

## **Abstrakt**

text

## **Summary**

text

## **Klíčová slova**

text

## **Key words**

text



**Bibliografická citace**

text

## 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 xx. x. 2017

.....

jméno

**Poděkování**

text

# Obsah

<b>ÚVOD</b>	<b>8</b>
<b>CÍL DIPLOMOVÉ PRÁCE</b>	<b>9</b>
<b>METODIKA PRÁCE</b>	<b>10</b>
Metody . . . . .	10
Postupy . . . . .	10
<b>1 TEORETICKÁ VÝCHODISKA</b>	<b>11</b>
1.1 Vývojové platformy low-code . . . . .	11
1.1.1 Příklady low-code platforem . . . . .	12
1.1.2 Cloudový výpočet . . . . .	12
1.2 Bussiness Intelligence . . . . .	15
1.2.1 Big data . . . . .	15
1.2.2 Data-mining . . . . .	17
1.3 Platforma pro pokročilou vizualizaci dat . . . . .	19
1.3.1 Single page aplikace . . . . .	20
1.3.2 Dynamická a interaktivní vizualizace dat . . . . .	22
1.3.3 Webová služba RESTful . . . . .	23
1.4 Server pro řízení přístupu a identity . . . . .	25
1.4.1 JSON Web Token . . . . .	25
<b>2 ANALYTICKÁ ČÁST</b>	<b>27</b>
2.1 Webové technologie . . . . .	27
2.1.1 Webový aplikační rámec . . . . .	27
2.1.2 Silné vs. slabé typování . . . . .	29
2.1.3 Responzivní aplikační rámec . . . . .	29
2.2 Single sign-on . . . . .	30
2.2.1 Standardy získání identity . . . . .	30
2.2.2 Autentikační servery . . . . .	31
2.3 Datová analýza . . . . .	32

2.3.1	Shluková analýza . . . . .	32
2.3.2	Rozhodovací stromy . . . . .	33
2.3.3	Dolování dat na webové aplikaci . . . . .	34
2.3.4	Ukládání a práce s velkým objemem dat . . . . .	36
2.3.5	Aplikace využívající datovou analýzu . . . . .	37
2.4	Databázové aplikační platformy . . . . .	38
2.5	Management vztahu se zákazníky (CRM) . . . . .	39
2.5.1	Monitorování webové aplikace . . . . .	40
2.6	Shrnutí . . . . .	41
<b>3</b>	<b>VLASTNÍ NÁVRHY</b>	<b>42</b>
3.1	Návrh aplikace . . . . .	42
3.1.1	Vedení vývojového týmu . . . . .	43
3.1.2	Popis aplikační funkčnosti . . . . .	43
3.2	Použité technologie . . . . .	44
3.2.1	Engine . . . . .	44
3.2.2	Uživatelské rozhraní . . . . .	45
3.2.3	Práce s grafy . . . . .	47
3.2.4	Vyhledávací jazyk . . . . .	48
3.2.5	Zabezpečení . . . . .	49
3.3	Dolování dat . . . . .	50
3.3.1	Heuristiky pro automatické linkován . . . . .	50
3.3.2	Nejčastěji používané entity . . . . .	52
3.3.3	Přibližné napovídání . . . . .	53
3.4	Analýza rizik . . . . .	53
3.5	Cenový model . . . . .	56
	<b>ZÁVĚR</b>	<b>58</b>
	<b>Seznam obrázků</b>	<b>59</b>
	<b>Seznam tabulek</b>	<b>60</b>
	<b>Seznam použitých zdrojů</b>	<b>61</b>

# ÚVOD

V aktuální době je velké množství dokumentů ve firmách spracovááno nástroji, které nepodporují do jisté míry spolupráci a často 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 provázání. Cílem této práce je poté analýza současných nástrojů a následně návrh a vytvoření specifického nástroje, který se bude zaměřovat ve velké části na jeho snadné používání uživateli.

Práce nás postupně provede několika kapitolami, kdy se nejdříve zaměříme na teoretická východiska, kde si vysvětlíme některé pojmy zabývající se práci s dokumenty pomocí webových aplikací, poté se zaměříme na aktuální stav nástrojů a aplikací zabývající se touto problematikou. Část popisu aktuálního stavu se budeme věnovat nástrojům, které nám mohou pomoci při samotném vývoji aplikace. Kdy na samotný návrh a vývoj aplikace se zaměříme v poslední kapitole, ve které si rozebereme také případná rizika vývoje takového nástroje.



## 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. Tyto celky poté zobrazit uživateli, nechat jej dále definovat dodatečné informace, provádět reporting 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 dodáváný 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 k získání logických informací ze zadaných informací. Platforma bude vyvíjena s možností škálovatelnosti a použití nad velkým objemem dat.

## Postupy

K vytvoření co nejprívětivější platformy budou využity zkušenosti a knižní publikace zabývající se tímto tématem. Dále budou analyzovány jednotlivé postupy zadávání dat uživatelů do takového 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 případném vypracování platformy je potřeba si objasnit a vysvětlit několik témat. Jsou to především *Vývojové platformy low-code*, výsledná platforma by měla splňovat tuto definici. Dále si objasníme pojmy *Business intelligence* (platforma bude z části pracovat s touto oblastí) a *Platforma pro pokročilou vizualizaci dat* – pro snadné používání uživatelského rozhraní. A vzhledem k tomu že výsledná platforma musí do určité části 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, tyto produkty začali vznikat, protože malé a střední podniky potřebovali 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 a toto všechno za minimálního použití ručního programování. Takovéto platformy neeliminují programování jako takové, ale napomáhají rychlejšímu vývoji, tak že poskytují vizuální nástroje a napomáhají konfiguraci datových modulů a pomáhají eliminovat problémy spojené s datovou integrací. [2]

### Výhody low-code platform

- **Produktivita:** Systémy mohou být vyvíjeny a nasazeny během menšího časového rozmezí, oproti klasickému programování. [3]
- **Reakční schopnost:** Vývojář může často zvolit různé druhy platform na kterých bude výsledný produkt fungvat, 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 na to co má aplikace dělat, a ne jak to má dělat, programátoři se mohou zaměřit na funkcionalitu a uživatelskou spokojenost. Při vývoji je možné se zaměřit také 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 pro mobilní platformy a také jako webové služby. Při spojení 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 funkční napříč všemi mobilními zařízeními. [5]

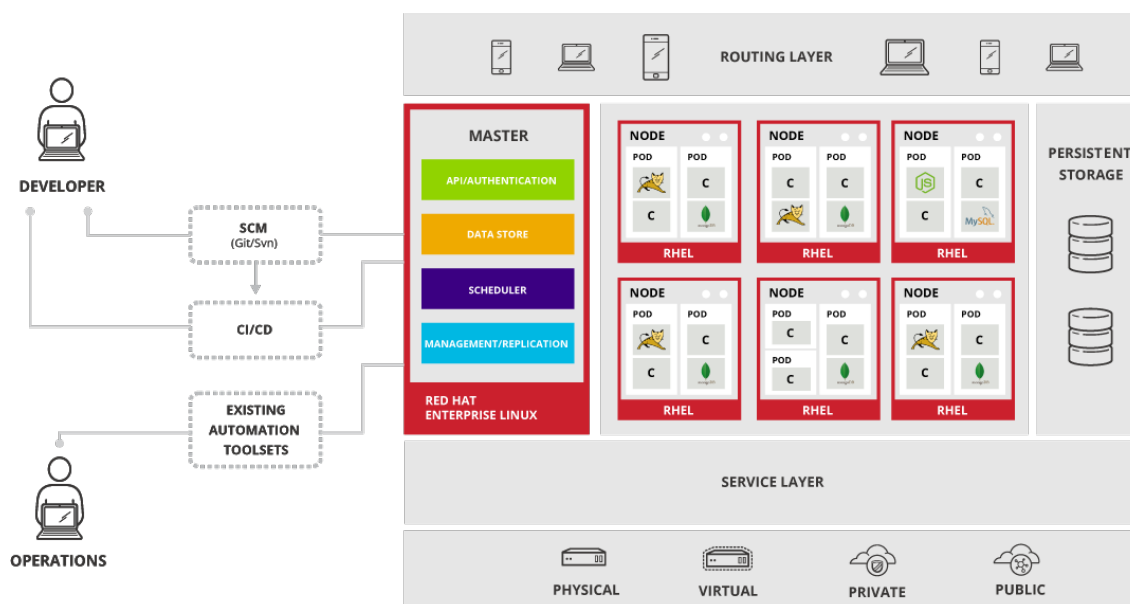
**Openshift** Platforma pro vývoj webovým a mobilních aplikací, postavená na kontejnerech, které zajišťují rychlý vývoj a možnost dedikovat vývojáře na vytvoření jednoduchých funkcionalit jako samostatné aplikace <sup>1</sup>, které za pomoci Openshiftu vytvoří velkou a komplexní aplikaci. Na Obrázku ?? můžeme vidět z jakých vrstev se skládá Openshift a můžeme vidět jednotlivé aplikace zázorněné v nodech, dále můžeme vidět jaké nástroje nabízí Openshift vývojářům (management zdrojových kódů – SCM a kontonualní integrace – CI/CD).[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 služba kekoliv na světě se člověk nachází. I když

---

<sup>1</sup>Takovýmto aplikacím se říká Microservice <https://smartbear.com/learn/api-design/what-are-microservices/>



Obrázek 1.1: Znoznění jednotlivých vrstev v platformě openshift.

poskytovatelé cloudových řešení mají často velice robustní bezpečnostní systém je na uživateli, který má uložené data v cloudu, aby zajistil jejich bezpečnost. To znamená že pokud data uniknou z cloudu díky špatnému zabezpečení v aplikaci, která je hostovaná, není chyba poskytovatele, ale firmy, která takovou aplikaci vydala.[7]

Pporovnání ceny cloudového výpočtu a klasického datového uložení není až tak jednoduché, záleží na několika faktorech. Vezměme si například lokaci datového uložení, pokud například cena elektické energie v místě datového uložení je velice levná firma nemusí být tolik tlačena do cloudového řešení. ale pokud 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 nám cloudové řešení vyplatí a kdy ne. Důležité je také uvědomit si, že 42 % nákladů na datové uložení jde do hardware a software (tyto náklady jsou rozloženy v průběhu času) a 58 % nákladů jde do topení, klimatizace, daní a samostatné práce. [7]

## Typy cloudového výpočtu

- **Veřejné** dostupné pro širokou veřejnost, jak zdarma, tak placené verze.
- **Soukromé** často používané firmami skupinami uživatelů, kteří potřebují zabezpečit data. Často velice drahé a ačasově nákladné řešení.

- **Komunitní** podobné soukromým, ale rozšířené mezi větší skupiny lidí.
- **Hybridní** vytvořené z jednoho a více druhů, privátního a nebo veřejného cloudu. Mezi jejich portfoliočastopatří zálohakekritickýmslužbám.
- **DaaS** data jako služba, pouze data uložené v cloudu.
- **PaaS** pro vývoj a hostování celého vývojového cyklu, často včetně možnosti nasazení výsledné aplikace.
- **IaaS** 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 často velice drahé nástroje. Někteří poskytovatelé nabízí také vyrovnání zatížení, to znamená že pokud je výsledná aplikace pod vysokým náporům uživatelů, automaticky se přiřadí prostředky, aby uživatelé nezaznamenali pád aplikace a bez nutnosti zasáhnout do nastavení služby.[9]

### Výhody PaaS

- **Méně kódu** – Díky možnosti projení 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 platform** – 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 Bussiness Intelligence

Nástroje pro bussiness Intelligence zahrnují jak samostatná data, tak časovou jednotku, takže můžeme nad těmito daty provádět predikci pomocí sofistikovaných nástrojů a výpočtů. Ze začátku bylo jednoduché provádět takové výpočty, protože jednoduše firma nesbírala takové množství dat. Aktuálně však není v lidských silách provádět takové výpočty nad tak obrovským množstvím dat. [11]

Pravděpodobně nejvíce rozšířeným aplikováním data-miningu 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 nejvíce **Homogenní** (stejně). Pokud sledované subjekty spadají do více skupin (toto se může hodit z mnoha důvodů) můžeme použít takzvané **Clustery**. Což jsou skupiny objektů, které si jsou co nejvíce podobné, ale objekty v jiné skupině se jim budou co nejvíce lišit. [11]

Samostatná data jako taková však nejsou to nejdůležitější, pokud nebudeme schopni vhodně interpretovat a využít data, která jsme nashromáždili jsou nám k ničemu, 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 těchct 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 a částečně strukturovaných dat, které mají potenciál k tomu aby z nich bylo

možné vydolovat nějaké zkruté 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, Velocity, Variety. Pro přesné definování můžeme vzít v úvahu také Veracity, Validity a Volatility.[13]

- **Volume** – Ve světě Big data máme na mysli opravdu velké množství dat.
- **Velocity** – Rychlost s jakou jsou data zpracována.
- **Variety** – Různorodostí dat máme na mysli jejich formát (*strukturovaná* – klasické RDBMS, *částečně strukturovaná* – emaily, zprávy ...; *nestrukturovaná* – multimediální obsah).
- **Veracity** – Věrohodostí rozumíme že data musí být očištěna od zbytečného šumu.
- **Validity** – Data musí být co nejpresnější a co možná nejvhodnější pro naše rozhodování.
- **Volatility** – Data musí být co nejaktuálnější pro přesnější predikci a jejich pozdější zpracování. [13]

Nejvhodnější místo pro uložení takového množství dat, je cloud, přesněji využít službu některých poskytovatelů privátních nebo veřejných IaaS. 1.1.2 Cloud je vhodný pro Big data převážně kvůli tomu, že jak uložení, tak práce s takovým objemem dat, požaduje velké množství distribuované počítačové síly. [14]

**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). [14]

**Pružnost** je potřeba zachovat co největší, převážně kvůli tomu že mohou nastat momenty, kdy máme velké množství dat, které se ale v čase mohou smršknout pouze na zlomek těchto dat. Pokud se tomu tak stane nechceme nadále platit za zbytečně nevyužitý prostor. [14]



**Sdružování prostředků** cloud dovoluje vytvářet skupiny sdílených prostředků. [14]

**Samooobsluha** většina poskytovatelů clouhových řešení nabízí možnost samostatně, bez nutnosti kontaktovat IT oddělení, navýšit zdroje, případně je odpojit (pokud již nejsou potřebné). Toto je často řešeno nějakým portálem, případně specifickými nástroji u uživatele daného cloudu. [14]

**Nízká pořizovací cena** znamená že často cloudové řešení nestojí uživatele tolik jako pořízení drahých datových skladů. [14]

**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. [14]

**Tolerance pádů** u cloudových řešení je nutnost a musí být u takovýchto řešení co nejnižší, aby byl zajištěn nepřetržitý chod. [14]

### 1.2.2 Data-mining

Pokud budeme potřebovat velkémnožství dat (Big data 1.2.1), musíme je nejdříve očistit od chybně zapsaných dat. Dokonce i když máme data ve standardní formě nemůžeme počítat s tím, že jsou tyto data naprosto bez chyb. Takto chybně zapsaná data mohou být nicméně důležitá pro náš systém.[15]

Dále některé atributy mohou naprosto chybět pro některé záznamy, pro vypořádání s takto chybějícími atributy můžeme zvolit jednu ze dvou taktik. [15]

1. Odstranění – jednoduše odstraníme celý záznam ve kterém se nám nachází nějaký chybějící atribut.
2. Náhrada – použijeme nejvíce frekventovanou, nebo nějakou výchozí hodnotu na místě ve kterém najdeme chybějící atributy. [15]

Na obrázku ?? vidíme jednotlivé fáze popsané v CRISP-DM <sup>2</sup> modelu, tyto fáze jsou:

**Pochopení podniku** na začátek je potřeba pochopit, co za problém se snažíme vyřešit pomocí dolování dat. Tento první krok je nejdůležitější, ze začátku budeme

---

<sup>2</sup>Zkratka znamená cross-industry process for data mining

mít pouze slabé pochopení tohoto kroku a v rámci procesu dolování dat se budeme vracet do tohoto bodu. [16]

**Pochopení dat** 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 některá prostě neexistují. Proto v rámci pochopení dat musíme zvážit případné rozšíření datové složky a zda se nám vyplatí investovat do dalších dat. [16]

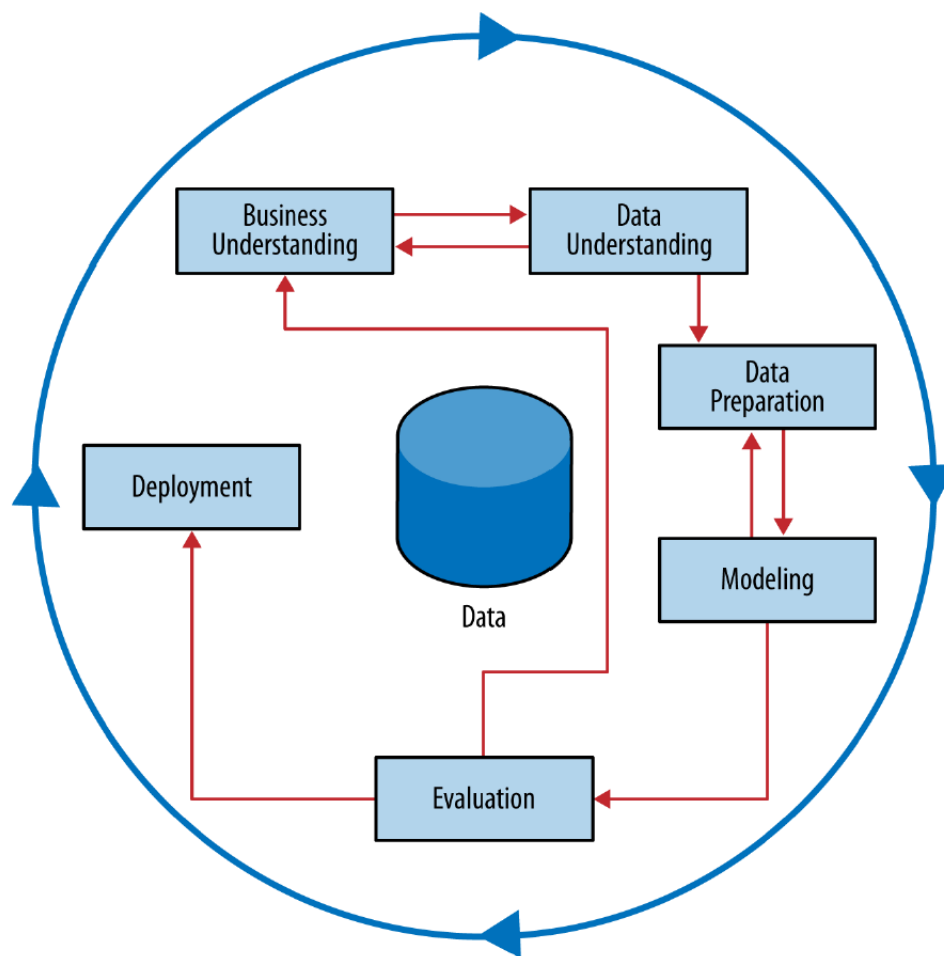
**Příprava dat** je dost možné, že tato část bude předcházet **pochopení dat**, protože jednoduše pro pochopení dat musíme nejdříve tyto data očistit od případných chybných a chybějících atributů. [16]

**Modelování** primární fáze ve které jsou použity techniky pro dolování dat. [16]

**Vyhodnocení** 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 celé kolečko znovu. Pro vyhodnocování používáme jak kvantitativní tak kvalitativní kritéria. [16]

**Nasazení** 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. [16]

**Extraction, Transformation, Loading** Extraction, Transformation, Loading ve zkratce **ETL** jsou procesy zodpovědné za naplnění dat do datových skladů. Nejdříve jsou data vyextrahována z různých uložišť. Těmito uložišti mohou být OLTP, starší systémy, webové stránky atd. Poté jsou data očištěna, zbavena duplikátů a chybějících atributů. Následně jsou data nahrána do datových skladů, kde se s nimi může dále pracovat.



Obrázek 1.2: Fáze CRISP-DM, referenční model.

### 1.3 Platforma pro pokročilou vizualizaci dat

Podniky často zjišťují že vizualizace dat je pro ně důležitým prvkem v odhalování problémů a monitorování podniku. Často se stává že klasické zobrazení dat v podobě reportů za použití tabulek nedokáže zaměřit celý problém, nebo často vede ke špatné analýze. Z toho plyne že je lepší použití grafických prvků jako (jednoduše řečeno grafů), pro ještě lepší usnadnění analýzy je vhodné použít dynamických a interaktivních grafických prvků. 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 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 stát že různí analytici mají přístup k různým datovým sadám, proto je potřeba zajistit pověřovací úroveň.
- **Varování pro podnik** – Pokud se stane že na obrazovce se nachází příliš mnoho dat může se často stát že analytici jednoduše nebudou 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. [17]

### 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. Celá těžká práce je založena 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 navigaci, to 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 <sup>3</sup>. [18]

### Výhody Single page aplikací

- **Vykreslování** – Klasické webové aplikace potřebují při většině uživatelských akcí překreslit celou stránku. To vede k pauze, kdy uživatel čeká na zpracování požadavku na serveru a poté 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. Pouze část stránky se vždy překreslí a často není potřeba k takové akci činnosti serveru.
- **Responzibilita** – Takovéto aplikace minimalizují reakční čas tím, že větší část logiky je přenesena k uživateli. Pouze zpracování dat, validaci a autentizaci má na starosti server. Jak jsou tato data zobrazena a případná filtrace je zpracována u uživatele pomocí SPA. Například pokud uživatel filtruje data pomocí výběru sloupce v grafu není potřeba provést získání dat, tím se nezatěžuje server a stránka zůstává nadále responzivní.
- **Notifikace** – Pokud takto napsaná aplikace musí čekat na server může uživateli dát nějakým grafickým prvkem najevo takovou skutečnost. Případně je možné uživatele upozornit na zprávy ze serveru za pomoci **Webových notifikací**<sup>4</sup>.
- **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.
- **Aktualizace** – V případě aplikací, které jsou uloženy u uživatele je často potřeba vydat aktualizací balíček a uživatel si ho musí nainstalovat. V případě

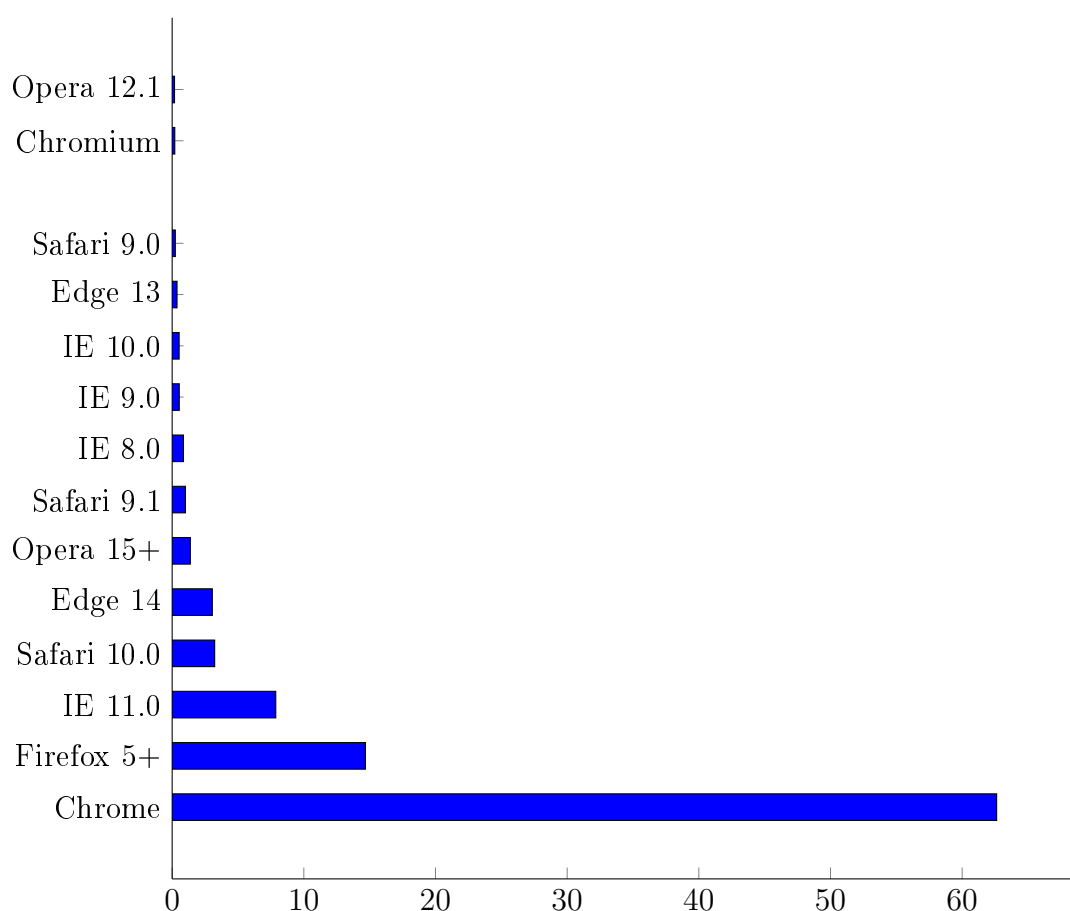
---

<sup>3</sup>Takovémuto přístupu se říká Service Worker a více se můžete dozvědět na <https://developers.google.com/web/fundamentals/getting-started/codelabs/offline/>

<sup>4</sup>Pro detailní popis můžete následovat [https://developer.mozilla.org/en-US/docs/Web/API/Notifications\\_API/Using\\_the\\_Notifications\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API/Using_the_Notifications_API).

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.

- **Multiplatformní** – Webové aplikace jsou dostupné takřka na všech platformách, od mobilních zařízení až po velkoplošné obrazovky. Limitací je snad pouze používání zastaralých prohlížečů, naštěstí uživatelé začínají používat modernější prohlížeče a chápou nutnost aktualizací. Jak lze vidět na grafu 1.3, tak modernější prohlížeč chrome je o hodně používanější než zastaralý prohlížeč Internet explorer ve verzi 9 a méně. [19]



Obrázek 1.3: Podíl vybraných prohlížečů mezi uživateli. Vypracováno na základě dat TODO

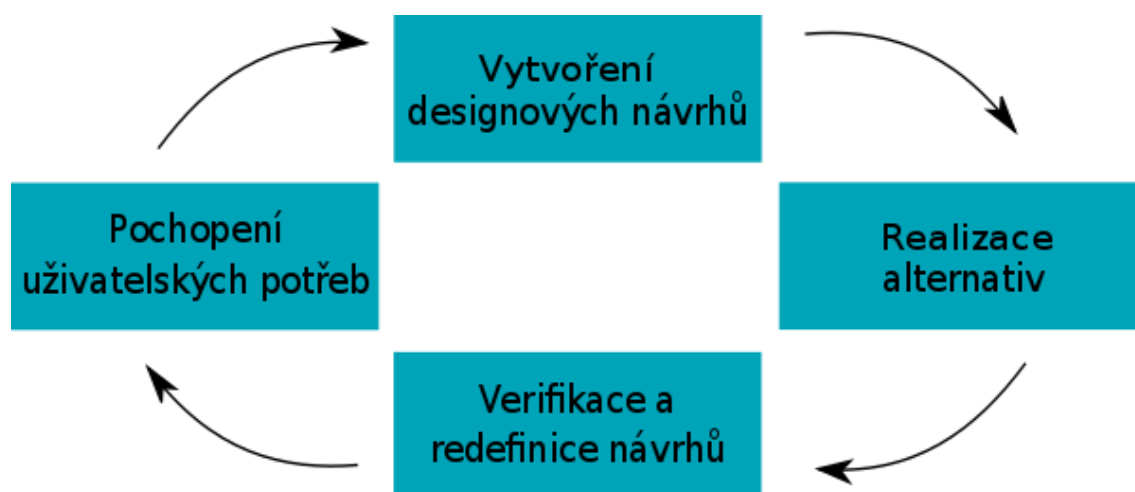
### 1.3.2 Dynamická a interaktivní vizualizace dat

Problém moderní doby je přehršel 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 zinteraktivnění takovýchto grafů. [20]

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 uživatel samotný se snaží vybrat a zpřehlednit data k obrazu svému. interaktivní vizualizace se změnila pouze nepatrně od roku 1996, kdy Ben Shneiderman z Univerzity Marilandu pronesl „Nejdříve náhled, poté zoom a filtrování, na závěr detail na vyžádání.” [20]

Při vytváření interaktivní vizualizaci dat se musíme zaměřit na 4 fáze návrhu a vývoje. **Vytvoření designových návrhů** – kdy navrhujeme co a jak bude vypadat v našem systému. **Realizace alternativ** – zapracujeme návrhy a vytvoříme první prototypy. **Verifikace a redefinice návrhů** – kdy předložíme uživateli námi vytvořené řešení a zapíšeme si jeho reakce a chování. **Pochopení uživatelských potřeb** – ze zápisků z předešlého kroku a z požadavků uživatele vyvedeme závěry a opět se vracíme do prvního kroku vytvoření návrhů. Celý koloběh si můžeme prohlédnout na obrázku 1.4. [21]



Obrázek 1.4: Cyklus návrhu grafických prvků.

### 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 bezstavová komunikace, kdy si server již nedrží stav o uživatelských akcích, ale odpovídá na všechny dotazy jako na nové sezení. Jedním kdo využívá tohoto bezstavového přístupu k uživatelům je architektní styl RESTful (zkráceně můžeme používat pouze REST). [22]

## **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. [22]

**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 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. [22]

**OAuth 1.0** Je zkratka Otevřené autentikace (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. [22]

**OAuth 2.0** Toto je vylepšený protokol a je momentálně nejvíce rozšířený. Ve zkratce to znamená to, že není nutné pro každé zařízení generovat specifický token, který je možno v případě napadení zařízení zrušit. Tento protokol využívá prostřed-



níka pro uchování klientských údajů. Je to v podstatě to, že služba požádá jiný server, který spravuje daného uživatele o přihlašovací token. Uživatel je přesměrován na adresu třetí serveru, kam zadá své údaje, server poté vyhodnotí správnost těchto údajů a službě která si vyžádala přihlašovací token odešle buď povolí nebo zakáže přihlášení. Výhodou je to, že pokud je uživatel 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. [22]

## 1.4 Server pro řízení přístupu a identity

Mnoho firem stále váhá s přechodem z vlastního řešení (jak databázových jednotek, tak aplikací) do cloudu. Hlavní problém je často zabezpečení, firmy jednoduše nechtějí věřit jiné firmě s důvěrnými informacemi, že je ochrání před případným napadením. Naštěstí mnoho poskytovatelů cloudů si toto uvědomuje a tak nemalou část financí investují do robustního autentikačního formátu. [23]

### Mechanismy používané pro zabezpečení dat v cloudu

- **RSA** – Pro přihlášení je potřeba HW token, který je synchronizován se servery cloudu.
- **Vícefaktorové** – 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. Výhoda je ta, že není nutnost vždy zadávat přihlašovací údaje, při přístupu k aplikaci je vyžádán od druhé služby takzvaný JSON web token. [23]

#### 1.4.1 JSON Web Token

Zkráceně JWT jsou 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). [24]

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

**Hlavička** obsahuje typ a hashovací algoritmus použitý pro dekodování podpisu.

**Náklad** zde se nachází data, která služba poskytuje. Často jsou zde přihlašovací údaje (uživatelské jméno a čas vypršení), dále může obsahovat jakákoliv data chceme, například pohlaví uživatele, věk, email, atd.

**Podpis** je text, který vznikne spojením hlavičky a nákladu, který je poté zhashován pomocí algoritmu, který je určen v hlavičce tokenu. [25]

## 2 ANALYTICKÁ ČÁST

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

Nejenom vývojovými technologiemi je potřeba se zabývat, ale také aktuálním stavem trhu a toho co uživatelé nejčasteji používají. Proto se v této části zaměříme také průzkumem trhu.

### 2.1 Webové technologie

Aktuální trend v používání webových prohlížečů můžeme vidět na grafu 1.3, kde jde jasně vidět že moderní prohlížeče v podání Chrome a Firefox převládají nad tolika vývojáři proklínaný a čím dál tím méně používaný Internet Explorer. dále si uživatelé již uvědomují, že aktualizace prohlížeče je pro zachování zabezpečení nutností a tak naštěstí webové prohlížeče pomalu začínají pořádně fungovat s novým standardem JavaScriptu nazvaný EcmaSript 6 (znáý též pod názvem ES2015 a ES6). [26]

Nový standard ES6 přináší mnoho vylepšení a mnoho optimalizací. Nicméně ani většina moderních prohlížečů nepodporuje 100 % tento standard, 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%). [27]

#### 2.1.1 Webový aplikační rámec

V aktuální době se mnoha vývojářům webových aplikací ověřili takzvané webové aplikační rámce. Dříve hojně využívaná knihovna **jQuery** má již mnoho nástupců jak v podobě knihoven, tak aplikačních rámců. Výhoda takových aplikačních rámců je že se stará v podstatě o veškerou těžkou a neustále se opakující práci a nechává programátorovi volnou ruku při realizaci samostatné aplikace. [28]

V předešlém odstavci bylo použito obou pojmů, jak JavaScriptové knihovny, tak aplikačního webového rámce. Tyto pojmy dost často vedou k hádkám a nedorozumění, 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. Každý jeden nástroj by představoval knihovnu.
- **Aplikační webový rámec** má na starosti veškerou práci a často zahrnuje přesně definovanou architekturu. Pokud si vezmeme příklad s kávou, aplikační webový rámec by byl kávovar, do kterého nalijeme vodu a nasypeme kávová zrna. [29]

**Angular.js** je plnohodnotný aplikační rámec, v aktuální době pravděpodobně nej-používanější.<sup>1</sup>

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 zpracování s aplikací. První datum vydání bylo v roce 2010 a v roce 2016 byla vydána verze 2, která umožňuje vytvářet uživatelské rozhraní jak pro webové, tak pro mobilní aplikace.[30]

**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 zesložitění a častým problémům s nastavením. [31]

**Vue.js** je progresivní rámec, pro vytváření uživatelských rozhraní. Jeho tvůrci kladli důraz na jeho jednoduchost použití a inspirovali se v **React.js**, výhodou oproti zmíněnému má 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 kompromisem mezi **Angular.js** – který může být příliš náročný pro pochopení, a **React.js** – který může vést k

---

<sup>1</sup>Přesná čísla se určit nedají, nicméně více informací o popularitě se můžete dočíst zde: <https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282#.ufk9gznd>

zesložitění při použití s více rozšířeními. [32]

### 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, jejich zástupci 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 nutné 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. [33]

**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. [34]

### 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.)

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

**Bootstrap** 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 různé rozlišení pro různé uživatele, tomuto se chtěl bootstrap vyhnout. Nyní nabízí velké množství doplňků a rozšíření, díky kterým z něj dělají rávem nejrozšířenější responzivní rámec. [36]

**Patternfly** se inspiroval bootstrapem a vytvořil vlastní responzivní rámec, dále nabízí několik widgetů pro zobrazování složitějších uživatelských dat. Jeho hlavním zaměřením je unifikovat jednotlivé aplikace ve firmě, tak aby měly stejný, ale unikátní design. Proto nabízí několik jednoduchých přístupů co a jak dělat pro dosažení co nejvíce podobného vzhledu napříč aplikacemi. [37]

## 2.2 Single sign-on

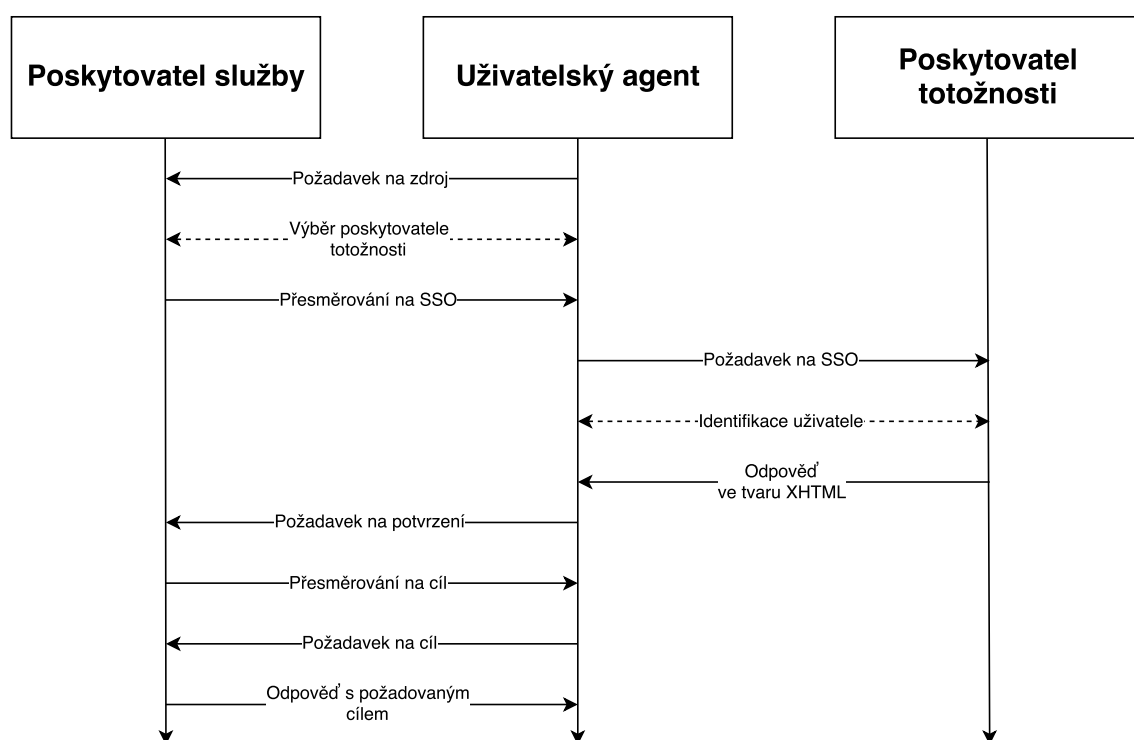
V aktuální době velká část společností nabízejících větší portfolio aplikací volí takzvanou službu single sign-on, kdy uživatel nemusí pokaždé zadávat své přihlašovací údaje, ale o jeho identifikaci se postará specializovaná služba. Většinou se stačí do této služby přihlásit jednou a dokud nevyprší čas 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 identitní vrstva postavená nad OAuth 2.0, poskytuje možnost identifikace uživatele a také získání základních informací. 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 2.1. 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 požadavake, o který si znovu požádá a server mu jej vrátí. [39]



Obrázek 2.1: Předání uživatelského požadavku při použití SAML2

### 2.2.2 Autentikační servery

Aktuální trh nabízí velké množství autentikačních 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.

**Keycloak** je open source řešení, které nabízí připojení jak 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]

**Sterling externí autentikační server** je 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** od firmy Microsoft nabízí možnost sdílet identifikační informace napříč důvěryhodným obchodním partnerům skrze extranet. Výhodou je že se nemusí aplikace 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. Pokud 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 zjednodušené proces organizace objektů do skupin, tak aby její členové byli nějakým způsobem co nejvíce podobní. Shlukovou analýzu můžeme rozdělit do dvou skupin **nehierarchické** a **hierarchické**.

**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 jemuž středu je nejbližší a



při každém běhu algoritmu se středy shluků přepočítávají jako aritmetické průměry všech bodů.

**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 nejlépe použít v případě že máme menší vzorek dat a chceme co nejlepší shluk.

Pro shrnutí je potřeba si uvědomit dvě věci, jaký je nejmenší počet shluků a naopak jaký je největší počet těchto shluků. Na první otázku je jednoduché odpovědět, je to v podstatě celý seznam datových bodů v jednom shluku (není nám to příliš užitečné, ale je to shluk dat). Na otázku největšího počtu shluků můžeme říct, že jím je počet všech datových bodů (toto opět není příliš užitečné). 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.

### 2.3.2 Rozhodovací stromy

Tento algoritmus 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 spojující vnitřní uzly, které mají opět několik hran projujících 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. Při výběru vlastnosti, která je nejvíce odlišná od ostatních příkladů v ostatních třídách se používá takzvané entropie, její výpočet lze vidět na 2.1, kde  $p_t$  je pravděpodobnost výskytu třídy  $t$  a  $t$  je počet tříd

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

Pro vytvoření stromu se vezme tabulka hodnot a pro každý atribut se vypočítá jeho entropie, ta znázorňuje homogenitu prvku. Pokud je prvek naprosto homo-

genní (nácházející se ve všech attributech) 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).

Příklad algoritmu, který se používá pro vytváření rozhodovacích stromů je takzvaný ID3, je založen na booleanovských hodnotách, používá hladové vyhledávání v prostoru <sup>2</sup> a strom je vytvářen odshora dolů. Vstupní parametry tohoto algoritmu jsou **příklady** – data na kterých je rozhodovací strom vytrénován **atributy** – nad nimiž ude strom testován a **cílový atribut** – tento atribut se bude vytvořený strom snažit predikovat. ID3 algoritmus byl několikrát vylepšen a upraven pro použití za různých podmínek jako například C4.5 (vhodné při chybejících, 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.

Na obrázku 2.2 můžeme vidět příklad stromu, který vznikl pro rozhodování zda jít hrát badminton, pro jeho vytvoření byl použit algoritmus ID3 s menším vzorkem dat.

### 2.3.3 Dolování dat na webové aplikaci

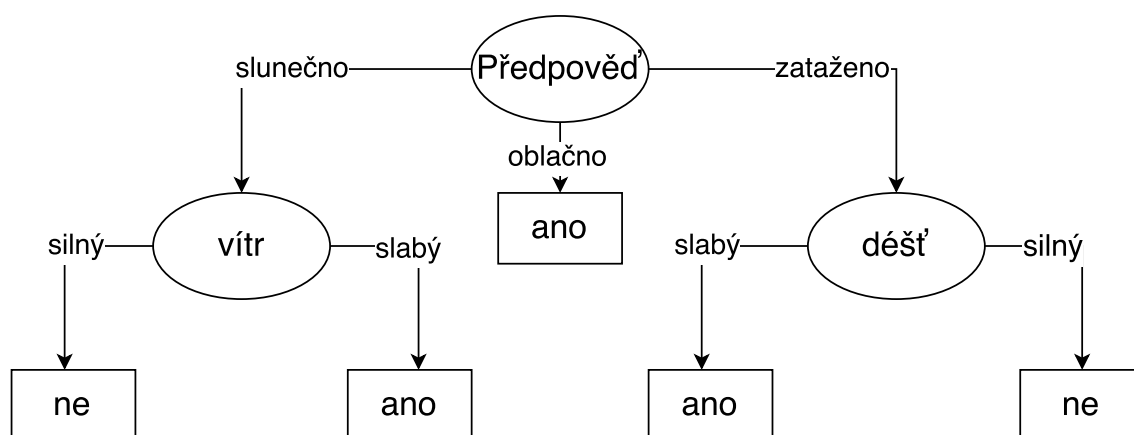
Toto získávání dalších informací z dat je v podstatě dolování z webových dokumentů, hyperlinků mezi nimi a získání informací z logů a webových stránek. [48]

Na webu lze využít několik možných technik dolování:

- **Webového obsahu** je proces získávání informací z webových dokumentů, mohou to být jak samotné stránky, tak videa, zvukových stop a obrázků. Hlavním zaměřením je však získání informací ze samotných textů na webovém dokumentu.

---

<sup>2</sup>Pro bližší informace o hladovém prohledávání <http://www.how2examples.com/artificial-intelligence/tree-search>



Obrázek 2.2: Příklad rozhodovacího stromu při použití algoritmu ID3.

- **Webové struktury** strukturu webových stránek si lze představit jako, uzlový graf, ve kterém jednotlivé uzly jsou samotné stránky a hrany spojující tyto stránky jsou vzájemné odkazy. Lze se také dále zanořit v jednotlivých dokumentech a ty znázornit pomocí stromové struktury <sup>3</sup>.
- **Webového použití** je název pro označení technik, díky kterým lze získat další informace z používání webových stránek. Tyto data lze získat z několika zdrojů **webový server** – záznamy, které vygeneroval server během jeho používání (IP adresy uživatelů, navštívenou stránku a čas navštívené stránky ...), **aplikační server** – použití moderních aplikačních serverů dovoluje hladší vytvoření podnikových aplikací, tyto servery nadále dovolují bližší sledování uživatelských interakcí, **data aplikací** – dále je možné v samotné aplikaci zapnout výpis dalších informací. [48]

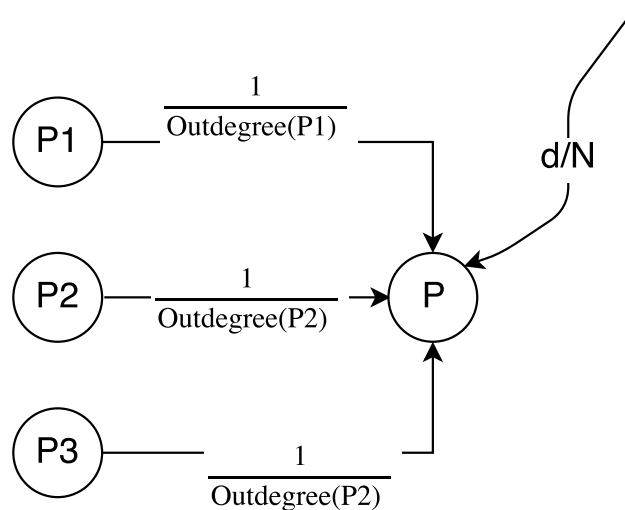
**PageRank** je označení pro stránkové ohodnocení, což má za následek vyšší skóre ve vyhledávacích nástrojích (jako je Google, Seznam, Yahoo, atd.). V jednoduchosti označují metriku pro ohodnocení dokumentů a zjištění jejich kvality. Takže hodnocení jednotlivých stránek záleží na hodnocení stránek, které odkazují na tuto stránku. Výpočet ohodnocení stránky  $p$  lze 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  $1 - d$

<sup>3</sup>Přesný popis tohoto zápisu je znám pod zkratkou DOM (Document object model) <https://www.w3.org/TR/DOM-Level-2-Core/introduction.html>

označuje pravděpodobnost že uživatel navštívá stránku z prokliknutého hyperlinku. [48]

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

Pro znázornění ohodnocení náhodné stránky se můžeme podívat na obrázek 2.3, kde na jednu stránku ukazují tři stránky **P1**, **P2** a **P3**.



$$PR(p) = d/N + (1 - d) \left( \frac{PR(P1)}{Outdegree(P1)} + \frac{PR(P2)}{Outdegree(P2)} + \frac{PR(P3)}{Outdegree(P3)} \right)$$

Obrázek 2.3: Příklad hodnocení stránky pomocí Markovova modelu náhodné stránky.

#### 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 tak 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 použít takzvané **Datové sklady** – ty mají možnost připojit se na několik databází (bez ohledu na jejich typ) a za pomoci sofistikovaných reportingových nástrojů vytahovat z těchto databází informace, které dále transformují a upravují.

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** <sup>4</sup> nebo **Elasticsearch** <sup>5</sup>

Pokud naopak využijeme datových skladů můžeme pro následnou vizualizaci využít takzvané OLAP kostky, což je sada 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 zdokonalovali a začínají být čím dál tím více uživatelsky přístupné. Firmy se snaží propojit několik takovýchto 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 takzvaný dashboard. Pro připojení na různé datové zdroje je použit nástroj Power BI gateway, který umožňuje, mimo jiné, připojit 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 je jednoznačně možnost rychle, a bez nutnosti pokročilých technik dolování dat, získat ukryté informace v zákaznických datech. Tohoto docílí inženýři ve firmě IBM tím, že vyvinuli velice silnou a rychlou umělou inteligenci, která má za úkol hledat skrytou podstatu v datech. <sup>6</sup> [44]

**MicroStrategy** platforma podporuje dashboardy, interaktivní reporty a dotazování, upozornění a mnoho dalšího potřebného pro vytváření strategie podniku a zjišťování informací z dat. Tato platforma nevyužívá klasické multidimenzionální OLAP kostky, ale využívá relační OLAP architekturu, což umožňuje možnost pou-

---

<sup>4</sup>Jednoduchý a otevřený indexační nástroj <http://lucene.apache.org/solr/>

<sup>5</sup>Nástroj, který vyhledává a dokáže také analyzovat data <https://www.elastic.co/>

<sup>6</sup>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](http://www.slate.com/blogs/future_tense/2014/02/14/watson_is_real_artificial_intelligence_despite_claims_to_the_contrary.html)

žití nástroje drill down v jakékoliv dimenzi. Výhodou této platformy pro programátory je jistě dodávaný vývojářský balíček, který umožňuje dalšímu upravování vytvořených reportů a grafů. [45]

**SAP** nabízí hned dvě řešení pro možné dolování informací z dat **SAP BusinessObjects** a **SAP HANA**. První ze jmenovaných je sada fron-end aplikací která nabízejí uživateli prohlížet, řadit a pracovat s BI datami. Druhým je platforma, 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 aplikace excel od firmy Microsoft, který v aktuální době používá více než 1,2 miliardy uživatelů <sup>7</sup>.

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 plnohodnou aplikaci, kterou by měl uživatel nainstalovanou na počítači a která by umožňovala uživateli pracovat pohodlněji s dokumenty (jak tomu je při použití aplikací od firmy Microsoft).

**Fusioo** je 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 které je možné jednoduše managovat tým a jeho aktivity.

**Ragic** tato platforma nabízí uživatelům možnost snadného přechodu z klasických excelových tabulek do datbázového světa bez nutnosti jejich pochopení. Hlavním tahákem této platformy je určitě její chytré a intuitivní vyhledávání a při zadávání

---

<sup>7</sup>Podle oficiální zprávy od Microsoftu v roce 2016 <http://www.windowcentral.com/there-are-now-12-billion-office-users-60-million-office-365-commercial-customers>

dat možnost zapnutí validace pro jednotlivé položky. Oproti konkurenci nabízejí neobvyklé reporty jako například TODO listy, náleky, kontingenční tabulky a mapy.

**Quickbase** je aplikace zaměřené 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, takovéto aplikace poté může zákazník dále nabízet ostatním uživatelům. Firma 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, nicméně 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 – jednotlivé záznamy mohou být navzájem propojené a získat tak další informace a uživatel také může definovat vzorce a formule. Nad konkurencí tento nástroj vede převážně díky možnosti snadno importovat a exportovat data.

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

## 2.5 Management vztahu se zákazníky (CRM)

Je v podstatě termín používaný pro označení strategií a technologií použitých společnostmi pro monitorování a zjišťování stavu co uživatelé dělají s jejich produkty – používají se jak k monitorování webových aplikací, tak 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 nechat CRM systém vygenerovat speciálně cílenou reklamu pro každého zákazníka.
- **Automatizace prodeje** eliminuje snahu prodat stejný produkt vícekrát různými zaměstnanci. 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 opakuje).
- **Geolokační technologie** některé CRM systémy dokonce disponují geolokační službou na základě které může podnik zjistit prodej určitého produktu například 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 firma díky tomuto nástroji získá možnost přilákat nové zákazníky díky zjištění chování stávajících zákazníků. [21]

**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** <sup>8</sup>. Služba nabízí základní grafy a reporty spolu s možností zobrazit je na dashboardu. Těchto druhů dashboardu může být několik druhů – základní, SEO analýza, sociální média, geografie, mobilní analýza, příchozí/odchozí a technický (uživatel si samozřejmě může vytvořit vlastní dashboard).

**KissMetrics** je placená alternativa ke Google analytics, cílí především na velké webové stránky a nabízí mnoho způsobů jak monitorovat zákazníky. Mezi přední výhodou patří možnost identifikace uživatele ještě před jeho přihlášením, kdy se ukládá

---

<sup>8</sup>Placená služba, která dovoluje zákazníkům předplatit si reklamu na webových stránkách.



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 <sup>9</sup>, 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.

## 2.6 Shrnutí

Aplikace zaměřující se na ukládání a práci s daty pomocí webového rozhraní se zaměřují převážně na větší podniky, případně nemají tolik intuitivní uživatelské rozhraní. Dále rozšiřitelnost dostupných aplikací není snadná a nastavení uživatelských práv je pouze v mizivém množství dostupných aplikací. Proto se dále zaměříme na vývoj samostatné aplikace, která spojí myšlenky z aktuálních aplikací a využije opensource nástroje, které umožní snadnější práci s uživateli.

---

<sup>9</sup>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 ...)

## 3 VLASTNÍ NÁVRHY

Nástroj, který je součástí této práce se zaměří z velké části na uživatelské rozhraní a do jisté části využije technologie a nástroje, které jsou dostupné a usnadní takto vývoj a nasazení tohoto nástroje. Dále se zaměříme na životaschopnost tohoto nástroje a způsobu jakým bude takto vytvořený nástroj nabízen veřejnosti. Rozsah firem, pro které bude tato aplikace doporučována je malé, až střední podniky, které pracují s větším množstvím dat a potřebují nějakým způsobem najít zkryté informace.

### 3.1 Návrh aplikace

Aplikace je postavena na základě aplikačního serveru WildFly, který pracuje s moduly napsanými v jazyce Java. Základní rozdělení aplikace je tedy **Engine** a **UI**, engine je připojen na databázi MongoDB, což je NoSql databáze. Výhodou tohtoto 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 samostaného enginu je možnost lépe řídit vývojářský tým, lépe rozdělit práci a v neposledn 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 tedy pomocí REST rozhraní a o zabezpečení se stará vrstva Keycloak, ve kterém jsou oba moduly registrovány a slouží jako takový středobod celého systému. Jak jsou jednotlivé části propojeny můžete vidět na obázku 3.1, kde hrany znamenají komunikační zprávy a uzly logické bloky.

## TODO

Obrázek 3.1: Diagram znázorňující schéma nástroje.

Pokud uživatel již disponuje velkým datovým uložištěm je ho možné připojit do systému jednoduchou konfigurací v rámci UI, systém prozkoumá tyto data, některá si nakopíruje a začne je uživateli nabízet podobně, jako kdyby je uživatel měl již dříve v systému. Podobně je tomu s dolováním dat – modul, který je zodpovědný za

samostatné dolování je možné připojit do systému, odkázat jej na datové uložště a spustit samostatné dolování dat. Vše je nastavitelné jak z uživatelského prostředí, tak pomocí volání na předem definované RESTové služby.

### 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, některé setkání, která jsou definována ve scrumu byla vyškrtána. Nejenom proto nelze vedení týmu, které bylo zvoleno označit jako čistokrevný scrum, ale pouze že jsme se při vedení inspirovali touto metodikou.

Nastavili jsme si třítydenní 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**<sup>1</sup>, který nám postupně přestal stačit a tak jsme využili opensource licence **Youtrack**<sup>2</sup>

### 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, které umožní snadné napovídání datových záznamů a popisu dat (v případě tabulky si toto lze představit že nám aplikace bude napovídat možné záhlaví tabulky a data v tabulce). Editace a nahlédnutí do záznamů bude povolena pouze určitému uživateli. Dále jednotlivé soubory bude uživatel moci seskupovat do takzvaných kolekcí, tímto dosáhneme

---

<sup>1</sup>Jednoduchý nástroj, který vizualizuje aktuální práci za použití jednoduchých tabulí <https://trello.com/>

<sup>2</sup>Tento nástroj je podobný Trello, nicméně dovoluje snadnější a pro tým důležitou vizualizaci práce <https://www.jetbrains.com/youtrack/>

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 aplikaci je možné 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, jak jsou tyto technologie propojeny a k čemu je to dobré.

### 3.2.1 Engine

Základní část celé aplikace a její středobod se skládá ze tří pomyslných částí **Data-báze**, **REST rozhraní** a **hlavní logika**. Díky tomuto rozdělení je možné snadně 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 Wildfly, díky jeho snadnému propojení do Keycloaku a převážně díky jeho robustnosti – největší předností je jednoduché napojení na velké množství různých databází. K načtení závislostí, sestavení a nasazení tohoto modulu byl použit nástroj maven. V podstatě pro jednoduchost si můžeme představit jako vstup databázi a jako výstup RESTové rozhraní.

- **Databáze** postavena nad technologií NoSql, 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** nástroji slouží pro snadnější přístup k jednotlivým datům a definuje snadno dostupné koncové uzly. Pokud bude potřeba nástroj rozšířit a hlavně zmapovat jednotlivé koncové uzly budeme moci použít například technologii Swagger <sup>3</sup>

---

<sup>3</sup>Slouží pro vizualizaci a zobrazení jaké data jednotlivý uzel očekává a jaké produkuje <http://swagger.io/>

### 3.2.2 Uživatelské rozhraní

Vzhledem k tomu, že hlavní část aplikace je nasazována na aplikační server Wildfly, také uživatelské rozhraní je sestaveno a možno nasadit jako samostatný modul. Samotný proces sestavení používá nástroj **Webpack**, který celý zdrojový kód rozdělí do dvou javascriptových souborů. Jeden obsahuje závislosti nutné pro chod aplikace a druhý obsahuje samostatný kód aplikace. V rámci usnadnění a jednoduššího přístupu k aplikaci jsme se rozhodli většinou část Javascriptových souborů umístit na veřejné CDN <sup>4</sup>.

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

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

#### Úkol číslo 1

Výběr kolekce a následně zjištění informací o této kolekci. Úroveň náročnosti **lehká**.

#### Úkol číslo 2

Výběr kolekce a dokumentu v kolekci pro který budou upraveny propojení. **středně těžká**

---

<sup>4</sup>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.

Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	25s	87837	787
2	38s	78	5415
3	43s	778	7507

Tabulka 3.1: The caption without a number

Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	25s	87837	787
2	38s	78	5415
3	43s	778	7507

Tabulka 3.2: The caption without a number

### Úkol číslo 3

Výběr kolekce a dokumentu v kolekci pro který se zobrazí práva. Úroveň náročnosti **těžká**

Číslo uživatele	Rychlost vykonání	Ohodnocení	Slovní popis
1	25s	87837	787
2	38s	78	5415
3	43s	778	7507

Tabulka 3.3: The caption without a number

Vzhledem na výsledky hodnocení a uživatelské podněty 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 uživatelů, 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

nějakým způsobem 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 navrženo nové uživatelské rozhraní, které lze vidět na 3.2.

## TODO

Obrázek 3.2: Návrh nového uživatelského rozhraní.

### 3.2.3 Práce s grafy

Díky tomu, že využíváme řebové 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 ovlává a je jednoduchá pro nastavení, nicméně složitější grafické útvary nejsou jednoduché na vytvoření. Základní uživatel disponuje několika druhy grafů z této knihovny. Pokud chce používat složitější grafické znázornění musí si více připlatit a v aplikaci se zpřístupní možnost používat složitější 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.

#### Složitější grafy

Jak již bylo nastíněno v předešlé sekci pro platící uživatele nabízí aplikace rozšířenou práci s grafy, převážně rozšíření 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 používána 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 nějaký způsobem větu, kterou odešleme na server a ten nám vrátí nejlépe sedící 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 na z těch je postupně tvořen dotaz na server. Opět, jak uživatel 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 (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 tomuto slouží tokenové vyhledávání, které na druhou stranu je složitější, ale přináší možnost hledat napříč velkým množstvím entit. Jako bonus, uživatel si 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 zamříme 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 roztrídí do binárního stromu. Jak lze například vidět na obrázku 3.3, 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 burger a které mají pizzu. Bohužel není vyhledávací algoritmus natolik přesný, aby vyhledal dokumenty ve kterých jsou tyto proměnné spojené.

# TODO

Obrázek 3.3: Binární strom pro roztrídění věty.

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*, 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 (prokliknutím do něj se dostane rovnou na vyhledané entity skrz uložený dotaz) a používat ve vyhledávání.

### 3.2.5 Zabezpečení

V mnoha moderních aplikacích dochází k problémům se zabezpečením, ať již je to nedostatečné zabezpečení vůči případným útokům, tak nechtěného dovolení přístupu různým uživatelům k datům, ke kterým by neměli mít přístup. Tento 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 autentikačních serverech 2.2.2.

V rámci tohoto nástroje je tento autentikační 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 autentikačním serverem keycloak, který má na starosti také tuto aplikaci). Pokud uživatel není přihlášen je vyzván k přihlášení, nabízíme možnost použít několik sociálních platforem – Github, Facebook, Google účet a Twitter, kromě klasického přihlášení pomocí emailu a hesla. Poté se již každý dotaz na server kontroluje tímto autentikačním serverem, takže je možné nastavit jednoduše pravidla uživatelům pro přístup jak k RESTovým službám, tak jednotlivým zdrojům za pomoci Keycloak serveru a vytvořením skupin.

Díky autentikačnímu 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 samostatný hlavní modul v rámci části dolování dat se zaměříme převážně na tři části. **Heuristiky pro automatické linkování** – pokud vznikne, nebo je upraven nějaký dokument je spuštěna řada automatických metod, které provedou případné automatické spojení, **Nejčastěji používané entity** – na mnoha místech se aplikace snaží usnadnit práci uživateli tím, že mu nabídne ty, které jsou nejčastěji používané a **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 do-**

**kument**, 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 tak nevytvoří 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 jednoduchost, snadnost a hlavně rychlost 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 tato funkce lze 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 definování jednoduchých funkcí uživatel bohužel ztrácí automatické znovunačtení záznamů v případě změn v propojených souborech, pokud chce načíst změny v takových záznamech musí si vyžádat načtení hodnot, které jsou uloženy do souboru.

## 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 na uživatelské požadavky obsahuje systém možnost zobrazit entity, které jsou nejčastěji používané daným uživatelem pro jednu organizaci a projekt. Pro zobrazení požadovaných entit jsme nejdříve vybrali algoritmus **zásobníku**, ale tento model se nám neosvědčil protože se často stávalo, že často používané entity mizeli a méně časté se nacházeli v nabídce, proto jsme do systému naimplementovali možnost přepnutí na takzvaný **zásobník 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.

- **Zásobník** 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 3.4, kdy zásobník obsahuje entity označené 1, 2, 3, 4 a 5. Uživatel otevřel nově entitu s označením c a entity v zásobníku jsou tedy (od spodu zásobníku) c, 1, 2, 3 a 4.

TODO

Obrázek 3.4: Zásobník, plnění.

- **Zásobník s počítadlem** je v podstatě rozšířená implementace zásobníku. Pokud je některá entita otevíraná častěji je u ní zvyšován počet otevření, entita s největším počtem otevření je na spodu zásobníku a entita s nejmenším počtem na vrchu. Pokud se otevře nový dokument je mu zvýšen počet otevření

a pokud překročí toto číslo nejvyšší prvek na zásobníku, tato nově otevřená entita nahradí tu, která byla na vrcholu zásobníku. Jak je to provedeno v rámci aplikace můžeme vidět na obrázku 3.5, 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 dvakrát. Nově tedy zásobník obsahuje entity d, 2, 3, 4 a 1.

## TODO

Obrázek 3.5: Rozšířený zásobník, plnění.

### 3.3.3 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. 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 nabídnout 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 *Rez*, *taue*, *auae* a *Res*. Uživatel zadá do hledání **Restauaes** a systém mu nabídne entitu **Restaurace**.

## 3.4 Analýza rizik

Vzhledem k tomu, že vytvořit takto náročnou aplikaci není triviální jsme si na začátku vývoje vytvořili analýzu možných rizik na která můžeme narazit během vývoje a při distribuci. Také jsme si tyhle 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

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

Po vytvoření základních tabulek pro odhadnutí rizik, jsme jednotlivá rizika zapsali do tabulky a ohodnotili jsme je možnou pravděpodobností a případně jaký by mělo dopad na funkčnost, samotné nasazení nebo celkové nepřijetí uživateli. Kde jako mezní jsme si určili hodnotu 14 bodů při ohodnocení pravděpodobnosti a dopadu rizika.

Č.	Scénář	Hrozba	Pravděp.	Dopad	Hodnocení
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 ovládatelné	3	3	9
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

Č.	Scénář	Hrozba	Pravděp.	Dopad	Hodnocení
6	Nezájem investorů	Aplikace nazujme případné investory, což povede k nedostatku zdrojů v prvotním spuštění	3	5	15

Tabulka 3.4: Možné scénáře rizik při vývoji a nasazení

Jak lze vidět na tabulce 3.5, tak pouze 2 rizika si vyžadují speciální pozornost, protože překračují námi definovanou hranici rizika, zaměříme se také na riziko s číslem **5**, které je by mělo také kritický dopad na chod aplikace.

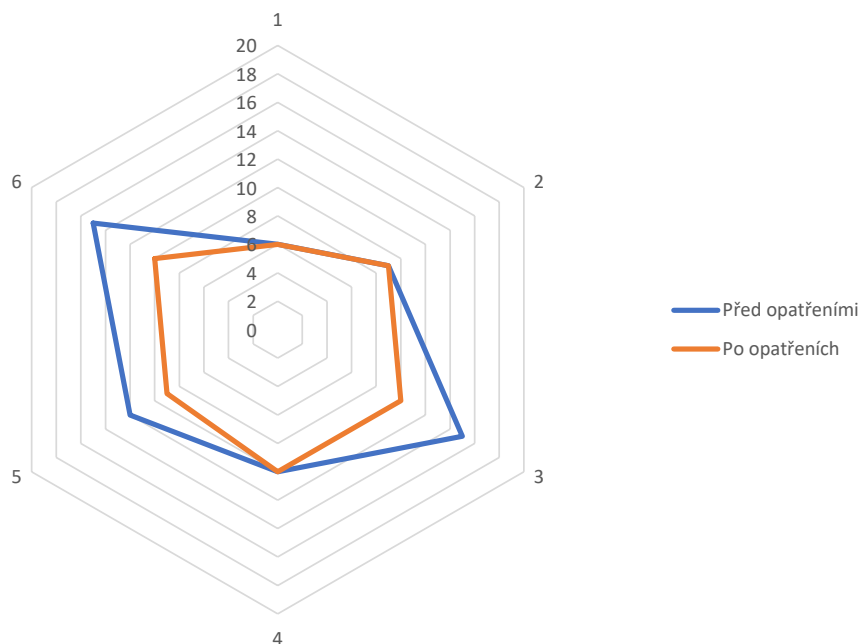
Č.	Opatření	Cena	Pravděp._	Dopad_	Hodnocení_
3	Při vývoji použijeme nástroj, který zabezpečí data.	Cena nástroje pro zabezpečení	2	5	10
5	Použití indexovacího nástroje	Cena nástroje pro indexaci dat	3	3	9
6	Dedikování člověka pro komunikaci s investory	Cena člověka pro komunikaci	2	5	10

Tabulka 3.5: Opatření pro vyhnutí rizikům.

Z možných opatření vyplývá že všem potencionálním technickým rizikům, které by měli 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 na kterém lze vidět že žádné riziko po nepřekračuje hodnotu 14 bodů, které jsme si

nastavili jako kritický bod. Nezájímali jsme se reizikům, které nemají tak vysokou pravděpodobnost, že by nastali, nicméně i na ty je si potřeba dávat pozor a proto je dobré, že jsme na některé z nich narazili již nyní.



Obrázek 3.6: Graf znázorňující rizika před a po zavedení opatřeních.

### 3.5 Cenový model

Jak již bylo řečeno několikrát v předešlém 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 třech skupin – **základní**, **střední** a **nejvyšší** (označováno též jako stříbrný, zlatý a platinový zákazník). Zákazníci si sami vyberou jaké funkcionality se jim více hodí, kdy první 2 měsíce mají placení zdarma na vyzkoušení. Takže zákazník dostane na začátku plnou podporu všech funkcí systému bez nutnosti platit jakoukoliv částku, až v případě že by chtěl školení, případně podporu v těchto dvou měsících je s ním vytvořena smlouva, která může pokrývat



například tyto dva zkušební měsíce. Pokud nechce na začátku zákazník nic platit, nemusí, dostane návod jak aplikaci používat a menší představení co s aplikací dělat a má možnost dva měsíce bezplatně testovat náš produkt.

Po uplynutí dvouměsíční lhůty si s námi zákazník sjedná schůzku, kde se domluví bližší detaily přechodu do systému. Na začátku zákazník platí za služby jako například převod celé databáze, školení uživatelů, podpora při problémech a výpadcích atd. Každý tento úkon je ohodnocen individuálně na základě velikosti databáze a době strávené na přechodu pod náš 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 počtu a druhu aktivovaných funkcí a do jaké ze tří skupin se zákazník řadí.

Vzhledem na to, že vývoj aplikace probíhá formou opensource tak nemáme možnost, aby si schopnější uživatelé nevzali naši aplikaci a nespustili si ji sami na svém vlastním stroji, tomuto se nijak nebráníme a je nám jasné že takové případy budou vznikat, proto musíme vsázet 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 informaci a následně nabízet uživateli jednoduché rozhraní pro práci s těmito daty.

Nejdříve bylo potřeba se seznámit s několika teoretickými pojmy, které byli důležité pro pochopení celého problému vývoje a návrhu takto rozsáhlé platformy. Vzhledem k tomu, že takováto platforma bude muset být často nasazena u několika zákazníků zároveň a že práce musí být plynulá a nemělo by záležet kde a jak ji uživatelé používají, proto bylo zvoleno použití webových technologií a k nim přidruženým nástrojům.

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

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

Během saní této práce byl již nástroj představen několika případným investorům, kteří vyslovili velký zájem a chtěli by do něj přenést svá data. Někteří investoři sami chválí myšlenku této platformy a jsou spokojeni s jednoduchostí a snadným používáním.

## Seznam obrázků

1.1	Znázornění jednotlivých vrstev v platformě openshift. . . . .	13
1.2	Fáze CRISP-DM, referenční model. . . . .	19
1.3	Podíl vybraných prohlížečů mezi uživateli. Vypracováno na základě dat TODO . . . . .	22
1.4	Cyklus návrhu grafických prvků. . . . .	23
2.1	Předání uživatelského požadavku při použití SAML2 . . . . .	31
2.2	Příklad rozhodovacího stromu při použití algoritmu ID3. . . . .	35
2.3	Příklad hodnocení stránky pomocí Markovova modelu náhodné stránky. . . . .	36
3.1	Diagram znázorňující schéma nástroje. . . . .	42
3.2	Návrh nového uživatelského rozhraní. . . . .	47
3.3	Binární strom pro roztřídění věty. . . . .	49
3.4	Zásobník, plnění. . . . .	52
3.5	Rozšířený zásobník, plnění. . . . .	53
3.6	Graf znázorňující rizika před a po zavedení opatření. . . . .	56

## Seznam tabulek

3.1	The caption without a number . . . . .	46
3.2	The caption without a number . . . . .	46
3.3	The caption without a number . . . . .	46
3.4	Možné scénáře rizik při vývoji a nasazení . . . . .	55
3.5	Opatření pro vyhnutí rizikům. . . . .	55

## 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] TODO data science business
- [12] TODO predictive analytics
- [13] TODO big data analytics
- [14] TODO big data for dummies
- [15] BRAMER, Max. *Principles of data mining*. 3rd edition. 2016. ISBN 978-1-4471-7306-9.
- [16] TODO data mining principles
- [17] TODO the forrester wave advanced data visualization adv platforms q3 2012
- [18] TODO Serverless Single Page Apps
- [19] TODO 1617290750Single
- [20] TODO interactive-data-reily
- [21] 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.
- [22] ALLAMARAJU, Subrahmanyam. *RESTful Web services cookbook*. Sebastopol, CA.: O'Reilly, c2010. ISBN 978-0596801687.
- [23] TODO cloud security
- [24] TODO <https://scotch.io/tutorials/the-anatomy-of-a-json-web-token>
- [25] TODO RFC <https://tools.ietf.org/html/rfc7519>
- [26] TODO <https://benmccormick.org/2015/09/14/es5-es6-es2016-es-next-whats-going-on-with-javascript-versioning/>
- [27] TODO <https://kangax.github.io/compat-table/es6/>
- [28] TODO <https://medium.com/javascript-scene/top-javascript-frameworks-topics-to-learn-in-2017-700a397b711#.fng4s5tee>
- [29] TODO <http://voidcanvas.com/framework-vs-library-one-better/>
- [30] TODO <https://docs.angularjs.org/guide/introduction>
- [31] TODO <http://www.developer.com/services/what-is-react.js.html>
- [32] TODO <https://vuejs.org/v2/guide/>
- [33] TODO <http://haroldrv.com/2015/02/typescript-making-javascript-strongly-typed/>
- [34] TODO <https://code.facebook.com/posts/1505962329687926/>

- flow-a-new-static-type-checker-for-javascript/
- [35] TODO <https://material.io/guidelines/#introduction-goals>
  - [36] TODO <https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/>
  - [37] TODO <https://www.patternfly.org/get-started/frequently-asked-questions/>
  - [38] TODO <http://openid.net/connect/>
  - [39] TODO <http://www.gigya.com/blog/the-basics-of-saml/>
  - [40] TODO <http://www.keycloak.org/about.html>
  - [41] TODO [https://www.ibm.com/support/knowledgecenter/en/SS4T7T\\_2.4.1/com.ibm.help.seasimplementationguide.doc/SEAS\\_About\\_SEAS.html](https://www.ibm.com/support/knowledgecenter/en/SS4T7T_2.4.1/com.ibm.help.seasimplementationguide.doc/SEAS_About_SEAS.html)
  - [42] TODO <https://msdn.microsoft.com/en-us/library/bb897402.aspx>
  - [43] TODO <https://powerbi.microsoft.com/en-us/what-is-power-bi/>
  - [44] TODO <https://www.ibm.com/analytics/watson-analytics/us-en/>
  - [45] TODO <http://searchbusinessanalytics.techtarget.com/definition/MicroStrategy>
  - [46] TODO <http://searchsap.techtarget.com/essentialguide/Business-Objects-or-HANA-A-guide-to-SAP-analytics-environments>
  - [47] TODO <http://searchcrm.techtarget.com/definition/CRM>
  - [48] TOOD minig book
  - [49] MARZ, Nathan a James WARREN. *Big data: principles and best practices of scalable real-time data systems*. ISBN 978-1617290343.
  - [50] HAN, Jiawei, Micheline KAMBER a Jian PEI *Data mining: concepts and techniques*. 3rd ed. Haryana, India ; Burlington, MA: Elsevier, 2012. ISBN 9789380931913.
  - [51] 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.