VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

PLATFORMA PRO DEFINICI A ZPRACOVÁNÍ DAT
PLATFORM FOR DEFINING AND PROCESSING OF DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Karel Hala
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Kříž, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Karel Hala**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Jiří Kříž, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu 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:

Platforma pro definici a zpracování dat

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je vytvořit platformu, která bude schopná zpracovat data zadaná uživatelem, analyzovat je a na základě vnitřní logiky a informace nesené v těchto datech je uložit do logických celků. Tyto celky poté zobrazit uživateli, nechat jej dále definovat dodatečné informace, provádět reporting případně zobrazit v přehledných grafech.

Základní literární prameny:

ALLAMARAJU, Subrahmanyam. RESTful Web services cookbook. Sebastopol,CA.: O'Reilly, c2010. ISBN 978-0596801687.

HAN, Jiawei, Micheline KAMBER a Jian PEI. Data mining: concepts and techniques. 3rd ed. Haryana, India, Burlington, MA: Elsevier, 2012. ISBN 978-9380931913.

HARTSON, H. Rex a Pardha S. PYLA. The UX Book: process and guidelines for ensuring a quality user experience. Boston: Elsevier, c2012. ISBN 978-0123852410.

MARZ, Nathan a James WARREN. Big data: principles and best practices of scalable real-time data systems. ISBN 978-1617290343.

PROVOST, Foster. Data science for business: what you need to know about data mining and data-analytic thinking. Sebastopol: O'Reilly, c2013. Data Science/Business. ISBN 978-1449361327.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17.

V Brně, dne 28. 2. 2017

L. S.

doc. RNDr. Bedřich Půža, CSc.
ředitel

doc. Ing. et Ing. Stanislav Škapa, Ph.D.
děkan

Abstrakt

Tato diplomová práce se zabývá vývojem platformy pro ulehčení práce s velkým množstvím dat. V rámci této práce jsou vysvětleny některé pojmy a technické souvislosti potřebné pro bližší pochopení vývoje webové aplikace. Dále jsou zde navrženy postupy, jak usnadnit uživateli zadávání a práci s větším objemem dat. Platforma je koncipována tak, aby bylo možné jednoduše a rychle rozšířit jakoukoliv část.

Summary

This diploma thesis deals with creating platform which serves for easy manipulation with large data set. There are numerous technical knowledge described in this thesis to understand web development. Later there are proposed approaches of how to make as easy as possible for user to define and work with large data sets. Platform is written and created in a way, that it is easy to extend eny part of it.

Klíčová slova

big data, podpůrný nástroj, webová aplikace, uživatelské rozhraní

Key words

big data, supportive tool, web application, user interface

Bibliografická citace

HALA, K. *Platforma pro definici a zpracování dat.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 82 s. Vedoucí diplomové práce Ing. Jiří Kříž, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů jsou úplné a že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 24. 5. 2017

.....

jméno

Poděkování

Rád bych poděkoval panu Ing. Jiřímu Křížovi, Ph.D. za vedení a konzultace během této práce. Dále bych chtěl poděkovat rodině a partnerce za morální podporu.

Obsah

ÚVOD **9**

CÍL DIPLOMOVÉ PRÁCE **10**

METODIKA PRÁCE **11**

Metody 11

Postupy 11

1 TEORETICKÁ VÝCHODISKA **12**

1.1 Vývojové platformy low-code 12

1.1.1 Příklady low-code platforem 13

1.1.2 Cloudový výpočet 14

1.2 Business Intelligence (BI) 16

1.2.1 Big data 17

1.2.2 Data mining 19

1.2.3 Extraction, Transformation, Loading 22

1.3 Platforma pro pokročilou vizualizaci dat 22

1.3.1 Single page aplikace 23

1.3.2 Dynamická a interaktivní vizualizace dat 25

1.3.3 Webová služba RESTful 26

1.4 Server pro řízení přístupu a identity 28

1.4.1 JSON Web Token 28

2 ANALYTICKÁ ČÁST **30**

2.1 Webové technologie 30

2.1.1 Webový aplikační rámec 30

2.1.2 Silné vs. slabé typování 32

2.1.3 Responzivní aplikační rámec 32

2.2 Single sign-on (SSO) 35

2.2.1 Standardy získání identity 35

2.2.2 Autentikační servery 36

2.3 Datová analýza 37
2.3.1 Shluková analýza 38
2.3.2 Rozhodovací stromy 39
2.3.3 Dolování dat na webové aplikaci 40
2.3.4 Ukládání a práce s velkým objemem dat 42
2.3.5 Aplikace využívající datovou analýzu 43
2.4 Databázové aplikační platformy 44
2.5 Management vztahu se zákazníky (CRM) 45
2.5.1 Monitorování webové aplikace 46
2.6 Shrnutí 48
3 VLASTNÍ NÁVRHY **49**
3.1 Návrh aplikace 49
3.1.1 Vedení vývojového týmu 51
3.1.2 Popis aplikační funkčnosti 52
3.2 Použité technologie 52
3.2.1 Engine 52
3.2.2 Uživatelské rozhraní 53
3.2.3 Práce s grafy 57
3.2.4 Vyhledávací jazyk 58
3.2.5 Zabezpečení 60
3.3 Dolování dat 61
3.3.1 Heuristiky pro automatické linkován 61
3.3.2 Nejčastěji používané entity 63
3.3.3 Přibližné napovídání 65
3.4 Analýza rizik 66
3.5 Cenový model 68
ZÁVĚR **70**
Seznam obrázků **71**
Seznam tabulek **72**
Seznam rovnic **73**

Seznam grafů **73**

Seznam zkratek **74**

Seznam použitých zdrojů **74**

ÚVOD

V aktuální době je velké množství dokumentů ve firmách zpracováváno nástroji, které do jisté míry nepodporují spolupráci a často tak dochází k vytváření velkých a nepřehledných dokumentů, které se časem zvětšují a po několika letech se musí kompletně přepsat a případně zrušit.

S nástupem internetu tento problém částečně vymizel díky tomu, že se nyní dají takové dokumenty velice snadno sdílet, nicméně stále je zde problém, že velká část nástrojů pracujících s dokumenty nenabízí jednoduché a přehledné provázání. Proto se v rámci této práce zaměříme na možná řešení některých problémů, které vznikají při sdílení dokumentů a při jejich propojování.

Cílem této práce je tedy analýza současných nástrojů a následně návrh a vytvoření specifického nástroje, který se bude zaměřovat z velké části na jeho snadné používání uživateli.

Diplomová práce nás postupně provede několika kapitolami, kdy se nejdříve budeme věnovat teoretickým východiskům, kde si vysvětlíme některé pojmy zabývající se prací s dokumenty pomocí webových aplikací, poté se zaměříme na aktuální stav nástrojů a aplikací zabývajících se touto problematikou. Část popisu aktuálního stavu se budeme věnovat nástrojům, které nám mohou pomoci při samotném vývoji aplikace. Na samotný návrh a vývoj aplikace se zaměříme v poslední kapitole, ve které si rozebereme také případná rizika vývoje takového nástroje. Od čtenáře se očekávají alespoň základní technické znalosti vývoje aplikací a prací s webovými technologiemi.

CÍL DIPLOMOVÉ PRÁCE

Hlavním cílem této práce je vytvořit platformu, která bude schopná zpracovat data zadaná uživatelem, analyzovat je a na základě vnitřní logiky a informace nesené v těchto datech je uložit do logických celků. Aplikace poté umožní tyto celky zobrazit uživateli, nechá jej dále definovat dodatečné informace, provádět reporting a případně zobrazit v přehledných grafech.

Platforma se bude zaměřovat převážně na uživatelské rozhraní, tak aby její používání bylo co nejintuitivnější a nejjednodušší.

METODIKA PRÁCE

Metody

Nejdůležitějším zaměřením této platformy je uživatelská přívětivost a jednoduchost na používání, proto bude při vývoji kladen důraz na spokojenost uživatelů. Tohoto bude dosaženo použitím agilních metodik při vývoji, kdy bude postupně dodávaný produkt předáván úzkému kruhu uživatelů, kteří se budou vyjadřovat k uživatelskému rozhraní. Platforma bude psána jako webová aplikace, která bude přistupovat do databáze přes rozhraní napsané v jazyce Java.

V části zpracování dat bude použito několik ETL metodik a data mining technik, které povedou ke získání logických informací ze zadaných dat. Platforma bude vyvíjena s možností škálovatelnosti a použití nad velkým objemem dat.

Postupy

K vytvoření co nejpřívětivější platformy budou využity naše zkušenosti a knižní publikace zabývající se tímto tématem. Dále budou analyzovány jednotlivé postupy zadávání dat uživatelů do tohoto systému, které povedou ke zpřehlednění a zjednodušení používání.

Pro komunikaci se serverem bude použit standard REST, který usnadní komunikaci se serverem a umožní případné navázání nových aplikací. V případě, že bude vytvořena mobilní aplikace pro získávání dat, nebude nutné psát znovu stejnou nebo podobnou logiku.

Pro zabezpečené přihlášení do aplikace bude použit autentizační server, který bude zajišťovat vytváření a správu uživatelů spolu s jejich právy.

1 TEORETICKÁ VÝCHODISKA

Pro plné pochopení, výběru a případné vypracování platformy je potřeba si objasnit a vysvětlit několik témat. V prní řadě to jsou především *Vývojové platformy low-code* – výsledná platforma by měla splňovat tuto definici. Dále si objasníme pojemy *Business inteligence* – platforma bude z části pracovat s touto oblasti a *Platforma pro pokročilou vizualizaci dat* – pro snadné používání uživatelského rozhraní. A vzhledem k tomu že výsledná platforma musí do určité míry pracovat s uživatelskými právy a spravovat uživatele, objasníme si pojem *Server pro řízení přístupu a identity*

1.1 Vývojové platformy low-code

Vývojové platformy low-code jsou celkem nový pojem – jsou to v podstatě platformy, které napomáhají rychlému vytvoření, nasazení a zapojení aplikace za použití co nejmenšího množství ručního programování. Tyto produkty začaly vznikat, protože malé a střední podniky potřebovaly vytvořit rychle a za použití menšího počtu vývojářů aplikace, které mohou být nadále rychle spravovány. [1]

Toto v podstatě znamená, že vývojáři mohou rychle měnit software na základě uživatelských požadavků, což má za následek spokojenější uživatele, uživatelsky přívětivější software. Tyto platformy neeliminují programování jako takové, ale podporují rychlejší vývoj tak, že poskytují vizuální nástroje a napomáhají konfiguraci datových modulů a díky tomu eliminují problémy spojené s datovou integrací. [2]

Výhody low-code platforem

- **Produktivita:** Systémy mohou být vyvíjeny a nasazeny během menšího časového rozmezí, oproti klasickému programování. [3]
- **Reakční schopnost:** Vývojář může často zvolit různé druhy platforem na kterých bude výsledný produkt fungovat – od mobilních aplikací, až po webové služby. [3]
- **Spolehlivost:** Aplikace mohou být aktualizovány mnohem rychleji, což má za následek jejich stabilitu a spolehlivost. [3]
- **Úspora času a peněz:** Vývojáři mohou vytvořit mnohem více funkcionality

za kratší čas, z čehož plyne, že si firma může dovolit menší počet programátorů. [3]

- **Zaměření na samotný vývoj:** Zaměřením se na to co má aplikace dělat, a ne jak to má dělat, se programátoři mohou zaměřovat na funkcionalitu a uživatelskou spokojenost. Při vývoji je také možné cílit více na uživatelské požadavky mnohem rychleji. [3]

1.1.1 Příklady low-code platforem

Microsoft PowerApps

Vývojová platforma od firmy Microsoft, která dovoluje vytvořit během několika málo kliknutí aplikaci jak pro mobilní platformy, tak jako webové služby. Při spojením této platformy a aplikace Power BI vzniká velice robustní vývojářský nástroj, díky kterému je možné rychle integrovat produkční data do aplikace, kterou budou uživatelé rádi používat. [1]

Zoho Creator

Výhodou této platformy je využití techniky „`drag-and-drop`", která umožňuje vytvářet aplikace a převážně jejich uživatelské rozhraní bez nutnosti psát jakýkoliv kód. [4]

Rollbase

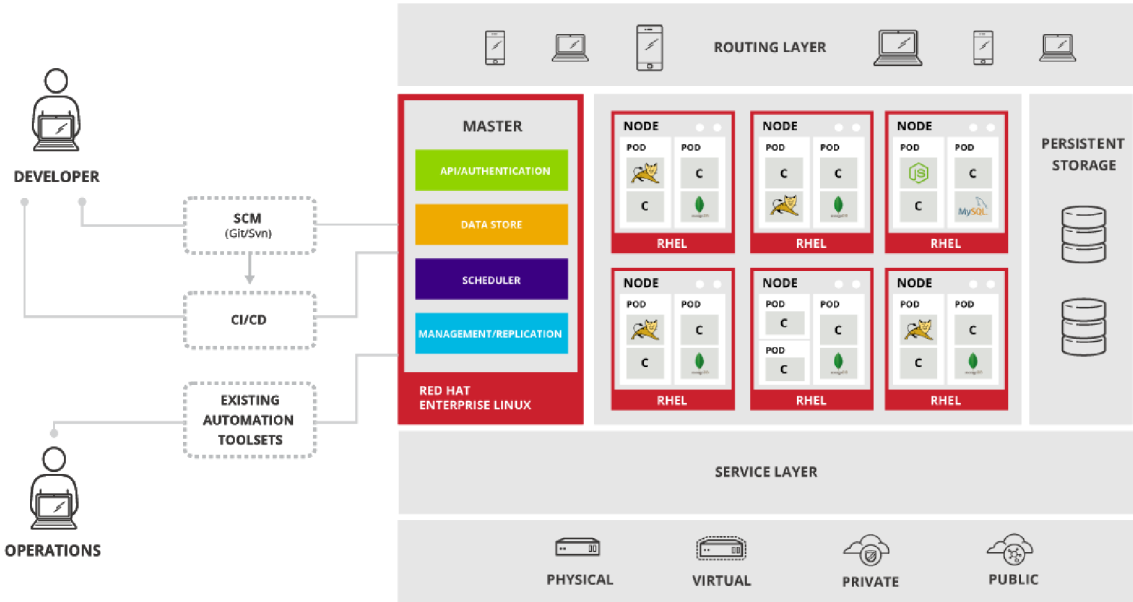
Při používání této platformy vývojář jako první definuje objekty, jejich vlastnosti a vztahy mezi těmito objekty. Po překonání tohoto kroku máme již plně funkční webovou aplikaci, která je použitelná napříč všemi mobilními zařízeními. [5]

Openshift

Platforma pro vývoj webovým a mobilních aplikací, postavená na kontejnerech, které zajišťují rychlý vývoj a umožňují dedikovat vývojáře na vytvoření jednoduchých funckionalit, které fungují jako samostatné aplikace[1]. Tyto malé aplikace zaměřující sena jedoduchou funkci se poté za pomocí Openshiftu spojí dohromady a vytvoří jednu velkou platformu. Na Obrázku 1.1 můžeme vidět, z jakých vrstev se

[1] Takovým aplikacím se říká Mikroservise `https://smartbear.com/learn/api-design/what-are-microservices`.

skládá Openshift a můžeme pozorovat jednotlivé aplikace znázorněné v nodech, dále můžeme sledovat, jaké nástroje nabízí Openshift vývojářům (management zdrojových kódů – SCM a kontinuální integrace – CI/CD).[6]



Obrázek 1.1: Znázornění jednotlivých vrstev v platformě Openshift. Zdroj: [6]

1.1.2 Cloudový výpočet

Výpočet pomocí cloudu je další fáze ve vývoji internetu, cloud v tomto použití znamená, že všechno potřebné pro vývoj a hostování aplikací, až po samostatné stroje, je možné nabídnout jako službu kdekoliv na světě. I když poskytovatelé cloudových řešení mají často velice robustní bezpečnostní systém, je na uživateli, který má uložená data v cloudu, aby zajistil jejich bezpečnost. To znamená, že pokud data uniknou z cloudu kvůli špatnému zabezpečení v aplikaci, která je hostovaná, není to chyba poskytovatele, ale firmy, která takovou aplikaci vydala.[7]

Porovnání ceny cloudového výpočtu a klasického datového uložiště není jednoduché – záleží totiž na několika faktorech. Vezměme si třeba lokaci datového uložiště, pokud je například cena eklektické energie v místě datového uložiště velice levná, nemusí firma být tolik tlačena do cloudového řešení. Pokud ale k těmto datům přistupuje velké množství uživatelů z různých koutů světa může se stát že námi poskytované služby budou neresponzivní a uživatelé mohou odejít ke konkurenci. V tomto momentě je potřeba zvážit, zda se cloudové řešení vyplatí či nikoliv.

Důležité je také uvědomit si, že 42 % nákladů na datové uložiště je vynaloženo na hardware a software (tyto náklady jsou rozloženy v průběhu času) a 58 % nákladů jde do topení, klimatizaci, daně a samostatné práce. [7]

Typy cloudového výpočtu

- **Veřejné** – dostupné pro širokou veřejnost, jak zdarma, tak placené verze.
- **Soukromé** – často používané firmami či skupinami uživatelů, kteří potřebují zabezpečit data. Většinou velice drahé a časově nákladné řešení.
- **Komunitní** – podobné soukromým, ale rozšířené mezi větší skupiny lidí.
- **Hybridní** – vytvořené z více druhů – kombinují privátní a veřejný cloud. Mezi jejich portfolio často patří záloha ke kritickým službám.
- **DaaS (Data as a Service)** – data jako služba, pouze data uložené v cloudu.
- **PaaS (Platform as a Service)** – platforma jako služba, pro vývoj a hostování celého vývojového cyklu, často včetně možnosti nasazení výsledné aplikace.
- **IaaS (Infrastructure as a Service)** – infrastruktura jako služba, opravdové, nebo virtuální počítače nabízené uživatelům. [8]

Platforma jako služba – PaaS

Platforma jako služba – tento pojem označuje službu, která zahrnuje kompletní škálu nástrojů sloužících pro vývoj aplikací (od databází, přes aplikační rámce a testovací nástroje až po nasazení a překlad aplikace). Výhoda této služby je převážně v tom, že všechno je přístupné přes internet a často jako webová aplikace, takže není nutné kupovat velice drahé nástroje. Někteří poskytovatelé nabízí také vyrovnání zatížení, což znamená že pokud je výsledná aplikace pod vysokým náporem uživatelů, automaticky se přiřadí prostředky, aby uživatelé nezaznamenali pád aplikace. [9]

Výhody PaaS

- **Méně kódu** – díky možnosti propojení několika menších aplikací dohromady není potřeba psát stejný kus kódu pořád dokola.
- **Nové možnosti bez potřeby nabírání nových lidí** – díky PaaS dostane tým do rukou sofistikovanější nástroje.
- **Vývoj pro více platforem** – výhodou mnoha poskytovatelů služby PaaS je

možnost překladu aplikací pro různé platformy (několik mobilních platforem a webová aplikace).

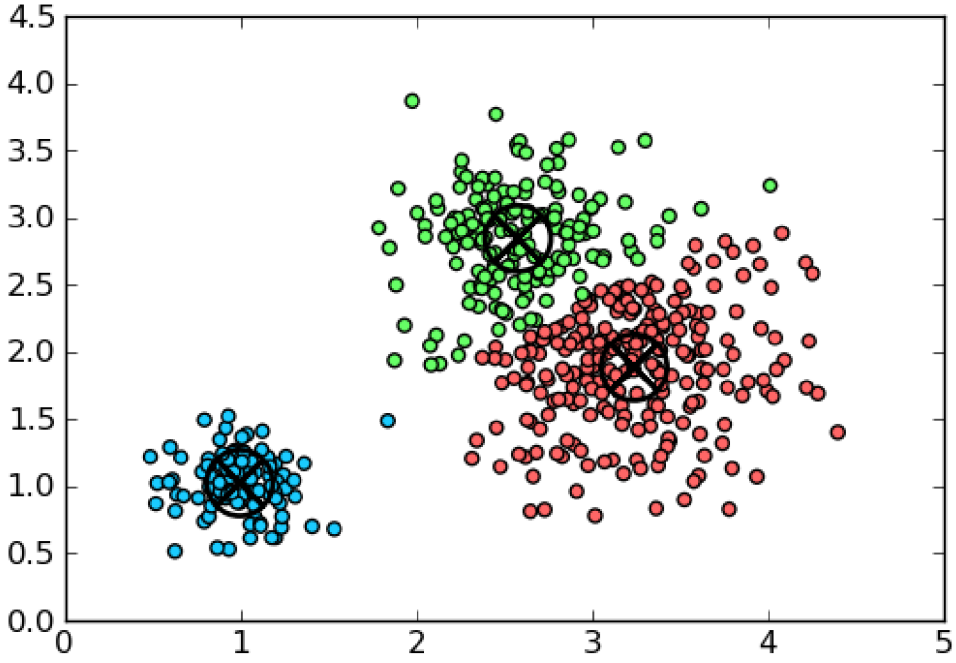
- **Propojení geograficky nesourodých týmů** – pokud je tým rozdělen po různých částech světa, služby PaaS dovolují takto rozděleným týmům pracovat efektivněji.
- **Efektivní životní cyklus aplikace** – v rámci integrovaného prostředí se často nachází funkce pro podporu životního cyklu aplikace (sestavení, nasazení, otestování, správa, aktualizace...). [10]

1.2 Business Intelligence (BI)

Pojem Business Intelligence zahrnuje aplikace, infrastrukturu, nástroje a praktiky, které napomáhají analyzovat informace ke zlepšení a optimalizaci výkonu při rozhodování. Nástroje pro Business Inteligence zahrnují jak samostatná data, tak časovou jednotku, takže můžeme nad těmito daty provádět predikci pomocí sofistikovaných nástrojů a výpočtů. Ze začátku bylo jednoduché provádět výpočty, protože jednoduše firma nesbírala takové množství dat. Aktuálně však není v lidských silách provádět výpočty nad tak obrovským množstvím dat. [11]

Pro dolování dat se používají techniky Data miningu – což je proces, při kterém se prochází velké množství dat, ve kterých se hledají spojitosti pro předpověď vývoje určitého segmentu. Pravděpodobně nejvíce rozšířeným aplikováním dolování dat je marketing – sledování nakupování a chování zákazníků. V této oblasti je možné vybrat si každého zákazníka, a v případě že máme dostatek dat, cílit na něj lépe reklamu. [11]

Pro plné využití BI nástrojů potřebujeme sledované subjekty rozdělit do několika skupin, tyto skupiny musí být co nejvíce **heterogenní** (rozdílné) vůči sobě, a subjekty v rámci jedné skupiny musí být na druhou stranu co nejvíc **homogenní** (stejné). Pokud sledované subjekty spadají do více skupin můžeme použít takzvané **clustery**, což jsou skupiny objektů, které si jsou co nejvíce podobné, ale objekty v jiné skupině se od nich budou co nejvíce lišit. [11] Jak takové clustery vypadají můžeme vidět na obrázku 1.2, kde jsou tři druhy skupin do kterých jsou jednotlivá data rozdělována.



Obrázek 1.2: Příklad dělení dat do clusterů. Zdroj: [11]

Samotná data jako taková však nejsou to nejdůležitější, pokud nebudeme schopni je vhodně interpretovat a využít data, která jsme nashromáždili. Z toho důvodu vzniklo strojové učení. V podstatě to znamená, že pokud předáme stroji dostatečně velké množství dat a nastavíme správně parametry, tak nám mohou stroje umožnit rychlejší interpretaci takových dat. Ale na předpovědi je nutno nahlížet s odstupem a nebrat je příliš vážně a přesně. Naštěstí například pro úspěšný prodej není potřeba přesných předpovědí, stačí pouze vědět kdy a komu poslat přesně cílenou reklamu. Pokud se systém trefí do těchto kritérií, je velká pravděpodobnost, že si námi zacílený zákazník pořídí produkt, který se mu snažíme prodat. [12]

1.2.1 Big data

Označení Big data může dostat jakékoliv množství strukturovaných, nestrukturovaných či částečně strukturovaných dat, které mají potenciál k tomu, aby z nich bylo možné vydolovat nějaké skryté informace. Ve zkratce to znamená, že data začnou být velkými v momentě, kdy jejich zpracování tradičními metodami je časově a technologicky složité.[13]

Big data lze definovat pomocí pravidla **3V** – Volume (obsah), Velocity (rychlost), Variety (různorodost). Pro přesné definování můžeme vzít v úvahu také Veracity (věrohodnost), Validity (přesnost) a Volatility (aktuálnost).[14]

- **Volume** – ve světě Big data máme na mysli opravdu velké množství dat, často v řádech terabytů.
- **Velocity** – rychlost, s jakou jsou data zpracována.
- **Variety** – různorodostí dat máme na mysli jejich formát (*strukturovaná* – klasické relační databáze; *částečně strukturovaná* – emaily, zprávy, atd; *nestrukturovaná* – multimediální obsah).
- *Veracity* – věrohodností rozumíme že data musí být očištěna od zbytečného šumu.
- *Validity* – data musí být co nejpřesnější a co možná nejvhodnější pro naše rozhodování.
- *Volatility* – důraz na aktuálnost dat pro přesnější predikci a jejich pozdější zpracování. [14]

Nejvhodnější místo pro uložení takového množství dat, je cloud, přesněji je žádoucí využít službu některých poskytovatelů privátních nebo veřejných IaaS. Cloud je vhodný pro Big data převážně kvůli tomu, že jak uložení, tak práce s takovým objemem dat, požaduje velké množství distribuované počítačové síly. Výhodami pro uložení velkého množství dat jsou škálovatelnost, pružnost, sdružování prostředků, samoobsluha, nízká pořizovací cena, platba za chodu a tolerance pádů. [15]

Škálovatelnost se zaměřením na hardware znamená, že pro Big data je potřeba přejít z relativně malého výpočetního výkonu na velký během několika chvil bez nutnosti změnit architekturu. Pokud budeme mluvit o softwaru je zapotřebí zachovat stejnou jednotku síly, jak se zvětšuje hardware (při zvýšení objemu dat by mohlo dojít ke značnému poklesu výkonu, pokud na to není systém připraven). [15]

Pružnost je potřeba zachovat co největší – převážně kvůli tomu, že mohou nastat momenty, kdy máme velké množství dat, které se ale v čase mohou smrsknout pouze na zlomek těchto dat. Pokud se tomu tak stane, nechceme nadále platit za zbytečně nevyužitý prostor. [15]

Sdružování prostředků – cloud dovoluje vytvářet skupiny sdílených prostředků. [15]

Samoobsluha – většina poskytovatelů cloudových řešení nabízí možnost samostatně, bez nutnosti kontaktovat IT oddělení, navýšit zdroje, případně je odpojit (pokud již nejsou potřebné). Toto je často řešeno portálem, případně specifickými nástroji u uživatele daného cloudu. [15]

Nízká pořizovací cena – znamená, že často cloudové řešení nestojí uživatele tolik jako pořízení drahých datových skladů. [15]

Platba za chodu se používá u poskytovatelů cloudu jako způsob platby za zdroje, které využíváme. Takže je dost možné, že v průběhu používání cloudu budeme platit různé částky. [15]

Tolerance pádů u cloudových řešení je nutnost a musí být u takovýchto řešení co nejnižší, aby byl zajištěn nepřetržitý chod. [15]

1.2.2 Data mining

Pod pojmem Data mining si můžeme představit získávání skrytých informací ve velkém množství dat. Pokud budeme potřebovat velkého množství dat (Big data 1.2.1), musíme je nejdříve očistit od chybně zapsaných údajů. Dokonce i když máme data ve standardní formě (zjednodušená a optimalizovaná struktura databázových tabulek), nemůžeme počítat s tím, že jsou tyto informace naprosto bezchybné.[16]

Dále některé atributy mohou naprosto chybět pro některé záznamy, pro vypořádání s takto chybějícími parametry můžeme zvolit jednu ze dvou taktik. [16]

1. Odstranění – jednoduše odstraníme celý záznam, ve kterém se nachází nějaký chybějící atribut.
2. Náhrada – použijeme nejvíce frekventovanou, nebo nějakou výchozí hodnotu, na místě ve kterém najdeme chybějící atributy. [16]

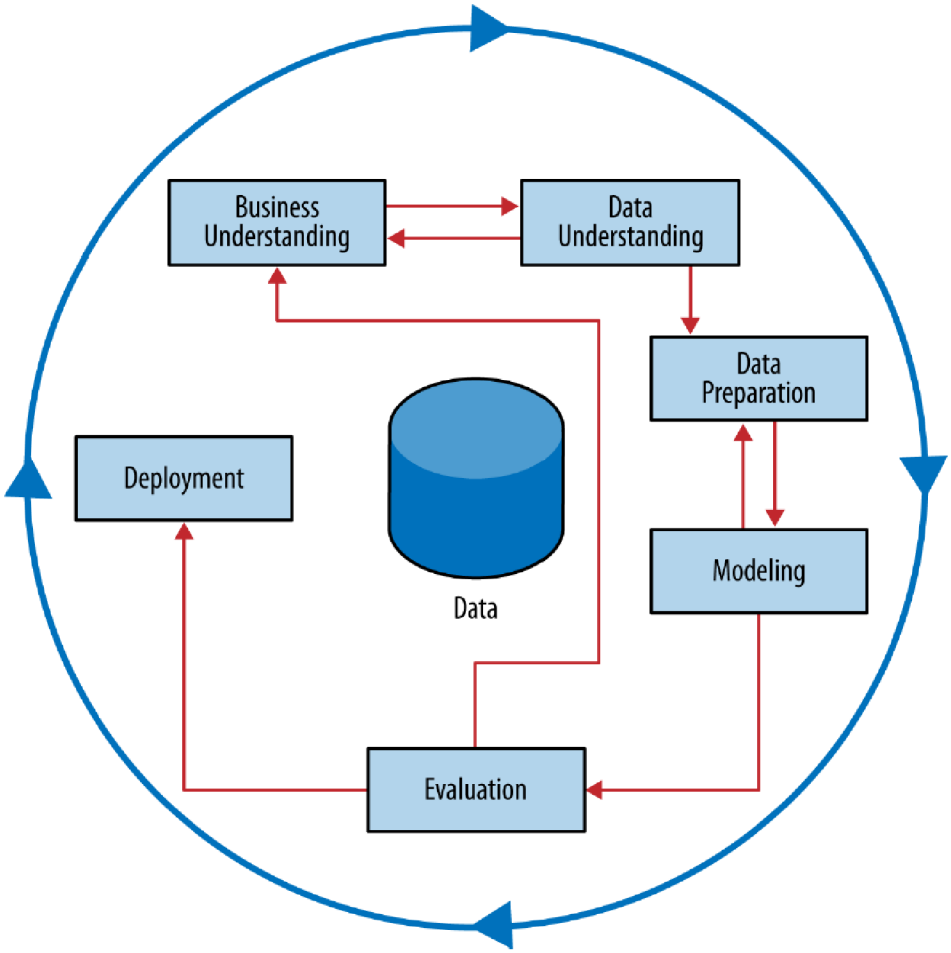
Na obrázku 1.3 vidíme jednotlivé fáze popsané v CRISP-DM[2] modelu, tyto fáze jsou:

- **Pochopení podniku (Business understanding)** – na začátek je potřeba pochopit, co za problém se snažíme vyřešit pomocí dolování dat. Tento první

[2]Zkratka znamená cross-industry process for data mining.

krok je nejdůležitější, ze začátku budeme mít pouze slabé pochopení tohoto kroku a v rámci procesu dolování dat se budeme vracet do tohoto bodu. [20]

- **Pochopení dat (Data understanding)** – často narazíme na data, která nebudou naprosto sedět zadanému problému, proto se musíme zaměřit na jejich silné a slabé stránky. Často se stává že historická data neodpovídají aktuálním problémům podniku. Další složkou, která se projeví do dat, je samozřejmě cena – některé datové sady jsou takřka zadarmo, další se dají pořídit a některé prostě neexistují. Proto v rámci pochopení dat musíme zvážit případné rozšíření datové složky, a zda se nám vyplatí investovat do dalších dat. [20]
- **Příprava dat (Data preparation)** – je dost možné, že tato část bude předcházet **pochopení dat**, protože pro pochopení dat musíme nejdříve tato data očistit od případných chybných a chybějících atributů. [20]
- **Modelování (Modeling)** – primární fáze ve které jsou použity techniky pro dolování dat. [20]
- **Vyhodnocení (Evaluation)** – po aplikování modelování se nacházíme ve fázi, kde musíme ověřit, zda výsledný model je to, co podnik potřebuje. Pokud tomu tak není, vracíme se zpátky na pochopení podniku a opakujeme celý proces znovu. Pro vyhodnocování používáme jak kvantitativní, tak kvalitativní kritéria. [20]
- **Nasazení (Deployment)** – po vyhodnocení, že námi zvolený model je v pořádku a že techniky zvolené pro dolování dat byly korektní, aplikujeme výsledek na reálné využití. Bez ohledu na to, zda nasazení proběhlo v pořádku, vracíme se zpátky do první fáze – pochopení podniku. Další proces dolování dat může vyvodit jiný model, který může být vhodnější pro daný problém. [20]



Obrázek 1.3: Fáze CRISP-DM, referenční model. Zdroj: [20]

1.2.3 Extraction, Transformation, Loading

V případě práce s velkým objemem dat je musíme nejdříve očistit a upravit, tak aby byly co nejlépe použitelné pro náš případ, takovému procesu se říka Extraction, Transformation, Loading ve zkratce **ETL** (jsou to procesy zodpovědné za naplnění dat do datových skladů). Nejdříve jsou data vyextrahována z různých uložišť (Extraction – E). Těmito uložišti mohou být například OLTP (Online transaction processing – detailní a aktuální data v transakčních databázích), starší systémy, webové stránky atd. Poté jsou data očištěna, zbavena duplikátů a chybějících atributů (Transformation – T). Následně jsou data nahrána do datových skladů, kde se s nimi může dále pracovat (Loading – L).

1.3 Platforma pro pokročilou vizualizaci dat

Podniky často zjišťují, že vizualizace dat je pro ně důležitým prvkem v odhalování problémů a monitorování podniku. Ve velké většině případů se stává, že klasické zobrazení dat v podobně reportů za použití tabulek nedokáže zaměřit celý problém, nebo často vede ke špatné analýze. Z toho plyne že je lepší využít grafických prvků (jednoduše řečeno grafů). Pro ještě větší usnadnění analýzy je vhodné doplnění dynamickými a interaktivními grafy. Mezi tyto dynamické a interaktivní prvky patří dashboardy, grafy a tabulky, které se automaticky aktualizují, pokud se jejich datová sada změní.

Pokročilá vizualizace dat vs. statické grafy

- **Dynamické datové sady** – při změně datových sad (databází) se automaticky graf překreslí.
- **Vizuální dotazování** –jednoduchá manipulace s grafy; to znamená, že například kliknutím na sloupec se provede akce, která má za následek překreslení grafu.
- **Několik dimenzí, propojené vizualizace** – klasický graf nedokáže zobrazit závislosti mezi několika dimenzemi, proto je vhodné zobrazit provázané grafy, které reagují v závislosti na navigaci v jedné dimenzi. Například – počet prodaných kusů v čase, při výběru specifického měsíce a roku se automaticky

překreslí graf na počet prodaných kusů za den.

- **Animované vizualizace** – pokud má dimenze velké množství hodnot, je vhodné použít animaci pro jednoduché ovládání a znázornění.
- **Zosobnění** – analytici mohou mít různé pochopení dat a proto je dobré je nechat pracovat individuálně s daty. Často se také může stávat že různí analytici mají přístup k různým datovým sadám, proto je potřeba zajistit pověřovací úrovně.
- **Varování pro podnik** – v případě, že se stane že na obrazovce nachází příliš mnoho dat, může se stávat, že analytici nebudou jednoduše schopni odhalit problém včas. Proto by platforma pro pokročilou vizualizaci dat měla nabízet nějaký druh upozorňování. Tato upozorňování však nesmí být pouze grafického typu, ale také ve formě nějaké zprávy v případě, že se uživatel nedívá přímo na vizualizaci. [21]

1.3.1 Single page aplikace

Tradiční přístup k vývoji webové aplikace je že máme databázi připojenou do serverového back-endu ke kterému jsme schopní se dostat pomocí webového rozhraní. Veškerá práce se odehrává na serveru – to znamená navigace, veškerá logika a práva. To často vede k zatížení serveru a má za následek neresponzivní a pomalé aplikace. Z toho důvodu byl vyvinut nový způsob zobrazování aplikace – takzvané **Single page aplikace** (zkráceně pouze SPA). Tyto aplikace využívají Javascript k vykreslení stránky a k navigaci, což znamená méně dotazů na server při přechodu mezi jednotlivými částmi aplikace. V případě, že server budeme využívat pouze jako zdroj informací, je možné uživateli doručit intuitivní a snadno použitelnou aplikaci, kdy například ani pád serveru, nefunkční nebo pomalé připojení, nemusí znamenat pád aplikace[3]. [22]

Výhody Single page aplikací

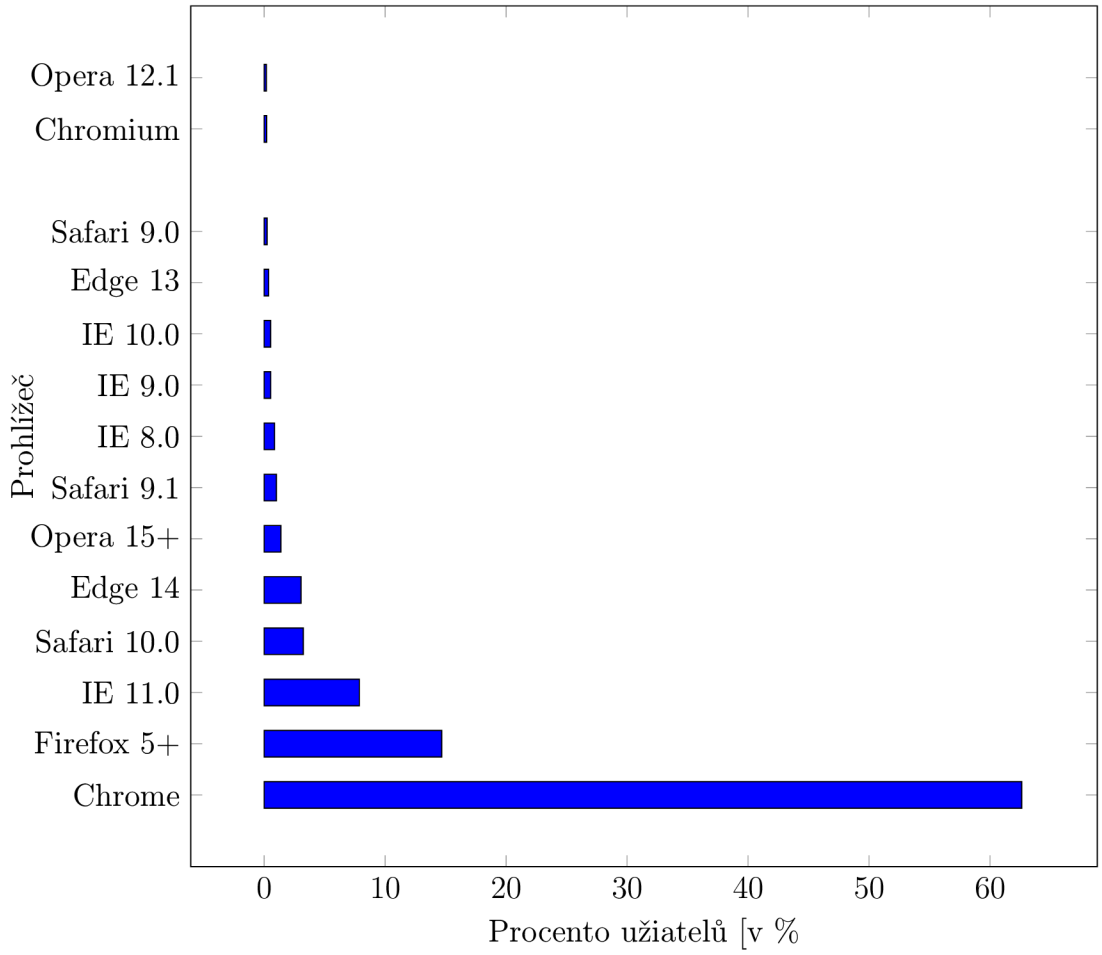
- **Vykreslování** – klasické webové aplikace potřebují při většině uživatelských akcí překreslit celou stránku. To vede k pauze, kdy uživatel čeká nejdříve na

[3]Takovému přístupu se říká Service Worker a více se lze dozvědět na `https://developers.google.com/web/fundamentals/getting-started/codelabs/offline`.

zpracování požadavku na serveru a poté na vykreslení celé stránky webovým prohlížečem. Pokud je uživatel připojen pomalým připojením, nebo je server zaneprázdněný, může tato pauza vykreslování trvat řádově několik vteřin. V případě použití SPA to není potřeba. Vždy se překreslí pouze část stránky a často není potřeba k takové akci činnosti serveru. [24]

- **Responsibilita** – takovéto aplikace minimalizují reakční čas tím, že větší část logiky je přenesena k uživateli. Server má na starosti pouze zpracování dat, jejich zobrazení a případná filtrace (řazení) je provedeno u uživatele. Například pokud uživatel filtruje data pomocí výběru sloupce v grafu není potřeba provést získání dat, tím se nezatěžuje server a stránka zůstává nadále responzivní. [24]
- **Notifikace** – pokud aplikace musí čekat na server může uživateli dát nějakým grafickým prvkem najevo. Případně je možné uživatele upozornit na zprávy ze serveru za pomocí **Webových notifikací**[4]. [24]
- **Přístupnost** – díky tomu, že jsou SPA napsány jako webové aplikace je možné se k nim dostat odkudkoliv, pokud má uživatel připojení k danému serveru, kde je takováto aplikace uložena. [24]
- **Aktualizace** – v případě aplikací, které jsou uloženy u uživatele je často potřeba vydat aktualizační balíček a uživatel si ho musí nainstalovat. V případě SPA tomu tak není, aktualizace se provádí takřka automaticky stačí na server nahrát novou verzi aplikace. Jediný problém je, že si uživatel musí obnovit stránku. [24]
- **Multiplatformnost** – webové aplikace jsou dostupné takřka na všech platformách, od mobilních zařízení až po velkoplošné obrazovky. Limitací je snad pouze používání zastaralých prohlížečů, naštěstí však uživatelé začínají používat modernější prohlížeče a chápou nutnost aktualizací. [24] Jak lze vidět na grafu 1.1, tak modernější prohlížeč Chrome je o hodně používanější než zastaralý prohlížeč Internet Explorer ve verzi 9 a méně.

[4]Detailní popis lze nalézt zde: `https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API/Using_the_Notifications_API`.



Graf 1.1: Podíl vybraných prohlížečů mezi uživateli. Data čerpána z: [25]

1.3.2 Dynamická a interaktivní vizualizace dat

Problém moderní doby je velké množství informací, proto se lidé snaží najít způsob snadné vizualizace velkého množství dat. Jedním z možných způsobů, jak vizualizovat data je pomocí grafů (doslovně je to mapování informací na vizuální objekty). Pokud použijeme webových technologií, získáme silný nástroj v podobě zatraktivnění takovýchto grafů. [26]

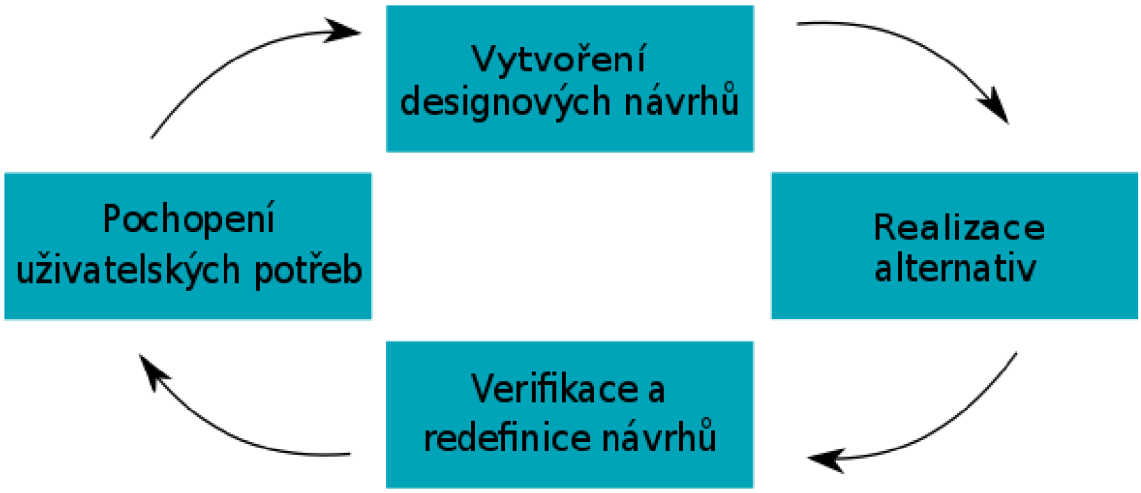
Statická vizualizace je často nepoužitelná a nedostatečná, proto je vhodné použít více grafů najednou, to může vést k nepřehlednosti a ke zmatení. Proto vznikla technologie interaktivní vizualizace, kdy se samotný uživatel snaží vybrat a zpřehlednit data k obrazu svému. Interaktivní vizualizace se změnila pouze nepatrně od roku 1996, kdy Ben Shneiderman (návrhář uživatelských rozhraní) z univerzity v Marylandu pronesl: „Nejdříve náhled, poté zoom a filtrování, na závěr detail na

vyžádání." [26]

Při vytváření interaktivní vizualizaci dat se musíme zaměřit na 4 fáze návrhu a vývoje.

1. **Vytvoření designových návrhů** – navrhneme co a jak bude v našem systému vypadat.
2. **Realizace alternativ** – zapracujeme návrhy a vytvoříme první prototypy.
3. **Verifikace a redefinice návrhů** – kdy předložíme uživateli námi vytvořené řešení a zapíšeme si jeho reakce a chování.
4. **Pochopení uživatelských potřeb** – ze zápisků z předešlého kroku a z požadavků uživatele vyvedeme závěry a opět se vracíme do prvního kroku vytvoření návrh. [24]

Celý koloběh si můžeme prohlédnout na obrázku 1.4.



Obrázek 1.4: Cyklus návrhu grafických prvků. Zdroj: [24]

1.3.3 Webová služba RESTful

Od počátku webových služeb server používal takzvané sezení (anglicky *session*) – uživateli se při každém načtení stránky přiřadil speciální token, který si server uložil a pracoval s ním pro přístup k datům. Server takto často například ukládal historii navštívených stránek na dané webové stránce, případně filtrovaná data atd. Bohužel ukládání těchto dat vedlo k zahlcení paměti serveru a dlouhým odpovědím ze serveru (kdy server musel často poskládat velké množství dat dohromady). Na základě tohoto se zavedla bez stavová komunikace, kdy si server již nedrží stav o uživatelských

akcích, ale odpovídá na všechny dotazy jako na nové sezení. Jedním, kdo využívá tohoto bez stavového přístupu k uživatelům je architektický styl RESTful (zkráceně můžeme používat pouze REST). [27]

Autentizace

Největším problémem při přístupu k datům při použití REST je rozhodně způsob autentizace. Server si nedrží přihlášené sezení, ale musí nějakým způsobem odlišit práva jednotlivých uživatelů. Na základě toho vzniklo několik druhů autentizace při použití REST. [27]

Autentizační způsoby

- **Základní přihlášení – Basic Auth**: Uživatel si nejdříve vytvoří účet (nebo je mu přidělen) a na základě toho zná své uživatelské jméno a heslo. Při každém dotazu na server uživatel pošle toto jméno a heslo a server vyhodnotí, zda mu pošle data, jaká uživatel chce. Tento přístup je jednoduchý, ale velice nebezpečný. Jméno a heslo se posílá jako text, který je pouze spojen pomocí **Base64** do jednoho textu. Pokud server používá nezabezpečené připojení HTTP, útočník může toto heslo a jméno jednoduše získat. Při použití zabezpečeného připojení tento problém mizí. [27]
- **OAuth 1.0**: Je zkratka Otevřené autentizace (Open authentication) ve verzi jedna. Při této metodě odpadá problém, kdy uživatel používá více zařízení k přístupu na danou stránku a jedno zařízení by bylo napadeno a uživatel by byl nucen změnit heslo pro všechny zařízení (kde by byl nucen znovu provést přihlášení). Tento problém je řešen tak, že každé zařízení má speciální přihlašovací token, a pokud jedno zařízení začne vykazovat špatné chování, uživatel je schopen tomuto zařízení zrušit přístup. [27]
- **OAuth 2.0**: Jedná se o vylepšený protokol a je momentálně nejvíce rozšířený. Ve zkratce to znamená to, že není nutné pro každé zařízení generovat specifický token, ale v případě napadaní takového zařízení mu odepřít přístup. Tento protokol využívá prostředníka pro uchování klientských údajů. V podstatě to znamená, že služba požádá jiný server, který spravuje daného uživatele o přihlašovací token. Uživatel je přesměrován na adresu třetího serveru, kam

zadá své údaje, server poté vyhodnotí správnost těchto údajů a službě, která si vyžádala přihlašovací token, odešle buď povolení nebo zákaz přihlášení. Výhodou je, že pokud je uživatel již jednou přihlášen na třetím serveru, není nutné se znovu přihlašovat. A také pokud uživatel již nevěří službě, která má přístup k přihlašovacímu tokenu, jednoduše jí zakáže přístup. [27]

1.4 Server pro řízení přístupu a identity

Mnoho firem stále váhá s přechodem z vlastního řešení (jak databázovými jednotkami, tak i aplikacemi) do cloudu. Hlavní problém je často zabezpečení – firmy jednoduše nechtějí věřit jiné firmě, že je ochrání před případným napadením. Naštěstí mnoho poskytovatelů cloudů si toto uvědomuje, a tak nemalou část financí investují do robustního autentizačního formátu. [28]

Mechanismy používané pro zabezpečení dat v cloudu

- **RSA** – pro přihlášení je potřeba HW token, který je synchronizován se servery cloudu.
- **Více faktorové** – pokud se uživatel chce přihlásit dostane email nebo zprávu s pinem, který po jednom přihlášení, nebo po určitém čase, ztrácí platnost.
- **Single sign-on** – uživatel je přihlášen ke druhé službě a ta jej autorizuje vůči aplikaci, kterou se chystá použít. Výhodou je, že není nutnost vždy zadávat přihlašovací údaje – při přístupu k aplikaci je vyžádán od druhé služby takzvaný JSON web token. [28]

1.4.1 JSON Web Token

Zkráceně JWT je standard pro autentizaci napříč aplikacemi. Takový token nezávisí na programovacím jazyce – nese veškeré informace (přihlašovací údaje a svůj podpis) a je jednoduše použitelný (může být součástí hlavičky nebo součástí dotazu). [29]

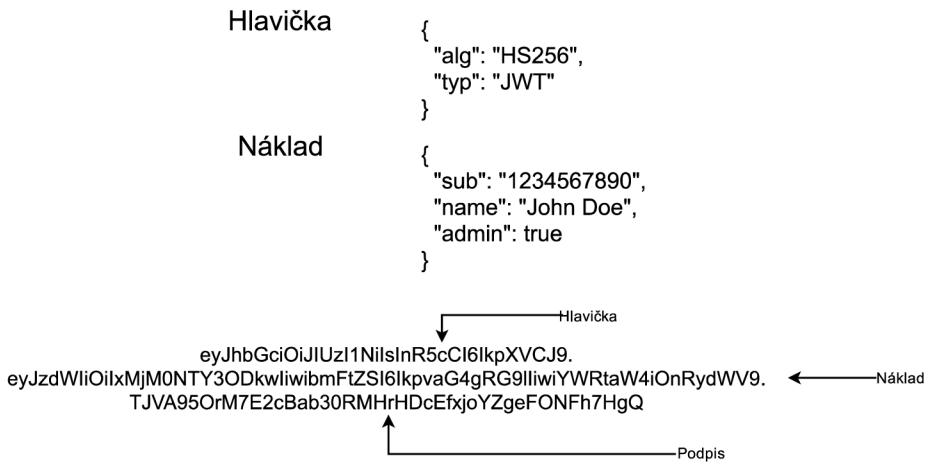
JWT je text, který je zakódování pomocí base64 a rozdělen do 3 částí:

- **Hlavička** obsahuje typ a hashovací algoritmus použitý pro dekódování podpisu.
- **Náklad** zde se nachází data, která služba poskytuje. Často jsou zde přihlašovací údaje (uživatelské jméno a čas vypršení), dále může obsahovat jakákoliv

data – například pohlaví uživatele, věk, email, atd.

- **Podpis** je text, který vznikne spojením hlavičky a nákladu, který je poté zahashován pomocí algoritmu a který je určen v hlavičce tokenu. Součástí tohoto algoritmu také může být tajné heslo. [30]

Příklad, jak vypadá zakódovaný a odkódovaný token můžeme vidět na obrázku 1.5, kde obsah dat, která JWT nese, jsou jméno uživatele, zda je administrátor a čas tokenu. Jak lze vidět, tak součástí hlavičky je také algoritmus použitý pro zakódování.



Obrázek 1.5: Příklad zakódovaného a odkódovaného JWT.

2 ANALYTICKÁ ČÁST

Pro definici a zpracování dat musíme před samotným vývojem platformy provést průzkum aktuálních technologií zaměřujících se na vývoj webových aplikací a jejich výběr. Výběr technologií, které nám pomohou při vývoji je velice podstatný, protože nám urychlí dodání a co je důležitější, pozdější úprava bude značně jednodušší pokud zvolíme technologie, které nám pozdější vývoj usnadní.

Nejenom vývojovými technologiemi je potřeba se zabývat, ale také je nutné věnovat se aktuálnímu stavu trhu a tomu co uživatelé nejčastěji používají. Proto se v této části zaměříme také průzkum trhu.

2.1 Webové technologie

Aktuální trend v používání webových prohlížečů můžeme vidět na grafu 1.1, kde jde jasně vidět že moderní prohlížeče v podání Chrome a Firefox převládají nad čím dál méně používaným Internet Explorerem. Dále si uživatelé již uvědomují, že aktualizace prohlížeče je pro zachování zabezpečení nutností, a tak webové prohlížeče pomalu začínají pořádně fungovat s novým standardem Javascriptu nazvaným EcmaSript 6 (známý též pod názvem ES2015 a ES6). [31]

Nový standard ES6 přináší mnoho vylepšení a mnoho optimalizací. Nicméně ani většina moderních prohlížečů nepodporuje tento standard na 100 % – například prohlížeč **Google Chrome** ve verzi 57 podporuje 97 % nového standardu, obdobně je na tom **Edge** (verze 15 podporuje 95 %) a **Firefox** (verze 52 zvládá 94 %). [32]

2.1.1 Webový aplikační rámec

V aktuální době se mnohým vývojářům webových aplikací ověřily takzvané webové aplikační rámce. Dříve hojně využívána knihovna **jQuery** má již mnoho nástupců jak v podobě knihoven, tak i aplikačních rámců. Výhoda aplikačních rámců je že se stará v podstatě o veškerou složitou a neustále se opakující práci a nechává programátorovi volnou ruku při realizaci samotné aplikace. [33]

V předešlém odstavci bylo použito dvou pojmů, jak Javascriptové knihovny, tak aplikačního webového rámce. Tyto pojmy dost často vedou k hádkám a nedorozuměním, kdy valná většina programátorů nerozumí rozdílům mezi aplikačním rámcem

a knihovnou.

- **Javascriptová knihovna** slouží ke konání jednoho úkolu. Vezměme si například výrobu kávy – můžeme si postavit vodu na oheň ohřát ji, rozemlít zrnka kávy a tento prášek zalít horkou vodou. Každá jedna činnost by představovala knihovnu. [34]
- **Aplikační webový rámec** má na starosti veškerou práci a často zahrnuje přesně definovanou architekturu. Pokud použijeme příklad s kávou, aplikační webový rámec by byl kávovar, do kterého nalijeme vodu a nasypeme kávová zrna. [34]

Mezi aktuálně nejvíce rozšířené aplikační rámce patří ty, které podporují Single page appliacation a jsou to:

- **Angular.js** – plnohodnotný aplikační rámec, který je v aktuální době pravděpodobně nejpoužívanější.[1]. Je to strukturální webový aplikační rámec, pro dynamické webové aplikace. Jeho přední výhodou je, že se soustřeďuje na tvoření jednostránkových aplikací, takže často snižuje počet dat nutných pro načtení a pracování s aplikací. První datum vydaní bylo v roce 2010 a v roce 2016 byla vydána verze 2, která umožňuje vytvářet uživatelské rozhraní jak pro webové, tak pro mobilní aplikace.
- **React.js** – není plnohodnotný aplikační rámec, jako **Angular.js**, nicméně je velkým hráčem na poli vývoje webových aplikací. Jeho výhodou je jeho jednoduchost (nastavení aplikace a počáteční vývoj je velice jednoduchý), nicméně nedokáže tolik věcí co plnohodnotný rámec. Samostatný ovšem není příliš vhodný, potřebuje několik rozšíření, které z něho udělají silnějšího hráče, to ale vede k jeho zkomplikování a častým problémům s nastavením.
- **Vue.js** – progresivní rámec pro vytváření uživatelských rozhraní. Jeho tvůrci kladli důraz na jednoduchost jeho použití a inspirovali se v **React.js**. Výhodou oproti zmíněnému React.js je použití s webovou stránkou, kdy se nesnaží obcházet její vykreslování, ale pracuje přímo s html elementy. Dále je tento rámec také zaměřen na práci s jednostránkovou aplikací, takže je výborným

[1] Přesná čísla se určit nedají, nicméně více informací o popularitě se můžete dočíst zde: `https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282#.ufk9gznfd`.

kompromisem mezi **Angular.js** – který může být příliš náročný pro pochopení, a **React.js** – který může vést ke zkomplikování při použití s více rozšířeními.

2.1.2 Silné vs. slabé typování

Pro vývoj responzivních webových aplikací se používá Javascript, který je ale slabě typovaný. To znamená, že proměnná nemá předem určený pevný typ a může ho během chodu aplikace měnit. To může mít za následek chyby spojené s očekávaným typem, který může být chybný. Takže například při očekávání čísla budeme sčítat, ale aplikace nám během jejího chodu vrátila text. Takový problém může vést až k pádu, případně zamrznutí aplikace. Proto vznikl způsob jak přivést silné typování do slabě typovaných jazyků. Příkladem těchto technik jsou **Typescript** a **Flow**.

Typescript je podmnožinou Javascriptu, to znamená, že veškeré soubory jsou přeloženy do předem vybrané verze specifikace a ty jsou poté spuštěny v prohlížeči. Není tedy třeba nutit uživatele do používání jiného prohlížeče. Velkou výhodou je také to, že se tým okolo Typescriptu snaží co nejvíce sledovat trend vývoje Javascriptu a tak přináší mnoho novinek ještě před tím, než je začnou používat prohlížeče. Takže je možné použít velkou část nových technologií bez nutnosti psát ne zrovna příjemně čitelný kód.

Flow je další způsob, jak přivést silné typování do světa Javascriptu. Oproti **Typescriptu** má výhody, že problémy s implicitní deklarací, které mohou nastat během používání, aplikace jsou lépe vyhodnoceny a programátor je o tomto informován již během překladu aplikace. Ale jeho nevýhodou je, že nesleduje takovou měrou nejnovější trendy a je často pozadu (někdy schválně, protože pro překlad novějších definic do staršího použití existují další nástroje – jako například **Babel**[2]).

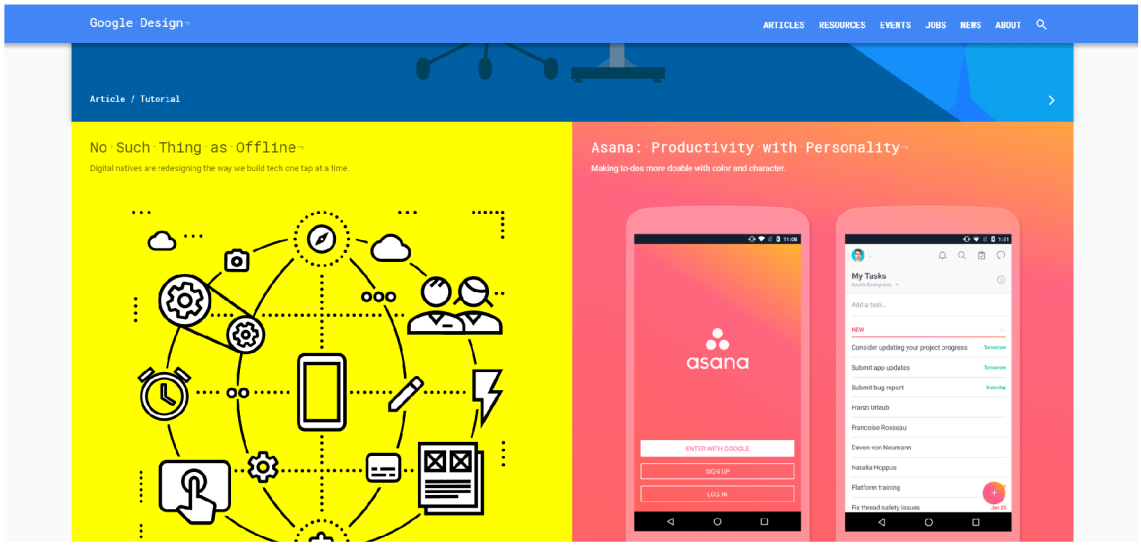
2.1.3 Responzivní aplikační rámec

Pro usnadnění používání webových aplikací na jakémkoliv zařízení vzniká aktuálně mnoho knihoven a aplikačních rámců, které mají za úkol sjednotit design napříč několika aplikacemi a také jejich responzibilitu. To znamená, že stejná aplikace se bude chovat a vypadat stejně bez ohledu na to, na jakém zařízení ji otevřeme (mobilní zařízení, tablet, počítač, televize, atd.)

[2]Více o této technologii můžete dozvědět zde: `https://babeljs.io/`.

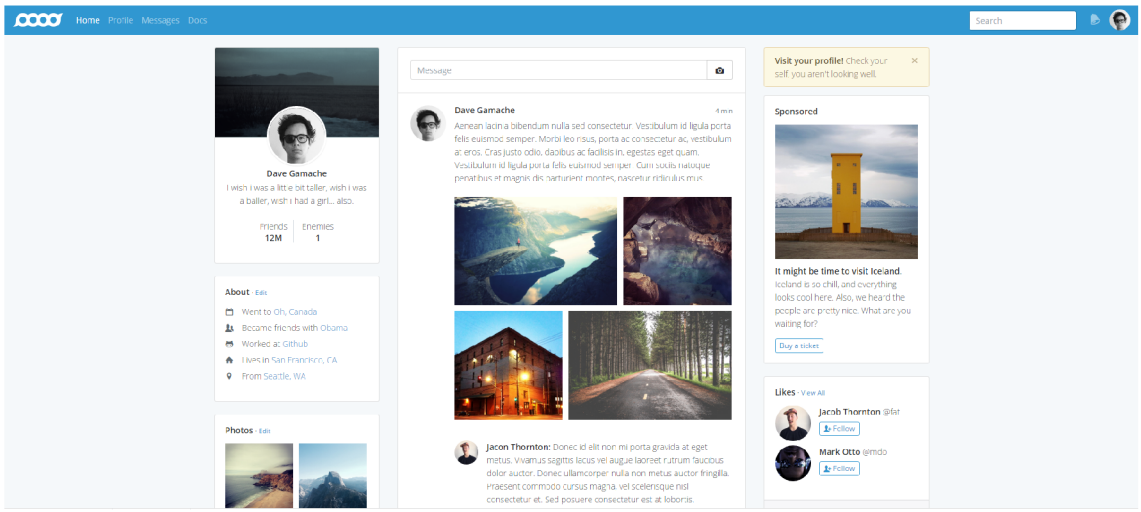
- **Material design** – vznikl na popud zjednodušení a zpřehlednění uživatelského rozhraní, jeho hlavním zaměřením je dotek, hlas a kliknutí. Definuje tři pravidla:
  1. **Material je metafora** – chytře využívat prostor a pohyb jednotlivých elementů.
  2. **Tučné, grafické, záměrné** – základními stavebními bloky jsou typografie, mřížka, místo, barva a použití obrázků.
  3. **Pohyb má význam** – pokud uživatel vykoná nějakou akci design mu napoví jaká akce se stane pohybem. [35]

  Příklad aplikace napsané pomocí Materialu můžeme vidět na obrázku 2.1.

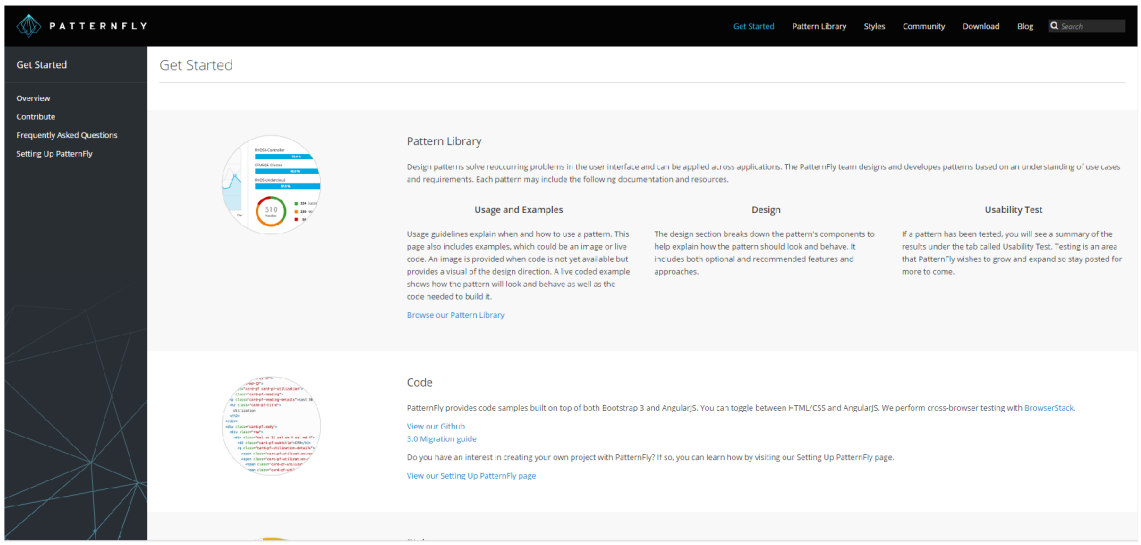


Obrázek 2.1: Příklad tématu pro aplikaci napsanou technologií Material. Zdroj: [35]

- **Boostrap** – jeden z prvních responzivních rámců, který přivedl sjednocení uživatelského rozhraní napříč všemi platformami s největším důrazem na mobilní platformy. Dost často jsou vytvořeny webové aplikace, které nerespektují různá rozlišení pro různé uživatele – tomuto se chtěla firma stojící za aplikací Twitter vyhnout, a tak vytvořila responzivní design s názvem Bootstrap. Nyní nabízí velké množství doplňků a rozšíření, díky kterým je z něj dělají právem nejrozšířenější responzivní rámec. [36] Příklad aplikace napsané pomocí Bootstrapu můžeme vidět na oficiálním tématu na obrázku 2.2.



Obrázek 2.2: Příklad tématu pro aplikaci napsanou technologií Bootstrap. Zdroj: [36]



Obrázek 2.3: Příklad tématu pro aplikaci napsanou technologií patternfly. Zdroj: [37]

- **Patternfly** – společnost RedHat se inspiroval Bootsrapem a vytvořila vlastní responzivní rámec s ázvem Patternfly, který dále nabízí několik widgetů pro zobrazování složitějších uživatelských dat. Jeho hlavním zaměřením je unifikovat jednotlivé aplikace ve firmě, tak aby měli stejný, ale unikátní design. Proto nabízí několik jednoduchých přístupů co a jak dělat pro dosažení co nejvíce podobného vzhledu napříč aplikacemi. [37] Příklad aplikace napsané pomocí Patternfly můžeme vidět na obrázku 2.3.

2.2 Single sign-on (SSO)

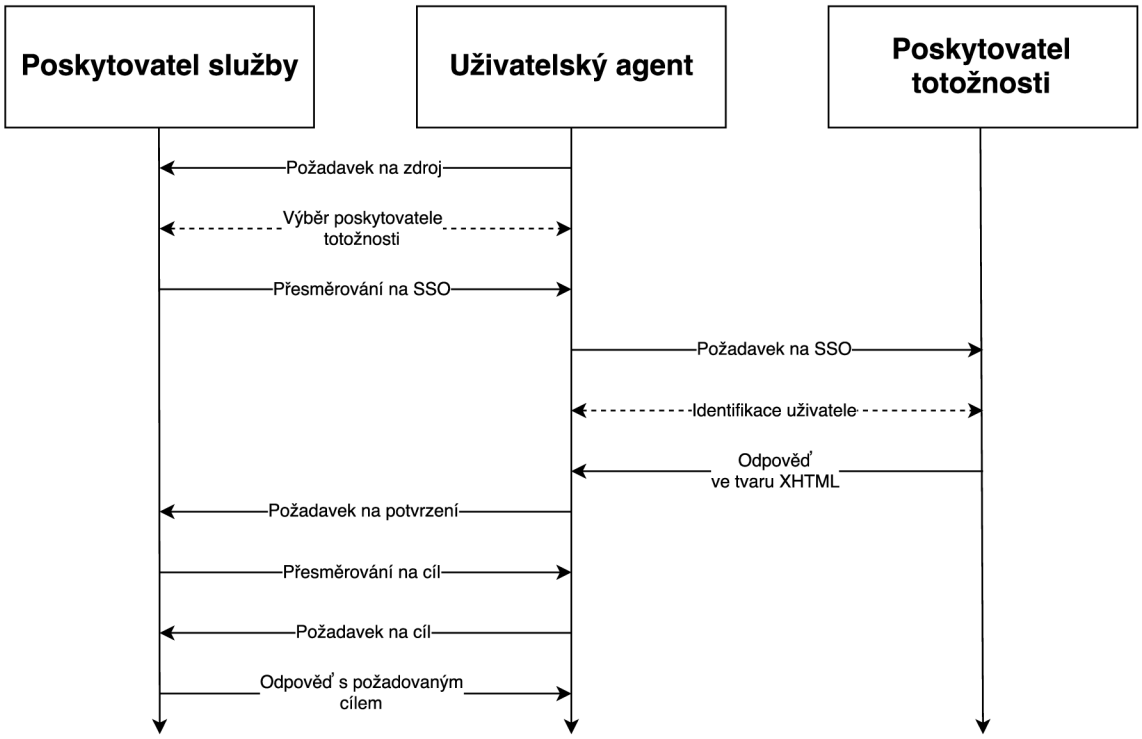
V aktuální době velká část společností nabízejících větší portfolio aplikací volí takzvanou službu single sign-on, kdy uživatel nemusí pokaždé zadávat své přihlašovací údaje, ale o jeho identifikaci se postará specializovaná služba. Většinou se stačí do této služby přihlásit jednou, a dokud nevyprší čas od posledního použití aplikace, která používá tuto službu, nebo není uživatel schválně odhlášen, nemusí se znovu přihlašovat. Jako protiklad je postaven Single sign-off, kdy při odhlášení v jedné aplikaci je uživatel automaticky odhlášen ze všech ostatních aplikací, toto napomáhá dalšímu zabezpečení.

2.2.1 Standardy získání identity

V aktuální době jsou pravděpodobně nejrozšířenější dva standardy použití SSO – novější **OpenID Connect** a starší, ale robustnější **SAML**.

OpenID Connect je v podstatě identifikační vrstva postavená nad OAuth 2.0, poskytuje možnost identifikace uživatele a také získání základních informací. Jedná se o modernější a v aktuální době více používaný způsob – převážně kvůli jeho jednoduchosti a nižšímu zatížení serveru. [38]

SAML se skládá ze dvou částí – poskytovatele služby a poskytovatele totožnosti (tím může být například Facebook, Google, Github, atd.). Jakým způsobem je uživatel přihlášen do služby, můžeme vidět na obrázku 2.4. Nejdříve uživatel vytvoří požadavek na poskytovatele služby, ten vygeneruje SAML požadavek v URL, aplikace přesměruje uživatele na poskytovatele totožnosti, ten ověří přihlašovací údaje, přepošle zpět na poskytovatele služby požadavek na potvrzení, dále je uživatel přesměrován na cílový požadavek, o který si znovu požádá a server mu jej vrátí. [39]



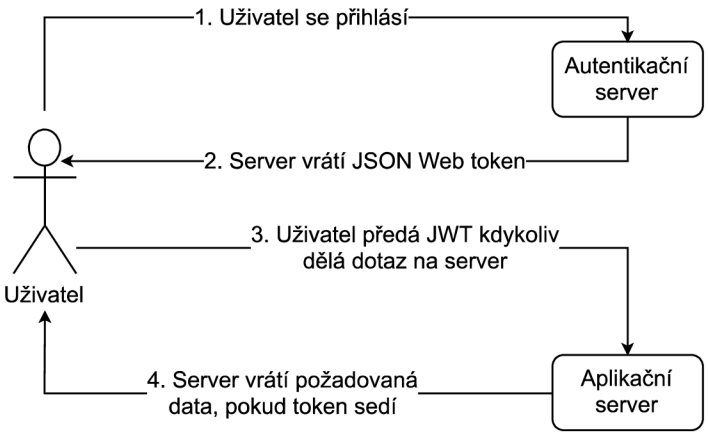
Obrázek 2.4: Předání uživatelského požadavku při použití SAML2. Zdroj: [39]

2.2.2 Autentikační servery

Aktuální trh nabízí velké množství autentizačních serverů, které spravují uživatele a poskytují přihlášení do více aplikací. Některé jsou dokonce volně k použití bez nutnosti zakoupit licenci – je nutné pouze spravovat servery, na kterých takováto služba poběží. Mezi tyto servery patří například **Keycloak**, **Sterling externí autentizační server** a **Active Directory Federation Services**.

Keycloak

Keycloak je open-source software řešení, které nabízí připojení i přes sociální sítě a možnost získání uživatelských účtů z LDAP, Active Directory a nebo přímo z databáze. Jeho velkou premisou je jednoduchost správy uživatelů, včetně definice přístupových práv pro jednotlivé části aplikace. [40] Jak uživatel získá JWT při použití nástroje Keycloak můžeme vidět na obrázku 2.5, kde si uživatel zažádá o nový token, kterým se poté autorizuje u aplikačního serveru.



Obrázek 2.5: Znázornění získání JWT z nástroje Keycloak. Zdroj: [40]

Sterling externí autentizační server

Sterling externí autentizační server označuje řešení od IBM nabízející rozšířené autentizace a validace služeb pro IBM produkty. Nabízí napojení například na LDAP a možnost SSH autentizace. [41]

Active Directory Federation Services

Active Directory Federation Services od firmy Microsoft nabízí možnost sdílet identifikační informace napříč důvěryhodným obchodním partnerům skrze extranet. Výhodou je že se aplikace nemusí starat o všechny uživatele, ale pouze si vyžádá od spřátelené aplikace jeho práva. [42]

2.3 Datová analýza

Pokud firma disponuje velkým množstvím dat, je vhodné nad těmito daty najít nějaké souvislosti, které jí dopomůžou k vytváření dalšího zisku, případně pouze pro pochopení daného zákazníka, nebo skupiny zákazníků. Jestliže je těchto dat opravdu hodně, můžeme je označit za Big Data (popsáno v 1.2.1), poté procesům

k získání dalších informací říkáme dolování dat (blíže popsáno v 1.2.2).

2.3.1 Shluková analýza

Pro bližší pochopení velkého množství dat je vhodné použít metodu shlukové analýzy, což je zjednodušený proces organizace objektů do skupin tak, aby prvky v jedné skupině byly nějakým způsobem co nejvíce podobné. Shlukovou analýzu můžeme rozdělit do dvou skupin **nehierarchické** a **hierarchické**.

Nehierarchické

Nehierarchické metody mohou být tvrdé nebo jemné. Tvrdé seskupují data na základě specifické shlukové vlastnosti – pokud patří do tohoto shluku nemůže patřit do jiného. Jemné na druhou stranu nedefinují přesnou hranici – patří do tohoto shluku, ale má některé vlastnosti datových bodů jiných shluků. Pro dělení můžeme použít metodu K-means, která přiřadí každý bod do shluku jehož středu je nejblíže a při každém běhu algoritmu se středy shluků přepočítávají jako aritmetické průměry všech bodů. [13]

Hierarchické

Hierarchické metody můžeme použít ve dvou způsobech – ze shora dolů a odspodu nahoru. V prvním případě vezmeme všechny datové body a označíme je za shluk, který poté rozdělíme na další shluky, které dále dělíme. Opět můžeme využít K-means pro dělení shluků. Druhý způsob (odspodu nahoru) je nejlepší použít v případě že máme menší vzorek dat a chceme co nejlepší shluk. [13]

Shrnutí

Při použití shlukové analýzy musíme odpovědět na dvě otázky.

1. **Jaký je nejmenší počet shluků?** – Na první otázku je jednoduché odpovědět, je to v podstatě celý seznam datových bodů v jednom shluku.
2. **Jaký je největší počet shluků?** – Odpovědí může být počet všech datových bodů.

Proces shlukování je tedy rozdělování těchto dat do shluků větších jak jedna a menších než počet prvků, proces ukončíme v momentě, kdy jsme spokojeni s celkovým počtem shluků, případně velikostí těchto skupin. [20]

2.3.2 Rozhodovací stromy

Algoritmus rozhodovacích stromů slouží k získání pravidel a vztahů v datovém souboru pomocí větvení. Díky použití stromů je tento algoritmus rychlý a výhodný pro použití s počítačem. Složení stromu je takové, že na vrcholu je jeden uzel, kterému se říká kořen, ze kterého vede několik hran spojujících vnitřní uzly, které mají opět několik hran propojujících další uzly. Každý uzel musí mít právě jednoho předka (ne více ani méně), kromě kořenového uzlu, který nemá žádného předka. Výhodou je jednoduchost, efektivnost a možnost použít i pro velký objem dat. Na druhou stranu pokud nám budou některá data chybět, nebo budeme mít spojitá data rozhodovací stromy budou mít problémy. [16] Při výběru vlastnosti, která je nejvíce odlišná od ostatních příkladů v ostatních třídách se požívá takzvané entropie – její výpočet lze vidět na 2.1, kde p_t je pravděpodobnost výskytu třídy t a t je počet tříd.

$$E(S) = -\sum_{t=1}^{T}(p_t log_2 p_t) \tag{2.1}$$

Rovnice 2.1: Vzorec pro výpočet entropie v rozhodovacích stromech. Zdroj: [16]

Pro vytvoření stromu se vezme tabulka hodnot a pro každý atribut se vypočítá jeho entropie – ta znázorňuje homogenitu prvku. Pokud je prvek naprosto homogenní (nacházející se ve všech atributech), jeho entropie je 0, naopak pokud je prvek naprosto heterogenní (nachází se pouze v jednom atributu), jeho entropie je 1. Při prvotním vytvoření rozhodovacího stromu je tedy potřeba mít co nejlepší trénovací data, nicméně velké množství dat může mít za následek příliš složitý strom, který je opět nepoužitelný (proto je potřeba dbát na vhodnou velikost učebních dat). Pokud nám vznikne složitý a ne příliš efektivní rozhodovací strom, je vhodné použít takzvané prořezávání, kdy odstraníme nadbytečné podstromy (velice vhodné v případě, že jsme použili velký vzorek dat pro vytvoření stromu). [17]

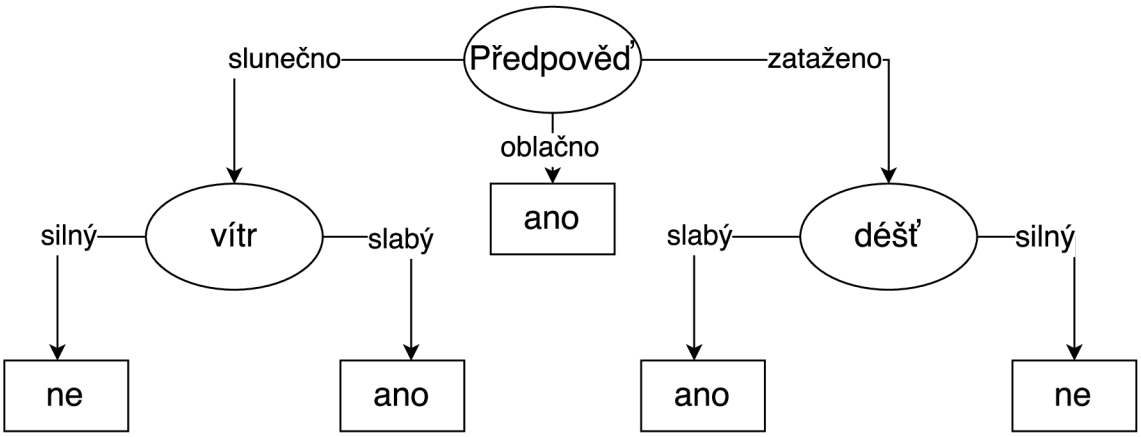
Příklad algoritmu, který se používá pro vytváření rozhodovacích stromů je takzvaný ID3. Ten je založen na booleovských hodnotách, používá hladové vyhledávání v prostoru a strom je vytvářen odshora dolů. Vstupní parametry tohoto algoritmu

jsou:

- **příklady** – data, na kterých je rozhodovací strom vytrénován.
- **atributy** – nad nimiž bude strom testován.
- **cílový atribut** – tento atribut se bude snažit predikovat vytvořený strom. [18]

ID3 algoritmus byl několikrát vylepšen a upraven pro použití za různých podmínek jako například C4.5 (vhodné při chybějících, nebo spojitých datech), C5 (vylepšený C4.5), pro vytvoření binárních rozhodovacích stromů je například vhodné použít klasifikační a regresní stromy (CART), případně pro práci s opravdu velkým množstvím dat je možné použít algoritmus SPRINT. [18]

Na obrázku 2.6 můžeme vidět příklad stromu, který vznikl pro rozhodování zda jít hrát badminton. Pro jeho vytvoření byl použit algoritmus ID3 s menším vzorekem dat.



Obrázek 2.6: Příklad rozhodovacího stromu při použití algoritmu ID3. Zdroj: [18]

2.3.3 Dolování dat na webové aplikaci

Dolování dat na webové aplikaci je získávání informací z webových dokumentů (případně webových stránek), hyperlinků mezi webovými dokumenty a stránkami a získávání informací z logů (serveru).

Na webu lze využít několik možných technik dolování:

- **Dolování webového obsahu** – proces získávání informací z webových dokumentů – mohou to být jak samotné stránky, tak videa, zvukové stopy a obrázky. Hlavním zaměřením je však získání informací ze samotných textů na

webovém dokumentu.

- **Dolování webové struktury** – strukturu webových stránek si lze představit jako, uzlový graf, ve kterém jsou jednotlivé uzly samotné stránky a hrany spojující tyto stránky jsou vzájemné odkazy. Lze se také dále zanořit v jednotlivých dokumentech a ty znázornit pomocí stromové struktury[3].
- **Dolování webového používání** – název pro označení technik, díky kterým lze získat další informace z používání webových stránek. Tato data lze získat z několika zdrojů:
  1. **Webový server** – záznamy, které vygeneroval server během jeho používání (IP adresy uživatelů, navštívenou stránku a čas navštívené stránky ...).
  2. **Aplikační server** – použití moderních aplikačních serverů dovoluje hladší vytvoření podnikových aplikací, tyto servery nadále dovolují bližší sledování uživatelských interakcí.
  3. **Data aplikací** – získávání dalších ifnormací o pohybu a chování uživatele na základě serverových logů (technické informace vypsané serverem s časovým razítkem a případně navštívenou stránkou). [18]

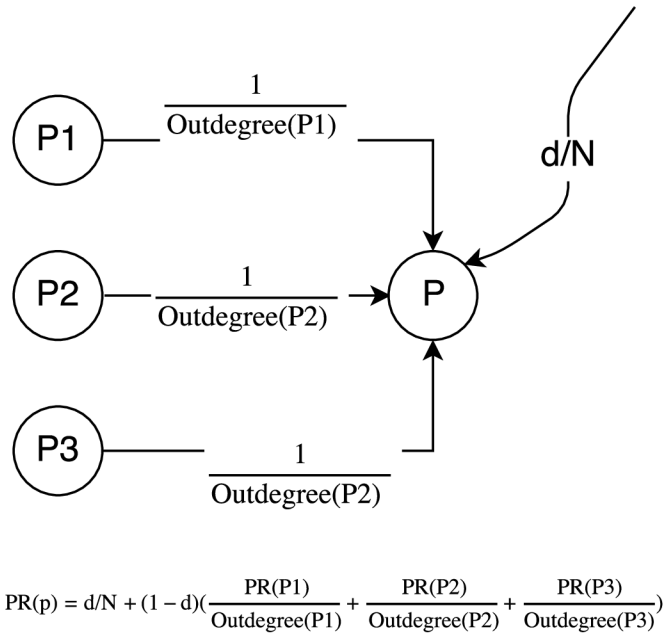
PageRank slouží k hodnocení webových stránek, které pomáhá k tomu, aby daný web byl na vyšších příčkách ve vyhledávačih jako Google, Seznam, Yahoo, atd. Označuje tedy metriku pro ohodnocení dokumentů a zjištění jejich kvality. Výpočet ohodnocení stránky p lze získat použitím rovnice 2.2, kde n je počet odchozích uzlů, $Outdegree(q)$ je počet hyperlinků na stránce q, d znamená pravděpodobnost, že uživatel zadá stránku přímo bez prokliknutí z jiné stránky a $1 - d$ označuje pravděpodobnost, že uživatel navštíví stránku z prokliknutého hyperlinku. [18]

$$PR(p) = d/n + (1 - d) \sum_{(q,p) \in G} \left(\frac{PR(q)}{Outdegree(q)}\right) \tag{2.2}$$

Rovnice 2.2: Vzorec pro výpočet ohodnocení stránky. Zdroj: [16]

[3]Přesný popis tohoto zápisu je znám pod zkratkou DOM (Document object model).

Pro znázornění ohodnocení náhodné stránky se můžeme podívat na obrázek 2.7, kde na jednu stránku ukazují tři stránky **P1**, **P2** a **P3**.



Obrázek 2.7: Příklad hodnocení stránky pomocí Markovova modelu náhodné stránky. Zdroj: [18]

2.3.4 Ukládání a práce s velkým objemem dat

Při použití většiny technik na dolování dat je potřeba toto dolování spustit nad velkým množstvím dat, proto vzniklo několik technik jak na ukládání, tak na práci s velkým objemem dat. Pro ukládání se často příliš nehodí klasická relační databáze, a proto se používají například takzvané NoSql databáze. Jednou z technik, jak propojit několik datových uzlů, je použití takzvaných **Datové sklady** – ty mají možnost připojit se na několik databází (bez ohledu na jejich typ) a za pomocí sofistikovaných reportingových nástrojů vytahovat z těchto databází informace, které se dále transformují a upravují. [19]

V případě použití NoSql databáze je potřeba data nějakým způsobem provázat (indexovat), aby s nimi šlo pracovat ve velice rychlém časovém úseku bez nutnosti při každém dotazu brát data přímo z databáze. Mezi indexovací nástroje patří například

nástroje **Solr**[4] nebo**Elasticsearch**[5]

Pokud naopak využijeme datových skladů, můžeme pro následnou vizualizaci využít takzvané OLAP kostky, což jsou sady nástrojů pro práci s daty (mezi tyto nástroje patří například pivoting, řezy, drill down/up atd.)

2.3.5 Aplikace využívající datovou analýzu

V průběhu využívání dolování dat a využívání datové analýzy se nástroje a aplikace postupně zdokonalovaly a začínají být čím dál více uživatelsky přístupné. Firmy se snaží propojit několik takových aplikací a dát uživatelům silné nástroje pro snadné a rychlé získání dodatečných informací z velkého množství dat. Často se objevuje možnost připojit různé zdroje dat do těchto aplikací.

Power BI – je sada obchodních analytických nástrojů od firmy Microsoft, tyto nástroje vynikají především možností připojení na velké množství datových zdrojů a také možností publikovat vytvořené reporty na webové rozhraní (na takzvaný dashboard). Pro připojení na různé datové zdroje je použit nástroj Power BI gateway, který umožňuje (mimo jiné), připojení na různé SQL databáze a analytické modely. [43]

IBM Watson Analytics – hlavní motto tohoto nástroje je poskytnutí silných a pokročilých analytických technik bez přílišné komplexnosti. Výhodou oproti ostatním nástrojům a aplikacím, které se věnují datové analýze, je jednoznačně možnost rychle (a bez nutnosti pokročilých technik dolování informací) získat ukryté informace v zákaznických datech. Tohoto docílili inženýři ve firmě IBM tím, že vyvinuli velice silnou a rychlou umělou inteligenci[6], která má za úkol hledat skrytou podstatu v datech. [44]

MicroStrategy – platforma podporující dashboardy, interaktivní reporty a dotazování, upozornění na události a mnoho dalších technik potřebných pro vytváření strategie podniku a zjišťování informací z dat. Tato platforma nevyužívá klasické

[4]Jednoduchý a otevřený indexační nástroj `http://lucene.apache.org/solr/`.

[5]Nástroj, který vyhledává a dokáže také analyzovat data `https://www.elastic.co/`.

[6]Více o této umělé inteligenci pojmenované Watson se můžete dočíst zde: `http://www.slate.com/blogs/future_tense/2014/02/14/watson_is_real_artificial_intelligence_despite_claims_to_the_contrary.html`.

multidimenzionální OLAP kostky, ale využívá relační OLAP architekturu, což umožňuje použití nástroje drill down v jakékoliv dimenzi. Výhodou této platformy pro programátory je jistě dodávaný vývojářský balíček, který umožňuje další upravování vytvořených reportů a grafů. [45]

SAP – informační systém, který nabízí hned dvě řešení pro možné dolování informací z dat – **SAP BusinessObjects** a **SAP HANA**. První ze jmenovaných je sada fron-end aplikací, která nabízí uživateli možnost prohlížet, řadit a pracovat s BI datami. Druhým řešením je platforma SAP HANA, která má za úkol zpracovávat velké množství dat v reálném čase. Jako bonus firma SAP nabízí vývojářský balík, kterým si uživatelé mohou upravit tuto platformu a rozšířit tak její využití. [46]

2.4 Databázové aplikační platformy

Mnoho firem v současné době nabízí různá řešení platforem, které zobrazují a umožňují práci s daty bez nutnosti instalace sofistikovaného programu u uživatele, ale za pomoci webového rozhraní. Tyto aplikace jsou převážně inspirovány úspěchem programu Excel od firmy Microsoft, který v aktuální době používá více než 1,2 miliardy uživatelů[7].

Firma Microsoft začala v nedávné době využívat možnost sdílet dokument mezi uživateli, kteří v reálném čase vidí změny prováděné na daném dokumentu. Podobnou funkci nabízí také firma Google, nicméně již nenabízí plnohodnotnou aplikaci, kterou by měl uživatel nainstalovanou na počítači a která by umožňovala uživateli pracovat pohodlněji s dokumenty (jak tomu je při použití programu od firmy Microsoft).

Fusioo – webová aplikace, která nabízí jednoduchou integraci v rámci týmu pro správu důležitých informací. Je zde možné si nastavit dashboard, který nabízí metriky, grafy a upozornění. Jeho velkou výhodou je možnost provázanosti do kalendáře, díky kterému je možné jednoduše spravovat tým a jeho aktivity.

Ragic – tato platforma nabízí uživatelům možnost snadného přechodu z klasických excelových tabulek do databázového světa bez nutnosti jejich pochopení. Hlavním

[7]Podle oficiální zprávy od Microsoftu v roce 2016 `http://www.windowscentral.com/there-are-now-12-billion-office-users-60-million-office-365-commercial-customers`.

tahákem této platformy je určitě její chytré a intuitivní vyhledávání a při zadávání dat také možnost zapnutí validace pro jednotlivé položky. Oproti konkurenci nabízejí neobvyklé reporty jako například TODO listy, nálepky, kontingenční tabulky a mapy.

Quickbase – aplikace zaměřená na pokročilé uživatele, nenabízí příliš jednoduchý způsob zadání dat, nicméně dovoluje upravovat data v připojené databázi za pomoci nástroje, který připomíná Access od firmy Microsoft. Tato platforma se zaměřuje převážně na vývoj aplikací bez nutnosti psát kód – takové aplikace poté může zákazník dále nabízet ostatním uživatelům. Firma stojící za aplikací Quickbase nabízí možnost definovat přístupová práva k jednotlivým dokumentům, skrze nastavení přístupových práv pro jednotlivé skupiny. Aplikace disponuje jedním velice zajímavým nástrojem Ganttovým diagramem, který napomáhá k rozvržení práce v rámci týmu a naplánování vývoje.

Knack – velice jednoduchá, a snadno použitelná aplikace, která dovoluje uživatelům definovat databázi a následně ji spravovat online pomocí webového rozhraní. V rámci definování dat je možné nastavit jejich strukturu (určit jednotlivé typy pro záznamy), propojení (záznamy mohou být navzájem propojené) a získat tak další informace. Dále uživatel může také definovat vzorce a formule. Nad konkurencí tento nástroj vede převážně díky možnosti snadno importovat a exportovat data.

Nintex workflow – platforma, která nabízí automatizaci procesů v rámci firmy, což znamená, že zákazník může propojit jednotlivé aplikace (které jsou kompatibilní s aplikací Nintex) a systémy do určité posloupnosti úkolů, které dostanou zaměstnanci. Výhodou této platformy je především možnost propojení do velkého portfolia firemních aplikací – například NetSuite, Microsoft Dynamics a SAP, případně je možné vyvolat akci v podobě odeslání emailu nebo SMS.

2.5 Management vztahu se zákazníky (CRM)

CRM (Client Relationship Management) je termín používaný pro označení strategií a technologií používaných společnostmi pro monitorování a zjišťování stavu, co uživatelé dělají s jejich produkty – používají se jak k monitorování webových aplikací, volání, chatování, mailů a sociálních sítí. Mezi funkce CRM patří **automatizace marketingu**, **automatizace prodeje**, **automatizace kontaktního centra**

a **geolokační technologie**. [47]

- **Automatizace marketingu** napomáhá marketingovému oddělení některé repetitivní úkoly provádět automaticky. Jako například při zavedení nového produktu není nutné psát každému zákazníkovi speciální email, ale je možně nechat CRM systém vygenerovat speciálně cílenou reklamu pro každého zákazníka.
- **Automatizace prodeje** eliminuje snahu prodat stejný produkt různými zaměstnanci víckrát. Napomáhá ve sledování, kdo a s jakou úspěšností se snaží daný produkt komu prodat.
- **Automatizace kontaktního centra** usnadňuje komunikaci se zákazníky, dovoluje například nahrát zprávu, která se ozve všem zákazníkům s určitým problémem nebo dotazem (pokud se například problém nebo dotaz opakují).
- **Geolokační technologie** – některé CRM systémy dokonce disponují geolokační službou, na základě které může podnik zjistit prodej určitého produktu napříč státy. [47]

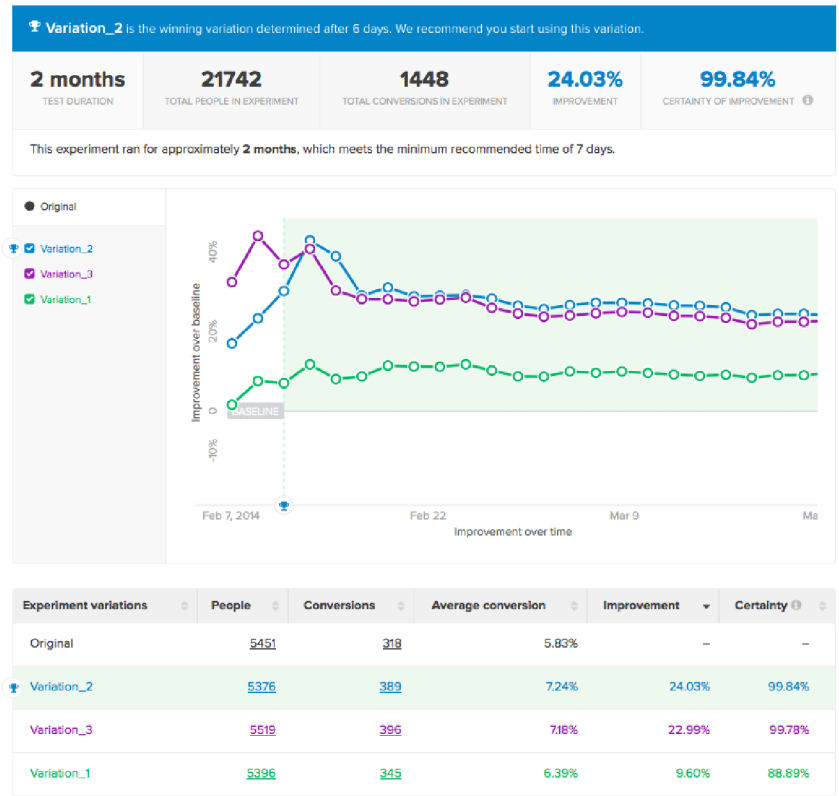
2.5.1 Monitorování webové aplikace

V rámci zjišťování chování zákazníků na stránce může firma sáhnout po speciálním softwaru, který dovoluje jejich přesné monitorování. Díky tomuto monitorování může firma zjistit například v jakém kroku nákupu produktu zákazník odešel ze stránky a hlavně díky tomuto nástroji firma získá možnost přilákat nové zákazníky díky zjištění chování stávajících zákazníků. [24]

Google Analytics je poskytován zdarma od firmy Google, poskytuje statistické a základní nástroje analýzy používání webové stránky. Kromě toho, že je produkt zdarma má další výhody v podobě napojení na další nástroje od firmy Google, jako například **AdWords**[8]. Služba Google Analytics nabízí základní grafy a reporty spolu s možností zobrazit je na dashboardu. Těchto druhů dashboardu může být několik typů – základní, SEO analýza, sociální média, geografie, mobilní analýza, příchozí/odchozí a technický (uživatel si samozřejmě může vytvořit vlastní dashboard).

[8]Placená služba, která dovoluje zákazníkům předplatit si reklamu na webových stránkách.

KissMetrics je placená alternativa ke Google analytics, cílí především na velké webové stránky a nabízí mnoho způsobů jak monitorovat zákazníky. Mezi přední výhodu patří možnost identifikace uživatele ještě před jeho přihlášením, kdy se ukládá jeho pohyb do anonymního účtu, a pokud se tento uživatel v budoucnu identifikuje, všechna anonymní data se automaticky spárují s tímto účtem. Tento nástroj také nabízí mnohem jednodušší možnost nastavení A/B testování[9], kdy není potřeba vytvořit dvě různé URL pro jednu stránku. Dále tento nástroj nabízí velice jednoduché nastavení sledování chování uživatelů na jednotlivých stránkách, kdy je velice snadné nastavit například sledování času stráveného na stránce, dobu vyplňování formuláře, pohyb kurzoru po stránce atd. Na obrázku 2.8 můžeme vidět výsledky A/B testování v nástroji KissMetrics.



Obrázek 2.8: Data získána z A/B testování v nástroji KissMetrics. Zdroj: [48]

[9] A/B testování je v podstatě vytvoření dvou různých variant jedné stránky a testování, která je více úspěšnější mezi uživateli (více stráveného času, koupě produktu, více kliknutí na stránce ...).

2.6 Shrnutí

Aplikace věnující se ukládání a práci s daty pomocí webového rozhraní se zaměřují převážně na větší podniky a obvykle nemají tolik intuitivní uživatelské rozhraní. Vylepšování dostupných aplikací není snadné a nastavení uživatelských práv je pouze v mizivém množství dostupných aplikací. Proto se dále zaměříme na vývoj samotné aplikace, která spojí myšlenky z aktuálních aplikací a využije open source nástroje, které umožní snadnější práci s uživateli.

3 VLASTNÍ NÁVRHY

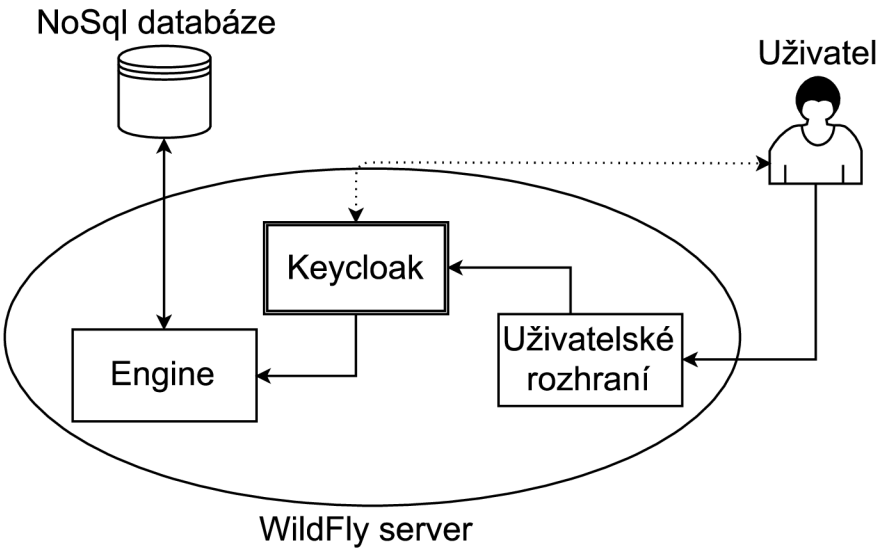
Nástroj, který je součástí této práce se zaměří z velké části na uživatelské rozhraní a do jisté míry využije technologie a nástroje, které jsou dostupné a usnadní tak vývoj a nasazení nového nástroje. Dále se budeme věnovat životaschopnosti tohoto nástroje a způsobem, jakým bude vytvořený nástroj nabízen veřejnosti. Firmy, pro které bude tato aplikace doporučována jsou malé, až střední podniky, které pracují s větším množstvím dat a potřebují nějakým způsobem najít skryté informace. Celá aplikace je vyvíjena v rámci open-source licence, proto je možné najít celý její kód (všech částí) na adrese: `https://github.com/Lumeer`.

3.1 Návrh aplikace

Aplikace je postavena na základě aplikačního serveru WildFly, který pracuje s moduly napsanými v jazyce Java. Základní rozdělení aplikace je na **Engine** (veškerá serverová logika) a **UI** (webové uživatelské rozhraní). Engine je připojen na databázi MongoDb, což je NoSql databáze. Výhodou tohoto nastavení je vysoká pružnost v rámci zapsaných dat a není tedy nutné přesně definovat závislosti v rámci databáze (každý zákazník si může definovat jiné důležité atributy pro každý projekt).

Další výhodou oddělení uživatelského rozhraní a samostatného Enginu je možnost lépe řídit vývojářský tým, lépe rozdělit práci a v neposlední řadě, také možnost později vytvořit klienta nezávislého na dosavadním uživatelském rozhraní – například vytvoření mobilní aplikace, která se bude specializovat na určitou část systému. Celý systém spolu komunikuje pomocí REST rozhraní a o zabezpečení se stará vrstva Keycloak, ve které jsou oba moduly registrovány a slouží jako takový středobod celého systému. Jak jsou jednotlivé části propojeny můžete vidět na obrázku 3.1, kde hrany znamenají komunikační zprávy a uzly logické bloky.

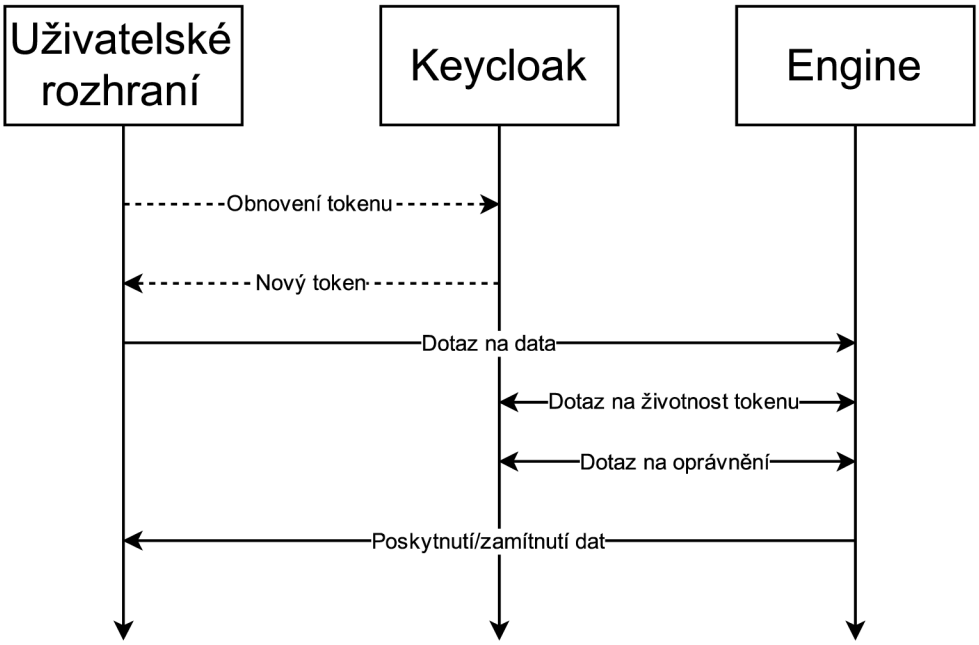
Pro znázornění jak dochází k získání dat se můžeme podívat na diagram 3.2. Když uživatel vyvolá jakoukoliv akci, která vyžaduje získání dat ze serveru, uživatelské rozhraní se nejdříve zeptá Keycloak serveru, zda má aktivní token stále validní a pokud ne, autentizační server mu vytvoří nový. Poté uživatelské rozhraní pošle dotaz na samotný engine (v tomto dotazu se nachází autentizační token), kde se zkontroluje opět životnost tokenu, a zda má uživatel práva pro daná data. Pokud je



Obrázek 3.1: Diagram znázorňující schéma nástroje.

vše v pořádku, server vrátí data, pokud ne, odpoví chybovou hláškou. Výhodou takové komunikace je, že nezáleží na implementaci uživatelského rozhraní, to může být napsáno jak pomocí webových technologií (Javascript), tak jako mobilní aplikace.

Pokud uživatel již disponuje velkým datovým uložištěm je možné ho připojit do systému jednoduchou konfigurací v rámci UI. Systém prozkoumá tato data, některá si nakopíruje a začne je uživateli nabízet podobně, jako kdyby je uživatel měl již dříve v systému. Stejně je tomu s dolováním dat – modul, který je zodpovědný za samostatné dolování, je možné připojit do systému, odkázat jej na datové uložiště a spustit dolování dat. Vše je nastavitelné jak z uživatelského prostředí, tak pomocí volání na předem definované RESTové služby.



Obrázek 3.2: Diagram znázorňující schéma nástroje.

3.1.1 Vedení vývojového týmu

Pro snadný a rychlý vývoj bylo nutné zvolit vhodné vedení vývojového týmu, který se podílel na vývoji celé aplikace. Ze zkušeností bylo určeno, že rychlého vývoje se dosáhne použitím agilní metody vedení týmu (inspirováno scrumem). Nicméně, protože jednotliví členové týmu se nenacházeli v dostatečné vzdálenosti, a nebyl tento projekt veden jako hlavní úkol jednotlivých členů, některá setkání, která jsou definována ve scrumu byla vyškrtnuta. Nejenom proto nelze vedení týmu, které bylo zvoleno, označit jako čistokrevný scrum, ale pouze jsme se při vedení inspirovali touto metodikou.

Nastavili jsme třítýdenní zveřejnění jednotlivých částí a každý týden jsme se scházeli, abychom si ujasnili na čem jaký člen týmu dělá a zda nemá nějaký problém. Použili jsme také nástroje určené ke snadnějšímu rozdělování práce, kdy jsme začali s používáním nástroje **Trello**[1], který nám postupně přestal stačit, a tak jsme využili

[1]Jednoduchý nástroj, který vizualizuje aktuální práci za použití jednoduchých tabulí `https://trello.com/`.

opensource licence **Youtrack**[2]

3.1.2 Popis aplikační funkčnosti

Před samotným popisem co a jak je propojeno, si musíme určit, co vlastně bude tato aplikace dělat a důvod jejího vzniku. Hlavním důvodem vzniku této aplikace je usnadnění editace často složitých záznamů (souborů) pomocí moderních technologií, záznamy můžeme rozumět například tabulky, grafy, kontingenční tabulky, atd. Mezi jednotlivými záznamy mohou vznikat a zanikat propojení, která umožní snadné napovídání datových záznamů a popisu dat (v případě tabulky si toto lze představit tak, že nám aplikace bude napovídat možné záhlaví tabulky a data v tabulce). Editování a nahlížení do záznamů bude povoleno pouze určitému uživateli. Dále jednotlivé soubory bude uživatel moci seskupovat do takzvaných kolekcí. Tímto dosáhneme jisté abstrakce nad soubory tak, aby uživatel přibližně tušil již při prvním pohledu na záznam, jaká data v něm mohou být. Vzhledem k tomu, že aplikace bude cílena na větší množství uživatelů, je možné aplikaci přepínat mezi různými projekty a organizacemi, takže uživatel může mít několik záznamů se stejným jménem ve stejné kolekci pod různými projekty a organizacemi.

3.2 Použité technologie

Může se zdát, že pro vytvoření tohoto nástroje bylo použito velké množství technologií, proto se následující kapitola bude věnovat jednotlivým částem aplikace – jaké technologie přesně používá a jak jsou tyto technologie propojeny.

3.2.1 Engine

Základní část celé aplikace a její středobod se skládá ze tří pomyslných částí **Databáze**, **REST rozhraní** a **hlavní logika**. Díky tomuto rozdělení je možné snadno a hlavně rychle přidat novou funkcionalitu, dále je také možné rozdělit jednotlivé části vývojářům, kteří se mohou zaměřit na menší úkoly a pracovat tak efektivněji.

Hlavní logika byla naprogramována v jazyce Java a jako aplikační server poté použit Wildfly, díky jeho snadnému propojení do Keycloaku a také díky jeho robust-

[2]Tento nástroj je podobný Trellu, nicméně dovoluje snadnější a pro tým důležitou vizualizaci práce `https://www.jetbrains.com/youtrack/`.

nosti – největší předností (aplikačního serveru wildfly) je jednoduché napojení na velké množství různých databází. K načtení závislostí, sestavení a nasazení tohoto modulu byl použit nástroj Maven.

- **Databáze** – postavena nad technologií NoSql a přesněji použit databázový nástroj MongoDb. V pozdějším použití budeme pravděpodobně muset sáhnout po nějakém indexovacím nástroji jako je například Elasticsearch nebo Solr, který nám umožní snadnější hledání napříč složitější databázovou strukturou.
- **REST rozhraní** – slouží pro snadnější přístup k jednotlivým datům a definuje jednoduše dostupné koncové uzly. Pokud bude potřeba nástroj rozšířit a hlavně zmapovat jednotlivé koncové uzly budeme moci použít například technologii Swagger[3]

3.2.2 Uživatelské rozhraní

Hlavní část (Engine) funguje jako samostaný modul, který je nasazen do aplikačního serveru Wildfly, obdobně je tomu i v případě uživatelského rozhraní (samostatný modul, který je nasazena na Wildfly). Samotný proces sestavení používá nástroj **Webpack**, který celý zdrojový kód rozdělí do dvou Javascriptových souborů. Jeden obsahuje závislosti nutné pro chod aplikace a ve druhém je samostatný kód aplikace. V rámci usnadnění a jednoduššího přístupu k aplikaci jsme se rozhodli velkou část Javascriptových souborů umístit na veřejné CDN[4].

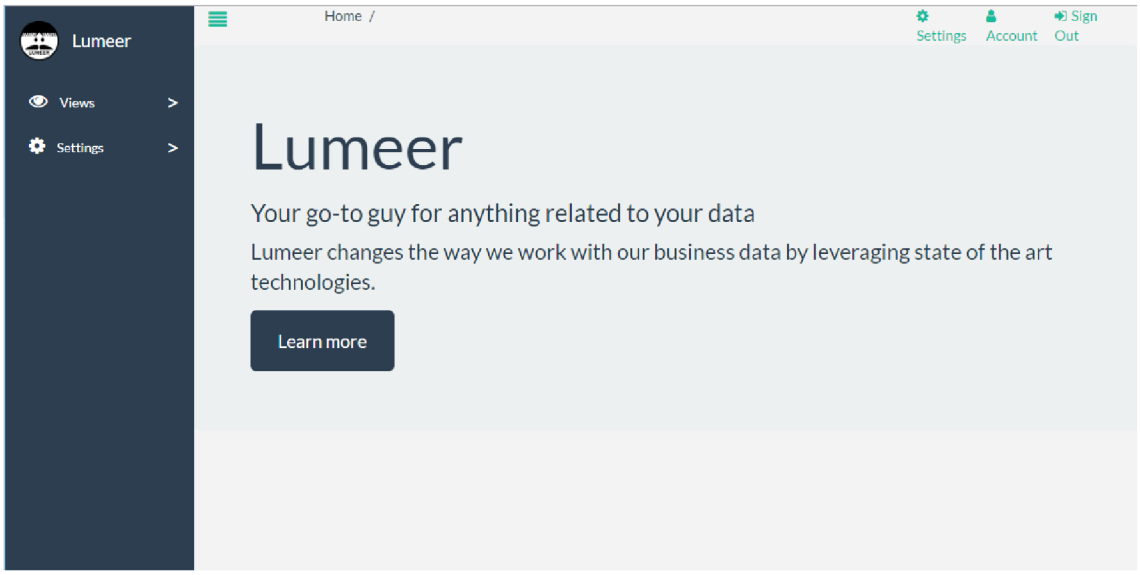
Vzhledem k rozsáhlosti a složitosti uživatelského rozhraní bylo rozhodnuto, že se použije aplikační framework s podporou jednostránkové aplikace. Po prozkoumání alternativ byl následně vybrán aplikační rámec **Angular 2**. S výběrem tohoto aplikačního rámce jde ruku v ruce také výběr jazyka ve kterém je uživatelské rozhraní napsáno, což je silně typovaný jazyk **Typescript**.

Uživatelské rozhraní se postupně upravovalo a byla snaha o co možná nejjednodušší používání této aplikace, proto byla první verze rozhraní představena několika méně zkušeným uživatelům počítače, kterým se dali jednoduché úkoly a byl sledován

[3]Slouží pro vizualizaci a zobrazení jaká data jednotlivý uzel očekává a jaké produkuje: `http://swagger.io/`.

[4]CDN je v podstatě systém serverů, které nabízejí JS a CSS soubory uživatelům s ohledem na jejich geografickou lokaci, tak aby měli tyto soubory co nejdříve.

jejich postup a následně jim byla položena otázka, zda byl úkol snadný. Systém jim byl pouze zevrubně vysvětlen slovy, že se jedná o aplikaci, která obsahuje kolekce souborů, které jsou mezi sebou propojeny. Jak vypadalo uživatelské rozhraní před těmito úkoly se můžete podívat na obrázku 3.3.



Obrázek 3.3: Design aplikace před představením uživatelům.

Úkol číslo 1

Výběr kolekce a následné zjištění informací o této kolekci. Úroveň náročnosti: **lehká**. Na výsledky hodnocení lze vidět v tabulce 3.1.

Úkol číslo 2

Výběr kolekce a dokumentu v kolekci pro který budou upraveny propojení. Náročnost: **středně těžká**. Výsledky hodnocení lze vidět v tabulce 3.2.

Úkol číslo 3

Výběr kolekce a dokumentu v kolekci, pro který se zobrazí práva. Úroveň náročnosti: **těžká**. Výsledky hodnocení jsou k vidění v tabulce 3.3.

Vzhledem k výsledkům hodnocení a uživatelským podnětům se tým rozhodl přepracovat UI do více uživatelsky přívětivého způsobu. Hlavní výtkou všech uživatelů bylo příliš mnoho navigačních prvků na stránce, což vede k nepřehlednosti a uživatelé tak nevědí, na co kliknout a co použít. Na druhou stranu většina uživa-

Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	25s	Snadný	Úkol se mi zdál ze začátku těžký, ale postupně jsem přišel na to kam kliknout.
2	38s	Středně těžký	Úkol se mi zádl celkem těžký a byl jsem docela zmatený s aplikací.
3	43s	Těžký	Vlbec jsem nechápal kam kliknout a potřeboval jsem pomoc.

Tabulka 3.1: Vyhodnocení úkolu číslo 1.

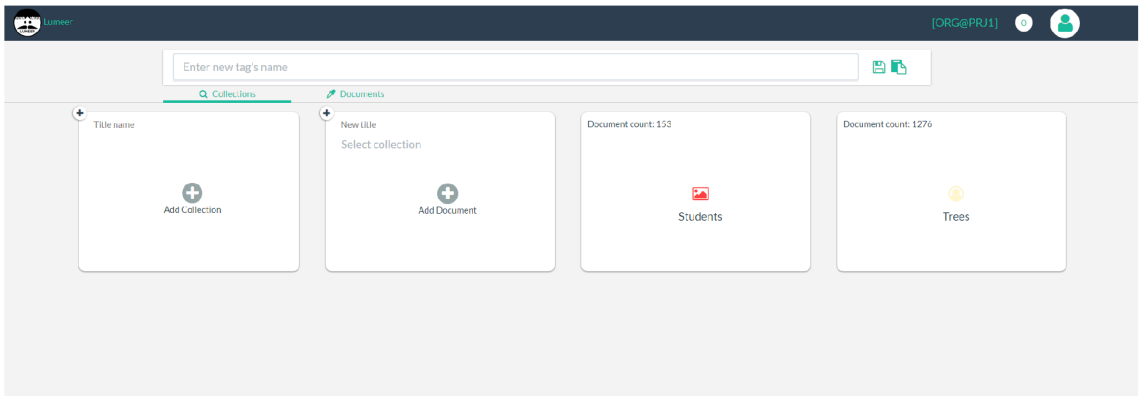
Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	40s	Středně těžký	I po předhozím pochopení aplikace jsem měl s aplikací celkem problém.
2	48s	Těžký	Vůbec jsem něvěděl kam kliknout a co dělat.
3	1m 15s	Těžký	Potřeboval jsem hned od začátku pomoci a potřeboval jsem úkol několikrát vysvětlit a pomoci co dělat.

Tabulka 3.2: Vyhodnocení úkolu číslo 2.

Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	30s	Středně těžký	Vzhledem k tomu že tento úkol navazoval na předešlý, hned jsem věděl kde přibližně hledat.
2	52s	Těžký	Úkol navazoval na druhý a proto jsem přibližně věděl co a kam kliknout.
3	1m 10s	Těžký	Úkol byl podobný jako druhý úkol a proto jsem nepotřeboval takovou pomoc.

Tabulka 3.3: Vyhodnocení úkolu číslo 3.

telů, kterým byla aplikace představena zdůraznili výhodu vyhledávání a filtrování v rámci kolekcí, dokumentů a propojení. Z těchto poznatků bylo tedy určeno, že nejlepší bude zvýraznit vyhledávací formulář, který bude na všech stránkách aplikace. Dále také bude třeba sjednotit všechny prvky aplikace do jednoho vyhledávání tak, aby uživatelé měli přehledně a snadno dostupné například dokumenty a hned vedle nich propojení. Z těchto požadavků bylo tedy navrhnuto nové uživatelské rozhraní, které lze vidět na obrázku 3.4.



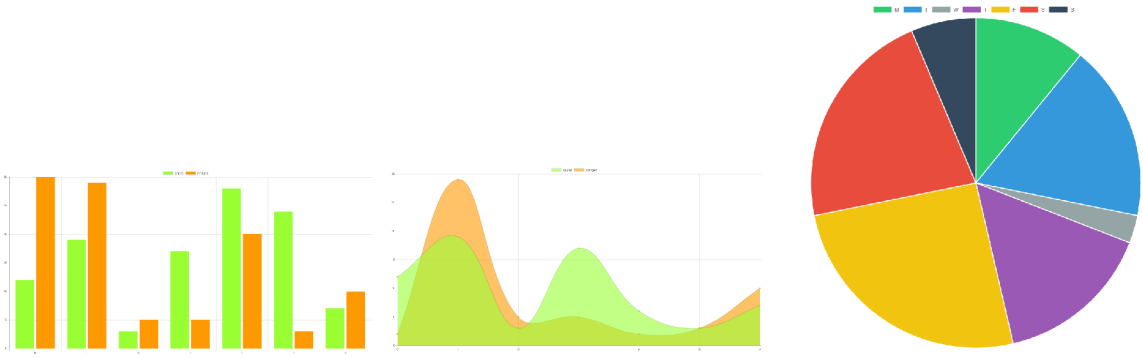
Obrázek 3.4: Návrh nového uživatelského rozhraní.

3.2.3 Práce s grafy

Díky tomu, že využíváme Webové technologie, můžeme jednoduše vytvořit grafy pro znázornění některých datových sad. Aplikace využívá více knihoven pro práci s daty, aby umožnila uživateli jednoduchou práci a možnost měnit typ grafů, upravovat datovou sadu, atd.

Jednoduché grafy

Pro znázornění a práci s jednoduchými grafy jsme využili knihovny **chart.js**. Tato knihovna se snadno ovládá a je jednoduchá pro nastavení; nicméně složitější grafické útvary nejsou jednoduché na vytvoření. Uživatel disponuje několika druhy grafů z této knihovny. Pokud chce používat komplexnější grafické znázornění musí si více připlatit a v aplikaci se zpřístupní možnost používat náročnější grafy, které jsou vytvořeny v knihovně **D3**. V základních operacích s grafy nabízí aplikace jednoduché sloupcové, spojnicové, koláčové a radarové grafy. Uživatel si může zvolit, z jakých datových sad bude čerpat data, jak je namapuje na grafy a případně možnost exportu grafu do pdf a obrázku.



(a) Sloupcový (b) Spojnicový (c) Koláčový

Obrázek 3.5: Použité jednoduché grafy.

Složitější grafy

Jak již bylo nastíněno v předešlé sekci, pro platící uživatele nabízí aplikace rozšířenou práci s grafy, převážně komplexnějších typů grafů, se kterými může uživatel pracovat. Také možnost exportovat grafy do formátu, který přečte aplikace Microsoft Excel. Ale hlavní výhodou oproti jednoduchým grafům má použití knihovny D3 – ta nabízí

velké možnosti vytvoření jak grafů, tak různých grafických prvků. Tato knihovna je používaná nadstavbou **C3**, která definuje základní grafy, které je jednoduché vytvořit, ale je značně rozšiřitelná. Také v rámci složitější práce s grafy nabízíme uživateli možnost importovat data z různých formátů (CSV, XLS, XLSX atd.) a poté zvolit dokument, do kterého se tato data vloží, případně pouze pracovat s těmito daty a vytvořit graf, který poté může uživatel uložit jako samostatnou entitu.

3.2.4 Vyhledávací jazyk

Vzhledem k tomu, že součástí požadavků na aplikaci bylo také možnost jednoduše filtrovat a hledat v jednotlivých entitách, museli jsme přijít s vyhledávacím jazykem, který pokryje jak složité vyhledávací dotazy, tak je pro uživatele jednoduchý pro pochopení.

Z pohledu uživatelského rozhraní

Pomocí uživatelského rozhraní může uživatel zadat vyhledávací text dvojím způsobem – **Fulltext**, nebo **tokeny**.

Pokud zvolí fulltextové vyhledávání, budou postupně uživateli napovídány jednotlivé texty napříč všemi entitami, tyto texty vytvoří nějakým způsobem větu, kterou odešleme na server a ten nám vrátí nejvhodnější výsledky. Součástí této funkcionality je také postupné překreslování vyfiltrovaných entit, které se objevují pod vyhledávacím polem.

V případě, že se uživatel rozhodne vyhledávat pomocí tokenů, usnadní serveru práci s odhadem co chce vyhledat a díky tomu dostane lépe sedící výsledky. Tokenizace probíhá tak, že si uživatel vybírá z předem definovaných prvků a z těch je postupně tvořen dotaz na server. Jak uživatel opět vytváří dotaz, jsou mu předávány prvky, které s velkou pravděpodobností hledá. Při použití tokenů uživatel získá také možnost hledat napříč několika entitami za použití logických operátorů, matematických operandů (větší, rovno, menší než) a textových (podobnost) operandů.

Pokud se tedy podíváme na tyto dvě metody, uživatel má možnost jednoduchého a rychlého hledání napříč aplikací pomocí fulltextového vyhledávání, nicméně toto vyhledávání není příliš vhodné pro složité dotazy. K tomu pak slouží tokenové vyhledávání, které je sice složitější, ale na druhou stranu přináší možnost hledat na-

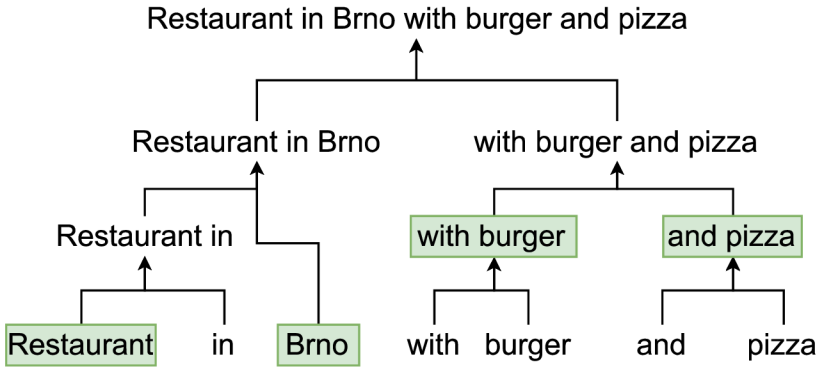
příč velkým množstvím entit. Jako bonus si uživatel může tokenizované vyhledávání uložit jako filtr se jménem a následně v tokenizovaném filtru použít toto zástupné jméno. Může tedy postupně upravovat vnořené filtry, které může dále rozšiřovat a upravovat.

Z pohledu serveru

Pokud se zaměříme na vyhledávání z pohledu serveru musíme opět rozdělit tyto vyhledávací funkce na dvě části, stejně jako je tomu v případě uživatelského rozhraní.

Fulltextové vyhledávání je složitější pro server převážně kvůli tomu, že jako dotaz bude mít vždy jeden dlouhý řetězec, který musí rozpadnout do jednotlivých prvků a nad těmi provést výsledné vyhledávání v rámci databáze. Server funguje tak, že si větu nejdříve rozdělí do jednotlivých slov a ty roztřídí do binárního stromu – jak lze například vidět na obrázku 3.6, kde je věta *Restaurant in Brno with burger and pizza*. Server zpracovává binární strom od spodu a vždy, když se mu povede vyhledat některý prvek, označí větev za vyřešenou a již ji dále nezpracovává. Takže při použití tohoto příkladu server zjistí, že uživatel hledá kolekci s názvem Restaurant, která obsahuje atribut Brno, dále server vyfiltruje všechny restaurace, které mají buger a které mají pizzu. Bohužel není vyhledávací algoritmus natolik přesný, aby vyhledal dokumenty ve kterých jsou tyto proměnné spojené.

Pokud uživatel použije tokenového, vyhledávání usnadní tak serveru následné hledání, protože nemusí složitě rozdělovat větu a skládat ji do strojově srozumitelného vyhledávání. Pokud například vezmeme opět větu *Restaurant in Brno with burger and pizza*, tak ta by v tokenové verzi vypadala tak, že by vyhledávací dotaz obsahoval čtyři tokeny každý s informací o jednotlivých požadavcích. Tokenizované vyhledávání si uživatel může uložit do takzvaných pohledů, které může následně vyhledávat. Kliknutím na pohled se dostane rovnou na vyhledané entity skrz uložený dotaz.



Obrázek 3.6: Binární strom pro roztřídění věty.

3.2.5 Zabezpečení

V mnoha moderních aplikacích dochází k problémům se zabezpečením, ať již je to nedostatečné zabezpečení vůči případným útokům, nebo nechtěné dovolení přístupu různým uživatelům k datům, ke kterým by neměli mít přístup. Vytvářený nástroj se na toto téma snaží co nejvíce zacílit a použít řešení, které nebude příliš komplexní k nastavení pro zákazníka a zároveň bude poskytovat silné zabezpečení. Proto byl zvolen nástroj od firmy RedHat **Keycloak**, jehož fungování je popsáno v sekci o autentizačních serverech 2.2.2.

V rámci tohoto nástroje je autentizační server nastaven tak, že před zapnutím aplikace je ověřeno přihlášení uživatele (díky nástroji Keycloak nemusí být uživatel přihlášen pouze k této aplikaci, ale k jakékoliv aplikaci spravované autentizačním serverem Keycloak, který má na starosti také tuto aplikaci). Pokud uživatel není přihlášen je vyzván k přihlášení, nástroj kromě klasického přihlášení pomocí emailu a hesla nabízí také možnost použít několik sociálních platforem – Github, Facebook, Google účet a Twitter. Poté se již každý dotaz na server kontroluje tímto autentiza-

čním serverem, takže je možné jednoduše nastavit pravidla uživatelům pro přístup jak k RESTovým službám, tak k jednotlivým zdrojům za pomoci Keycloak serveru a vytvořením skupin.

Díky autentizačnímu serveru se tedy v aplikaci nemusíme přímo starat o jednotlivé uživatele a nemusíme řešit zasílání hesel, celkovou správu a extra nastavení na serveru.

3.3 Dolování dat

Pokud vezmeme v úvahu hlavní modul starající se o dolování dat, můžeme ho rozdělit na tři části:

1. **Heuristiky pro automatické linkování** – pokud vznikne, nebo je upraven nějaký dokument je spuštěna řada automatických metod, které provedou případné automatické spojení.
2. **Nejčastěji používané entity** – na mnoha místech se aplikace snaží usnadnit práci uživateli tím, že mu nabídne ty, které jsou nejčastěji používané.
3. **Přibližné napovídání** – technika, která umožní napovídat uživateli možné výsledky při hledání bez nutnosti vyplnění celého jména.

3.3.1 Heuristiky pro automatické linkován

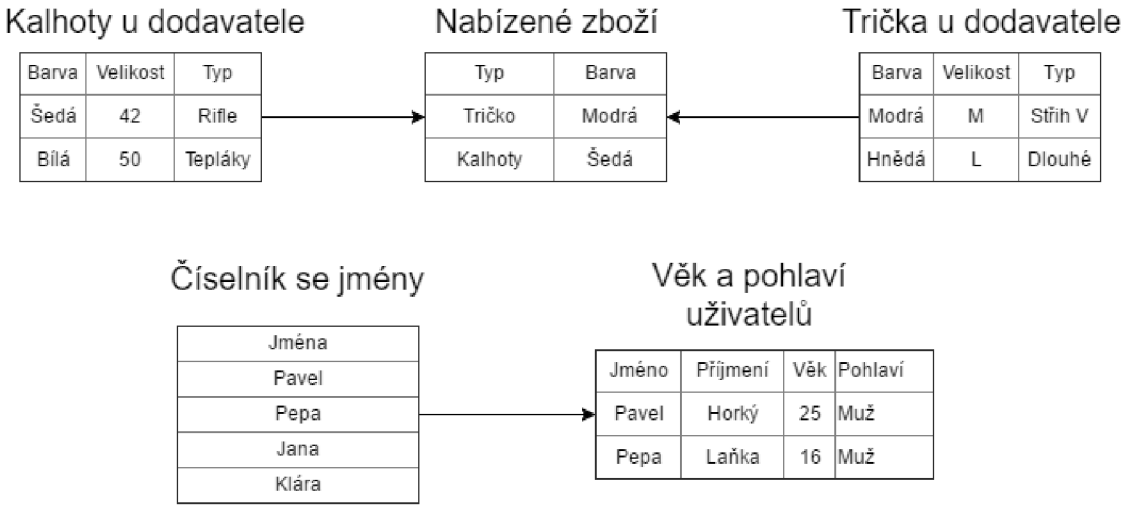
Systém disponuje několika heuristikami, které se snaží vyhledávat skrytý význam v jednotlivých dokumentech a napomáhat tak uživateli při propojování dokumentů. Vezměme si například dva dokumenty – jeden obsahuje jména, příjmení a telefonní čísla, nazvěme ho **uživatelský dokument**. Druhý bude například **studentský dokument**, ten bude obsahovat spojená jména a příjmení studentů spolu s jejich celkovým prospěchem. Po vytvoření studentského dokumentu nemusí mít uživatel tušení, že existuje uživatelský dokument a nevytvoří tak manuální propojení, nicméně systém si označí tyto dva dokumenty jako potencionální spojení a při zadávání dat do dokumentu studentů nám bude napovídat jména a příjmení z uživatelského dokumentu. Propojení těchto dokumentů tedy vznikne automaticky. Uživatel o tomto propojení nemusí být nijak informován, ale na stránce detailu studentského dokumentu uvidí informativní hlášku, která mu sdělí, že vzniklo nové propojení. Tento link může dále uživatel upravit, případně ho může odstranit a systém mu již nebude

napovídat jména a příjmení.

Toto automatické propojení vzniká na základě dvou heuristických technik a následného vyhodnocení pomocí složení dvou sloupců z jedné tabulky do druhé. Tímto docílíme jednoduchosti a hlavně rychlosti zadávání nových záznamů. Velkou výhodou této techniky je, že sloupce, které jsou označeny jako propojené a získávané z různých tabulek, se automaticky obnovují, takže v případě změny jména v souboru uživatelů se tato změna automaticky projeví také ve studentském dokumentu (opět lze tuto funkci vypnout pro celý dokument, případně pro jednotlivé záznamy).

Nad takto nově vzniklými propojeními lze provádět samozřejmě stejné operace jako v případě klasických propojení – tedy definovat propojovací řetězec a jednoduché funkce. Nicméně při vytvoření jednoduchých funkcí uživatel bohužel ztrácí automatické znovunačtení záznamů. Pokud chce načíst změny v takových záznamech, musí si vyžádat načtení hodnot, které jsou následně uloženy do souboru.

Příklady takových propojení můžeme vidět na obrázku 3.7, kde první tabulka s názvem Nabízené zboží získává data ze dvou tabulek a poté další tabulka o (Věk a pohlaví uživatelů) čerpá jména z dokumentu, který obsahuje všechna možná jména.



Obrázek 3.7: Příklad automaticky prolinkovaných dokumentů.

Heuristika 1

V případě první heuristiky se kontrolují pouze záhlaví tabulky, takže pokud vezmeme v úvahu definovaný příklad, v momentě vytvoření dokumentu se provede kontrola

nad podobnými soubory, a pokud se najde shoda, propojení se automaticky vytvoří.

Heuristika 2

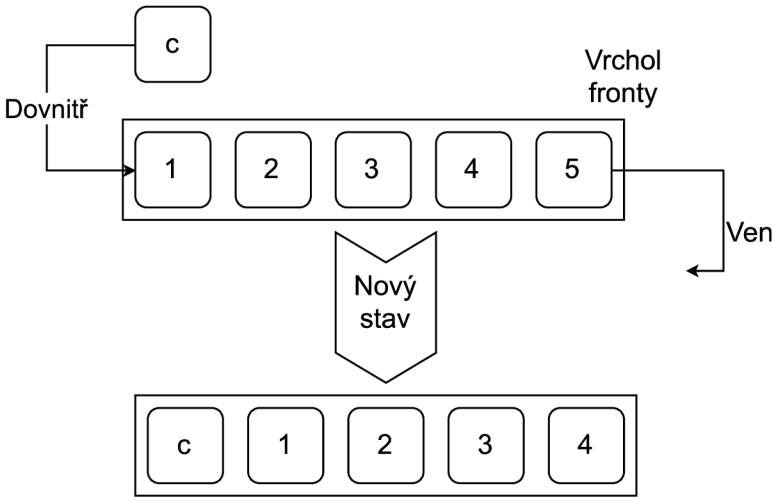
Tato heuristika je složitější a znamená velkou zátěž pro systém, proto je potřeba dopředu provést nastavení tabulky při vytváření. Funguje tak, že při vytvoření nové tabulky uživatel může definovat tuto tabulku jako zdrojové data a kdykoliv se bude vytvářet nový soubor, provede se kontrola nad zdrojovými tabulkami, zda některá neobsahuje příslušný záznam. Vzhledem k náročnosti na systém jsme se rozhodli, že tuto vlastnost odstupňujeme pro jednotlivé druhy zákazníků.

3.3.2 Nejčastěji používané entity

S přihlédnutím k uživatelským požadavkům obsahuje systém možnost zobrazit entity, které jsou nejčastěji používané daným uživatelem pro jednu organizaci a projekt. Pro zobrazení požadovaných entit jsme nejdříve vybrali algoritmus **fronta**, ale tento model se nám neosvědčil, protože se často stávalo, že nejpoužívanější entity mizely z rychlého výberu a méně časté se nacházely v nabídce, proto jsme do systému naimplementovali možnost přepnutí na takzvanou **frontu s počítadlem** – možnost pouze pro zákazníky s kvalitnější podporou. Přepnutí těchto technik se poté nachází v uživatelském nastavení, takže každý uživatel je schopen si toto řazení změnit.

Fronta

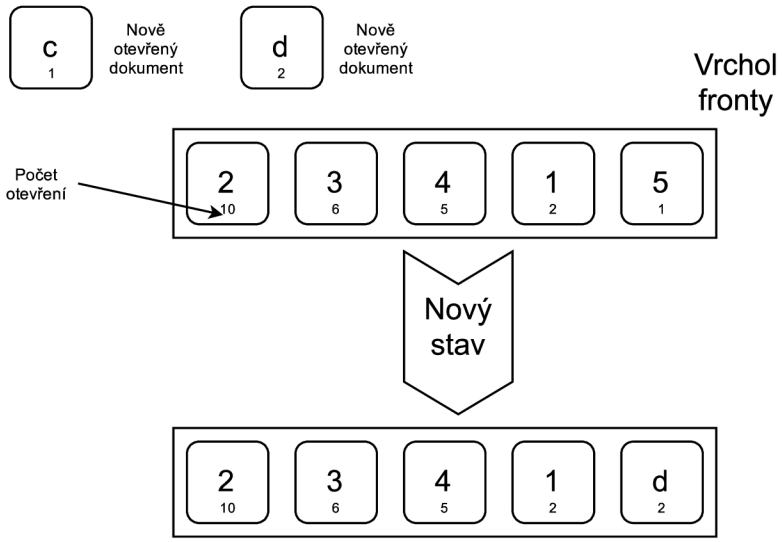
Fronta funguje na stylu omezeného počtu záznamů, které mohou být zobrazeny, a pokud je tento počet překročen, poslední záznam, který byl zaktivován, je odstraněn. Toto řešení je dostačující, nicméně může nastat to, že entita, která je často otevíraná, zmizí z tohoto seznamu kvůli otevření několika stejných entit. Chování v aplikaci lze vidět na obrázku 3.8, kdy fronta obsahuje entity označené 1, 2, 3, 4 a 5. Uživatel otevřel nově entitu s označením c a entity v zásobníku jsou tedy (od spodu fronty) c, 1, 2, 3 a 4.



Obrázek 3.8: Příklad plnění jednoduché fronty.

Fronta s počítadlem

Fronta s počítadlem je v podstatě rozšířená implementace fronty. Pokud je některá entita otevíraná častěji, je u ní zvyšován počet otevření. Entita s největším počtem otevření je na spodu fronty a entita s nejmenším počtem navrchu. Pokud se otevře nový dokument, je mu zvýšen počet otevření, a pokud překročí toto číslo nejvrchnější prvek ve frontě, tato nově otevřená entita nahradí tu, která byla na vrcholu fronty. Provedeno v rámci aplikace můžeme vidět na obrázku 3.9 – nejčastěji používaná entita je označena 2 a nejméně používaná je označena 5; dále uživatel otevřel nově dokument s označením c a poté otevřel dokument d, který byl předtím otevřen jednou. Nově tedy zásobník obsahuje entity 2, 3, 4, 1 a d.



Obrázek 3.9: Příklad plnění rozšířené fronty s počítadlem.

3.3.3 Přibližné napovídání

Přibližné napovídání pracuje nad takzvanou fuzzy logikou a funguje na principu, že zadaný řetězec nemusí být vždy přesně zadán, aby uživateli byl dodán výsledek, který požaduje. Přibližné napovídání v aplikaci přibližné napovídání funguje tak, že v momentě, kdy uživatel zadá hledaný řetězec, jsou mu zobrazeny záznamy, které se nejblíže shodují k danému řetězci (případně obsahují prvních pár písmen), ideálně jsou tyto řetězce shodné. Aplikace nabízí možnost zobrazit uživateli entity, které mají v názvu nebo v možných řetězcích jedno písmeno jiné. Pokud uživatel následně vybere entitu, je k ní tento hledaný řetězec přidán. Entita tedy obsahuje seznam možných hledaných řetězců, které se dále mohou lišit o jedno písmeno. Například pokud budeme hledat entitu s názvem **Restaurace**, tato entita dále obsahuje seznam možných hledaných řetězců – například *Rez*, *taue*, *auea* a *Res*. Uživatel tak zadá do hledání **Restaues** a systém mu nabídne entitu **Restaurace**.

3.4 Analýza rizik

Vzhledem k tomu, že vytvořit náročnou aplikaci není triviální, na začátku vývoje jsme si vytvořili analýzu možných rizik, na která můžeme narazit během vývoje a při distribuci. Také jsme si tato rizika ohodnotili a pro některé, které by měli vysoký dopad na aplikaci, jsme si připravili riziková opatření.

Následující tabulky znázorňují pravděpodobnost výskytu a dopad rizika.

Pravděpodobnost	Hodnota
0 - 20 %	1
20 - 40 %	2
40 - 60 %	3
60 - 80 %	4
80 - 100 %	5

Dopad	Hodnota
Velmi nízký	1
Nízký	2
Střední	3
Vysoký	4
Velmi vysoký	5

Tabulka 3.4: Ohodnocení pravděpodobnosti a dopadu rizika.

Po vytvoření základních tabulek pro odhadnutí rizik jsme jednotlivá rizika zapsali do tabulky a ohodnotili jsme je možnou pravděpodobností a dopadem na funkčnost, nasazení nebo nepřijetí uživateli. jako mezní hodnotu jsme si určili 14 bodů při ohodnocení pravděpodobnosti a dopadu rizika – výsledek je možné vidět v tabulce 3.6.

Č.	Scénář	Hrozba	Pravděp.	Dopad	Hodnocení[5]
1	Pomalý vývoj	S příchodem zákazníků budou přibývat požadavky na systém, které nemusíme stíhat přidávat	2	3	6
2	Uživatelé nebudou rozumět aplikaci	Uživatelské rozhraní nebude intuitivní a snadno ovladatelné	3	3	9

Č.	Scénář	Hrozba	Pravděp.	Dopad	Hodnocení[5]
3	Nezabezpečená data	Kdokoli se dostane k uživatelským datům	3	5	15
4	Napadení aplikace	Napadení aplikace třetí stranou	2	5	10
5	Pomalá práce s daty	Při velkém objemu dat může nastat zamrzání aplikace	4	3	12
6	Nezájem investorů	Aplikace nezujme případné investory, což povede k nedostatku zdrojů v prvotním spuštění	3	5	15

Tabulka 3.5: Možné scénáře rizik při vývoji a nasazení.

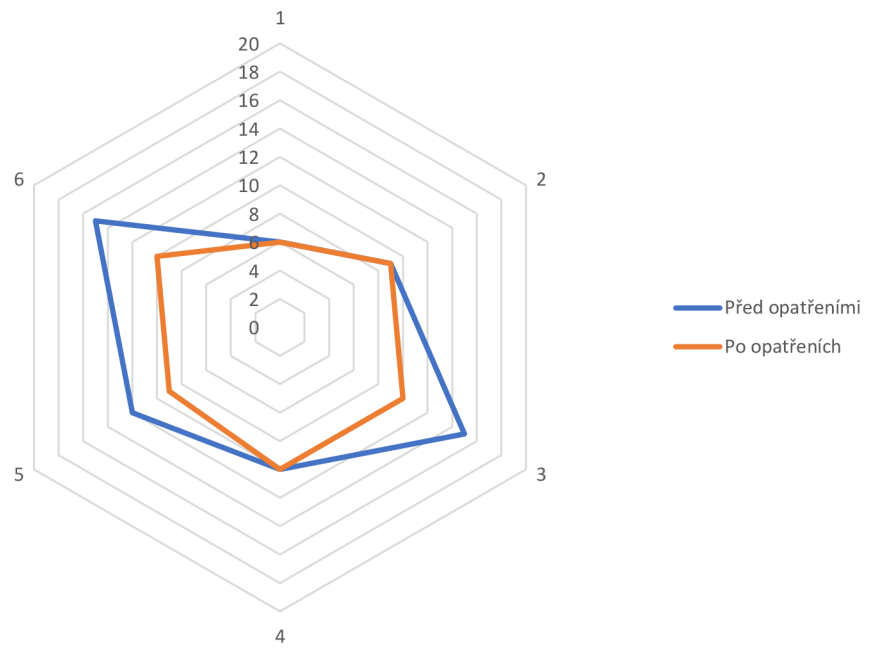
Jak lze vidět v tabulce 3.6, tak pouze 2 rizika si vyžadují speciální pozornost, protože překračují námi definovanou hranici. Zaměříme se také na riziko s číslem **5**, které by mělo také kritický dopad na chod aplikace.

Č.	Opatření	Cena	Pravděp._	Dopad_	Hodnocení_[6]
3	Při vývoji použijeme nástroj, který zabezpečí data	Cena nástroje pro zabezpečení	2	5	10
5	Použití indexovacího nástroje	Cena nástroje pro indexaci dat	3	3	9
6	Dedikování člověka pro komunikaci s investory	Cena člověka pro komunikaci	2	5	10

Tabulka 3.6: Opatření pro vyhnutí se rizikům.

Z možných opatření vyplývá že všem potencionálním technickým rizikům, ktera by měla vysoký dopad, se dá vyhnout použitím vhodných technologií a v případě rizika **6**, které by znamenalo možný nezájem investorů, musíme dedikovat člověka pro komunikaci a vytváření povědomí o tomto nástroji.

Pro znázornění zavedení opatření pro vyhnutí rizik jsme zvolili terčový graf 3.1, na kterém lze vidět, že žádné riziko nepřekračuje hodnotu 14 bodů, kteréou jsme si nastavili jako kritický bod. Nevěnovali jsme se rizikům, která nemají tak vysokou pravděpodobnost, že by nastala, nicméně i na ta je třeba dávat si pozor, a proto je dobré, že jsme na některá z nich narazili již na začátku.



Graf 3.1: Rizika před a po zavedení rizikových opatření.

3.5 Cenový model

Jak již bylo několikrát řečeno v předchozím textu, aplikace se bude nabízet v několika cenových relacích, kdy více platící zákazníci dostanou na výběr větší množství funkcí a také větší podporu od vývojového týmu.

Cenový model můžeme shrnout do tří skupin – **základní**, **střední** a **nejvyšší** (označováno též jako stříbrný, zlatý a platinový zákazník). Zákazníci si sami vyberou, jaké funkcionality se jim více hodí, přičemž první 2 měsíce mají zdarma na vyzkoušení. Takže zákazník dostane na začátku plnou podporu všech funkcí systému bez nutnosti platit jakoukoliv částku. V případě, že by chtěl školení, případně lepší podporu v těchto dvou měsících, je s ním vytvořena speciální smlouva, která může pokrývat tyto dva zkušební měsíce. Pokud nechce na začátku zákazník nic platit, nemusí, dostane návod jak aplikaci používat a menší ukázku toho, co s aplikací dělat a má možnost dva měsíce bezplatně testovat tento produkt.

Po uplynutí dvouměsíční lhůty si zákazník sjedná schůzku, kde se domluví bližší detaily přechodu do systému. Na začátku zákazník platí za služby jako například převod celé databáze, školení uživatelů, podporu při problémech a výpadcích atd. Každý úkon je ohodnocen individuálně na základě velikosti databáze (případně počtu uživatelů a jejich požadavků) a době strávené na přechodu pod systém.

Pokud zákazník souhlasí se všemi podmínkami a je ochotný začít používat aplikaci, je s ním také vytvořen speciální platící plán, ve kterém záleží na počtu uživatelů, kteří budou mít přístup k systému a k funkcionalitám, které jednotliví uživatelé požadují. Tento platící plán je opět individuální a nejvíce se mění v závislosti na počtu a druhu aktivovaných funkcí a na skupině zákazníka.

Vzhledem k tomu, že vývoj aplikace probíhá formou open-source, nemáme možnost, aby si schopnější uživatelé nevzali vyvíjenou aplikaci a nespustili si ji sami na svém vlastním stroji, tomuto se nijak nebráníme a je jasné, že takové případy budou vznikat, proto musíme sázet na to, že zákazníci budou chtít platit za podporu a za přidané hodnoty, na které mají nárok v rámci cenových modelů.

ZÁVĚR

Cílem této práce bylo vytvořit platformu, která bude schopná pracovat s daty, vyhodnocovat jejich vnitřní informace a následně nabízet uživateli jednoduché rozhraní pro práci s těmito daty. Tohoto cíle bylo dosaženo – platformu je možné používat, pracovat s ní a nahrávat do ní data, nicméně je možné ji dále vylepšovat (rozšiřovat ji a přidávat do ní další funkční moduly).

Nejdříve bylo potřeba seznámit se s několika teoretickými pojmy, které byly důležité pro pochopení celého problému vývoje a návrhu rozsáhlé platformy. Vzhledem k tomu, že taková platforma bude muset být nasazena u několika zákazníků zároveň a že práce musí být plynulá, nemělo by záležet kde a jak ji uživatelé používají. Proto bylo zvoleno použití webových technologií a k nim přidružených nástrojů, které umožňují snazší přístup pro větší množství uživatelů.

V rámci návrhu a vývoje platformy došlo několikrát k testování u různě schopných uživatelů, což mělo za následek částečnou změnu uživatelského rozhraní do podoby, která je jednoduše pochopitelná a snadná na používání.

Hlavní výhodou tohoto nástroje je jeho snaha o pochopení vnitřních informací v datech, které uživatel poskytne. Nad těmito daty se poté provádí několik funkcí, které nám pomohou s následným zpracováním tak, že uživateli jsou automaticky napovídány hodnoty, které jsou již v systému zapsány. Tímto se značně usnadňuje uživateli práce jak při zadávání nových hodnot, tak práce s již existujícími datovými sadami.

Během psaní této práce byl nástroj již představen několika případným investorům, kteří o něj vyslovili zájem a chtěli by do něj přenést svá data. Někteří investoři sami chválí myšlenku nástroje, který kombinuje webové technologie a práci s tabulkami. Nejvíce si však pochvalují jednoduchost a snadné používání vytvářené aplikace.

Seznam obrázků

1.1 Jednotlivé vrstev v platformě openshift 14
1.2 Příklad dělení dat do clusterů 17
1.3 Fáze CRISP-DM, referenční model 21
1.4 Cyklus návrhu grafických prvků 26
1.5 Příklad zakódovaného a odkódovaného JWT. 29

2.1 Příklad tématu pro aplikaci napsanou technologií Material 33
2.2 Příklad tématu pro aplikaci napsanou technologií Bootstrap 34
2.3 Příklad tématu pro aplikaci napsanou technologií Patternfly 34
2.4 Předání uživatelského požadavku při použití SAML2 36
2.5 Znázornění získání JWT z nástroje Keycloak 37
2.6 Příklad rozhodovacího stromu při použití algoritmu ID3 40
2.7 Příklad hodnocení stránky pomocí Markovova modelu náhodné stránky 42
2.8 Data získána z A/B testování v nástroji KissMetrics 47

3.1 Diagram znázorňující schéma nástroje. 50
3.2 Diagram znázorňující schéma nástroje. 51
3.3 Design aplikace před představením uživatelům. 54
3.4 Návrh nového uživatelského rozhraní. 56
3.5 Použité jednoduché grafy. 57
3.6 Binární strom pro roztřídění věty. 60
3.7 Příklad automaticky prolinkovaných dokumentů. 62
3.8 Příklad plnění jednoduché fronty. 64
3.9 Příklad plnění rozšířené fronty s počítadlem. 65

Seznam tabulek

3.1 Vyhodnocení úkolu číslo 1. 55
3.2 Vyhodnocení úkolu číslo 2. 55
3.3 Vyhodnocení úkolu číslo 3. 56
3.4 Ohodnocení pravděpodobnosti a dopadu rizika. 66
3.5 Možné scénáře rizik při vývoji a nasazení. 67
3.6 Opatření pro vyhnutí se rizikům. 67

Seznam rovnic

2.1 Vzorec pro výpočet entropie v rozhodovacích stromech 39
2.2 Vzorec pro výpočet ohodnocení stránky 41

Seznam grafů

1.1 Podíl vybraných prohlížečů mezi uživateli 25

3.1 Rizika před a po zavedení rizikových opatření. 68

Seznam zkratek

BI Business Intelligence.

CI/CD Continuous integration/Continuous delivery = Kontinuální integrace/Kontinuální nasazení.

CRM Customer relationship management = Management vztahu se zákazníky.

DaaS Data as a Service = Data jako služba.

ETL Extraction, Transformation, Loading = Vytažení, očištění, načtení.

IaaS Infrastrcure as a Service = Infrastruktura jako služba.

JWT JSON Web Token = JSON webový token.

OLAP Online Analytical Processing = Online analytické zpracování.

OLTP Online transaction processing = Online zpracování transakcí.

Paas Platform as a Service = Platforma jako služba.

REST Representational State Transfer = Reprezentativní přenos stavu.

SAML Security Assertion Markup Language = Bezpečnostní značkovací jazyk zabezpečení.

SCM Software configuration management = Správa konfigurace softwaru.

SPA Single page application = Aplikace jedné stránky.

SSO Single sing only = Pouze jedno přihlášení.

UI User interface = Uživatelské rozhraní.

Literatura

[1] MARVIN, Rob. Building an App With No Coding: Myth or Reality. *Pcmag*[online]. Ziff Davis, LLC. PCMag Digital Group, 2016 [cit. 2017-03-11]. Dostupné z: `http://www.pcmag.com/article/345661/building-an-app-with-no-coding-myth-or-reality`

[2] RUBENS, Paul. Use Low-Code Platforms to Develop the Apps Customers Want. *CIO* [online]. IDG Communications, 2014 [cit. 2017-03-11]. Dostupné z: `http://www.cio.com/article/2845378/development-tools/use-low-code-platforms-to-develop-the-apps-customers-want.html`

[3] MARVIN, Rob. How low-code development seeks to accelerate software delivery. *SD Times* [online]. BZ Media LLC., 2014 [cit. 2017-03-11]. Dostupné z: `http://sdtimes.com/low-code-development-seeks-accelerate-software-delivery/`

[4] Zoho Creator REVIEW. *Finances Online* [online]. [cit. 2017-03-11]. Dostupné z: `https://reviews.financesonline.com/p/zoho-creator/`

[5] CIOT, Thierry. What is a Low-Code Platform? *Progress* [online]. 2016 [cit. 2017-03-11]. Dostupné z: `https://www.progress.com/blogs/what-is-a-low-code-platform`

[6] OpenShift Origin Overview. *OpenShift Origin* [online]. [cit. 2017-03-11]. Dostupné z: `https://docs.openshift.org/latest/architecture/index.html`

[7] HURWITZ, Judith. *Cloud computing for dummies.* Hoboken, NJ: Wiley Pub., c2010. ISBN 978-0470484708.

[8] ROME, C. H. *The cloud computing Book: The ultimate guide to mastering cloud computing.* Fifth edition. Bernemouth: Imagine Publishing, 2015.

[9] CHANDRASEKARAN, K. *Essentials of Cloud Computing.* Maiami: CRC Press, 2014. ISBN 978-1482205435.

[10] Co je PaaS?: Platforma jako služba. *Microsoft Azure* [online]. [cit. 2017-03-12]. Dostupné z: `https://azure.microsoft.com/cs-cz/overview/what-is-paas/`

[11] FOSTER PROVOST & TOM FAWCETT. *Data science for business what you need to know about data mining and data-analytic thinking.* Sebastopol, CA: O'Reilly Media, 2013. ISBN 9781449374280.

[12] SIEGEL, Eric. *Predictive analytics: the power to predict who will click, buy, lie, or die.* Revised and Updated Edition. ISBN 9781119145684.

[13] PRASAD, Y. Lakshmi. *Big data analytics made easy.* 1. McNichols Road, Chetpet Chennai: Notion Press, 2016. ISBN 978-1-946390-72-1.

[14] MARZ, Nathan a James WARREN. *Big data: principles and best practices of scalable real-time data systems.* ISBN 978-1617290343.

[15] NUGENT, Alan, Fern HALPER a JUDITH HURWITZ AND MARCIA KAUFMAN. *Big data for dummies.* Hoboken, N.J: John Wiley, 2013. ISBN 9781118644171.

[16] BRAMER, Max. *Principles of data mining.* 3rd edition. 2016. ISBN 978-1-4471-7306-9.

[17] HAN, Jiawei, Micheline KAMBER a Jian PEI *Data mining: concepts and techniques.* 3rd ed. Haryana, India ; Burlington, MA: Elsevier, 2012. ISBN 9789380931913.

[18] W. CHU, Wesley, LIN, T. Y., ed. *Foundations and Advances in Data Mining.* Springer-Verlag. ISBN 9783540808190.

[19] SADALAGE, Pramod J. a Martin FOWLER *NoSQL distilled: a brief guide to the emerging world of polyglot persistence.* Upper Saddle River, NJ: Addison-Wesley, c2013. ISBN 978-0321826626.

[20] WITTEN, I. H., Eibe. FRANK a Mark A. HALL. *Data mining: practical machine learning tools and techniques.* 3rd ed. Burlington, MA: Morgan Kaufmann, c2011. Morgan Kaufmann series in data management systems. ISBN 978-0-12-374856-0.

[21] EVELSON, Boris a Noel YUHANNA. *The Forrester WaveTM: Advanced Data Visualization (ADV) Platforms.* Corn park Drive, Cambridge, 2017. Forrester Research, Inc.

[22] RADY, Ben. *Serverless Single Page Apps: Fast, Scalable, and Available.* 1. United States of America: Pragmatic Bookshelf, 2016. ISBN 978-1-68050-149-0.

[23] MIKOWSKI, Michael S. a Josh C. POWELL. *Single page web applications: JavaScript end-to-end*. 1. United States of America: Manning Publications, 2013. ISBN 9781617290756.

[24] HARTSON, H. Rex. a Pardha S. PYLA. *The UX Book: process and guidelines for ensuring a quality user experience*. Boston: Elsevier, c2012. ISBN 978-0123852410.

[25] *Desktop Browser Version* [online]. [cit. 2017-05-23]. Dostupné z: `http://gs.statcounter.com/browser-version-partially-combined-market-share/desktop/worldwide/#monthly-201612-201703-bar`

[26] MURRAY, Scott. *nteractive data visualization for the web*. Sebastopol, CA: O'Reilly Media, 2013. ISBN 978-1-449-33973-9.

[27] ALLAMARAJU, Subrahmanyam. *RESTful Web services cookbook*. Sebastopol,CA.: O'Reilly, c2010. ISBN 978-0596801687.

[28] OOMMEN JOSEPH, Allen, Jaspher W. KATHRINE a Rohit VIJAYAN. Cloud Security Mechanisms for Data Protection: A Survey. In: *International Journal of Multimedia and Ubiquitous Engineering*. Karunya University Coimbatore, India: School of Computer Science and Technology, 2014, s. 81-90. ISSN 1975-0080.

[29] SEVILLEJA, Chris. Scotch: Developers bringing fire to the people. *The Anatomy of a JSON Web Token*. [online]. [cit. 2017-05-03]. Dostupné z: https://scotch.io/tutorials/the-anatomy-of-a-json-web-token

[30] RFC 7519: JSON Web Token (JWT) *IETF Tools*. [online]. [cit. 2017-05-11]. Dostupné z: `https://tools.ietf.org/html/rfc7519`

[31] ES5, ES6, ES2016, ES.Next: What's going on with JavaScript versioning? *Benmccormick.org*. [online]. [cit. 2017-05-11]. Dostupné z: `https://benmccormick.org/2015/09/14/es5-es6-es2016-es-next-whats-going-on-with-javascript-versioning/`

[32] *ECMAScript6 Compatibility table*. [online]. [cit. 2017-05-11]. Dostupné z: `http://kangax.github.io/compat-table/es6/`

[33] *Top JavaScript Frameworks & Topics to Learn in 2017*. [online]. 2016 [cit. 2017-05-11]. Dostupné z: `https://medium.com/javascript-scene/`

top-javascript-frameworks-topics-to-learn-in-2017-700a397b711#.fng4s5tee

[34] Framework vs Library: which one is better? *Void Canvas*. [online]. 2017 [cit. 2017-05-11]. Dostupné z: http://voidcanvas.com/framework-vs-library-one-better/

[35] *Material design: Introduction*. [online]. [cit. 2017-05-11]. Dostupné z: https://material.io/guidelines/#introduction-goals

[36] *What is Bootstrap and How Do I Use It?*. [online]. [cit. 2017-05-11]. Dostupné z: https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/

[37] *Patternfly: Frequently Asked Questions*. [online]. [cit. 2017-05-11]. Dostupné z: https://www.patternfly.org/get-started/frequently-asked-questions/

[38] *OpenID: Welcome to OpenID Connect*. [online]. [cit. 2017-05-11]. Dostupné z: http://openid.net/connect/

[39] *The Basics of SAML*. [online]. [cit. 2017-05-11]. Dostupné z: http://www.gigya.com/blog/the-basics-of-saml/

[40] *Keycloak: about* . [online]. [cit. 2017-05-11]. Dostupné z: http://www.keycloak.org/about.html

[41] *IBM: About Sterling External Authentication Server*. [online]. [cit. 2017-05-11]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SS4T7T_2.4.1/com.ibm.help.seasimplementationguide.doc/SEAS_About_SEAS.html

[42] *Active Directory Federation Services*. [online]. [cit. 2017-05-11]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb897402.aspx

[43] *What is Power BI?* . [online]. [cit. 2017-05-11]. Dostupné z: https://powerbi.microsoft.com/en-us/what-is-power-bi/

[44] *BM Watson Analytics: Analytics made easy* . [online]. [cit. 2017-05-11]. Dostupné z: https://www.ibm.com/analytics/watson-analytics/us-en/

[45] *What is MicroStrategy?* . [online]. [cit. 2017-05-11]. Dostupné z: http://searchbusinessanalytics.techtarget.com/definition/MicroStrategy

[46] *Business Objects or HANA? A guide to SAP analytics environments.* [online]. [cit. 2017-05-11]. Dostupné z: http://searchsap.techtarget.com/essentialguide/Business-Objects-or-HANA-A-guide-to-SAP-analytics-environments

[47] *What is customer relationship management (CRM).* [online]. [cit. 2017-05-11]. Dostupné z: http://searchcrm.techtarget.com/definition/CRM

[48] *Kissmetrics Academy: LEARN HOW KISSMETRICS CAN OPTIMIZE YOUR MARKETING.* [online]. [cit. 2017-05-11]. Dostupné z: https://blog.kissmetrics.com/google-analytics-and-kissmetrics/