měnícím se technologickým trendům a zůstala konkurenceschopná.

Společnost má silné schopnosti v oblasti řízení projektů, což většinou zajišťuje úspěšné dodání projektů v rámci rozpočtu, dohodnutého času a požadavků zákazníků.

Společnost prokazuje schopnost inovovat a přinášet kreativní řešení pro své zákazníky. Společnost podporuje kulturu inovací a kreativity, což umožňuje rychlejší přizpůsobení se tržním změnám.

Společnost také efektivně řídí znalosti, aby zachovala a sdílela klíčové know-how a zkušenosti mezi zaměstnanci. Systém řízení znalostí zahrnuje interní databáze, dokumentaci a školení, které pomáhají přenášet důležité informace a zkušenosti mezi jednotlivými týmy a generacemi zaměstnanců.

2.1.5 SWOT analýza

Silné stránky

Globální dosah: Společnost má pobočky ve více než 40 zemích, což jí umožňuje efektivně reagovat na místní potřeby zákazníků a získávat přístup k novým trhům.
Diversifikovaná nabídka služeb: Společnost nabízí širokou škálu služeb, včetně konzultací, technických služeb, cloudových služeb a outsourcingu, což jí umožňuje odlišit se od konkurence a snižuje riziko závislosti na jednom tržním segmentu.
Odborné znalosti a zkušenosti: Společnost má tým odborníků s hlubokými znalostmi a zkušenostmi v různých průmyslových odvětvích, což jí umožňuje poskytovat vysoce kvalitní a přizpůsobené služby svým zákazníkům.
Silná značka a reputace: Společnost má dobře známou značku a vynikající reputaci na trhu IT konzultací a služeb, což přitahuje nové zákazníky a udržuje loajalitu stávajících klientů.
Strategické akvizice a partnerství: Společnost aktivně hledá akvizice a partnerství, která rozšiřují její dovednosti a služby, zvyšují zákaznickou základnu a posilují její konkurenceschopnost.

Slabé stránky

Závislost na velkých klientech: Společnost může být závislý na několika velkých klientech, což může způsobit nestabilitu v příjmech, pokud by některý z těchto klíčových klientů ukončil své smlouvy nebo snížil objem svých zakázek.
Vysoké náklady na služby: Společnost je obecně vnímána jako společnost s vyššími náklady na služby ve srovnání s některými konkurenty. To může být nevýhodou při soutěžení o cenově citlivé zakázky.
Komplexní organizační struktura: Společnost má velkou a komplexní organizační strukturu, což může někdy způsobovat byrokratické zpoždění a omezenou rychlost při rozhodování a provádění změn.
Konkurence s nízkorozpočtovými poskytovateli: Společnost čelí rostoucí konkurenci od nízkorozpočtových poskytovatelů služeb, kteří mohou nabízet srovnatelné služby za nižší ceny, což může ohrozit tržní podíl společnosti.

Příležitosti

Růst poptávky po digitální transformaci: Stále více společností usiluje o digitální transformaci svých procesů a operací, což představuje obrovskou příležitost pro společnost poskytovat své konzultační a technické služby.

Cloudové služby: Cloud computing je rychle rostoucí trh s vysokým potenciálem. Společnost rozšiřuje své služby v oblasti cloudových řešení a snaží se získat podíl na tomto rostoucím trhu.

Kybernetická bezpečnost: Vzhledem k rostoucím hrozbám kybernetické bezpečnosti a potřebě společností chránit svá data a systémy, se kybernetická bezpečnost stává stále důležitějším odvětvím. Společnost se snaží rozšířit svoji nabídku v oblasti kybernetické bezpečnosti a poskytovat špičková řešení svým klientům.

Rozvoj umělé inteligence a strojového učení: Umělá inteligence a strojové učení jsou klíčovými technologiemi budoucnosti. Společnost zkoumá příležitosti, jak investovat do výzkumu a vývoje těchto technologií a začlenit je do svých služeb a produktů.

Růst na rozvojových trzích: Rozvojové trhy, jako jsou Asie, Latinská Amerika a Afrika, nabízejí obrovský potenciál pro růst IT služeb a konzultací. Společnost může expanzí do těchto regionů získat přístup k novým trhům a zákazníkům.

Hrozby

Silná konkurence: Společnost čelí silné konkurenci od ostatních velkých IT společností v tomto odvětví. Tyto společnosti soutěží o tržní podíl a mohou ovlivnit ceny a ziskovost.
Technologie: Rychlý vývoj technologií a průmyslových inovací představuje hrozbu pro společnost, pokud nedokáže udržet krok s těmito změnami a inovovat své produkty a služby.
Nízkorozpočtoví konkurenti: Společnost čelí rostoucí konkurenci od nízkorozpočtových poskytovatelů služeb, kteří mohou nabízet srovnatelné služby za nižší ceny. To může ohrozit tržní podíl a ziskovost společnosti.
Změny v právním prostředí: Změny v právním prostředí mohou mít vliv na společnost, zejména v oblasti ochrany dat, kybernetické bezpečnosti a mezinárodního obchodu. Společnost musí zajistit, aby byla v souladu se všemi platnými zákony a nařízeními.
Nedostatek talentů: IT odvětví je charakteristické vysokou poptávkou po odbornících s technickými dovednostmi a zkušenostmi. Společnost čelí hrozbě nedostatku kvalifikovaných pracovníků, což může omezit jejich schopnost růst a poskytovat služby.

Interní	
Silné stránky	**Slabé stránky**
Globální dosah	Závislost na velkých klientech
Diversifikovaná nabídka služeb	Vysoké náklady na služby
Odborné znalosti a zkušenosti	Komplexní organizační struktura
Silná značka a reputace	Konkurence s nízkorozpočtovými poskytovateli
Strategické akvizice a partnerství	
Externí	
Příležitosti	**Hrozby**
Růst poptávky po digitální transformaci	Silná konkurence
Cloudové služby	Technologie
Kybernetická bezpečnost	Nízkorozpočtoví konkurenti
Rozvoj umělé inteligence a strojového učení	Změny v právních a regulatorních prostředích
Růst v rozvojových trzích	Nedostatek talentů

Obrázek 10 SWOT analýza

(Zdroj: vlastní zpracování)

Tabulka 1: IFE EFE matice (Zdroj: vlastní zpracování)

	Položka	Důležitost	Hodnocení	Váha	Celková váha
S	Globální dosah	0.15	4	0.6	1.99
	Diversifikovaná nabídka služeb	0.13	3	0.39	
	Odborné znalosti a zkušenosti	0.09	4	0.36	
	Silná značka a reputace	0.1	4	0.4	
	Strategické akvizice a partnerství	0.08	3	0.24	
W	Závislost na velkých klientech	0.2	2	0.4	0.9
	Vysoké náklady na služby	0.1	2	0.2	
	Komplexní organizační struktura	0.05	2	0.1	
	Konkurence s nízkorozpočtovými poskytovateli	0.1	2	0.2	
O	Růst poptávky po digitální transformaci	0.1	4	0.4	1.44
	Cloudové služby	0.07	2	0.14	
	Kybernetická bezpečnost	0.1	3	0.3	
	Rozvoj umělé inteligence a strojového učení	0.1	3	0.3	
	Růst v rozvojových trzích	0.15	2	0.3	
T	Silná konkurence	0.15	3	0.45	1.22
	Technologie	0.08	2	0.16	
	Nízkorozpočtoví konkurenti	0.11	3	0.33	
	Změny v právních a regulatorních prostředích	0.05	2	0.1	
	Nedostatek talentů	0.09	2	0.18	

2.2 Zdrojová aplikace – Service desk

Jako zdrojová aplikace je v kontextu této práce myšlena aplikace, která produkuje data. Bude se používat jako podpůrná aplikace pro zaměstnance oddělení supportu při řešení problémů incident managementu, nebo i dalších oblastí v budoucnu.

Bude umožňovat vytvořit záznam o vzniku, průběhu a uzavření incidentu či jiné aktivity. Tato aplikace je v současné době ve vývoji, avšak jak již bylo zmíněno, vzhledem k potřebě škálovatelnosti aplikací a udržitelnosti kódu produktů je aplikace vyvíjena také jako mikroslužba.

2.2.1 Architektura aplikace

Jedná se tedy o mikroslužbu, která bude postavena, na rozdíl od v dnešní době majoritně rozšířené objektově orientované architektuře, na principu event-driven architektury. Vnitřní fungování zdrojové aplikace však není pro tuto práci klíčové, jelikož formát dat, která budou produkována to dále nijak neovlivňuje.

2.2.2 Formát produkovaných dat

Data, která budou produkována zdrojovou aplikací jsou ve formátu JSON. Na následujícím obrázku je možné vidět ukázku těchto dat. Data jsou strukturována do několika částí, první z nich je eventType, tento atribut určuje, o jaký typ eventu se jedná, dalším atributem jsou data, pod tímto atributem se nachází struktura odesílaných dat, posledním atributem jsou metadata, ten obsahuje záznam o čase vzniku eventu, kdo ho vyvolal a k jakému streamu se tato změna pojí.

Obrázek 11: Ukázka dat zdrojové aplikace

(Zdroj: Vlastní zpracování)

2.3 Cílová aplikace - Analytics

2.3.1 Současný stav

V současné době neexistuje mezi aplikacemi žádné propojení, avšak částečně by se jako současný stav dalo popsat propojení mezi Analytics a jinými aplikacemi v ekosystému společnosti.

Propojení mezi těmito aplikacemi je v současné době v podobě importu dat do databáze Analytics přímo ze zdrojových databází jednotlivých aplikací. K napojení na jednotlivé databáze je využíván příslušný JDBC konektor.

Aplikace Analytics je schopna importovat a používat ke svému provozu všechny nejvíce rozšířené relační databáze. Mezi tyto databáze patří MariaDB, MySQL, MSSQL, Oracle a Columnstore.

2.4 Změny dat v relačních databázích

Vzhledem k tomu, jak fungují relační databáze a jak jsou v nich data ukládána je dostupný vždy aktuální stav jednotlivých záznamů a jejich atributů. To však není úplně vhodné a výhodné v případě, že chceme mít přehled o tom, jak byl záznam měněn v historii. Tato potřeba vzniká jak z bezpečnostních důvodů, tak z důvodů ekonomických, kvůli kterým často chceme analyzovat průběh například incidentu. Tento stav nemá v relačních databázích jednoznačné řešení, avšak je možné ho řešit několika způsoby, nejčastěji je možné se setkat s takzvaným „Audit trail“ způsobem.

Audit trail, by se dal popsat jako systém sledování změn v databázi, který zaznamená jakékoliv úpravy dat. Pokaždé, když je nějaký záznam upraven, vytvoří se v samostatné auditní tabulce záznam, který obsahuje informace o tom, jaká byla provedena změna, kdy byla provedena a kým byla provedena. Záznam může obsahovat celý záznam nebo také jen data, která byla změněna.

Auditní tabulky tak umožňují sledovat historii toho, jak byli data v minulosti měněna. Analýza těchto dat tak může odhalit neoprávněné změny dat nebo chyby v systému.

2.5 Změny dat v Event driven databázi

Tato architektura ukládání dat není tak rozšířená jako ukládání dat v relačních databázích, avšak nabízí mnoho pozitivních funkcionalit. Vzhledem k tomu, že data nejsou přepisována tak, aby zaznamenávali pouze současný stav, ale každá změna dat je nativně zaznamenána a data jsou tak uložena jako stream po sobě jdoucích událostí, plyne z toho několik výhod.

Mezi hlavní výhody tedy patří nativně dostupná možnost sledovat průběh toho, jak byla data měněna v čase. Tyto změny jsou navíc ukládány jako neměnné záznamy, proto tento způsob ukládání dat přináší i vyšší stupeň bezpečnosti.

Tento způsob ukládání dat má však i svá úskalí, a to především v tom, že architektura aplikace, která taková data produkuje musí být těmto datům přizpůsobena a může být i komplexnější než běžná objektově orientovaná architektura.

2.6 Analýza dostupných message brokerů

Jelikož se očekává, že aplikace ServiceDesk bude komplexní a bude využívána velkým množstvím uživatelů, jeví se jako výhodné z pohledu výkonu a škálovatelnosti systému částečně oddělit ukládání dat od samotné aplikace. Dalším důvodem je to, že zdrojová aplikace je postavena na event-driven architektuře. V tomto případě přichází na řadu implementace message brokeru.

2.6.1 RabbitMQ

RabbitMQ patří mezi nejrozšířenějších message brokery vůbec. Je hojně využíván pří vývoji jak malými společnostmi, tak velkými konglomeráty. Mezi jeho hlavní přednosti patří jeho výkonnost, škálovatelnost a relativně snadná konfigurace, která i přesto nabízí širokou škálu nastavení. [27]

I přes svou na první pohled jednoduchost se jedná o jeden z nejsofistikovanějších message brokerů. Skládá se ze čtyř základních částí, publisher, subscriber, exchange a queues - front, a pracuje na principu toho, že jednotlivé zprávy jsou Publisherem odesílány do Exchange, která zprávy distribuuje mezi jednotlivé fronty, se kterými je spojena, a to automaticky nebo na základě směrovacího klíče. Z těchto front jsou poté zprávy čteny příslušnými Subscribery. [27]

Základní typy Exchanges:

- AMQP default – výchozí exchange, která je automaticky napojena na všechny fronty, a směrování jednotlivých zpráv je řízeno pouze na základě směrovacího klíče.
- amq.direct – Zprávy jsou směrovány striktně na základě směrovacího klíče a tato exchange je vhodná především pro přímé odesílání zpráv do jedné fronty.
- amq.fanout – Zprávy jsou směrovány automaticky do všech front napojených na tuto exchange. Směrovací klíč je v tomto případě ignorován.
- amq.topic – Zprávy jsou směrovány na základě směrovacího klíče a jsou odeslány do všech front, které jsou na tuto exchange napojeny a zároveň se směrovací klíč shoduje

Všechny typy front pracují na principu FIFO, first in first out, tedy že zprávy jsou čteny v pořadí, v jakém byli do fronty přidávány. Každá zpráva, která je takto z fronty přečtena a není označena atributem ohledně jejího zpracování je z fronty tímto čtením odebrána. Pokud zpráva obsahuje atribut pro potvrzení o zpracování zprávy, a zpráva není potvrzena, je možné vytvořit frontu kam je tato zpráva následně přesunuta. [27]

Základní typy front:

- Classic – výchozí typ fronty, který poskytuje všechny základní nastavení jako je perzistence zpráv, prioritizaci zpráv nebo třeba TTL (time-to-live), což určuje dobu životnosti zprávy, pro kterou je zpráva relevantní, tato fronta má nejlepší optimalizaci, a proto nabízí nejvyšší propustnost a latenci

- Quorum – je speciální typ fronty, který je určen pro případy, kdy je nutné dosáhnout nejvyšší možné spolehlivosti a bezpečnosti, data jsou rovněž ukládána perzistentně, jak ve frontě typu „Classic“, ale jsou ukládána pomocí speciálního algoritmu Raft, tato bezpečnostní opatření se však mohou projevit na propustnosti a latenci systému

- Lazy – je variantou výchozí fronty, který je určen pro případy, kdy nejsou zprávy zpracovávány okamžitě, ale je nutné je udržovat v paměti po delší, pro tento případ jsou zprávy rovnou automaticky ukládány na disk namísto jejich udržování v operační paměti

- Priority – je variantou výchozí fronty, která navíc kromě základních funkcí umožňuje zprávy zpracovávat podle priority, která je uvedena v hlavičce zprávy

Kromě výše uvedených funkcionalit poskytuje RabbitMQ, bez nutnosti další konfigurace, poměrně rozsáhlé možnosti sledování výkonu a tím umožňuje případnou optimalizaci práce s tímto message brokerem. [28]

Podporované protokoly komunikace:

- AMQP
- STOMP
- MQTT

2.6.2 Apache Kafka

Apache Kafka je dalším z velice rozšířených message brokerů. Jeho hlavní předností je vysoká propustnost a nízká latence, avšak jeho primární případy užití jsou spíše než pro komunikaci mezi systémy distribuování zpráv velkému množství subscriberů. Další nevýhodou je relativně velká komplexnost celého systému, jehož konfigurace je poměrně rozsáhlá. Naopak zajímavou výhodou tohoto message brokeru je, že zprávy nejsou procesem čtení odebrány, což umožňuje kdykoliv v budoucnu jejich opětovné zpracování v pořadí, v jakém byli vytvořeny.[29]

Apache Kafka má podobnou strukturu jako RabbitMQ, a skládá se také ze čtyř základních částí. Tyto části jsou producer, brokers, topics, consumers. Producera a consumera není nutné znovu popisovat, jelikož byli popsány již výše. Brokers zde poté zastává stejnou roli jako exchange, a to že směruje zprávy k jednotlivým frontám – v tomto případě jsou označovány jako topics. [29]

Na rozdíl od exchanges, brokers mají pouze jeden typ, ale umožňuje provádět téměř totožné konfigurace.

Základní funkcionality Brokers:

- Message storage – ukládání zpráv do jednotlivých topiců
- Replication – vytváření duplikátů příchozích zpráv, pro dosáhnutí nejvyšší možné bezpečnosti
- Topic management – veškerá konfigurace topiců je prováděna prostřednictvím brokers, což umožňuje na jednu stranu jednodušší správu, avšak může to mít vliv na velikost celého systému a tím druhotně i na propustnost a latenci

Message routing – směrování zpráv je prováděno na základě směrovacích klíčů, stejně jako u RabbitMQ[29]

Podporované protokoly komunikace:

- TCP

2.6.3 Implementace vlastního message brokeru

Jako jedna z variant se nabízí i implementace vlastního message brokeru a jeho optimalizace přesně na míru. Avšak tato varianta byla zavrhnuta již na počátku, jelikož vývoj, údržba a optimalizace takové aplikace by byla vzhledem k zdarma dostupným alternativám finančně neúnosná a prakticky zbytečná.

2.7 Souhrn analýz

Z analýz v první části této kapitoly je patrné, že společnost má potenciál pro růst a vývoj nových aplikací a že si to vzhledem ke své stabilitě může dovolit. Z analýzy také plyne, že data založená událostech přirozeně zaznamenávají historické změny a po transformaci je možné je využívat pro analytické účely. V poslední řadě na základě analýzy vyplývá, že RabbitMQ se jeví jako nejvhodnější volba z dostupných message brokerů.

3 VLASTNÍ NÁVRH ŘEŠENÍ, PŘÍNOS PRÁCE

Tato kapitola se věnuje vlastnímu návrhu a implementaci aplikace, analýze rizik a ekonomickému zhodnocení aplikace. Popisuje také implementaci a konfiguraci sekundárních aplikací nutných pro chod aplikace. Kromě toho také obsahuje analýzu rizik a návrh protiopatření k těmto rizikům.

3.1 Vybrané technologie

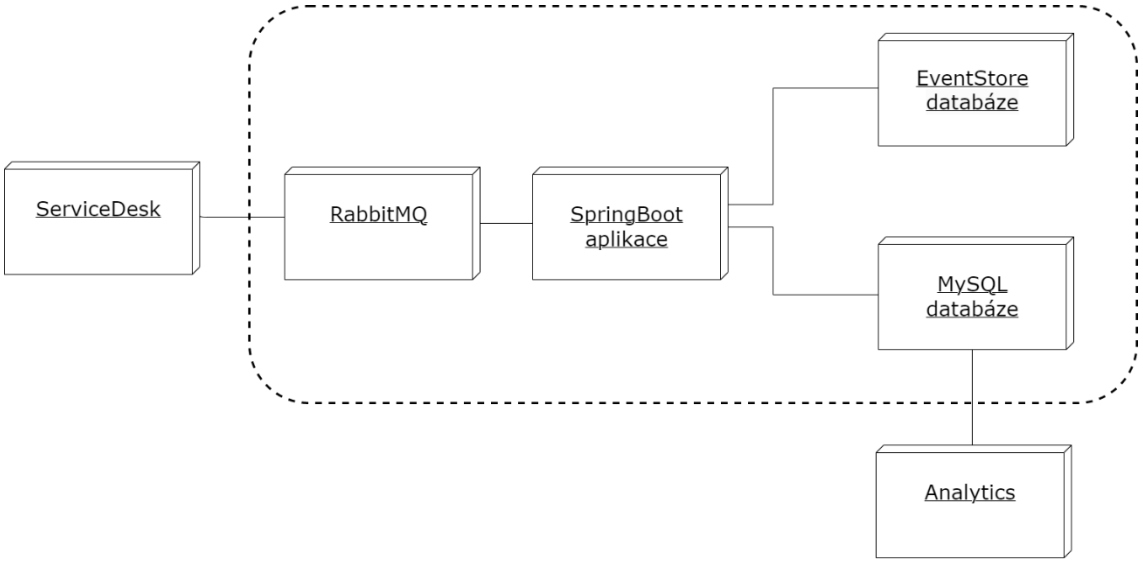
Na základě analýzy bylo rozhodnuto, že mezi aplikaci ServiceDesk a vyvíjenou aplikaci bude umístěn message broker – konkrétně message broker RabbitMQ, který se jeví jako nejvhodnější pro tento typ aplikace.

Jako nejvhodnější úložiště pro data se jeví databáze EventStore, která byla pro tento typ dat a architekturu přímo vyvinuta.

Jelikož aplikace Analytics umožňuje import dat do několika typů relačních databází, jako je MariaDB, MySQL, MSSQL, Oracle a Columnstore bylo zvoleno, že data budou importována do databáze MySQL, jelikož se jedná o prakticky nejrozšířenější databázi v současné době a vytvořené řešení je poté relativně snadno přenositelné i pro ostatní databáze.

Architektura aplikace

Na obrázku níže je možné vidět ohraničený návrh architektury celé implementace.



Obrázek 12: Architektura implementovaného celku

(Zdroj: Vlastní zpracování)

3.2 Analýza rizik

Realizace každého projektu s sebou nese vždy jistá rizika. V případě vývoje webové aplikace jsou to především rizika v podobě špatně zvolených technologií nebo špatného plánování. Kromě těchto rizik však existuje i mnoho dalších.

Následující tabulky obsahují slovní a číselné ohodnocení kritérií analýzy rizik.

Tabulka 2: Hodnota pravděpodobnosti (Zdroj: Vlastní zpracování)

Hodnota	Procentuálně	Slovní hodnocení
1 - 2	0% - 19%	Velmi nepravděpodobné
3 - 4	20% - 39%	Nepravděpodobné
5 - 6	40% - 59%	Pravděpodobné
7 - 8	60% - 79%	Více pravděpodobné
9 - 10	80% - 100%	Velmi pravděpodobné

Tabulka 3: Dopad rizika (Zdroj: Vlastní zpracování)

Hodnota	Slovní hodnocení
1 - 2	Bezvýznamný
3 - 4	Málo významný
5 - 6	Významný
7 - 8	Velmi významný
9 - 10	Kritický

Tabulka 4: Hrozby (Zdroj: Vlastní zpracování)

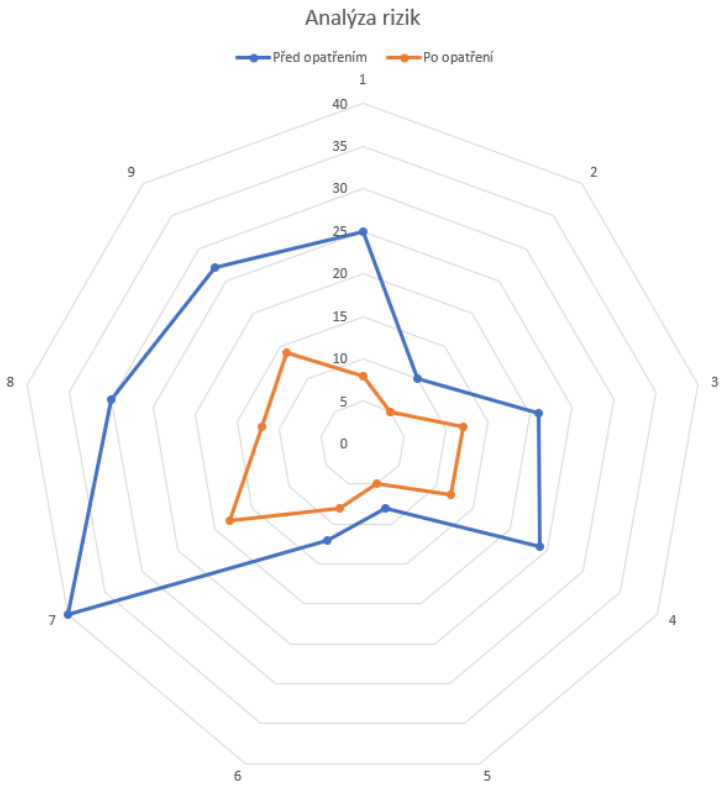
Číslo	Hrozba	Scénář	Pravděpodobnost	Dopad	Hodnota rizika
1	Špatné odhadnutí časové náročnosti projektu	Opoždění dokončení realizace projektu	5	5	25
2	Neefektivní řízení projektu.	Navýšení ceny projektu	2	5	10
3	Nesprávná nebo nefunkční integrace s existujícími systémy.	Nefunkční části systému	3	7	21
4	Nedostatečné školení zaměstnanců	Neschopnost řádně využívat aplikaci	4	6	24
5	Významná změna právního systému ohledně práce s daty	Stát zavede významnou změnu, která bude vyžadovat změnu systému	1	8	8
6	Nedostatečná podpora ze strany vedení společnosti.	Bude obtížně prosadit zahájení vývoje dané aplikace	2	6	12
7	Špatný odhad rozpočtu projektu	Zvýšení nákladů	5	7	35
8	Nedostatečné testování systému.	Uživatelé objeví chybu v systému až po delším časovém období používání	5	6	30
9	Nedostatečná ochrana dat a zabezpečení systému.	Data uživatelů uniknou	3	9	27

Tabulka 5: Hrozby s opatřeními (Zdroj: Vlastní zpracování)

Číslo	Hrozba	Opatření	Pravděpodobnost	Dopad	Hodnota rizika
R1	Špatné odhadnutí časové náročnosti projektu	Pečlivá analýza a plánování s dostatečnou rezervou	2	4	8
R2	Neefektivní řízení projektu.	Zvolení zkušeného vedoucího, který bude jasně koordinovat průběh projektu	1	5	5
R3	Nesprávná nebo nefunkční integrace s existujícími systémy.	Důkladná analýza před zahájením a průběžné testování	2	6	12
R4	Nedostatečné školení zaměstnanců	Naplánování důkladného školení všech zaměstnanců + vytvoření manuálu	2	6	12
R5	Významná změna právního systému ohledně práce s daty	Provádění periodické analýzy právního prostředí	1	5	5
R6	Nedostatečná podpora ze strany vedení organizace.	Otevřená komunikace a vysvětlení výhod aplikace	2	4	8
R7	Špatný odhad rozpočtu projektu	Vytvoření dostatečné finanční rezervy při plánování projektu	3	6	18
R8	Nedostatečné testování systému.	Vytvoření podrobného plánu testování	2	6	12
R9	Nedostatečná ochrana dat a zabezpečení systému.	Analýza současné situace a aktualizace bezpečnostních pravidel dle aktuálních standardů	2	7	14

Jak je vidět v tabulkách výše, u většiny hrozeb by bylo možné po zavedení protiopatření snížit primárně pravděpodobnost výskytu a až sekundárně dopad těchto hrozeb.

Výsledné porovnání rizik před a po zavedené opatření je možné vidět na následujícím grafu.



Graf 1: Analýza rizik před a po opatřeními

(Zdroj: Vlastní zpracování)

3.3 Implementace a konfigurace sekundárních aplikací

Pro běh aplikace je nutné implementovat několik již zmiňovaných aplikací. Pro jejich implementaci, vzhledem k firemním zvyklostem, byl využit docker. Jelikož všechny aplikace nabízí svůj oficiální docker image, bylo možné vytvořit docker-compose, nástroj, který umožňuje konfiguraci, spuštění a správu několika kontejnerů současně. Docker-compose v průběhu let vydal několik verzí, které poskytují různá vylepšení. Jelikož požadavky pro běh těchto aplikací jsou standartní, je možné použít nejnovější verzi 3.8.

Jak můžeme vidět na obrázcích níže, je nutné při vytváření souboru docker-compose stanovit několik mandatorních argumentů pro běh kontejnerů.

- **container_name –** určuje jméno pod kterým bude možné kontejner rozlišit mezi ostatními běžícímu kontejnery.
- **image –** určuje verzi aplikace, kterou chceme používat
- **environment –** určuje proměnné které je nutné stanovit a které ovlivňují chod celého kontejneru
- **ports –** určuje na kterých portech zařízení bude aplikace dostupná
- **volumes –** určuje lokaci, kde jsou uložena data kontejneru, která jsou perzistentní

3.3.1 MySQL

Pro běh kontejneru s databází je nutné stanovit následující atributy. Jak je na obrázku níže, pro běh aplikace byla zvolena verze „latest“, tedy nejnovější verze. Dále bylo nutné zvolit následující enviromentální proměnnou a nastavit volumes, kde budou skladována data kontejneru, ale také kde je umístěn SQL skript pro vytvoření databáze:

- MYSQL_ROOT_PASSWORD – heslo pro root uživatele, který má možnost upravovat a měnit veškeré aspekty databáze

```yaml
services:
  mysql:
    container_name: MySQL_DB
    image: "mysql:latest"
    environment:
      MYSQL_ROOT_PASSWORD: password
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql
      - ./mysql-init:/docker-entrypoint-initdb.d
```

Obrázek 13: Konfigurace MySQL kontejneru

(Zdroj: Vlastní zpracování)

Na obrázku níže je možné vidět skript pro vytvoření schéma analyticsDB pro data z EventStore databáze.

```sql
CREATE DATABASE analyticsDB;
CREATE USER 'EventStoreService'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON analyticsDB.* TO 'EventStoreService'@'%';
FLUSH PRIVILEGES;
```

Obrázek 14: SQL skript pro inicializaci databáze

(Zdroj: Vlastní zpracování)

Pro běžný pro provoz se se nehodí využívat root uživatele, z bezpečnostních důvodů, proto pro pozdější import dat byl vytvořen uživatel „EventStoreService“, heslo bylo nastaveno „password“ pro účely vývoje a testování, pro produkční účely bude samozřejmě změněno. Jak je také na obrázku výše vidět, je definován port 3306, který je určen jak pro komunikaci, tak pro připojení přes grafické rozhraní, například MySQL WorkBench.

3.3.2 RabbitMQ

Pro běh kontejneru s message brokerem je nutné stanovit následující atributy. Jak je na obrázku níže, pro běh aplikace byla zvolena verze „3-management“, jedná se o nejnovější stabilní verzi. Dále bylo nutné zvolit následující enviromentální proměnnou, která obsahuje relativní odkaz na konfigurační soubor, který bude popsán níže.

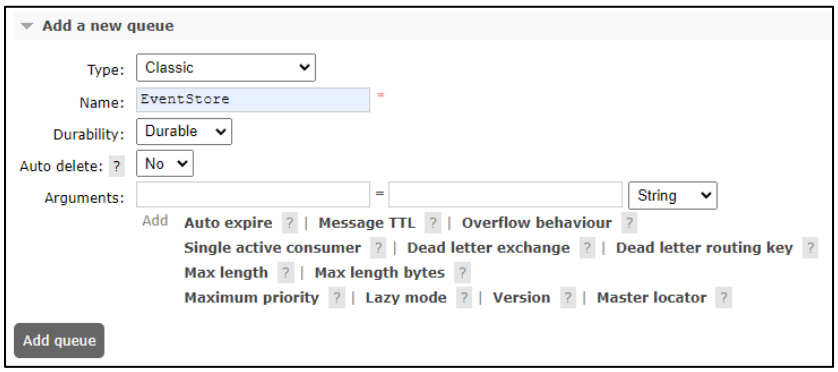
```yaml
rabbitmq:
  container_name: RabbitMQ
  image: "rabbitmq:3-management"
  ports:
    - "5672:5672"
    - "15672:15672"
  volumes:
    - rabbitmq-data:/var/lib/rabbitmq
    - ./rabbitmq-init:/etc/rabbitmq/definitions
  environment:
    RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS:
    "-rabbitmq_management load_definitions
    '/etc/rabbitmq/definitions/definitions.json'"
```

Obrázek 15: Konfigurace RabbitMQ kontejneru

(Zdroj: Vlastní zpracování)

Jak je vidět na obrázku výše, RabbitMQ má definovány dva porty. První port, 5672, je určen pro protokol AMQP, jež byl již popsán. Druhý port, 15672, je určen pro připojení do administrace přes webové rozhraní, kde je možné vytvářet jednotlivé fronty a provádět veškerá nastavení. Jako poslední je na obrázku výše vidět importování konfiguračního souboru.

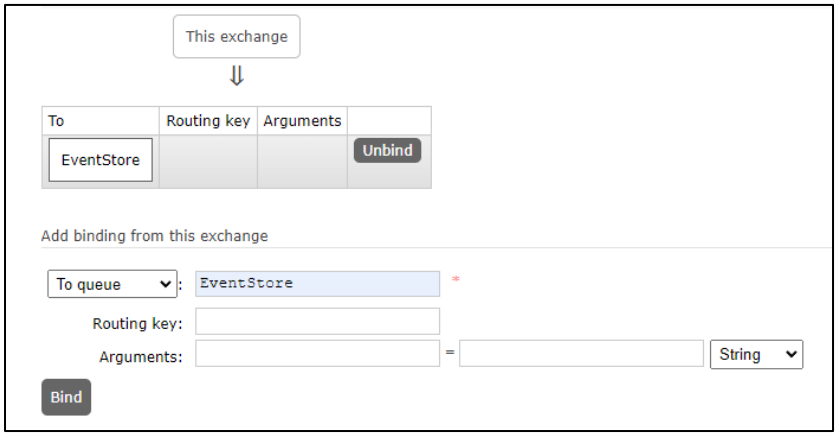
RabbitMQ také vyžaduje dodatečnou konfiguraci. Konkrétně se jedná o vytvoření Queue – fronty pro zprávy z aplikace ServiceDesk a její napojení na exchange. Toto nastavení lze provést dvěma způsoby.



Obrázek 16: Ruční vytvoření Queue - fronty v administraci RabbitMQ

(Zdroj: Vlastní zpracování)

Typ Queue – fronty byl zvolen na základě analýzy, „Classic“, což přesně odpovídá požadavkům aplikace. Dalším krokem je poté vytvořenou frontu navázat k odpovídající Exchange. V tomto případě na základě analýzy je použita exchange „amq.fanout“.



Obrázek 17: Vytvoření propojení mezi frontou a exchange

(Zdroj: Vlastní zpracování)

Druhým způsobem je vytvoření fronty a daného propojení pomocí konfiguračního JSON souboru při spouštění kontejneru. Jak je vidět na obrázku níže, soubor poté obsahuje definici fronty, která se skládá z názvu fronty, parametru „vhost“, který agreguje exchanges, fronty a nastavení pro případ že je RabbitMQ využíván vícero službami, parametrem „durable“, který zajišťuje, že zprávy nejsou smazány při restartu systému a parametru „auto_delete“, který určuje, že fronta není po vyprázdnění automaticky smazána.

Obrázek 18: Vytvoření Queue - fronty konfiguračním souborem

(Zdroj: Vlastní zpracování)

Úplně stejným způsobem je poté vytvořena a napojena fronta „LiveData“, ze které budou zprávy zpracovávány a automaticky vkládány do databáze analyticsDB.

3.3.3 EventStore

Pro běh kontejneru s databází EventStore je nutné stanovit následující atributy. Jak je vidět na obrázku níže, pro běh aplikace byla zvolena verze „latest“, jedná se o nejnovější verzi. Dále bylo nutné zvolit následující enviromentální proměnné:

- EVENTSTORE_CLUSTER_SIZE – počet uzlů na kterých databáze funguje, hodnota 1 byla stanovena na základě analýzy
- EVENTSTORE_RUN_PROJECTIONS - určuje, jaké projekce jsou vytvořeny
- EVENTSTORE_START_STANDARD_PROJECTIONS - určuje, zda jsou projekce vytvořeny při startu databáze
- EVENTSTORE_INSECURE – tento atribut určuje, že databáze nevyžaduje autentifikaci, toto nastavení je podle doporučení změněno až při nasazení do produkčního prostředí, jelikož pro zabezpečený běh je nutné vygenerovat certifikát

```
eventstore:
  container_name: EventStore_DB
  image: "eventstore/eventstore:latest"
  environment:
    EVENTSTORE_CLUSTER_SIZE: "1"
    EVENTSTORE_RUN_PROJECTIONS: "All"
    EVENTSTORE_START_STANDARD_PROJECTIONS: "true"
    EVENTSTORE_INSECURE: "true"
    EVENTSTORE_EXT_TCP_PORT: "1113"
    EVENTSTORE_HTTP_PORT: "2113"
    EVENTSTORE_ENABLE_EXTERNAL_TCP: "true"
    EVENTSTORE_ENABLE_ATOM_PUB_OVER_HTTP: "true"
  ports:
    - "1113:1113"
    - "2113:2113"
  volumes:
    - eventstore-data:/var/lib/eventstore
```

Obrázek 19: Konfigurace EventStore kontejneru

(Zdroj: Vlastní zpracování)

Poslední čtyři proměnné úzce souvisí s definovanými porty a zpřístupňují přes ně komunikaci. Port 2113 je využívám pro přístup do administrace databáze přes webové rozhraní, je také využíván pro HTTP API dotazy. Port 1113 je poté využíván pro TCP komunikaci protokolem gRPC.

3.4 SpringBoot aplikace

Jelikož celý ekosystém společnosti je vyvíjen na platformě Java, konkrétně verze Java 17, kombinací jazyků Java a Kotlin, tak pro vyvíjenou aplikaci byl zvolen framework SpringBoot, který je ve společnosti hojně využíván. Jako programovací jazyk byl zvolen Kotlin. Toto rozhodnutí umožní aplikaci integrovat do ekosystému a její budoucí vývoj a správu.

Souborová struktura aplikace je rozdělena do několika částí. Mezi základní náleží adresáře controller, model, repository, service a resources.

Controller - Jedná se o adresář, kde jsou umístěny soubory, určené pro komunikační rozhraní, které obsluhuje interakci mezi klientem – prohlížečem a serverovou částí aplikace
Model – adresář obsahuje definice datových tříd, v tomto případě pro databázi EventStore a MySQL
Repository – adresář obsahuje definice pro komunikaci mezi aplikací a databází
Service – adresář obsahuje soubory, které vykonávají veškeré vnitřní operace
Resources – adresář obsahuje HTML definice pro frontendovou část aplikace

3.4.1 Generátor dat

Jelikož aplikace je nutné při vývoji vždy testovat jak z hlediska funkčnosti, tak z hlediska výkonu, je proto před samotnou implementací aplikace nutné vytvořit také generátor dat založených na událostech. Generátor generuje náhodná data pro události popsané v kapitole Formát produkovaných dat. Spustit generování dat a určit počet vygenerovaných záznamů lze z frontendové části aplikace.

Samotný generátor lze rozdělit do několika částí. První část se věnuje navázání komunikace s message brokerem. Pro tuto komunikaci je využívána knihovna *com.rabbitmq.client*, která poskytuje rozhraní ke komunikačnímu protokolu AMQP. Pro vytvoření připojení je nutné poskytnout přihlašovací údaje a adresu kde je RabbitMQ spuštěn, v tomto případě „localhost“, stejně tak by zde mohla být zapsána IP adresa 127.0.0.1. Číslo portu není v tomto případě nutné zadávat, jelikož je využíván výchozí port číslo 5672.

```
val factory = ConnectionFactory()
factory.host = "localhost"
factory.username = "guest"
factory.password = "guest"

val connection = factory.newConnection()
val channel = connection.createChannel()
```

Obrázek 20: Připojení k RabbitMQ

(Zdroj: Vlastní zpracování)

Druhá část se věnuje samotnému generování náhodných dat a jejich publikace do RabbitMQ fronty. Jako první je nutné stanovit jaká exchange bude využita – na základě analýzy byla zvolena exchange „amq.fanout“, která nejvíce vyhovuje potřebám aplikace. Dále je možné vidět, že ukázka kódu obsahuje dva vnořené for cykly. První cyklus projde iteracemi na základě čísla v proměnné *numberOfMessages*, tento parametr je předán v requestu a zadává ho uživatel. Na začátku je možné vidět funkci generateDates, která vrací 9 náhodných datumů v rozmezí 20 dnů. Cyklus dále obsahuje tři části, vygenerování dat pro vytvoření incidentu, cyklus, který se opakuje sedmkrát pro vygenerování dat pro simulování změn v incidentu a vygenerování dat pro uzavření incidentu. Celkem 9 stavů není náhodné číslo, na základě firemních analýz jde o průměrný počet změn u jednotlivých incidentů.

```
val exchange = "amq.fanout"
for (i in 1..numberOfMessages) {
    val listOfDates = generateDates()
    var dateTime = listOfDates[0].toString()

    val createEventData = createRecordData(i, dateTime)
    channel.basicPublish(exchange, "", null, createEventData.toByteArray())


    for (k in 0..6) {
       val dateTime = listOfDates[k+1].toString()
       val status =  STATUSES[k]
       val updateEventData = updateRecordData(i, status, dateTime)
       channel.basicPublish(exchange, "", null, updateEventData.toByteArray())
    }

    dateTime = listOfDates[9].toString()
    val closeEventData = closeRecordData(i, dateTime)
    channel.basicPublish(exchange, "", null, closeEventData.toByteArray())
}
```

Obrázek 21: Generování dat a publikace do RabbitMQ fronty

(Zdroj: Vlastní zpracování)

Funkce createRecordData, updateRecordData, closedRecordData jsou analogicky velmi podobné, avšak jsou odlišné v tom, jaká generují data. Na následujícím obrázku je možné vidět ukázku generování dat funkcí updateRecordData. Jak je možné vidět na obrázku níže, náhodná hodnota pro jednotlivé proměnné je vždy vybrána stejnou funkcí getRandomValue, které přijímá jediný parametr, a to list možných hodnot, kterých může nabývat daná proměnná a z nich vybere a vrátí jednu náhodnou hodnotu.

```
fun updateRecordData(incidentId: Int, status: String, dateTime: String) : String {
    val impact : String = getRandomValue(IMPACT_LEVELS)
    val priority : String = getRandomValue(PRIORITY_LEVELS)
    val category : String = getRandomValue(CATEGORIES)
    val team : String = getRandomValue(TEAMS)
    val userId : String = getRandomValue(USER_IDS)

    return """{
          "eventType": "IncidentUpdated",
          "data": {
            "incidentId": "$incidentId",
            "status": "$status",
            "impact": "$impact",
            "priority": "$priority",
            "category": "$category",
            "team": "$team"
          },
          "metadata": {
            "timestamp": "$dateTime",
            "userId": "$userId",
            "streamId": "incident-IR-$incidentId"
          }
        }"""
}
```

Obrázek 22: Funkce updateRecordData

(Zdroj: Vlastní zpracování)

Funkce getRandomValue je ukázkou toho, jak i jednoduchá funkce může být efektivní. Přijímá jeden parametr typu list a vrací náhodnou hodnotu ve formátu string – řetězec znaků.

```
fun getRandomValue(list: List<String>): String {
    val randomIndex = (list.indices).random()
    return list[randomIndex]
}
```

Obrázek 23: Funkce pro výběr náhodné hodnoty

(Zdroj: Vlastní zpracování)

Vstupní parametr poté může vypadat jako na obrázku níže. Konkrétně se jedná o list možných hodnot, kterých může nabývat atribut status u incidentu.

```
val STATUSES = listOf(
    "Open",
    "Assigned",
    "In Progress",
    "On Hold",
    "Resolved",
    "Closed",
    "Reopened"
)
```

Obrázek 24: Množina přípustných hodnot statusu

(Zdroj: Vlastní zpracování)

3.4.2 RabbitMQ – čtení zpráv

Jak už bylo popsáno výše, zprávy ze zdrojové aplikace jsou posílány do fronty v RabbitMQ. Bylo by velmi nepraktické provádět například periodicky dotazy na danou frontu. Vedlo by to k neúměrnému zatěžování zdrojů a sítě, a navíc by zprávy nebyli zpracovávány okamžitě jak je to možné.

Pro tento případ je vhodné implementovat mechanismus, který toto umožňuje. Naštěstí je tento mechanismus již dostupný v knihovně v podobě listeneru. Ten v případě příchozí zprávy automaticky zavolá eventStoreConnector a funkci „main“, která obstará zpracování dané zprávy, která se nachází ve frontě EventStore.

```
import org.springframework.amqp.rabbit.annotation.RabbitListener

@Component
class RabbitMessageListener {

    @Autowired
    val eventStoreConnector: EventStoreConnector? = null
    @RabbitListener(queues = ["EventStore"])
    fun handleMessage(message: String) {
        eventStoreConnector?.main(message)
    }
}
```

Obrázek 25: RabbitMQ listener

(Zdroj: Vlastní zpracování)

Stejným způsobem je implementováno i odposlouchávání fronty „LiveData“. Rozdíl je v tom, že data z této fronty jsou poté rovnou vložena do analyticsDB, tak aby byla dostupná pro analyzování.

3.4.3 Zpracování zpráv z RabbitMQ a uložení do EventStore databáze

Každá příchozí zpráva je ve formátu „String“, tedy řetězec znaků. Naštěstí jsou zprávy ve standardizovaném formátu JSON, který je možné strojově zpracovat. Proto následující kód jako první krok převede textovou zprávu do JSON objektu. Z tohoto objektu je následně možné získat informace o tom, o jaký event se jedná, samotná data daného eventu a nakonec metadata, která obsahují unikátní identifikátor streamu, ke kterému data patří.

Předtím než je možné zapisovat do databáze je nutné vytvořit spojení s databází. Toto spojení je vytvořeno pomocí EventStoreDBClientu, který požívá pro komunikaci protokol gRPC. Před vytvořením připojení je nutné zadat adresu kde se databáze nachází, port a v tomto případě také parametr „?tls=false“, který umožňuje během vývoje pracovat s databází EventStore bez nutnosti autentifikace.

```
val setts: EventStoreDBClientSettings = EventStoreDBConnectionString.parseOrThrow("esdb://localhost:2113?tls=false")
val client: EventStoreDBClient = EventStoreDBClient.create(setts)
```

Obrázek 26: Vytvoření připojení k EventStore

(Zdroj: Vlastní zpracování)

```
fun main(message: String) {
    val messageJSON = JSONObject(message)
    val eventType = messageJSON.getString("eventType")
    val data = messageJSON.getJSONObject("data")
    val metadata = messageJSON.getJSONObject("metadata")
    val streamID = metadata.getString("streamId")

    when (eventType) {
        "IncidentCreated" -> evaluateCreatingData(data, eventType, streamID)
        "IncidentUpdated" -> evaluateUpdateData(data, eventType, streamID)
        "IncidentClosed" -> evaluateClosingData(data, eventType, streamID)
    }
}
```

Obrázek 27: Zpracování zpráv z RabbitMQ

(Zdroj: Vlastní zpracování)

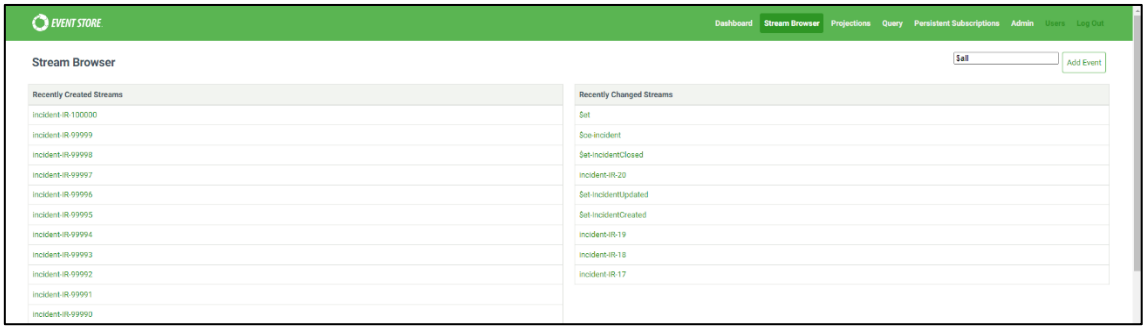
Vzhledem k tomu, jakým způsobem jsou data do databáze EventStore ukládána je nutné pro každý typ eventu implementovat vlastní funkci pro zápis. Jak je vidět na obrázku výše, na základě toho, o jaký se jedná event je poté zavolána funkce pro zapsání dat do databáze EventStore. Samotná funkce poté obsahuje vytvoření datového objektu navázaného k danému typu eventu. Následně je vytvořen objekt pomocí EventData.builder, který je poté vložen do databáze pomocí funkce appendToStream, která přijímá dva parametry, streamID, který slouží jako unikátní identifikátor streamu a samotná data.

```
fun evaluateCreatingData(data: JSONObject, eventType: String, streamID: String){
    val createEvent = IncidentCreatedEvent()

    createEvent.setData(data)
    val event = EventData.builderAsJson<Any>(eventType, createEvent).build()

    client.appendToStream(streamID, event).get()
}
```

Obrázek 28: Zapsání eventu do EventStore databáze

(Zdroj: Vlastní zpracování)

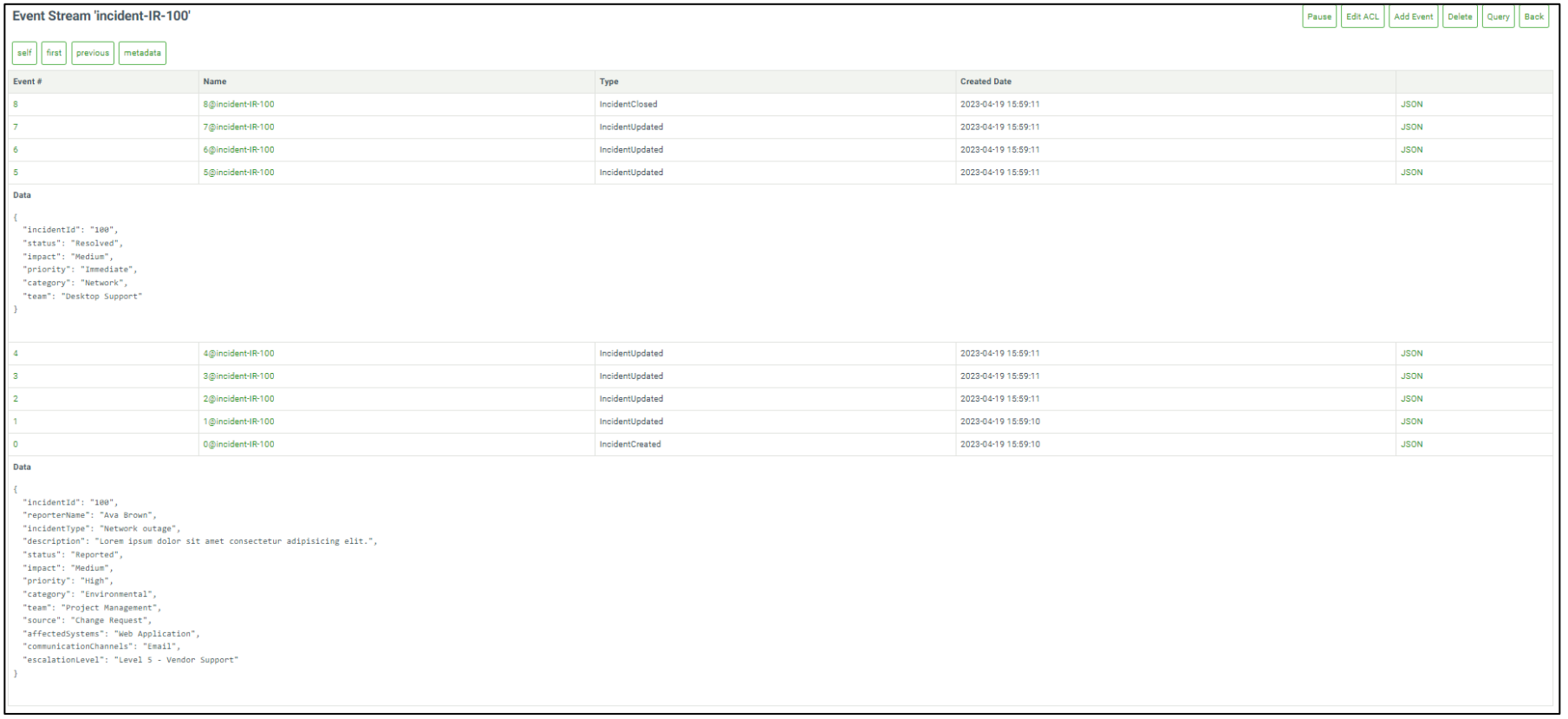
Administrace databáze EventStore je dostupná pomocí webového prohlížeče na portu 2113. Vložené incidenty v podobě streamů lze najít pod kartou „Stream Browser“. Toto zobrazení vypadá následovně.



Obrázek 29: EventStore - Stream Browser

(Zdroj: Vlastní zpracování)

V pravé části administrace je možné vidět naposledy upravené streamy a také automaticky vytvořené projekce. Například projekci „$ce-incident“, která bude využívána v následující části.



Obrázek 30: Detail dat incidentu v EventStore databázi

(Zdroj: Vlastní zpracování)

Na obrázku výše je vidět detailní pohled na to, jak je incident uložen v databázi EventStore. Jak již bylo zmíněno v kapitole Generátor dat, průměrný počet změn incidentu je devět. Na obrázku je také možné si všimnout, že každá změna incidentu vytvoří nový záznam ve streamu. To umožňuje projít celý životní cyklus jakéhokoliv incidentu.

3.4.4 Export dat z EventStore a import do MySQL

Proces exportu dat z EventStore databáze do databáze Analytics je klíčovou součástí vyvíjené aplikace. EventStore databáze však ukládá data jako stream eventů – událostí, naproti tomu Analytics umí pracovat pouze s databází relačního typu.

Před samotným exportem dat z databáze EventStore je nutné navázat spojení. Toho je docíleno analogickým způsobem jako v předešlé kapitole pomocí EventStoreDBClientu.

Čtení z databáze je možné provádět několika způsoby. Vzhledem k tomu, že nelze provádět SQL dotazy jako u relačních databází je nutné zvolit jiný způsob.

Jako první se nabízí možnost přečíst všechny streamy z databáze a filtrovat je v aplikaci, avšak takové řešení by bylo velice náročné jak časově, tak na zdroje a v případě velkého množství streamů by mohlo dojít i k vyčerpání dostupné operační paměti, což by vedlo ke zhavarování aplikace.

Další možností je číst jednotlivé streamy. Toho lze docílit pomocí unikátního identifikátoru streamu. Jak již bylo zmíněno, nelze provést SQL dotaz a získat list těchto unikátních identifikátorů, naštěstí lze využít jeden z nástrojů, které EventStore databáze nabízí, a tím je projekce, přesněji projekci podle kategorií, která je automaticky vytvořena. Jelikož unikátní identifikátor jednotlivých incidentů je vždy ve formátu „incident-IR-X“, kdy X odpovídá číslu incidentu, lze využít projekci „$ce-incident“. Prefix „$ce“ v tomto případě značí, že se jedná o automaticky vytvořenou systémovou projekci.

Na obrázku níže tedy můžeme vidět funkce readStreamByCategory, která vrátí list unikátních identifikátorů všech incidentů v databázi. Poté je pomocí for cyklu tento list iterován a každý incident v podobě streamu je jednotlivě přečten a uložen do relační databáze.

V kódu níže je také vidět mechanismus, který sleduje průběh exportu dat, který je využíván na frontendu aplikace. Tento mechanismus vychází z celkového počtu incidentů a aktuálně zpracovaných záznamů, díky čemuž je spočítáno procento zpracovaných záznamů. Toto procento je poté publikováno na frontend.

```
fun main() {

    val listOfIncidents: MutableSet<String> = readStreamByCategory()
    val numberOfIncidents: Double = listOfIncidents.size.toDouble()

    for((processedIncidents, incident) in listOfIncidents.withIndex()){
        readStreamByName(incident)
        val progress: Double = ((processedIncidents + 1)/numberOfIncidents) * 100

        if (progress % 1 == 0.0) {
            messagingTemplate?.convertAndSend("/topic/stream", progress)
        }
    }
}
```

Obrázek 31: Export dat z EventStore

(Zdroj: Vlastní zpracování)

Samotná funkce readStreamByCategory poté vypadá následovně. Je stanoven název streamu, v tomto případě se jedná o projekci „$ce-incident“, který obsahuje názvy všech incidentů v databázi. Je také nutné stanovit způsob čtení z databáze, v tomto případě je použit způsob čtení od inkrementálně od začátku a je přidán parametr „.resolveLinkTos()“, který zahrne do vrácených streamů i unikátní identifikátor každého incidentu. Vrácený objekt je poté iterován pomocí for cyklu a unikátní identifikátor je uložen do listu unikátních hodnot, což zajistí, že každí incident je v něm zastoupen pouze jedenkrát.

```
fun readStreamByCategory(): MutableSet<String>{
    val listOfIncidents = mutableSetOf<String>()

    val categoryStreamName = "\$ce-incident"
    val options = ReadStreamOptions.get()
        .forwards()
        .fromStart()
        .resolveLinkTos()

    val result = client.readStream(categoryStreamName, options).get()

    for (resolvedEvent in result.events) {
        listOfIncidents.add(resolvedEvent.event.streamId)
    }

    return listOfIncidents
}
```

Obrázek 32: Získání listu všech unikátních identifikátorů

(Zdroj: Vlastní zpracování)

Funkce readStreamByName vykonává dvě věci. První z nich je čtený jednotlivých streamů s konkrétními daty jednotlivých incidentů a jejich následné uložení do databáze. Aby byla plně využita možnost analyzovat průběh životního cyklu každého incidentu, a tak odhalovat a optimalizovat jeho průběh, je do relační databáze uložena každá změna incidentu. Změny jsou ukládány vždy jako záznam celého incidentu, aby bylo možné provádět analýzu těchto dat.

```
fun readStreamByName(streamName: String){

    val options = ReadStreamOptions.get().forwards().fromStart()

    val result: ReadResult = client.readStream(streamName, options).get()
    val objectMapper: ObjectMapper = jacksonObjectMapper()
    var savedIncident = Incident()

    for (resolvedEvent in result.events) {
        val recordedEvent = resolvedEvent.originalEvent
        val eventDataString = recordedEvent.eventData.decodeToString()

        val messageJSON = JSONObject(eventDataString)

        val keys = messageJSON.keys()

        var newIncident = savedIncident.copy()
        newIncident.created = recordedEvent.created
        while (keys.hasNext()) {
            val key = keys.next()
            val value = messageJSON.get(key)
            val field = newIncident.javaClass.getDeclaredField(key)
            field.isAccessible = true
            field.set(newIncident, value)
        }
          savedIncident = newIncident.copy()
          incidentRepository.save(newIncident)

    }
}
```

Obrázek 33: Čtení dat z EventStore a zápis do MySQL

(Zdroj: Vlastní zpracování)

Výsledná data v relační databázi poté vypadají následovně jak je vidět na obrázku níže, data jsou uložena v jedné tabulce, tabulce incidentů a v případě dalších oblastí, se kterými bude aplikace pracovat budou vytvořeny tabulky pro dané oblasti. To, že jsou data uložena v jedné tabulce, namísto jejich rozdělení do několika tabulek, jak tomu bývá v relačních modelech je svým způsobem výhodou, jelikož to zjednodušuje tvorbu datového modelu a umožňuje to data v databázi indexovat podle potřeby. Je také možné si všimnout toho, že obrázek obsahuje jen část atributů incidentu, kvůli jejich vyššímu počtu jich je vypsáno jen několik.

incident_id	created	resolution	resolution	root_cause	source	status	team
100	2023-04-01 13:59:10.926011	Empty	Empty	Empty	Change Request	Reported	Project Management
100	2023-04-03 13:59:10.985685	Empty	Empty	Empty	Change Request	Open	Database Administration
100	2023-04-04 13:59:11.046294	Empty	Empty	Empty	Change Request	Assigned	Database Administration
100	2023-04-05 13:59:11.059492	Empty	Empty	Empty	Change Request	In Progress	Application Development
100	2023-04-08 13:59:11.115618	Empty	Empty	Empty	Change Request	On Hold	Project Management
100	2023-04-11 13:59:11.165582	Empty	Empty	Empty	Change Request	Resolved	Desktop Support
100	2023-04-13 13:59:11.215759	Empty	Empty	Empty	Change Request	Closed	Desktop Support
100	2023-04-16 13:59:11.223624	Empty	Empty	Empty	Change Request	Reopened	Desktop Support
100	2023-04-19 13:59:11.229179	Configuration Change	Configuration Change	System Misconfiguration	Change Request	Closed	Desktop Support

Obrázek 34: Změny během životního cyklu incidentu

(Zdroj: Vlastní zpracování)

Na obrázku výše je také možné si všimnout několikrát hodnoty „Empty“, je to výchozí hodnota, která byla zvolena pro situace, kdy v daném záznamu události nebyla tato data a nebyly ani v předchozích záznamech.

3.4.5 Zápis dat do analytické databáze v reálném čase

Všechny nutné kroky pro zápis dat v reálném čase do databáze analytické aplikace jsou již v aplikaci implementovány a byli již popsány výše. Proto v této podkapitole dojde jen k jejich sumarizaci. Fronta „LiveData“ byla implementována spolu s frontou „EventStore“ při startu RabbitMQ, byl analogicky vytvořen event listener této fronty a daný stream je zpracován stejným způsobem jako na **Obrázek 33: Čtení dat z EventStore a zápis do MySQL** s tím rozdílem, že je zapsán až poslední stav, proto aby nedocházelo k duplicitám.

3.4.6 Výpis listu incidentů

Pro výpis incidentů na front-endu aplikace byla využita funkce ReadStreamByCategory, která byla popsána v kapitole 3.4.4 Export dat z EventStore a import do MySQL. Bylo možné ji beze změn využít, jelikož vrací list existujících incidentů.

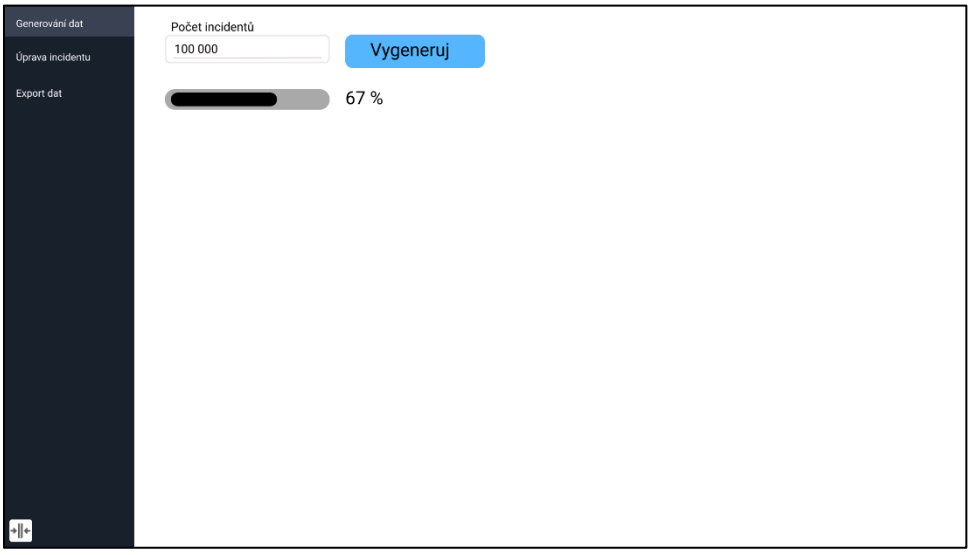
3.4.7 Vytvoření a úprava incidentu

Struktura dat vytvoření nového incidentu a úpravy již existujícího byla popsána v kapitole 2.2.2 Formát produkovaných dat, proto není nutné ji zde popisovat znovu. Pro vytvoření a úpravu incidentu bylo proto možné využít již existující kód popsaný v kapitole Generátor dat.

3.4.8 Frontend aplikace

I přesto, že většina funkcionalit se děje na pozadí aplikace, jsou části, které ke svému používání vyžadují webové rozhraní, jedná se o generování dat, export dat, zobrazení a úprava incidentu.

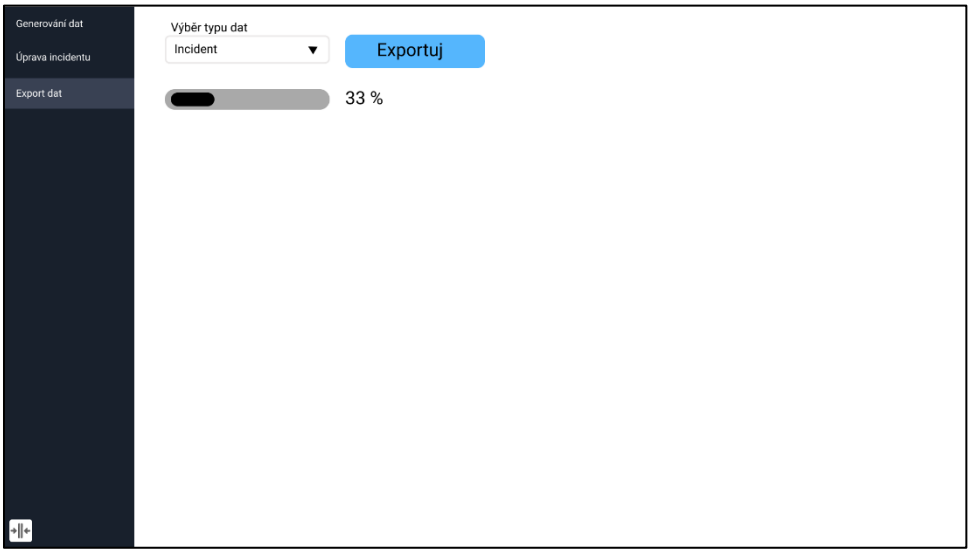
První je obrazovka pro generování dat. Umožňuje stanovit, kolik záznamů má být vygenerováno a sledovat průběh tohoto generování.



Obrázek 35: Generování dat - Front - end

(Zdroj: Vlastní zpracování)

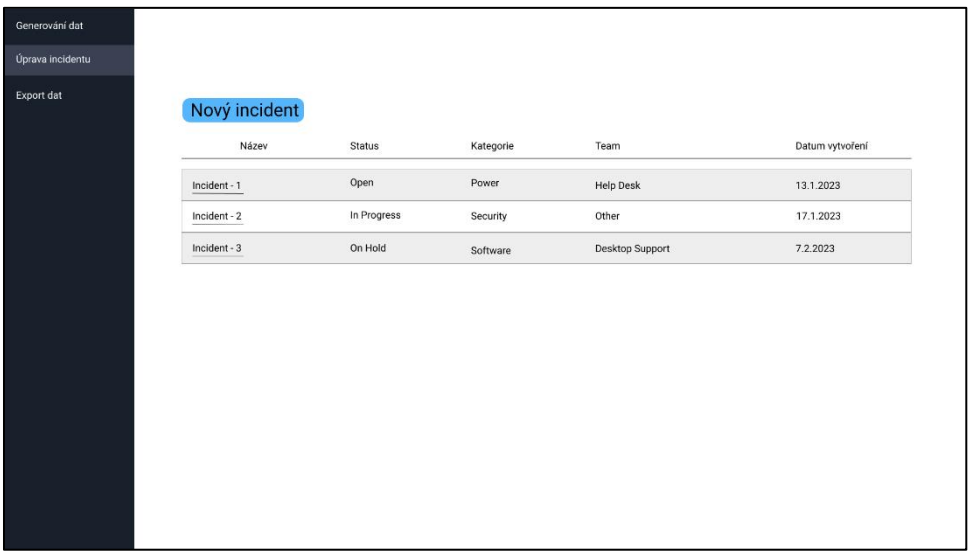
Druhá obrazovka je pro spuštění exportu dat z EventStore databáze a sledování průběhu toto exportu.



Obrázek 36: Export dat - Front-end

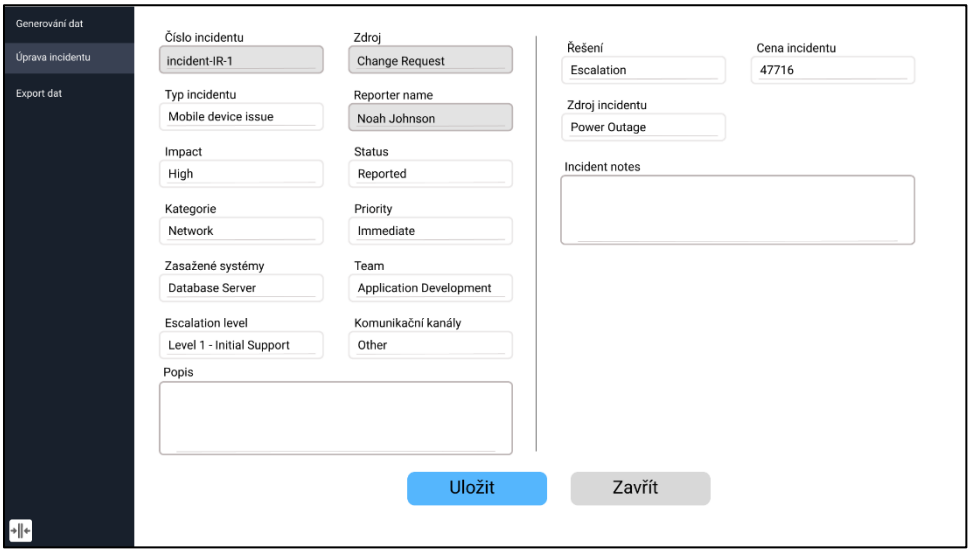
(Zdroj: Vlastní zpracování)

Třetí obrazovka umožňuje zobrazit seznam incidentů, detail incidentu a upravit ho. Tato poslední obrazovka je určena primárně pro testování, jelikož vytváření a úprava incidentů budou implementovány přímo v aplikaci ServiceDesk.



Obrázek 37: List incidentů – Front -end

(Zdroj: Vlastní zpracování)



Obrázek 38: Detail incidentu - Front-end

(Zdroj: Vlastní zpracování)

Websocket

Vzhledem k tomu, že jeden z požadavků bylo, aby aplikace byla interaktivní bylo nutné zvolit způsob aktualizace dat na frontendu. Toho lze docílit periodickými dotazy na aktuální stav pomocí REST api nebo GraphQL. Lze však implementovat i jiný mechanismus. Tímto mechanismem je použití technologie websocket, konkrétně implementací protokolu STOMP.

Implementace tohoto protokolu má dvě části. Tou první je vytvoření endpointů na backendu a následně vytvoření připojení z frontendové části aplikace.

Na následujícím obrázku je možné vidět registraci endpointu na backendu. Endpoint je registrován pod názvem „websocket“. Při komunikaci s endpointem z frontendové části aplikace byl zvolen prefix „/app“. Pro komunikaci z backendu na frontend byl zvolen prefix „/topic“. Tento endpoint slouží pouze k navázání spojení mezi klientem aplikace v prohlížeči a backendem.

Obrázek 39: Registrace endpointu na backendu

(Zdroj: Vlastní zpracování)

Po úspěšném vytvoření připojení je možné registrovat a komunikovat přes endpointy k tomu určené.

Prvním z nich je endpoint pro volání funkce na generování dat, popsanou v kapitole Generátor dat. Jak je možné vidět na obrázku níže, funkce přijímá jeden argument v podobě počtu záznamů, které mají být vygenerovány. Daný počet je z frontendové části odesílán v podobě stringu – řetězce znaků a je převeden na číselný formát. Funkce nevrací žádnou hodnotu.

Obrázek 40: Endpoint pro generování dat

(Zdroj: Vlastní zpracování)

Dalším je endpoint pro volání funkce „getStream“ pro export dat z EventStore databáze do MySQL databáze. Tato funkce nepřijímá žádný argument ani nevrací žádnou hodnotu.

```
@Controller
class WebSocketController {
    @Autowired val importDataToDatabase: ImportDataToDatabase? = null

    @MessageMapping("/getStream")
    fun getStrem(message: String){
        importDataToDatabase?.main()
    }

}
```

Obrázek 41: Endpoint pro export dat

(Zdroj: Vlastní zpracování)

Jakmile jsou vytvořeny na endpointy na backendu, je možné se k nim připojit z frontendové části aplikace. Jak již bylo řečeno, tohoto připojení je docíleno pomocí STOMP klientu poskytovaného knihovnou SockJS.

```
function connect() {
    var socket = new SockJS('/websocket');
    stompClient = Stomp.over(socket);
    stompClient.connect({}, function (frame) {
      stompClient.subscribe('/topic/messages', function (response) {});

      stompClient.subscribe('/topic/stream', function (status) {
          showStream(status.body);
      });
    });
}
```

Obrázek 42: Vytvoření připojení z frontendu

(Zdroj: Vlastní zpracování)

3.4.9 Rychlost a náročnost exportu dat do MySQL

Jedním z dílčích cílů této práce je zjistit, jak náročný je export historických dat z databáze EventStore do MySQL databáze. Náročnost je v tomto případě myšlena jak rychle, tedy doba trvání export a druhotně také náročnost na systémové zdroje.

Sledovat výkonnost Spring boot aplikace lze několika způsoby, avšak jedním z nejvhodnějších se jeví použití nástroje Prometheus pro sběr dat a následně nástroj Grafana pro vizualizaci těchto dat.

Prometheus

Pro použití nástroje Prometheus pro sběr dat je nutné pouze měřit systémové zdroje a zpřístupnit endpoint s těmito hodnotami. Pro měření se nejčastěji u SpringBoot aplikací používá knihovna Actuator. Poté již stačí v enviromentálních proměnných zpřístupnit tyto metriky.

```
management.endpoints.web.exposure.include = *
management.metrics.export.prometheus.enabled = true
```

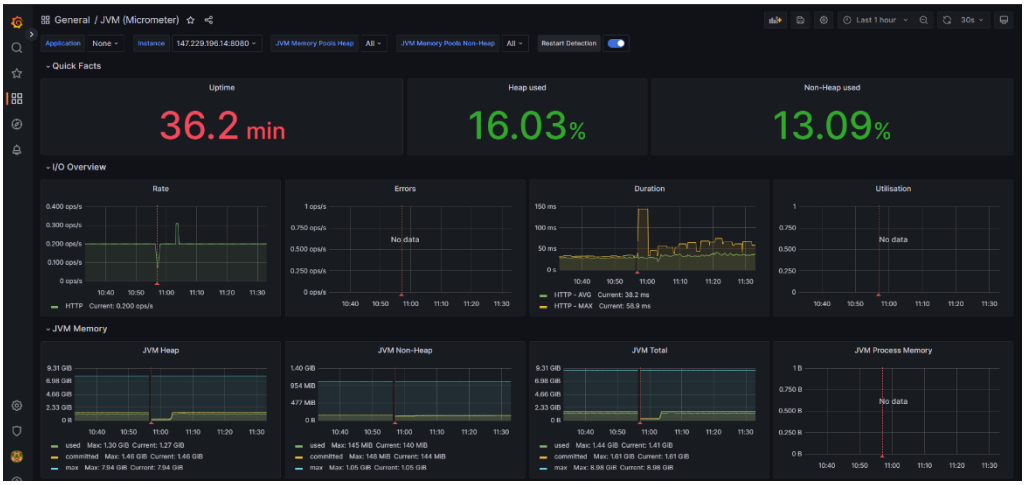
Obrázek 43: Konfigurace metrik

(Zdroj: Vlastní zpracování)

Grafana

Grafa slouží k vizualizaci dat o metrikách z aplikace. Je možné vytvořit vlastní dashboardy, avšak vzhledem k rozšířenosti Grafany a SpringBoot aplikací, je vhodnější (pro většinu případů) použít dashboard vytvořený komunitou.

Na obrázku níže je možné vidět část dashboardu JVM Micrometer, který je doporučen pro SpringBoot aplikace.



Obrázek 44: Grafana - JVM Micrometer

(Zdroj: Vlastní zpracování, [34])

Měření

Měření aplikace probíhalo na lokálním počítači s následující konfigurací.

Procesor: 12th Gen Intel(R) Core(TM) i5-12400 2.50 GHz
RAM: 8 GB*
Disk: Samsung SSD 980, M.2

*Hostující počítač má více RAM, ale aplikace měla pro běh k dispozici pouze 8 GB

Jako první bylo provedeno měření rychlosti exportu dat z EventStore databáze do MySQL databáze. Do databáze EventStore byl vygenerován reprezentativní vzorek 50 000 incidentů. Každý incident obsahoval 9 změn čili do databáze AnalyticsDB bylo vloženo 450 000 záznamů.

Tabulka 6: Měření rychlosti exportu dat 50 000 incidentů (Zdroj: Vlastní zpracování)

Číslo měření	**1**	**2**	**3**	**4**
Počet incidentů	50,000	50,000	100,000	50,000
Doba exportu	1:50:21	1:50:49	1:51:40	1:49:37

V tomto případě bylo využívání aplikací maximálně 0.6 GB operační paměti.

Druhé měření bylo provedeno s reprezentativním vzorkem 100 000 incidentů, které vygeneruje 900 000 záznamů v databázi.

Tabulka 7: Měření rychlosti exportu dat 50 000 incidentů (Zdroj: Vlastní zpracování)

Číslo měření	**1**	**2**	**3**	**4**
Počet incidentů	100,000	100,000	100,000	100,000
Doba exportu	3:39:25	3:37:16	3:42:41	3:38:07

V tomto případě bylo využívání aplikací maximálně 1.13 GB operační paměti.

3.4.10 Zabezpečení

Jak již bylo popsáno v předešlých kapitolách, aplikace bude integrována do ekosystému společnosti. Z tohoto důvodu je nutné v budoucnu zabezpečit MySQL databázi, EventStore databázi a RabbitMQ. Pro samotnou vyvíjenou aplikaci není nutné implementovat žádné bezpečnostní mechanismy, jelikož společnost pro autentifikaci do všech svých aplikací využívá technologii Keycloak.

3.4.11 Git

Vývoj celé aplikace proběhl pomocí verzovacího nástroje Git. Během vývoje byli vytvořeny a následně integrovány do hlavní větve následují větve.

Obrázek 45: Průběh vývoje v GITu

(Zdroj: Vlastní zpracování)

- Websocket – Implementace websocketu pro komunikaci mezi frontendem a backendem
- dataGenerator – Implementace generátoru dat
- eventStoreDB – Implementace importu dat z RabbitMQ do EventStore databáze
- importLegacyDataToDatabase – Implementace exportu dat z EventStore databáze do databáze AnalyticsDB
- LiveDataProcessing – Implementace zpracování dat v reálném čase a uložení do databáze AnalyticsDB
- exposeMetrics – Implementace publikování dat pro analýzu běhu aplikace

3.4.12 Kontejnerizace aplikace

Jelikož společnost se dlouhodobě snaží svoji práci i nástroje co nejvíce zefektivnit, tak i vyvíjená aplikace se tímto směrem bude ubírat. Z tohoto důvodu je aplikace umístěna do docker kontejneru pro snadnější a rychlejší nasazení.

Po vytvoření jar souboru, který obsahuje aplikaci je možné tento kontejner vytvořit pomocí následující konfigurace. Pomocí parametrů Xms je stanovena minimální požadovaná operační paměť a pomocí parametru Xmx maximální požadovaná operační paměť.

```
FROM openjdk:17
WORKDIR /app
COPY ./target/ServiceDeskBackend.jar /app/app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-Xmx8g", "-Xms512m", "-jar", "/app/app.jar"]
```

Obrázek 46: Kontejnerizace aplikace

(Zdroj: Vlastní zpracování)

3.5 Ekonomické zhodnocení

Obsahem této kapitoly bude provedení ekonomického zhodnocení analýzy, návrhu a implementace webové aplikace. Do ekonomického zhodnocení jsou započteny veškeré náklady na vývoj aplikace. Před zahájením této práce byla jedním z architektů aplikací ve společnosti stanovena předběžná kalkulace časové náročnosti práce, tato kalkulace se nachází v první tabulce v této kapitole. Reálná doba práce je poté vyjádřena ve druhé tabulce. Všechny uváděné částky jsou uváděny v korunách.

Jednotlivé kalkulace obsahují stejné položky. Jedná se o položky:

- Analýza současné situace – pojí se s analytickou částí této práce a analýzou teoretických východisek nutných pro budoucí návrh a implementaci webové aplikace.
- Návrh aplikace – zahrnuje návrh celé aplikace
- Implementace aplikace – zahrnuje průběh celého vývoje aplikace
- Testování aplikace – zahrnuje testování aplikace, jak její funkčnosti, tak následného měření výkonu

Tabulka 8: Plánovaná kalkulace nákladů (Zdroj: Vlastní zpracování)

Položka	Man-day	Počet hodin	Hodinová sazba	Celková částka
Analýza současné situace	10	80	300	24,000
Návrh aplikace	10	80	300	24,000
Implementace aplikace	50	400	300	120,000
Testování aplikace	10	80	300	24,000
Celkem	**80**	**640**		**192,000**

Jak je možné vyčíst z první tabulky, předpokládaná cena vývoje aplikace byla stanovena na 192 000 korun. Tato kalkulace počítá i s jistou časovou rezervou u každé položky.

Reálná kalkulace na základě zpracovávání této práce byla nakonec nižší. Jak je vidět v tabulce níže, reálný propočet ceny je nižší o zhruba 40 000 korun.

Tabulka 9: Reálná kalkulace nákladů (Zdroj: Vlastní zpracování)

Položka	Man-day	Počet hodin	Hodinová sazba	Celková částka
Analýza současné situace	6.25	50	300	15,000
Návrh aplikace	11.25	90	300	27,000
Implementace aplikace	40	320	300	96,000
Testování aplikace	7.5	60	300	18,000
Celkem	**65**	**520**		**156,000**

Jak je možné vidět v tabulce výše, doba potřebná na téměř všechny byla nižší než původní kalkulace, až s rozdílem položky „Návrh aplikace“, který si vyžádal navíc 10 hodin.

3.6 Přínosy práce

Jako hlavní přínosy jde uvést tři hlavní benefity, které aplikace přináší. Prvním z nich je možnost analyzovat data téměř v reálném čase, což umožňuje dělat operativní rozhodnutí založená na reálných datech. Dalším z nich je bezpečnost, jelikož data jsou ukládána v podobě neměnného záznamu, což umožňuje nejen sledovat jakoukoliv změnu v datech, i neoprávněnou, ale i to kdy a kým byla změna provedena a jaké měla následky. Posledním benefitem je to, že aplikace je oddělena od producenta dat a je uložena do kontejneru. To s sebou nese benefity v podobě odděleného vývoje, a že může být relativně snadno nasazena, spravována a škálovatelná. To společnosti snižuje provozní náklady.

Další nepominutelným benefitem je následná práce s daty, jelikož data jsou při exportu zapisována do jedné tabulky, je tím usnadněna následná příprava dat pro analýzu.

Jako poslední lze uvést přínos dalšího zajímavého produktu, respektive služby do portfolia. To může v budoucnu znamenat dodatečné zisky a případně může být i jazýčkem na vahách v případě, kdy se zákazník rozhoduje mezi produkty společnosti a produkty konkurence.

ZÁVĚR

Cílem této diplomové práce bylo analyzovat současnou situaci, navrhnout a implementovat webovou aplikaci ve firemním prostředí, která by umožňovala zpracovávat, transformovat a importovat data založená na událostech do analytického nástroje poskytovaného společností.

První kapitola se věnovala identifikaci a popisu všech důležitých nástrojů a technologií, které bylo v této práci využívány nebo byli zmiňovány v souvislosti s obsahem této diplomové práce.

Druhá kapitola se věnovala analytické části práce, byla v ní představena společnost, která se věnuje vývoji firemních informačních systémů a v rámci které byla tato diplomová práce zpracovávána. Dále obsahovala popis současné situace a popis způsobu sledování změn v datech v současné době. Jako poslední byli analyzovány v dnešní době nejrozšířenější message brokery.

Poslední kapitola se věnovala samotnému návrhu a implementaci webové aplikace, která vyžadovala implementaci některých dalších nástrojů, které byli nutné pro její provoz. Obsahem kapitoly je tedy návrh aplikace, samotná implementace včetně nutných konfigurací, analýza rizik spojených s implementací webové aplikace, ekonomické zhodnocení návrhu a přínosy práce.

SEZNAM POUŽITÝCH ZDROJŮ

[1] PESTLE analýza. Managementmania [online]. 2015 [cit. 2023-05-05]. Dostupné z: https://managementmania.com/cs/pestle-analyza

[2] McKinsey 7S. Managementmania [online]. 2015 [cit. 2023-05-05] Dostupné z: https://managementmania.com/cs/mckinsey-7s

[3] KORÁB, Vojtěch, Mária REŽŇÁKOVÁ a Jiří PETERKA. Podnikatelský plán. Brno: Computer Press, c2007. Praxe podnikatele. ISBN 978-80-251-1605-0

[4] SWOT ANALÝZA: JAK, a HLAVNĚ PROČ JI SESTAVIT. Magdalena Čevelová [online] 2011 [cit. 2023-05-05]. Dostupné z: https://www.cevelova.cz/proc-swot-analyza/

[5] *Extrakce, transformace a načtení (ETL)* [online]. In: . [cit. 2023-05-05]. Dostupné z: https://learn.microsoft.com/cs-cz/azure/architecture/data-guide/relational-data/etl

[6] Extract, transform, load. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-05]. Dostupné z: https://cs.wikipedia.org/wiki/Extract,_transform,_load

[7] What is Java technology and why do I need it?. *Java* [online]. [cit. 2023-05-05]. Dostupné z: https://www.java.com/en/download/help/whatis_java.html

[8] What is Java? Definition, Meaning & Features of Java Platforms. *Guru99* [online]. [cit. 2023-05-05]. Dostupné z: https://www.guru99.com/java-platform.html

[9] What Is Kotlin? A Brief Introduction. *Make use of* [online]. [cit. 2023-05-05]. Dostupné z: https://www.makeuseof.com/what-is-kotlin/

[10] What is Spring Boot?. *Educba* [online]. [cit. 2023-05-05]. Dostupné z: https://www.educba.com/what-is-spring-boot/

[11] 10 Best Practices to Follow for REST API Development. *Mind Inventory* [online]. [cit. 2023-05-05]. Dostupné z: https://www.mindinventory.com/blog/best-practices-rest-api-development/

[12] Working with JSON. *Mdn Web Docs* [online]. [cit. 2023-05-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

[13] GraphQL for Beginners: Introduction. *Articles by Victoria* [online]. 2021 [cit. 2023-05-05]. Dostupné z: https://lo-victoria.com/graphql-for-beginners-introduction

[14] What are message brokers?. *IBM* [online]. [cit. 2023-05-05]. Dostupné z: https://www.ibm.com/topics/message-brokers

[15] Messaging with RabbitMQ in Spring Boot Application. *Spring Cloud* [online]. 2022 [cit. 2023-05-05]. Dostupné z: https://www.springcloud.io/post/2022-03/messaging-using-rabbitmq-in-spring-boot-application/#gsc.tab=0

[16] Intro to Security and WebSockets. *Baeldung* [online]. 2022 [cit. 2023-05-05]. Dostupné z: https://www.baeldung.com/spring-security-websockets

[17] What Is Docker and How Does It Work? – Docker Explained. *Hostinger* [online]. 2023 [cit. 2023-05-05]. Dostupné z: https://www.hostinger.com/tutorials/what-is-docker

[18] Install Docker Desktop on Windows. *Docker docs* [online]. [cit. 2023-05-05]. Dostupné z: https://docs.docker.com/desktop/install/windows-install/

[19] Definování vícekontejnerové aplikace pomocí docker-compose.yml. *Microsoft* [online]. 2023 [cit. 2023-05-05]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/architecture/microservices/multi-container-microservice-net-applications/multi-container-applications-docker-compose

[20] Grafana OSS. *Grafana* [online]. [cit. 2023-05-05]. Dostupné z: https://grafana.com/docs/grafana/latest/introduction/

[21] What is Prometheus? - OVERVIEW. *Prometheus* [online]. [cit. 2023-05-05]. Dostupné z: https://prometheus.io/docs/introduction/overview/

[22] HTTPS V KOSTCE: CO TO JE, JAK FUNGUJE A JAK NA NĚJ PŘEJÍT. *Rascasone* [online]. 2021 [cit. 2023-05-05]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-https-http-ssl-tls

[23] What is AMQP Protocol ? How AMQP Protocol Works. *IoT Boys* [online]. [cit. 2023-05-05]. Dostupné z: https://iotboys.com/what-is-amqp-how-amqp-works-for-internet-of-things/

[24] Using Spring Boot for WebSocket Implementation with STOMP. *Toptal* [online]. [cit. 2023-05-05]. Dostupné z: https://www.toptal.com/java/stomp-spring-boot-websocket

[25] MQTT (MQ Telemetry Transport). *Tech Target* [online]. [cit. 2023-05-05]. Dostupné z: https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport

[26] Introduction to gRPC. *GRPC* [online]. [cit. 2023-05-05]. Dostupné z: https://grpc.io/docs/what-is-grpc/introduction/

[27] RabbitMQ and message brokers. *Fasthosts* [online]. [cit. 2023-05-05]. Dostupné z: https://www.fasthosts.co.uk/blog/rabbitmq-and-message-brokers/

[28] Using the Pika Python client. *RabbitMQ* [online]. [cit. 2023-05-05]. Dostupné z: https://www.rabbitmq.com/tutorials/tutorial-three-python.html

[29] What is Apache Kafka?. *IBM* [online]. [cit. 2023-05-05]. Dostupné z: https://www.ibm.com/topics/apache-kafka

[30] What is a monolithic application?. *Heptio* [online]. [cit. 2023-05-05]. Dostupné z: https://blog.heptio.com/what-is-a-monolithic-application-e375f5ad5ecb

[31] Monolithic vs. Microservices Architecture. *Medium* [online]. [cit. 2023-05-05]. Dostupné z: https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59

[32] What is Model-View and Control? Visual Paradigm [online]. [cit. 2023-05-05]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-model-view-controlmvc/

[33] Event Sourcing Versus Event-Driven Architecture. *Medium* [online]. [cit. 2023-05-05]. Dostupné z: https://medium.com/geekculture/event-sourcing-versus-event-driven-architecture-753aa5a5d0f6

[34] JVM (Micrometer). *Grafana* [online]. [cit. 2023-05-05]. Dostupné z: https://grafana.com/grafana/dashboards/4701-jvm-micrometer/

SEZNAM POUŽITÝCH OBRÁZKŮ

SEZNAM POUŽITÝCH TABULEK

SEZNAM POUŽITÝCH GRAFŮ