VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTĚNKA PRO GRAFICKÉ ZOBRAZOVÁNÍ FIREMNÍCH STATISTIK

GRAPHICAL REPRESENTATION OF COMPANY STATISTICS DASHBOARD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PAVLÍNA BORTLOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Studentka: **Bortlová Pavlína, Bc.**
Program: Informační technologie Obor: Management a informační technologie
Název: **Nástěnka pro grafické zobrazování firemních statistik**
Graphical Representation of Company Statistics Dashboard
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se s procesními, projektovými a byznysovými metrikami.
2. Prozkoumejte, jak by se daly tyto metriky použít pro firemní tým PnT DevOps (Product and Technologies Development and Operations).
3. Na základě výsledků průzkumu vyberte metriky, které by mohly být použity pro analýzu a zpracování dat z kolekce nástrojů používaných PnTDevOps, tak aby zobrazovaly požadované statistiky.
4. Navrhněte nástěnku, která bude zpracovávat statistiky z rozličných manažerských a servisních nástrojů (Jira, Bugzilla, ServiceNow, a případně další.). Pro každý nástroj vytvořte jednu, dle možností i více statistik (grafů).
5. Po dohodě s vedoucím implementujte moduly zpracovávající data z již zmíněných nástrojů. Za použití softwaru (Qlick, Tableau a vybrané další) vytvořte nástěnku s grafy pro zobrazení požadovaných statistik.
6. Použitelnost implementovaného řešení demonstrujte na vhodném vzorku dat vybraném po dohodě s vedoucí.
7. Zhodnoťte dosažené statistiky a použité metriky, zejména přínos vytvořeného řešení a diskutujte další možnosti budoucího rozšíření řešení.

Literatura:

- PARVIZ, Rad. *Metrics for Project Management*. Vienna: Management Conteps, Inc., 2006. ISBN 1-56726-166-3.
- KERZNER, Harold**.** *Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance:* John Wiley & Sons, Inc., 2017. ISBN 978-1119427285

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kreslíková Jitka, doc. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 30. října 2018

Abstrakt

Co je měřeno, může být vylepšováno. Proto firma Red Hat a tým *PnT DevOps* apeluje na vytvoření grafické vizualizace metrik a klíčových identifikátorů výkonnosti (KPIs), které bude k dispozici zákazníkům, inženýrům a zainteresovaným stranám k přehledu, jaký pokrok provází tým *PnT DevOps*.

Abstract

What is measured can be improved. That is why Red Hat and the *PnT DevOps* team are calling for a graphical visualization of metrics and key performance identifiers (KPIs) that will be available to customers, engineers, and stakeholders to review the progress of the *PnT DevOps* team.

Klíčová slova

Red Hat, nástěnka, byznysové metriky, KPI, DevOps, grafické zobrazení, Service Now, One Portal, MongoDB, Angular 7, TypeScript, html, Sass, DataHub, ngx-charts, API

Keywords

Red Hat, dashboard, business metrics, KPI, DevOps, graphical visualization, Service Now, One Portal, MongoDB, Angular 7, TypeScript, html, Sass, DataHub, ngx-charts, API

Citace

BORTLOVÁ, Pavlína. *Nástěnka pro grafické zobrazování firemních statistik*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Nástěnka pro grafické zobrazování firemních statistik

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní docentky Kreslíkové. Další informace mi poskytla společnost Red Hat a její zaměstnanci. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavlína Bortlová
31. července 2019

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Obsah

Kapitola 1

Úvod

Tato práce je projektem pro firmu Red Hat, ve které vznikla myšlenka mít nástroj, jež bude získávat data z manažerských a inženýrských nástrojů pro vytvoření centralizované nástěnky s byznysovými metrikami celého týmu *PnT DevOps (Product and Technologies Developement Operations)*. Tato myšlenka vznikla na popud manažerů týmu *Pnt DevOps Automation*, který je podtýmem *Pnt DevOps*. Cílem tohoto projektu je vytvořit nástěnku na jednom místě, která bude plně informovat jak zákazníky, inženýry, tak i zainteresované strany (stakeholders), o efektivitě a cílech, kterých chce tým *PnT DevOps* dosáhnout, a jak cíle splňuje.

Semestrální práce se zabývá teorií metrik a klíčových ukazatelů výkonnosti (KPIs - Key Performance Indicators), zkoumáním pojmu DevOps, rozborem týmu PnT DevOps a jeho podtýmů. Práce také rozebírá, jaké metriky a klíčové ukazatele výkonnosti používá společnost Red Hat (dále RH) a tým DevOps. Po zjištění dostatečných informací o metrikách a klíčových ukazatelů výkonnosti z teorie i z praxe DevOps týmu zapracujeme požadavky na byznysovou nástěnku. Jelikož bez dat, které tvoří metriky, by nebylo možné nástěnku vytvořit, budeme zde rozebírat nástroje, ze kterých získáváme data pro metriky, a celou nástěnku. Pro již zmíněný návrh nástěnky bude prostudováno které typy grafů mohou být použity pro určité typy metrik a klíčových ukazatelů výkonnosti. Z těchto požadavků bude vypracován návrh řešení jak získat data pro vytvoření metrik, kde tato data uložit, a následně jaké grafy použít pro metriky a klíčové ukazatele výkonnosti na vytvoření vizualizace byznysové nástěnky.

Kapitola 2

DevOps

Hlavní myšlenkou kapitoly je seznámit čtenáře s metodologií DevOps, která je hojně využívána pro svou efektivitu při stírání rozdílů mezi dvěma izolovanými světy vývojářů a administrátorů. První část kapitoly obsahuje rozbor pojmu (metodologie) „DevOps"(dále jen DevOps), její principy a pět hlavních faktorů, které nastiňují rozdíly mezi DevOps a agilní metodologií, jež je základem metodologie DevOps. V další části následuje popis týmu PnT Devops, jeho strukturu a seznámení s jeho podtýmy. Poslední část obsahuje příklady nástrojů, které tým DevOps každodenně používá, technologie, se kterými se setkává, a služby, které provozuje.

2.1 Pojem Devops

DevOps je složenina dvou anglických slov „development", což znamená „vývoj", a „operations", znamenající v překladu „operace", nebo také „administrativní činnosti". Vývojová část „Dev" je tvořena všemi, kteří vytvářejí kód od vývojářů až po designéry aplikací, kvality inženýry a testery. Operační částí „Ops" jsou myšleni všichni (administrátoři), kteří pracují se systémy a zajišťují jejich chod, jsou to například systémoví administrátoři nebo síťoví administrátoři. Devops je metodologie, kde administrátoři, členové části „Ops" a inženýři, členové části „Dev", spolupracují a zajišťují plynulý průběh životního cyklu složeného z návrhu, vývoje a administrace produktů nebo služeb [13]. DevOps může být interpretováno jako výsledek agilní metodiky, kde je agilní softwarový vývoj určen jako blízká spolupráce se zákazníkem, produktovým řízením(dále management), vývojáři a v některých případech i s inženýry kvality pro zrychlení iterací vývoje produktu. DevOps nahrazuje model, kde tým vyvíjející kód, dále tým, je oddělen od týmu, jež testuje tento kód, dále týmu, jež nasazuje kód do produkce a nakonec týmu, starajícího se o funkčnost kódu.

Jak už bylo zmíněno, nad DevOps můžeme smýšlet jako nad výsledkem agilní metodiky, ale pro lidi pracující v IT sféře může toto označení mít různé významy. Například takového významu se DevOps vyznačuje myšlenkou „zdrojový kód jako infrastruktura" (anglicky *treating your code as infrastructure*, také *infrastructure as code*), což je typ IT nastavení, kde inženýrské a administrátorské týmy řídí a poskytují technologické řešení, které nastaví celou infrastrukturu [17]. Dalším příkladem pochopení DevOps může být používání automatizace. Jiné zdroje uvádějí, že srdcem DevOps jsou tři myšlenkové směry[3][10]:

- **„princip proudu"** (anglicky *Flow principle* [18]), který vyžaduje rychlý a plynulý průběh projektu od vývoje, přes operační činnosti až po rychlé doručení hodnoty zákazníkovi. Pro uskutečnění tohoto principu napomáhají níže uvedené body:

  - kontinuální integrace (anglicky *Continuous Integration*),
  - kontinuální dodávání (anglicky *Continuous Delivery*),
  - kontinuální nasazování (anglicky *Continuous Deployment*),
  - mapování toku hodnot (anglicky *Value Stream Mapping (VSM)*),
  - Kanban plánovací systém,
  - teorie omezení (anglicky *Theory of Constraints (TOC)*) se snaží definovat omezení, která zpomalují nebo určují výkon systému [2].

- **„princip zpětné vazby“**(anglicky *Feedback principle*) jde o naslouchání, porozumění a reakci na potřeby zákazníků a zainteresovaných stran (anglicky *stakeholders*). Cílem je vytvořit a zajistit bezpečný a rezistentní systém práce. Systém práce by měl zahrnovat:
  - automatické testování (anglicky Automated testing),
  - recenze či posouzení produkčních změn kódu (anglicky *Peer review of production changes*),
  - monitorování a praktiky oznamování (anglicky *Monitoring and notification practices*),
  - vytvoření a udržování srozumitelných nástěnek a aktualizace stavů projektů (anglicky *'At a glance' dashboards and status updates*),
  - ukládání produkčních záznamů o činnostech (anglicky *Production logs*),
  - měření procesů (anglicky *Process measurements*),
  - procesy, který napomáhají se učit z předešlých událostí pomocí analýzy a diskuse (anglicky *Post-mortems*) [7],
  - udržování nepřetržité pohotovosti (anglicky *Shared on-call rotation*) při potížích např. s produkčními servery nebo s produktem samotným,
  - ukládání záznamů z managementu změn, incidentů, problémů a znalostí (anglicky *Change, Incident, Problem and Knowledge Management data*).

- **„princip kontinuálního experimentování a učení se“** (anglicky *Continuous experimentation and learning*) je principem třetím a posledním. Tento princip poukazuje jak je důležité vytvořit firemní kulturu, jež zahrnuje:
  - podporu pro neustálé zvyšování kvalifikace,
  - rozšiřování znalostí pomocí učení se a experimentování,
  - podpora pracovníků ve zdravém riskování,
  - možnosti se poučit z vlastních chyb.

Stejně jako agilní metodika DevOps je definován za pomocí **5 faktorů**:

- **Hodnoty** DevOps se odvíjejí od hodnot agilní metodiky, a tudíž jsou založeny na agilnímu manifestu s lehkým vylepšením, a to s cílem nejen vytvořit fungující software, ale také ho plně doručit zákazníkovi.

- **Principy**, které odstartovaly myšlenku DevOps byly iterativnost(anglicky *iterative*), inkrementálnost (anglicky *incremental*), opakovatelnost(anglicky *continuous*), automatizovanost(anglicky *automated*), sebeudržovatelnost(anglicky *self-service*), spolupráce (anglicky *collaborative*) a celostní pohled na systém (anglicky *holistic*). DevOps by mělo porovnat své procesy s těmito principy a zjistit kde samo přidává hodnotu firmě.
- **Metodami** mohou být například Scrum nebo Kanban, a jelikož DevOps vychází z agilní metodologie můžeme si býti jisti, že použitím agilních metod neuděláme chybu.
- **Praktikami** používanými v DevOps jsou kontinuální sestavení (anglicky *Continuous build*), kontinuální integrace (anglicky *Continuous integration*), kontinuální dodání (anglicky *Continuous delivery*), management automatického uvolňování (anglicky *Automated release management*) a inkrementální testování (anglicky *incremental testing*) [6].
- **Nástrojů**, které používají DevOps týmy, je velké množství, ať už se jedná o integrační nástroje, či nástroje pro automatizaci. Popis těchto nástrojů bude zmíněn v následujícím textu.

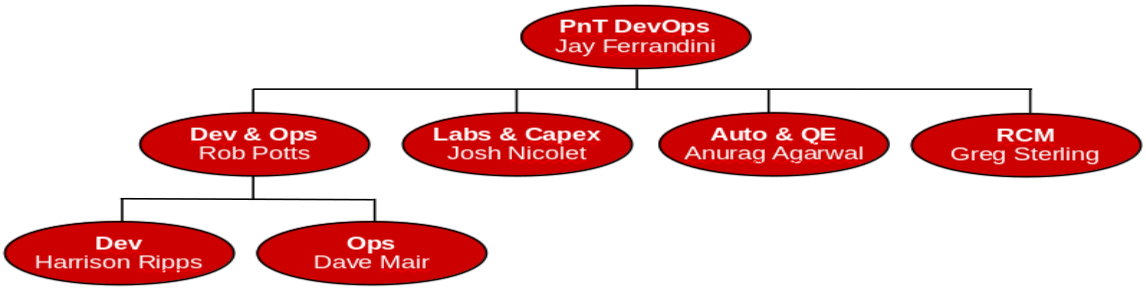
2.2 Tým PnT DevOps

V minulé kapitole je zmíněno jak by mělo vypadat fungování DevOps, kterými principy se vyznačuje a jaké faktory DevOps definují. Používání metodologie DevOps se může v malých a korporátních společnostech dosti lišit a důležitým faktorem, který ovlivňuje využití metodologie je společností zavedena. Pokud byla například společnost založena v posledních letech, mohla vzniknout na bázi DevOps metodologie a její struktura týmu bude čistě podle tři myšlenkových směrů zmíněných v sekci 2.1. Pokud však byla struktura společnosti nebo struktura týmů založena na jiném přístupu a společnost chce přejít na DevOps přístup, může se stát, že strukturu týmu nebude možno převést na čistý DevOps přístup.

V této práci se budeme zabývat společnosti Red Hat a týmem „PnT DevOps“(*Product and Technologies Development and Operations* dále PnT DevOps). Struktura tohoto týmu se po roky vytváří a přeskupuje až do dnešní řekněme hybridní podoby DevOps. Proč mluvíme o hybridní podobě DevOps? Společnost RH je společností korporátní, má okolo 12600 zaměstnanců a její tým PnT DevOps má kolem 250 inženýrů, což je velké množství pro vytvoření jednoho fungujícího týmu podle DevOps přístupu. Hybridním proto, že transformace staré struktury týmu (vývojáři a administrátoři) na strukturu podle metodologie DevOps, byla a je složitá a bude ještě potřebovat nějaký čas než se dostane do podoby DevOps. Tým Pnt DevOps se dnes skládá z pěti pilířů SysOps(systémové operace anglicky *system operations*), Development(vývoj), RCM(anglicky *release configuration management* nebo *release engineering*), Automation&QE(automatizace a inženýrství kvality anglicky *Automation and Quality engineering*) a Lab and Capital management, která je zobrazena na obrázku 2.1.

Tým **SysOps** je pověřen zajištěním odpovědností, a to tak, aby geograficky rozložené heterogenní prostředí bylo funkční a dostupné pro Red Hat organizaci. SysOps poskytuje globální pokrytí s operačními základnami umístěnými v Severní Americe, Austrálii, Číně a České republice. Toto geografické rozložení umožňuje týmu nabízet zákaznickou podporu 24-7.

[1] Dostupné na: https://mojo.redhat.com/docs/DOC-1048960#jive_content_id_Common_Tools_Used_In_DevOps



Obrázek 2.1: **Struktura PnT DevOps týmu.**[1]

Development tým rozvíjí interní nástroje a zásuvné moduly (anglicky *plugins*), které společnost RH každodenně používá k vytváření a dodávce RH softwaru.

Tým **RCM** udržuje nástroje, procesy a vnitřní nařízení společnosti, které umožňují definovat, vyvíjet a dodávat produkty konzistentním a podporovatelným způsobem.

Tým **Automation** staví automatizované služby a testy, které osvobozují inženýry DevOps od opakujících se chybných úkolů, také spolupracuje s inženýrskými týmy, aby vytvořili systém, který spojuje procesy a nástroje DevOps dohromady. Produkty Automation týmu odhalují efektivitu a umožňují inženýrům RH soustředit se na úspěšnost zákazníků. Tým **QE** se zaměřuje na kvalitu vnitřních nástrojů RH, které pomáhají distribuovat zákaznické produkty. Cílem týmu je poskytnout stabilní, rychlou a uživatelsky přívětivou sadu nástrojů (realease toolchain), aby se inženýři společnosti RH mohli zaměřit na spokojenost zákazníků.

2.3 Používané technologie, nástroje a služby

Obecné informace o DevOps nástrojích jsou získány z osobních zkušeností z firmy RH a ze stránek společnosti XebiaLabs[2], kde byla vytvořena periodická tabulka DevOps nástrojů. Společnost RH je novodobou společností, která se nebrání novým technologiím, takže se technologie nejen inovují, ale také vyvíjejí. RH představil několik nástrojů, jako je *Ansible*, používaný pro konfiguraci a vykonávání skriptů na vzdálených počítačích, nebo *OpenShift*, víceúčelová cloudová platforma, která je používaná pro vývoj, nasazení a řízení aplikací. Avšak RH nepoužívá jen své nástroje a technologie, ale díky dostupnosti open source technologií přebírá, inovuje a nastavuje je na míru požadavkům projektů. Tým PnT DevOps se zaměřuje na vývoj, automatizaci a údržbu vyvinutých nástrojů a pro tyto potřeby používá Python, Bash a jiné skriptovací jazyky. Pro testování Python skriptů se používají knihovny jako pytest, unittest, mock atp. a pro testování v ostatních jazycích vytvářejí inženýři a testeři své vlastní testy.

První a nejdůležitější nástroj pro programátora je nástroje pro ukládání a verzování kódu. Důležitost sdíleného úložiště je pro spolupracující programátory velice významná, a to z důvodu možnosti spolupráce více lidí na jednom kódu nebo projektu, a také proto, že verzování umožní uživatelům používat verzi softwaru, která je kompatibilní s jejich systémem nebo ostatními nástroji. Příklady těchto nástrojů jsou zobrazeny a popsány v tabulce 2.1.

[2] https://xebialabs.com/periodic-table-of-devops-tools/

Tabulka 2.1: **Nástroje pro uchovávání a verzování kódu**

Nástroje	Dostupnost	Popis
GitLab	Open source	GitLab je webový správce úložiště Git s funkcemi pro sledování problémů. GitLab je podobný nástroji GitHub, ale GitLab je na rozdíl od GitHub open source nástroj.
GitHub	Freemium	GitHub je webová hostingová služba Git úložiště, která nabízí všechny funkce kontroly nástroje Git a správu zdrojového kódu (SCM) a navíc přidává vlastní funkcionalitu. Na rozdíl od nástroje Git, který je jen nástrojem příkazového řádku, GitHub poskytuje webové grafické rozhraní a i desktopovou a mobilní integraci.
Subversion	Open source	Apache Subversion (často zkráceně SVN, po názvu příkazu svn) je systém pro správu verzí a revizí softwaru, distribuovaný jako open source pod licencí Apache.
BitBucket	Freemium	Bitbucket je webová hostingová služba pro projekty, které používají buď revizní systémy Mercurial nebo Git. Bitbucket nabízí jak bezplatné, tak i placené účty, které nabízejí funkcionalitu navíc.

Kontinuální integrace je soubor principů aplikovaný na každodenní práci vývojových týmů. Tým je zodpovědný za zajištění nasazení nové funkcionality do produkce nebo k zákazníkovi. Tento pracovní proces nasazování, podporovaný použitím verzovacích nástrojů, se skládá z vytvoření nového kódu umístěného do git úložiště a kontroly kódu od dalších programátorů, předtím než bude nová funkcionalita doručena zákazníkovi nebo nainstalována do produkce [12]. Příklady nástrojů pro kontinuální integraci jsou zmíněny v tabulce 2.2 a textu pod tabulkami.

Tabulka 2.2: **Nástroje pro kontinuální integraci**

Nástroje	Dostupnost	Popis
Jenkins	open source	Jenkins je nástroj kontinuální integrace napsaný v jazyce Java. Jenkins poskytuje kontinuální integrační služby pro vývoj softwaru. Podporuje nástroje SCM, včetně nástrojů Subversion, Git, Mercurial a dalších.
Bamboo	Placené	Bamboo je integrační server od společnosti Atlassian, výrobců JIRA a Confluence. Bamboo podporuje sestavení jakéhokoliv programovacího jazyka pomocí libovolného nástroje pro sestavení, zahrnující nástroje Ant, Maven, make a nástroje příkazové řádky. Zasílání sestavovacích (build) oznámení lze přizpůsobit podle typu události a přijímat pomocí e-mailu, okamžitých zpráv, RSS atd.
Travis CI	open source	Travis CI je open source hostovaná integrační služba, která se používá k sestavování a testování projektů hostovaných v GitHubu. Travis CI je nakonfigurován tak, že do kořenového adresáře úložiště GitHub je přidán soubor s názvem `.travis.yml`, což je textový soubor ve formátu YAML.

V tabulce 2.3 je popis příkladů nástrojů pro konfiguraci a zavádění softwaru hromadně do dvou a více zařízení najednou.

Tabulka 2.3: **Nástroje pro konfiguraci**

Nástroje	Dostupnost	Popis
Ansible	open source	Ansible je softwarová platforma pro konfiguraci a správu počítačů. Kombinuje zavádění softwaru s více uzly, nahodilé provádění úkolů a správu konfigurace. Spravuje uzly přes SSH nebo PowerShell a vyžaduje, aby byl na nich nainstalován Python (2.4 nebo novější). Moduly pracují s JSON a standardním výstupem a mohou být napsány v libovolném programovacím jazyce. Systém používá YAML k vyjádření opakovaně použitelných popisů systémů.
Puppet	Software jako služba (SaaS)[8]	Puppet je nástroj pro správu konfigurací. Je kompatibilní s Unix i Windows distribucemi a obsahuje vlastní deklarativní jazyk pro popis konfigurace systému.
CFEngine	open source	CFEngine je systém pro správu konfigurací. Jeho primární funkcí je poskytovat automatizovanou konfiguraci a údržbu rozsáhlých počítačových systémů včetně jednotného řízení serverů, stolních počítačů, spotřebních a průmyslových zařízení, vestavěných síťových zařízení, mobilních smartphonů a tabletových počítačů.

Kontejnery jsou velice moderní a využívanou technologií snad úplně ke každému použití. Poslední dobou jsou kontejnerová řešení velice využívána a to z důvodu usnadnění práce s různými softwarovými konfiguracemi a nízkou spotřebou fyzických zdrojů jako je paměť. V tabulce 2.4 jsou zobrazeny příklady kontejnerových řešení.

Tabulka 2.4: **Kontejnerové řešení**

Nástroje	Dostupnost	Popis
Kubernetes	open source	Kubernetes je systém pro správu kontejnerových aplikací, napříč několika zařízeními, a poskytuje základní mechanismy pro nasazení, údržbu a škálování aplikací.
Docker	open source	Docker je projekt, který automatizuje nasazení aplikací uvnitř softwarových kontejnerů. Docker využívá funkce izolace zdrojů jádra Linuxu, aby umožnila spuštění nezávislých „kontejnerů“ v jedné instanci systému Linux, a tím se vyhne režijním nárokům při spouštění a údržbě virtuálních strojů.

Tabulka 2.5: **Cloudové systémy**

Nástroje	Dostupnost	Popis
OpenShift	open source	OpenShift je víceúčelová cloudová platforma(PaaS) pro vývoj, nasazení a řízení aplikací.
OpenStack	open source	OpenStack je software pro vytváření soukromých a veřejných cloudových řešení[3].

Tabulka 2.6: **Nástroje na monitorování**

Nástroje	Dostupnost	Popis
Zabbix	open source	Zabbix je monitorovací řešení pro sítě a aplikace, které je určeno primárně pro firmy. Zajišťuje monitorování a sledování stavu různých síťových služeb, serverů a jiného síťového hardwaru.
Nagios	open source	Nagios je softwarová aplikace, která monitoruje systémy, sítě a infrastrukturu. Nagios také nabízí monitorovací a upozorňovací služby pro servery, přepínače(switches), aplikace a služby. Upozorňuje uživatele na chyby, které se objevily a také informuje uživatele, když je problém vyřešen.

Tabulka 2.7: **Nástroje pro spolupráci a komunikaci**

Nástroje	Dostupnost	Popis
Jira	placené	Jira je výkonná platforma, která kombinuje možnosti zaznamenávání problémů a správy projektů v jediné aplikaci. Použití nástroje Jira napomáhá plánování a organizování úkolů, projektů, pracovních postupů a shrnutí postupu práce.
ServiceNow	placené	ServiceNow je nástroj pro IT systémový management (ITSM), který se zabývá (Incident Management Problem Management, Change and Release Management, Request Management, Asset and Cost Management, Walk-Up Experience Knowledge Management, Configuration Management, Reports and Dashboards, Service Level Management, Benchmarks, Surveys and Assessments)

Kromě již zmíněných nástrojů RH dále využívá různé typy databází od multiplatformní relační databáze **MySQL**, přes bezschémovou databázi **Elasticsearch**, která je zároveň i vyhledávacím nástrojem s rozhraním *API*, až po **DataHub**[4], nástroj datového managementu, který je datovým úložištěm mnoha organizací jakou jsou OSN, Oxfordská univerzita, a další.

[3]Dostupné na: https://www.openstack.org/
[4]Dostupné na: https://datahub.io/

Velice důležitou informací pro společnost vyvíjející software je zpětná vazba zákazníka a oznamování chyb v produktech nebo službách, které jsou poskytovány. **Bugzilla**[5] je systém pro zaznamenávání bugů a problému v open source softwarových produktech. Tento systém napomáhá oznamovat vývojářům problémy, které se vyskytly u jejich produktů nebo služeb, a komunikovat se zákazníky o vzniklých problémech.

Další z RH nástrojů je **Errata**, která řídí proces přípravy a zasílání aktualizací softwaru pro produkty Red Hat. Tento proces zahrnuje přípravu, balení, testování, práce s dokumentací, podepisování balíků a konečné dodání zákazníkům.

[5] Dostupné na: https://www.bugzilla.org/docs/2.16/html/whatis.html

Kapitola 3

Metriky a klíčové ukazatele výkonnosti (KPIs)

Metriky a klíčové ukazatele výkonnosti jsou základními stavebními bloky vizualizací nástěnek, které upozorňují na vztahy mezi definovanými firemními cíli. Napomáhají zaznamenávat vývoj společnosti a jak se společnost inovuje. Předtím než se metriky a klíčové ukazatele výkonnosti (dále jen KPIs) začnou používat se, je nutnost dohody mezi manažery a zainteresovanými stranami, jaké specifické metriky a KPIs se budou používat a jak budou měřeny. Mezi oběma stranami musí proběhnout dohoda o tom, jaké metriky a KPI budou součástí nástěnkového informačního systému (anglicky *dashboard reporting system*), a specifikovat, jak budou změřená data interpretovaná.

Pojem **metrika** je asociován s numerickou reprezentaci byznysových dat ve vztahu k veličinám, například „prodej produktu za týden“, kdy „podej produktu v eurech“ je numerická reprezentace a „týden“ je vztahová veličina. Metriky jsou nástroje, které informují všechny zainteresované strany (stakeholders) o stavu měřeného systému.

Pojem **klíčový ukazatele výkonnosti** je typ metriky, která je vztažena k nějakému cíli. KPI nejčastěji reprezentuje rozdíl mezi předpokládaným výsledkem a jeho skutečnou hodnotou. Vztah mezi metrikou a KPI můžeme přiblížit na příkladu, kdy si určíme cíl prodeje „za týden prodat produkty za 10 000 euro“. V této definici bude metrikou „produkt za týden“ a cíl „10 000 euro“. KPI určuje splnění stanoveného cíle v procentech. Tudíž, pokud ve středu zjistíme, že byly prodány produkty za 8 000 eur, KPI pak ukazuje 80%ní splnění cíle.

3.1 Byznysové metriky

Výsledky byznysu záleží na metrikách, které měří výkonnost v průběhu času, a monitorují pokrok směrem k cíli. Pro byznysové metriky jsou specifické finanční a nefinanční metriky. Finanční metriky, získané z učetnictví firmy, napomáhají akcionářům a zainteresovaným stranám zhodnotit celkové finanční zdraví organizace. Nefinanční metriky se zaměřují na procesy, služby nebo měření kvality, které jsou taktéž důležité pro zdraví a úspěch organizace [15].
Nejpoužívanější typy finančních metrik jsou:

- **ukazatelé likvidity** - „ukazatele vyjadřující finanční schopnost hradit dluhové závazky z využitelné hotovosti“[1],
- **ukazatelé zadluženosti** - „ukazatele zadluženosti, patřící mezi poměrové ukazatele, bývají označovány také jako ukazatele dlouhodobé finanční stability“[2],
- **ukazatelé rentability** - „poměrové ukazatele finanční analýzy, vyjadřující využití aktiv při vytvořených nákladech za účelem dosažení přijatelný míry výnosu“[3].

Nefinančními měřeními výkonnosti jsou myšleny například:

- **měření spokojenosti zákazníka**,
- **měření angažovanosti a spokojenosti zaměstnanců**,
- **metriky kvality**,
- **měření vývoje a učení**.

3.2 Projektové metriky

Projektové metriky jsou specifické metriky pro vývoj softwarových produktů. Důvodem zjišťování těchto metrik je snaha kontrolovat a měřit rozsah projektu, cenu, časový plán projektu, spokojenost zákazníků a řízení projektů. Měření rozsahu projektu napomáhá vývojovým týmům pochopit, kolik času je potřeba na vývoj a kolik stojí distribuce a oprava chyb. Existuje zde několik metod pro odhadování rozsahu projektu, a to jsou **metoda funkčních bodu** (anglicky *function point analysis (FPA)*) a **počet řádků kódu** (anglicky *lines of code (LOC)*).

3.3 Procesní metriky

Metriky procesu měří všechny aspekty procesu vývoje softwaru. Cílem metrik je předpovědět a následně snížit odchylku od předpokládaného výsledku. Jako vlastnosti softwarového procesu můžeme chápat řízení podle cílů, řízení procesů a proces vyhodnocování výsledků a kontinuální zlepšování procesů. Pokud budeme měřit a vyhodnocovat softwarové procesy vytvářející softwarové produkty, napomůžeme tím k zlepšování a zkvalitňování těchto produktů. Pokud budeme mluvit o procesech ve společnosti RH, tak jimi budeme označovat projektové procesní metriky.[4]

[1] Dostupné na: https://www.patria.cz/slovnik/433/liquidity-ratios.html
[2] Dostupné na: https://managementmania.com/cs/ukazatele-zadluzenosti
[3] Dostupné na: https://www.patria.cz/slovnik/487/profitability-ratios-.html?culture=sk-SK

Kapitola 4

Nástěnka

Nástěnka je grafickým zobrazením přiměřeného počtu důležitých metriky a klíčových ukazatelů výkonnosti tak, že zainteresované strany a projektoví zástupci mohou vidět důležité informace, a napomoci tak k rozhodnutím. Čistá data jsou převedena na smysluplné informace, která jsou srozumitelně zobrazena na počítačovém displeji. Zajímavými fakty, které souvisejí s nástěnkami, jsou:

- Nástěnky jsou komunikační nástroje.
- Nástěnky poskytují informace o aktuální situaci a co to může znamenat do budoucna.
- Nástěnky nejsou do detailů rozepsané zprávy o aktuální situaci.
- Je důležité, aby se informace na nástěnky poukazovaly na budoucí situaci. Jinak bude analyzována jen minulost a nebude brán zřetel na směr, kterým metriky ukazují.
- Musíme klást důraz, aby zainteresované strany rozuměly, co je na nástěnce zobrazeno, co bylo měřeno a důležitosti sledování dané nástěnky.

Při vytváření nástěnky je nutné zvážit tři důležité charakteristiky:

- Cílové skupiny, které budou nástěnku sledovat.
- Typ nástěnky, která bude použita.
- Frekvenci aktualizací dat.

4.1 Typy nástěnek

Operační nástěnky (anglicky *operational dashboards*) monitorují hlavní operační procesy a jsou používány především pracovníky a vedoucími, kteří přímo komunikují se zákazníky, nebo řídí výrobu nebo dodávku firemních produktů nebo služeb. Nástěnka je aktualizována na denní bázi a klade se v ní více důraz na monitorování procesů než na analýzu a řízení procesů.
Taktické nástěnky (anglicky *tactical dashboards*) zaznamenávají procesy (na úrovních oddělení) a projekty, které jsou označeny za zajímavé pro omezenou skupinu lidí nebo segment organizace. Manažeři a byznys analytici používají taktické nástěnky pro porovnání výkonů jejich sféry působení nebo projektů, a následně pro předpovídání, vytváření plánů na rozpočet nebo shrnutí výsledků.

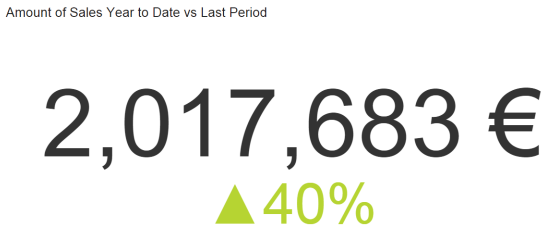
Strategické nástěnky (anglicky *strategic dashboards*) monitorují vykonávání strategických cílů a jsou často implementovány pomocí metodologií Balance Scorecard, TQM, Six Sigma. Cílem strategické nástěnky je seskupit celou organizaci kolem strategických cílů a určit cíl, kterým se organizace a všechny její části budou ubírat. Ve strategické nástěnce je kladen důraz více na řízení než na monitorování a analýzu.[9]

4.2 Typy vizualizace

Semaforová signalizace hlášení (anglicky *traffic light dashboard reporting*) je zřetelná a výstižná technika zobrazování výkonnosti projektů nebo systémů. Signalizace může nabývat různých významů, které však musí být předem definovány. Příklady významu signalizace:

- červená - Existuje problém, který může ovlivnit čas, peníze, kvalitu a rozsah. Je nutný zásah zainteresovaných stran.
- žlutá či oranžová - Upozornění na potenciální problém, který může vzniknout, pokud se situace nebude monitorovat.
- zelená - Vše funguje podle plánu a není nutnost zásahu od zainteresovaných stran.

Číselné grafy (anglicky *number chart*) se používají pro zobrazení aktuálních dat a poskytne přehled o konkrétním KPI. V obrázku 4.1 je příklad zobrazení KPI, který ukazuje nárůst objemu prodeje za rok k dnešnímu datu o proti minulému období. Toto zobrazení dat je zřejmě nejjednodušší pro zobrazení dat v rozmezí času a je velice nutné v něm popsat rozmezí, které určuje graf, tak, aby byl pro pozorovatele srozumitelný. Je doporučeno se dále vyvarovat použití nadměrného množství číselných grafů v nástěnce.



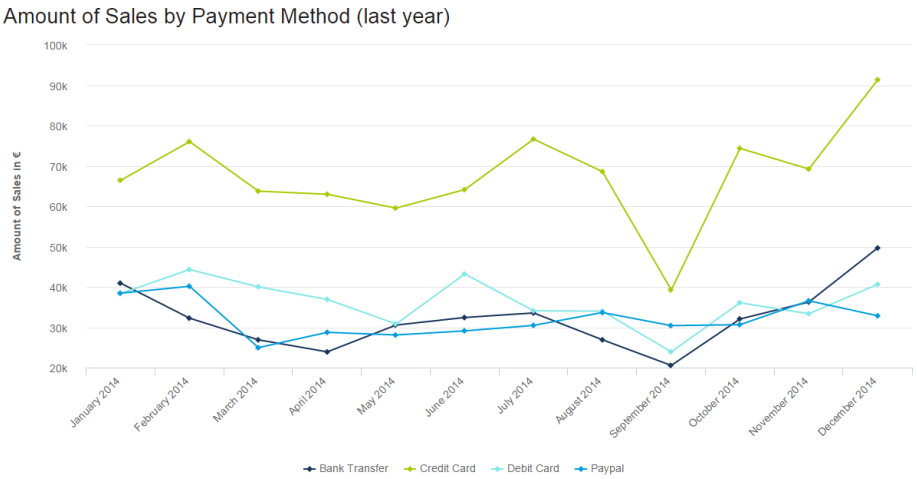
Obrázek 4.1: Ukázka číselného grafu[1]

Spojnicový graf (anglicky *line chart*) zobrazuje trendy, zrychlení/zpomalení a volatilitu. Tento typ vizualizace vyjadřuje vztahy, které se v čase mění. Ukázka spojnicového grafu je na obrázku 4.2, kde je zobrazena změna obejmu prodeje za použití různých platebních metod za rok 2014.

Tabulky sice nejsou oficiálním typem vizualizace, avšak se jedná o hojně využívané prostředky pro zobrazování dat v nástěnce. Pomocí tabulek můžeme zobrazit přesné měření, porovnávat je, seřadit data podle záhlaví tabulky a vytvořit součet dat. Na obrázku 4.3 jsou zobrazeny měsíční zisky za uplynulý rok 2014.

[1] K dispozici na: https://www.datapine.com/blog/wp-content/uploads/2015/05/number-chart-with-trend-indicator.png

[2] K dispozici na: https://www.datapine.com/blog/wp-content/uploads/2015/05/line-chart-amount-of-sales-by-payment-method.png



Obrázek 4.2: Ukázka spojnicového grafu[2]

Date	Revenue	ProductID
August 2015	€64.468,00	152
September 2015	€172.328,00	402
October 2015	€173.353,00	367
November 2015	€154.950,00	340
December 2015	€116.263,00	287
January 2016	€176.746,00	404
February 2016	€169.579,00	391
March 2016	€214.476,00	484
April 2016	€239.049,00	551
May 2016	€232.288,00	552
June 2016	€198.438,00	462
July 2016	€199.840,00	470
August 2016	€101.741,00	249
TOTAL	**€2.213.519,00**	**5.111**

Obrázek 4.3: Ukázka tabulky[3]

Pruhové grafy (anglicky *bar graphs*) se dělí na tři druhy: horizontální (anglicky *horizontal*), sloupcové (anglicky *column*) a skládané (anglicky *stacked*).

Horizontální grafy (anglicky *horizontal graphs*) jsou grafy určené nejlépe pro porovnávání a hodnocení, například nejlépe postavených firem nebo produktů na trhu. Na obrázku 4.4 jsou zobrazeny porovnání prodejů pěti nejlepších produktů.

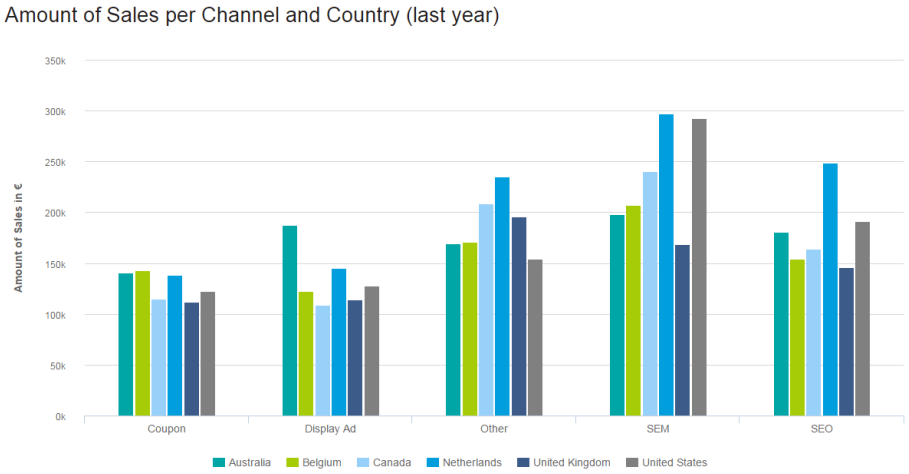
Sloupcové grafy (anglicky *column graphs*) znázorňují chronologické data, jako jsou například nárůsty za určité období, či různá porovnávání napříč kategoriemi. Na obrázku 4.5 je ukázka sloupcového grafu, který porovnává prodejní řetězce z pěti různých zemí[11].

[3]K dispozici na: https://www.datapine.com/blog/wp-content/uploads/2017/02/table-chart-datapine-1.png

[4]K dispozici na: https://www.datapine.com/blog/wp-content/uploads/2015/05/bar-chart-top-5-products-on-sales.png

[5]K dispozici na: https://www.datapine.com/blog/wp-content/uploads/2015/05/column-chart-amount-of-sales-by-country-and-channel.png

Obrázek 4.4: Ukázka sloupcového grafu[4]



Obrázek 4.5: Ukázka sloupcového grafu[5]

Kapitola 5

Požadavky

Cílem této práce je vytvořit nástěnku pro webovou stránku Red Hat One portal s vybranými byznysovými metrikami a klíčovými ukazately výkonnosti. Na začátku práce jsme specifikovali, že firma Red Hat bude mít zájem o projektové, procesní a byznysové metriky. Nakonec bylo rozhodnuto, že zaměření projektu se bude soustředit na byznysové metriky a ostatní metriky budou v nástěnce vynechány. I přesto, že se práce bude zaměřovat na byznysové metriky, obecná teorie o projektových a procesních metrikách byla sepsána do kapitoly 3, neboť ilustruje celkový pohled na metriky obecně a protože byla vypracována dříve než padlo rozhodnutí o vynechání zmíněných metrik. Webová stránka One portal je nástroj, který poskytuje informace o Pnt Devops týmech a technologiích, s kterými tyto týmy pracují uvnitř společnosti Red Hat. Jsou zde informace o projektech, na kterých týmy pracují, jaké nástroje vyvíjejí, dostupnost služeb o které se starají atd. Na obrázku 5.1 jsou zobrazeny metriky a KPIs (rozepsány níže), které jsou finálním návrhem pro zobrazení dat na nástěnce.

Požadované metriky:

- počet příchozích, otevřených a dokončených úkolů (Lab a System Operation týmů) za 90 dní (anglicky *number of incoming, open and closed tickets (Labs and System Operation teams) per 90 days*),
- aktuální počet členů PnT DevOps (anglicky *curent headcount per team*).

Požadované KPIs:

- úspěšnost odeslaní softwaru (*push success rate*)
- dostupnost systémů (*system uptime*)
- přesnost doručení softwaru na správné místo (*advisory delivery accuracy*)

Počet příchozích, otevřených a dokončených úkolů

Počet příchozích, otevřených a dokončených úkolů (tiketů) je metrika, která bude použita na dva pod týmy z týmu PnT DevOps, jelikož jen tyto dva týmy využívají software Service Now (je software určený k managmentu služeb[16]) pro správu úkolů. Tento graf bude zobrazovat tři stavy úkolů a to příchozí (definované podle data vytvoření), otevřený (definované podle absence data dokončení) a dokončený (definované podle data ukončení). Pomocí časové osy

bude zobrazen trend pro posledních 90 dní. Trend každého z již zmíněných týmů ukáže, jak efektivní jsou členové týmů v řešení úkolů, jestli je nutné přikročit k větší kontrole jejich práce.

Aktuální počet členů PnT DevOps

Metrika *aktuální počet členů PnT DevOps* bude zobrazovat aktuální rozdělení pracovních sil mezi týmy. Podle této metriky se manažer bude řídit pokud nastanou situace, kdy tým dostává velké množství práce od zainteresovaných stran nebo ostatních PnT DevOps týmů. Metrika mu nastíní situaci o rozložení sil mezi týmy a napomůže k rozhodnutí, jestli je potřeba najmou další pracovní síly.

Úspěšnost odeslání produktů

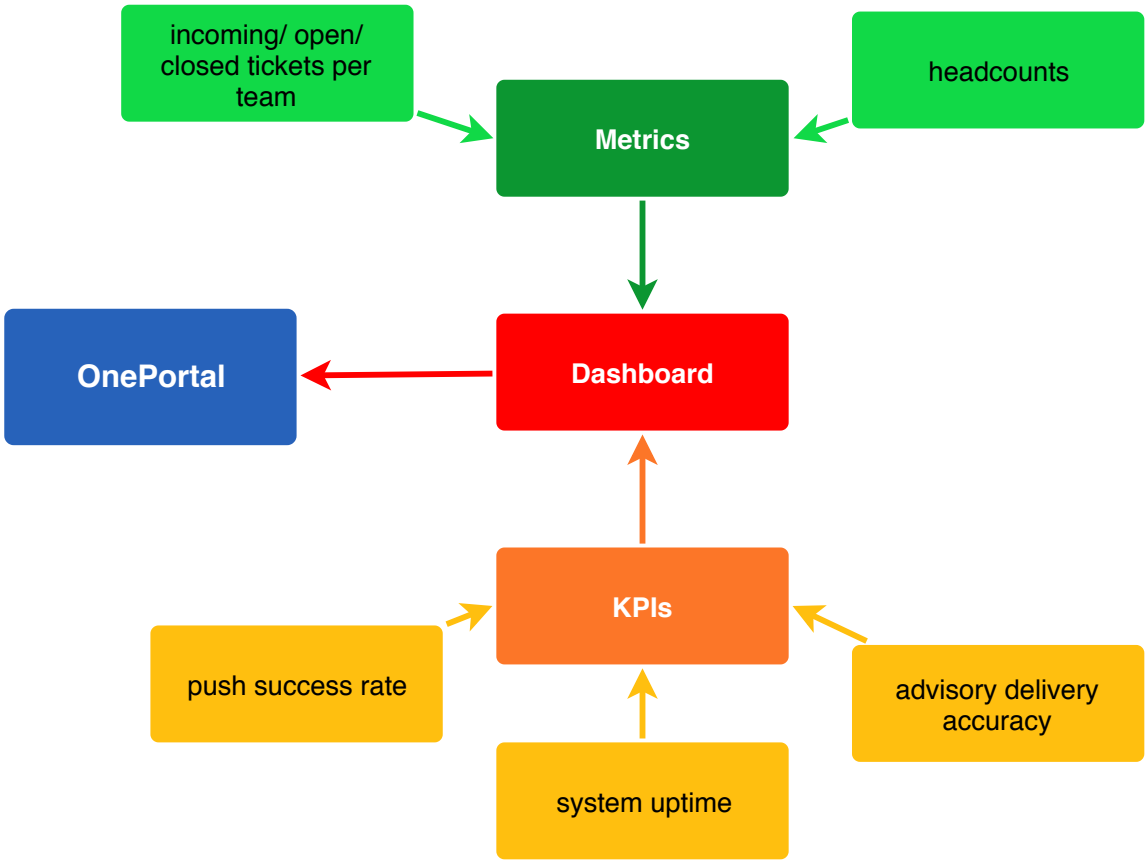
Úspěšnost odeslání produktů bude zobrazovat procentuální úspěšnost a neúspěšnost pokusu o odeslaní vydaného softwaru do Red Hat Network. Red Hat Network (dále zmíněno v textu jako RHN) je skupina systémově-spravovaných nástrojů, která umožňuje přístup uživatelům Red Hat softwaru k aktualizacím, seznamům změn, opravám chyb atd.[5]. Tento graf bude zobrazovat posledních 90 dní od aktuálního dne. Neúspěch pokusu o odeslání obsahu může nastat například pokud je systém nedostupný, pokud byly použity nesprávné příkazy k odeslání obsahu nebo když byl obsah odeslán do špatného adresáře a také pokud první nebo předchozí pokusy o odeslání stejného obsahu selhaly. Tento klíčový ukazatel výkonnosti by měl nastínit, jak spolehlivé jsou nástroje, které umožňují odeslání softwarového obsahu.

Dostupnost systému

Dostupnost systému bude zobrazovat monitorování jedenácti interních systémů, které jsou používané ke správě Red Hat softwaru. Pokud je některý z těchto systémů nedostupný mohlo by to ohrozit vydávání nového softwaru, opravy chyb a aktualizací, na které zákazníci čekají. Zobrazení hodnot grafu bude zabarveno podle stupnice 0% - 90% červenou barvou, 90% - 95% žlutou barvou a 95% - 100% barvou zelenou. Barevnost grafu napomůže odhalit spolehlivost systémů za posledních 90 dní.

Přesnost doručení softwaru na správné místo

Přesnost doručení softwaru na správné místo by měl poukázat s jak úspěšné je software doručen na správné místo určení a to do adresáře v RHN viz výše. Indikátor neúspěšnosti odeslaného softwaru může být nedostupnost obsahu softwaru nebo adresáře, nebo přepsání obsahu jiného adresáře. Tento graf bude zobrazovat data za posledních 90 dní.



Obrázek 5.1: Návrh metrik a KPIs na nástěnce

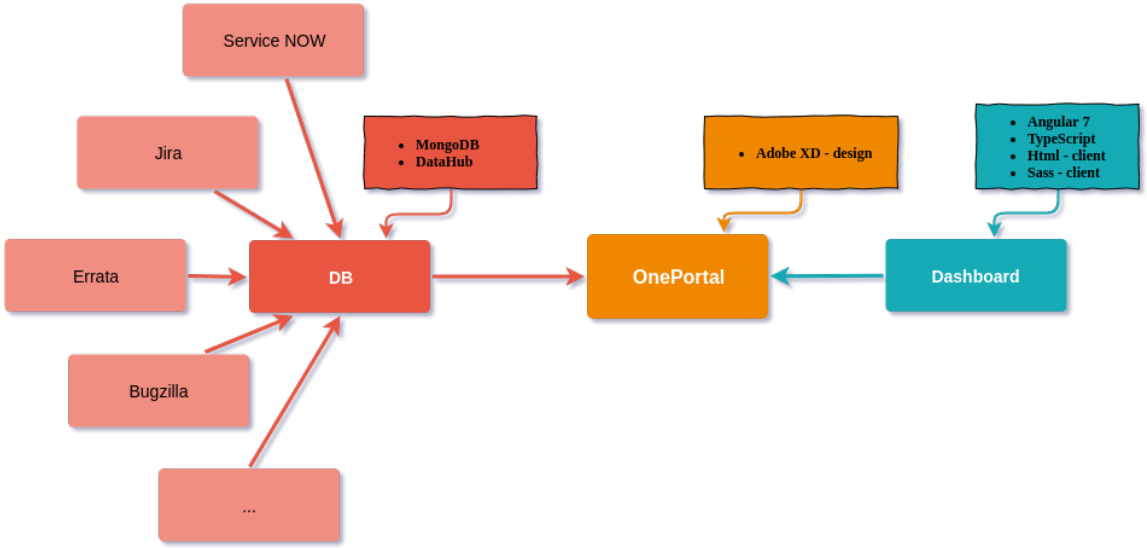
Kapitola 6

Návrh

Myšlenkou návrhu je mít centralizovanou nástěnku pro zobrazení byznysových metrik. Celá nástěnka je součástí projektu *One portal*, který má sdružovat všechny data týmu PnT DevOps na jednom místě a také je srozumitelně reprezentovat návštěvníkům webových stránek. Grafika celé nástěnky následuje rady a doporučení, které jsou součástí projektu *One Portal*. Pro potřebu nástěnky jsou přebrány vytvořené grafické návrhy a objekty, které se musí vyskytovat na všech webových stránkách patřící pod projekt One portal. Grafické návrhy jednotlivých grafů a tabulek jsou součástí této práce a budou později začleněny do projektu *One portal*. Zdrojový kód nástěnky je produktem diplomové práce, bude začleněn a zaveden do produkce, jako funkční celek projektu.

Zmíněný projekt, *One Portal*, využívá softwarový systém Node.js pro psaní webového serveru a společně s technologií GraphQL (dotazovací jazyk pro API, který běží na straně serveru a vykonává dotazy definované pomocí systémových typů[14]), Angular 7 (platforma pro vytváření webových nebo mobilních aplikací), TypeScriptem, HTML a Sass, vytváří prostředí pro fungování aplikace na bázi server-klient. Server se stará o získávání dat z databází a API a dává tato data k dispozici klientovi, který data dále zpracovává. Klient zobrazuje design webových stránek pomocí HTML, Sass a TypeScriptu a také zpracovává a propaguje data ze serveru.

Nástěnka získává data z nástrojů používaných týmem PnT DevOps a je přístupná pro všechny zaměstnance Red Hat. Na obrázku 6.1 je zobrazeno z jakých nástrojů jsou nebo budou sbírána data pro vytvoření metrik. Například to jsou projektové nástroje Jira a Confluence, z monitorovacích nástrojů jsou to Zabbix nebo Nagios nebo nástroje Errata a Bugzilla zmíněné v sekci 2.3. Některá potřebná data už jsou již sbírána týmem PnT DevOps a uložena v databázích Elasticsearch, MongoDB, DataHub nebo v LightBlue. Další potřebná data budou uložena v databázi MongoDB a za pomocí GraphQL budou získávána pro zpracování klientem. Druhým přístupem pro získání dat je pomocí API (rozhraní pro programování aplikací, anglicky *Application Programming Interface*). Přes API budeme získávat živá data, která budou zpracována na straně serveru a použita na straně klienta pro vytvoření vizualizace grafu na „One Portal“ stránce. Pro každou metriku a KPI bude nejprve vytvořen design v *Adobe XD* softwaru, který bude součástí designu celé nástěnky. Poté bude každá metrika a KPI implementována jako samostatný modul kvůli tomu, že data pro každý graf budou sbírána z různých zdrojů a tento přístup ulehčí rychlost načítání celé stránky s nástěnkou.



Obrázek 6.1: Návrh architektury

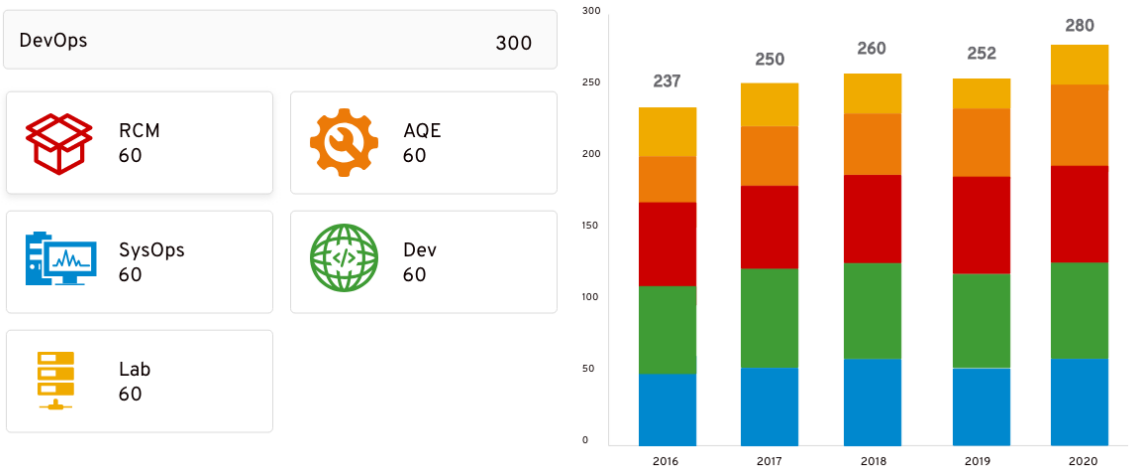
Jak již bylo zmíněno v kapitole 5, nástěnka bude obsahovat dvě metriky a tři klíčové ukazatele výkonnosti. Pro vybrání vhodného grafu pro metriky a KPIs, a správnou reprezentaci jejich dat byla použita příručka ze článku *Data Visualization Infographic: How to Make Charts and Graphs [1]*. V příručce jsou zmíněny čtyři přístupy návrhu, a to: vztah proměnných (*relationship*), porovnání (*comparison*), distribuce (*distribution*) a kompozice (*composition*) dat. Pro každou z metrik a klíčových ukazatelů výkonnosti (KPI) byla konzultována správná vizualizace a reprezentace dat, tak aby nejlépe vystihla potřebu managementu, trendy z minulých období a napomohla budoucímu vylepšení systémů, procesů atd..

6.1 Návrhy grafů

Aktuální počet členů týmu PnT DevOps

První metrikou je *aktuální počet členů v týmu PnT Devops*, pro kterou byly vybrány dvě vizualizace, a to vizualizace pomocí porovnání (*comparison*) sloupcovým grafem a číselný graf, který zviditelní aktuální počet členů v týmu. Na obrázku 6.2 je zobrazeno pět týmů, které tvoří tým PnT DevOps, a celkový počet členů v PnT DevOps. Na levé straně jsou zobrazeny číselné grafy, kde jsou specifikovány názvy týmů, počet členů týmů a barvy, kterými budou týmy reprezentovány ve sloupcovém grafu. Na straně pravé se nachází sloupcový graf, který poukazuje na rozdíly rozložení týmů. Data pro tuto metriku jsou sbírána za pomoci `ldapjs`[1], který získává data o uživatelích z adresářového serveru.

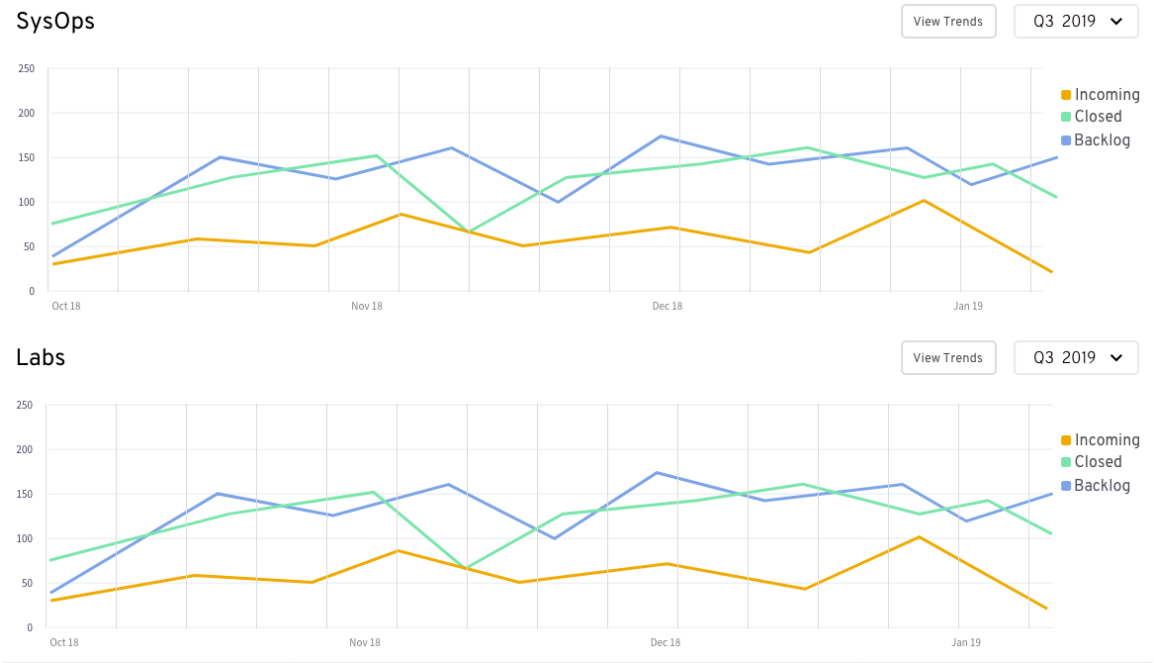
[1]K dispozici na: `http://ldapjs.org/`



Obrázek 6.2: Aktuální počet lidi v PnT DevOps

Počet příchozích, otevřených a dokončených úkolů

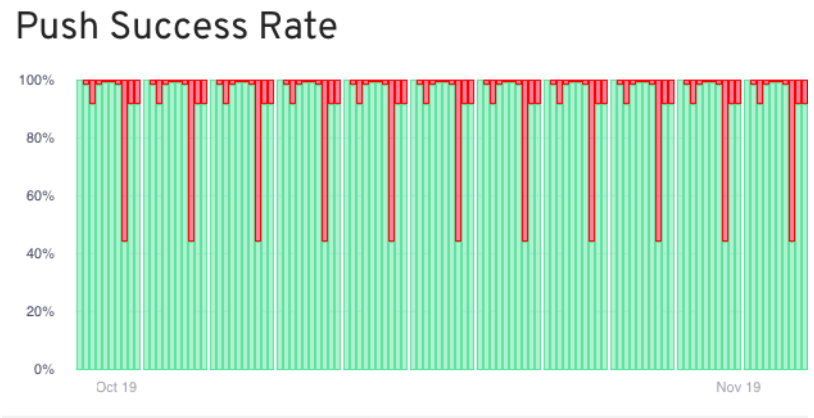
Další metrikou je *počet příchozích (incoming), otevřených (open) a dokončených (closed) úkolů (tiketů)*, které reprezentují porovnání stavů úkolů a jejích vývoj v předchozích 90 dnech. Počet příchozích a počet dokončených úkolů jsou nezávislá data, avšak počet otevřených úkolů závisí na tom, kolik v daný den bylo příchozích úkolů a kolik úkolů bylo dokončeno. Data pro tuto metriku jsou sbírána z nástroje Service Now, ze kterého jsou pomocí API (*application programming interface*) dotazu získávána data o týmech. Na obrázku 6.3 lze vidět liniový graf, který se skládá ze tří linek, které znázorňují aktuální data pro dva týmy PnT DevOps a to Pro System Operations tým a tým Labs. Jak už bylo zmíněno v předešlé kapitole návrh 6 tato metrika se zaměřuje pouze na tyto dva zmíněné týmy.



Obrázek 6.3: Počet příchozích, otevřených a dokončených úkolů

Úspěšnost odeslání produktů

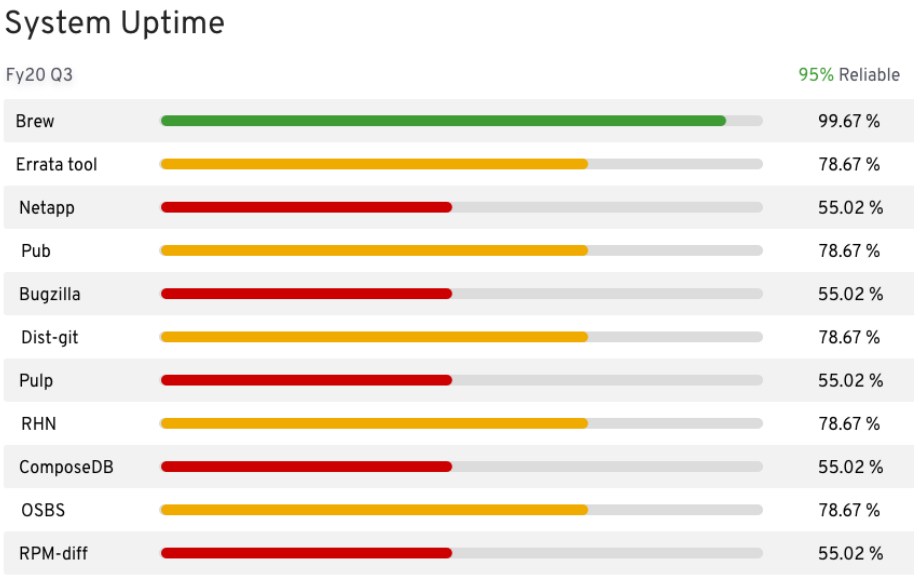
Velice důležitou metrikou pro úspěšnost a dobré přijetí produktů firmy Red Hat je úspěšnost odesílání produktů do nástroje RHN, zmíněného v předešlé kapitole. Data o této metrice jsou uložena v databázi DataHub a za pomocí přesně specifikovaného dotazu, jsou sbírána na serverové straně a zpřístupněna na url adrese. Tato url adresa je zpřístupněna pro klienta, který data zpracovává do formátu, který je kompozicí dat. Nejlepší volbou pro vizuální zobrazení je skládaný vertikální graf (*stacked vertical chart*). Na obrázku 6.4 je zobrazen návrh, jakým způsobem by data mohla být zobrazena. Skládaný graf je tvořen, z úspěchů (barva zelená) a neúspěchů (barva červená), které jsou měřeny každý den po dobu posledních 90 dní.



Obrázek 6.4: Úspěšnost odeslání produktů

Dostupnost systému

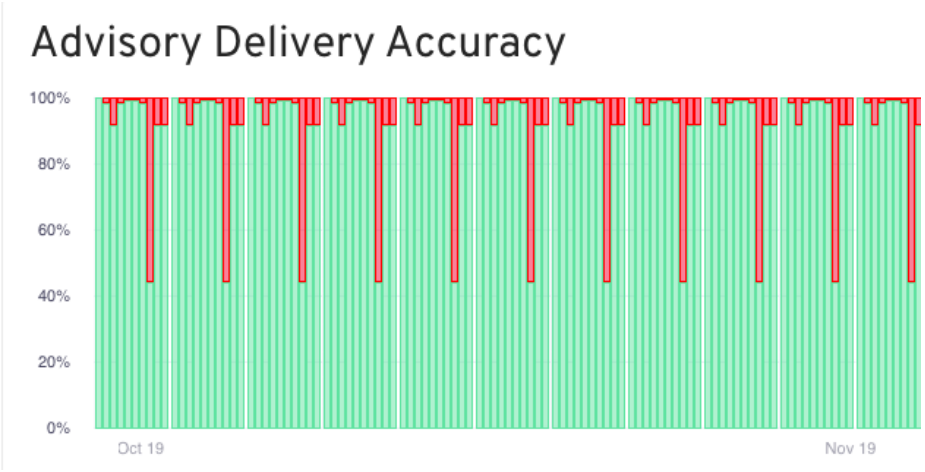
Klíčový ukazatel výkonnosti *dostupnost systému*, bude reprezentován horizontálním grafem a tabulkou s procentuálními hodnotami, která bude umístěna za grafem. Data pro metriku jsou sbírána z databáze DataHub, do které je zaslán dotaz s přesně specifikovaným časovým rámcem. Databáze vrátí odpověď s hodnotami, které jsou za posledních 90 dní. Na obrázku 6.5 je zobrazena barevnost grafu, při různých procentuálních hodnotách. Jak už bylo zmíněno v předešlé kapitole, data v grafu nabývají třech barev, a to zelené, žluté a červené. Tyto barvy určují, jak dostupné (spolehlivé) jsou systémy. Aby byla zajištěna dobrá viditelnost a srozumitelnost grafu, je barevná rozlišitelnost hodnot podpořena tabulkou, která se nachází těsně za grafem a zobrazuje hodnoty na přesnost dvou desetinných čísel.



Obrázek 6.5: Dostupnost systému

Přesnost doručení softwaru na správné místo

Tato metrika bude velice podobná zobrazení, ne-li stejná jako zobrazení metriky *úspěšnost odeslání produktů*. Pro získání dat je zaslán dotaz do databáze Datahub, kde jsou data uložena. Odpověď na dotaz do databáze, je jasně strukturovaná ve formátu JSON. Následně je vytvořen graf této metriky, opět skládaný graf, nabývající procentuálních hodnot, které zobrazují buď úspěch (zelená barva), nebo neúspěch (barva červená). Tato jednoduchost je potřebná pro zobrazení všech 90 dní v rámci jednoho grafu. Na obrázku 6.6 je zobrazen návrh, jak bude daný graf vypadat.



Obrázek 6.6: Přesnost doručení softwaru na správné místo

Kapitola 7

Implementace

Tato kapitola se zabývá implementací serveru, klienta, komunikací s databází MongoDB a vytvořením vizualizace dat. Nejprve bude specifikováno, jakým způsobem jsou získávána data na straně serveru, jak je s daty zacházeno a kde všude je možné data uložit. Následně bude uvedeno, jak klient získává svá data a jak tato data převádí do různých podob vizualizace. Pro každý graf nástěnky je vyčleněna podkapitola, která shrne, které data jsou pro tento graf potřeba, kde byla data získána, kde a jak jsou data uložena a proč. Dále bude též popsáno, jak tato data správně vizualizovat, aby sloužila správnému účelu, jelikož tabulka s daty není vždy vhodným způsobem vizualizace. K vizualizaci dat je použita knihovna `ngx-charts`[1], která nabízí využití okolo dvaceti typů grafů a umožňuje jejich úpravu, tak aby se co nejvíce přiblížila k předem vytvořenému návrhu.

7.1 Server

Server se zabývá získáváním dat, jedná se o data *živá* získaná pomocí API dotazu, či data pocházející z ostatních databází. Pokud jsou živá data sbírána v malém množství, tak je server dále naváže (anglicky `bind`) na webovou stránku, odkud si tato data klient získá. Pokud je nutno získávat větší množství dat, získávat je opakovaně, nebo si ukládat historii změn, použíje se správce úloh Cron, který automaticky spouští funkce pro získávání dat a zapisování je do databáze MongoDB. Všechny úkony, které jsou na serveru vykonány, jsou spouštěny ze souboru `server.ts`. V tomto souboru jsou volány objekty a funkce, které získávají data pro další propagaci směrem ke klientovi.

Uložení dat

Server ukládá data dvěma způsoby. Prvním způsoben je navázání (anglicky *binding*) dočasných dat na webovou stránku a druhým je ukládání dat do databáze MongoDB. MongoDB[2] je databáze založená na bázi dokumentů, které mají strukturu podobnou jazyku JSON.

Nejprve bude specifikován způsob navázání dat na webovou stránku. Na obrázku 7.1, je uveden příklad funkce `getPntStats`, která je navázána pomocí třídy Router z Angular knihovny `express` na adresu `/datahub/pnt-stats`. Nejprve se vykoná funkce `getPntStats`, pomocí které se server dotáže na data pro metriky. Tato data jsou v samotné funkci zpracována tak, aby ze serveru na klienta nebylo posíláno zbytečné množství nevyužitelných dat. Následně jsou data předána přes specifikovanou adresu (například /datahub/pnt-stats).

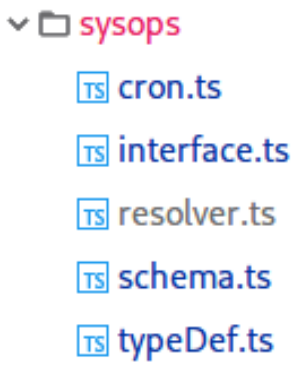
[1] Dokumentace na: https://swimlane.gitbook.io/ngx-charts/
[2] Získáno z: https://www.mongodb.com/what-is-mongodb

Poté má klient k dispozici adresu, odkud tato data získává. Výhodou tohoto přístupu je, že není nutno udržovat databázi, protože data na serveru zůstanou po dobu než se na ně klient dotáže. Avšak nevýhodou tohoto přístupu je, že se data načítají znovu při každém obnovení webové stránky.

```
constructor(router: Router) {
  this.router = router;
  // Binding the functions on class instance for all APIs
  this.router.get( path: "/datahub/pnt-stats", this.getPntStats.bind(this));
}
```

Obrázek 7.1: Navázání (binding) dat na webovou adresu

Pro ukládání dat do databáze používá tým *One Portal* strukturu adresáře, která napomáhá k jednoduchému pochopení, jak správně definovat a ukládat data. Na obrázku 7.2 lze vidět, které soubory jsou v adresáři `sysops` umístěny. Prvním souborem je `cron.ts`, ve kterém je uložena implementace získání a uložení dat. Definice datových typů, které jsou nezbytné pro uložení do databáze, se vyskytují v souborech `typeDef.ts`, `interface.ts` a `schema.ts`. Posledním souborem vyskytující se v adresáři je `resolver.ts`, kde jsou implementovány funkce, jež naplní data podle definice a zpřístupní je pro zobrazení na webové stránce serveru pomocí jazyku GraphQL nebo pro práci na straně klienta.



Obrázek 7.2: Struktura adresáře

Nejprve je vytvořena definice typů dat, které jsou uloženy v souboru `typeDef.ts`. Tyto definice jsou uspořádány do rozhraní v souboru `interface.ts` a jsou brány jako jeden dokument v databázi. Tyto dokumenty jsou ukládány do dynamického databázového schéma, které je vytvořeno z definice v souboru `schema.ts`. Aby bylo možné vytvořit dokument, nejprve je nutné v souboru `cron.ts` vytvořit funkce, které tato data získají a poté tyto funkce pomocí Cron funkce vykonat. Cron funkce (příklad zobrazen na obrázku 7.3) je umístěna v souboru `server.ts`, odkud jsou vykonávány všechny funkce pro sběr dat. Jakmile je Cron funkce spuštěna, jsou data uložena do databáze podle již dříve zmíněných definic typů a rozhraní. Jelikož je nutné nejen data uložit, ale také je z databáze získat, jsou v souboru `resolver.ts` implementovány funkce pro získání dat z databáze, které jsou později použity na straně klienta. Pro práci s databází je užitečná knihovna `mongoose`, která je vytvořena pro potřeby komunikace serveru `Node.js` s databází MongoDB.

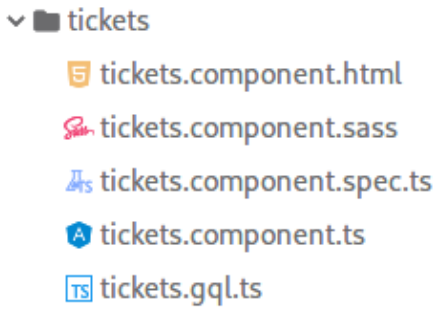
```
// Cron to Sync Labs Tickets with Service Now
schedule.scheduleJob( rule: "10 2 * * *", callback: function () {
  console.log("Fetching Data from Service Now for Labs tickets");
  const labsCron = new LabsCron(socket);
  labsCron.getTickets( numberOfDaysToLoad: 1);
});
```

Obrázek 7.3: Cron fuknce

7.2 Klient

Klient je hlavním tvůrce vizualizace webových stánek a k tomu se zde používá jazyk TypeScript, HTML a Sass. Nejen, že klient vytváří, jak bude daná webová stránka vypadat, ale také musí komunikovat se stranou serveru, kde získává data, která jsou následně na straně klienta přeměněna na vizualizace grafů nebo tabulek, ze kterých je tvořena nástěnka. Nástěnka, uložena v adresáři `src/app/modules/dashboard/business-operations`, má v projektu *One Portal* přesně danou strukturu. Každá metrika a KPI má svůj vlastní adresář a v něm čtyři nebo pět souborů. Na obrázku 7.4 je zobrazen příklad struktury adresáře. Adresář se skládá z HTML souboru, souboru `*.sass` pro vizuální úpravu stránky, souboru pro testování (`*.spec.ts`), souboru `*.component.ts`, kde se získávají a modifikují data, aby odpovídaly formátu, který je potřeba pro grafy. V některých případech se může vyskytovat soubor `*.gql.ts`, ve kterém je definována struktura dotazu na data umístěná v databázi MongoDB.

Komunikace se serverem je umístěna v souborech typu `service` (například soubor `dashboard.service.ts`). Tento soubor je důležitý, jelikož jsou v něm volány funkce pro komunikaci se serverem. Níže bude specifikováno, jakým způsobem se získávají data ze serveru.



Obrázek 7.4: Adresář metriky

Získání dat z databáze nebo webové adresy

Pro získání dat ze serveru je nutné znát jak server data ukládá. Prvním způsobem je zpřístupnění malého objemu dat pomocí webové adresy, například jako `/api/data`. Klient pak komunikuje se serverem pomocí knihovny HttpClient[3] a také pomocí objektu `Observable` z knihovny `RxJS`[4]. Pomocí již zmíněné knihovny HttpClient je vytvořen API dotaz na ad-

[3]Dokumentace HttPClient knihovny: https://angular.io/guide/http
[4]Dokumentace RxJS knihovny: https://angular.io/guide/rx-library

resu (např. /api/data). Za použití objektu `Observable` klient odposlouchává a čeká na odpověď od serveru v předem definované formě (např. ve formě JSON slovníku (anglicky *JSON dictionary*, dále v textu používáno v anglické verzi)) nebo na chybovou hlášku (`err`). Na obrázku 7.5 je reprezentována forma funkce, která komunikuje se serverem pro získání dat. Funkce `getPntStats` používá objekt `Observable`, který poslouchá na určené adrese (`this.path + pnt-stats`). Pokud klient v kódu zavolá funkci `getPntStats.subscribe()`, dostává data, která mu server odesílá.

```
getPntStats(): Observable<any> {
  return this.http.get<any>( url: this.path + 'pnt-stats', httpOptions);
}
```

Obrázek 7.5: Získání dat z adresy

Druhý způsob získání dat z databáze je použitím knihovny Apollo[5] (GraphQL klient), která je navržena pro získání dat z GraphQL databáze umístěné na serveru. Na obrázku 7.6 je zobrazena funkce `getTeamMembers` pro získání dat z databáze. Tato funkce vytváří dotaz (anglicky `query`) do databáze s hodnotou `getTeamMembersByID`. Klient čeká do té doby, než dostane odpověď z databáze, a nebo chybu (`err`). Struktura dat odpovědi databáze je přesně určena podle definice v souboru `headcount.gql.ts`.

```
getTeamMembers(id: string): Promise<any> {
  return this.apollo
    .watchQuery<any>( options: { query: getTeamMembersByID, variables: { _id: id } })
    .result()
    .then( onfulfilled: response => _.cloneDeep(response.data.getTeamMembersByID))
    .catch( onrejected: err => err);
}
```

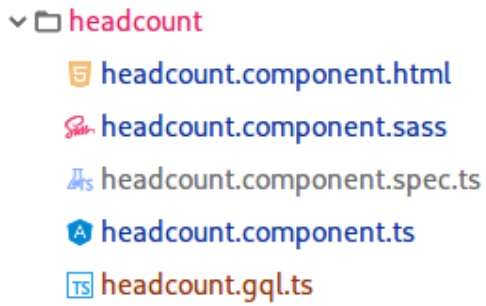
Obrázek 7.6: Získání dat z databáze

7.3 Aktuální počet členů týmu PnT DevOps

Data pro tuto metriku jsou již udržována v databázi týmem *One portal*, a tudíž není nutné implementovat jejich získání. Data v databázi MongoDB jsou získána z LDAP serveru, kde se nachází informace o zaměstnancích Red Hatu. Za předpokladu, že jsou data uložena v databázi, získáme data na straně klienta podle dříve definovaného postupu v části *Získání dat z databáze nebo webové adresy*. Struktura adresáře pro tuto metriku je zobrazena na obrázku 7.7. Nejdříve je definována struktura dat v souboru `headcount.gql.ts`, která je známá ze souboru na serveru, kde byla tato struktura vytvořena. V souboru `headcount.component.ts` je uložena implementace zpracování dat z databáze. Nejprve jsou zjištěny identifikátory týmu, které budou později použity pro získání počtu členů podtýmů PnT DevOps. Z důvodu asynchronního získávání dat z databáze se může stát, že se graf začne vykreslovat dříve než je potřeba. Této skutečnosti je zabráněno vytvořením funkce `getStats`, která zapouzdří volání asynchronní funkce `getTeamMembers` umístěné v souboru `dashboard.service.ts`, vyčká, až se všechny asynchronní funkce dokončí, a poté vrátí výsledek všech týmu dohromady. Funkce `getTeamMembers` zjistí podle argumentu "id"počet členů týmu. Výsledky z funkce `getStats` jsou dále modifikovány a uloženy do dvou polí

[5]Dokumentace Apollo knihovny: https://www.apollographql.com/docs/angular/

objektů `headcountStats` a `chartValues`. Pole `chartValues` udržuje strukturu pro použití vertikálního grafu a pole `headcountStats` definuje data pro vizualizaci číselných grafů. Definice grafů je umístěna v souboru headcount.component.html, kde je specifikován název *PnT DevOps Associates*. V tomto souboru je vytvořeno pět číselných a jeden vertikální graf. Vertikální graf je uložen a popsán pomcí `svg` formátu na obrázku 7.8. Číselné grafy obsahují ikonu týmu, název týmu a počet členů týmu.



Obrázek 7.7: Aktuální počet členů týmu PnT DevOps - struktura adresáře

```
<a class="chart">
  <ngx-charts-bar-vertical
          [view]="headcountChart.view"
          [results]="chartValues"
          [yAxis]="true"
          [legend]="false"
          [scheme]="headcountChart.colorScheme"
          [barPadding]="50"
  ></ngx-charts-bar-vertical>
</a>
```

Obrázek 7.8: Aktuální počet členů týmu PnT DevOps - definice vertikálního grafu

7.4 Počet příchozích, otevřených a dokončených úkolů

Implementace získání dat je umístěna ve struktuře adresáře `sysops` (příklad na obrázku 7.3) a `labs`. Tyto dva adresáře jsou vytvořeny z nutnosti uložení dat do databáze a pravidelného sbírání dat, protože načítání většího objemu dat zpomaluje načítání nástěnky. Nejprve je vytvořen soubor `typeDef.ts`, kde je určena struktura jednoho záznamu. Záznam je určen podle data, počtu příchozích, otevřených a dokončených úkolů (tickets) v daný den. Následně jsou v tomtéž souboru definovány dotazy (queries) a metody, které umožňují zobrazovat a upravovat data na serveru pomocí jazyka GraphQL.

V souboru `interface.ts` je uvedena struktura úkolu, která se následně využívá pro účely databáze. Pro vytvoření záznamu (dokumentu) v databázi MongoDB se používá soubor `schema.ts`, kde se vyskytuje objekt `Schema`, který je naplněn daty ze Service Now a následně uložen do databáze. V souboru `resolver.ts` jsou připraveny funkce pro další práci s databází. Nejdůležitějším souborem v tomto adresáři je soubor `cron.ts`, který implementuje získání dat. Funkce `getTickets` s argumentem `numberOfDaysToLoad`, jež specifikuje počet dní, používá třídu SnowApiHelper sbírající data z nástroje Service Now. Nejprve je vygenerováno pole dat (například ve tvaru 2019-04-01) podle argumentu `numberOfDaysToLoad`,

které je potřeba pro následné vytvoření API požadavku pro službu Service Now. Následně jsou pro každé datum z pole dat provedeny asynchronní funkce pro získání příchozích, otevřených a dokončených úkolů. Každá asynchronní funkce používá pomocnou metodu `getTicket`, z třídy `SnowApiHelper`, která vytváří požadavek o data ze Sevice Now. Po dokončení všech asynchronních požadavků jsou data navrácena funkcí `callback`, která následně uloží data do struktury úkolů `ISysOpsTickets` ze souboru `interface.ts`. Poté je tato struktura použita pro vytvoření nového objektu `new SysOpsStat`, kde objekt reprezentuje jeden záznam (dokument) v databázi. Zmíněný objekt je uložen pomocí metody `save` (`newSysOpsStat.save()`) do databáze MongoDB. Takto probíhá cyklus vytváření a ukládání záznamů den po dni pro počet dní, které byly specifikovány z argumentu `numberOfDaysToLoad`. Spuštění funkce pro získání a uložení dat do databáze je definováno v souboru `server.ts` umístěném v serverové části implementace.

Pro získání dat z databáze se na straně klienta používájí soubory specifikované názvem `service` (například `dashboard.service.ts`). V souboru `dashboard.service.ts` jsou data z databáze získána funkcí (například `getSysOpsTickets`), která pracuje s knihovnou `apollo-angular`. Metoda `watchQuery`[6], knihovny apollo-angular, zašle dotaz do databáze s argumentem `query`, který specifikuje v jaké formě jsou data získána (to vše je specifikováno v souboru `tickets.gql.ts`). Jak již bylo zmíněno výše, každá metrika a KPI má svou adresářovou strukturu. Pro tuto metriku je použit adresář `tickets`, který obsahuje soubory na obrázku 6.3. V souboru `tickets.component.ts` jsou zavolány metody `getSysOpsTickets` a `getLabsTickets` ze souboru `dashboard.service.ts`, které získají data z databáze. Každá s těchto metod nejprve získá data, která následně transformuje na pole objektů. Pole objektů, které udržuje data o úkolech (tickets), se nazývá `sIncoming` (pole příchozích úkolů), druhé pole `sOpen` (pole otevřených úkolů) a třetí pole `sClosed`. Každý objekt v těchto polích obsahuje klíč (key) `name`, ve kterém je uloženo datum, a klíč `value`, který udržuje počet úkolů (tickets). Po vytvoření polí objektů následuje funkce `updateListWithDates` pro hledání a doplnění chybějících klíčů `name` (což obsahuje datum) a jejich `value` hodnotou nula. Jakmile jsou doplněny pole o chybějící objekty, funkce `sortByDate` seřadí chronologicky všechny objekty v jednotlivých polí s daty. Doplnění objektů do polí zabraňuje nestandardnímu chování grafů. Před vytvořením konečné struktury dat (zobrazeno na obrázku 7.9) je nutné vybrat data jen z posledních 90 dní pomocí funkce `getQuarterList`. Poté už jsou všechny výsledky připravené pro zobrazení v grafu.

[6]Dokumentace watchQeury metody https://www.apollographql.com/docs/angular/basics/queries/

```
[
  {
    "name": "Germany",
    "series": [
      {
        "name": "2010",
        "value": 7300000
      },
      {
        "name": "2011",
        "value": 8940000
      }
    ]
  },

  {
    "name": "USA",
    "series": [
      {
        "name": "2010",
        "value": 7870000
      },
      {
        "name": "2011",
        "value": 8270000
      }
    ]
  }
]
```

Obrázek 7.9: Datová struktura liniového grafu [7]

Pro vizualizaci metriky je vytvořena struktura v souboru `tickets.component.html`, která určuje, jak bude část nástěnky s metrikou vypadat. Html soubor obsahuje název *Quarterly Average Tickets Interaction* a dva grafy, které určují metriky týmu SysOps a Labs. Graf metriky je obrázek formátu `svg` implementovaný knihovnou `ngx-charts`. Příklad implementace grafu je zobrazen na obrázku 7.10. Každý graf je definovaný argumenty `results`, ve kterém jsou specifikovaná data grafu, a `view`, které určuje velikost grafu podle rozměrů výšky a šířky. Pro zobrazení posisu os `x` a `y` jsou použity argumenty `xAxis` a `yAxis`, které jsou nastaveny na hodnotu "true". Argumentem `scheme` jsou určeny barvy, které budou použity na vytvoření linek grafu.

[7] Dostupné na: https://swimlane.gitbook.io/ngx-charts/examples/line-area-charts/line-chart#data-format

```
<a class="ticket-chart">
  <div class="ticket-team">SysOps</div>
  <ngx-charts-line-chart
          [view]="ticketColors.view"
          [results]="sysOpsStats"
          [xAxis]="true"
          [yAxis]="true"
          [scheme]="ticketColors.colorScheme"
          [legend]="true"
          [legendTitle]="'Tickets Per 90 days'"
  ></ngx-charts-line-chart>
</a>
```

Obrázek 7.10: Definice grafu

7.5 Úspěšnost odeslání produktů

Nejprve je serverem poslán požadavek na data na adresu DataHubu (`https://elasticsearch.datahub.redhat.com:30200`) s blíže specifikovanou adresou (`/pnt-devops-mars-pub-*/_search`), která upřesní, kde jsou data pro tento ukazatel uložena. Požadavek na data je upřesněn tělem (anglicky `body`), kde je specifikován časový rámec 90 dní a upřesnění, kde budou data hledána. Na obrázku 7.11 je část těla, které specifikuje v rozmezí, `range`, posledních 90 dní a v `query_string` podmínky odkud jsou vybrána data. V další části těla, zobrazené na obrázku 7.12, je specifikováno filtrování jen těch dat, které jsou potřebná pro graf úspěchů (*Success*) a neúspěchů (*Failed*). Odpověď získaná z DataHub je převedena do JSON slovníku (v jazyce TypeScript je JSON slovník nazýván objektem) a uložena do proměnné `pushSuccessRate`, která je součástí pole objektů `result`, jež udržuje všechny proměnné získané z DataHub. Proměnná `result` je přístupná klientovi, jelikož je součástí funkce navázané na webovou adresu, zmíněnou v sekci *Uložení dat*. Tímto navázáním je dokončeno získávání dat na serveru.

```
"query": {
  "bool": {
    "must": [
      {
        "query_string": {
          "query": "_type: push AND e2e_push: false AND (method:PushAdvisory OR
           method:PushStaged OR method:PushDocker OR method:PublishContentIsos)",
          "analyze_wildcard": true
        }
      },
      {
        "range": {
          "@timestamp": {
            "gte": ${moment(new Date()).subtract( amount: 90,  unit: "days").valueOf()},
            "lte": ${moment(new Date()).valueOf()},
            "format": "epoch_millis"
          }
        }
      }
    ],
    "must_not": []
  }
},
```

Obrázek 7.11: Úspěšnost odeslání produktů - Filtrování výsledků

Obrázek 7.12: Úspěšnost odeslání produktů - struktura výsledků

Jak již bylo zmíněno v části *Klient*, pro komunikaci se serverem je použit soubor `*.service.ts`, ve kterém jsou implementovány funkce pro komunikaci se serverem. Pro případ tohoto KPI je použita již implementovaná funkce `getPntStats` ze souboru `pnt.servis.ts`. Funkce `getPntStats` získává data (úspěšnosti odeslání produktu) předávanou webovou adresou `pnt-stat`. V souboru `push-success-rate.component.ts` je zavolána funkce `getPntStats` s metodou `subscribe`, která čeká až budou ze serveru předány všechny data. Poté, co klient obdrží data, jejíž struktura je zobrazena na obrázku 7.12, data jsou klientem zpracována tak, aby klient obdržel statistiky o úspěchu a neúspěchu za posledních 90 dní. Následně je vytvořena struktura dat pro graf zobrazena na obrázku 7.13. Klíč `name` obsahuje datum KPI a v `series` jsou specifikovány procentuální hodnoty úspěchu(Success) a neúspěchu (Failure). Tato struktura dat je povinná pro zobrazení dat skládaným vertikálním grafem (stacked vertical chart). Grafická implementace zobrazení grafu je umístěna v souboru HTML, kde je specifikován název grafu (Push success rate), odkaz na graf přístupný v nástroji Kibana[8] a samotný graf určený v `svg` formátu. Na obrázku 7.14 je definováno, jak bude graf vypadat. V dokumentaci [9] je vysvětleno, jak argumenty pro tento graf používat.

[8]Dostupné na: https://www.elastic.co/products/kibana

[9]Dostupné na: https://swimlane.gitbook.io/ngx-charts/examples/bar-charts/stacked-vertical-bar-chart

```
return {
  'name': timestamp,
  'series': [
    {
      'name': 'Success',
      'value': pushSuccessRateSuccessPercent

    },
    {
      'name': 'Failure',
      'value':  pushSuccessRateFailurePercent
    }
```

Obrázek 7.13: Úspěšnost odeslání produktů - struktura dat pro graf

```
<a class="push-chart">
  <ngx-charts-bar-vertical-stacked
          [view]="pushColors.view"
          [animations]="animations"
          [results]="pushStats"
          [xAxis]="true"
          [yAxis]="true"
          [scheme]="pushColors.colorScheme"
          [maxYAxisTickLength]="pushColors.maxYAxisTickLength"
          [maxXAxisTickLength]="pushColors.maxXAxisTickLength"
          [barPadding]="pushColors.barPadding"
          [legend]="true"
          [trimXAxisTicks]="true"
          [trimYAxisTicks]="true"
          [legendTitle]="'Per 90 days'"
  ></ngx-charts-bar-vertical-stacked>
</a>
```

Obrázek 7.14: Úspěšnost odeslání produktů - definice skládaného vertikálního grafu

7.6 Přesnost doručení softwaru na správné místo

Tento klíčový ukazatel výkonnosti má velice podobnou implementaci, jako výše uvedený ukazatel úspěšnosti odeslání produktů. Data jsou (stejně jako výše) sbírána z DataHub, ale jejich zdroj je definovaná adresou `/pnt-devops-mars-errata-*/_search`. Požadavek, kterým jsou data získána z DataHub, je na stejném principu jako na obrázku 7.11. Rozdíl oproti předešlému KPI je ve `query_string`, kde je specifikován dotaz (query) jako `NOT product_id:116`. Odpověď na požadavek má téměř stejnou strukturu výsledku, jako je zmíněno na obrázku 7.12. Tento výsledek je však uložen v proměnné `advisoryDeliveryAccuracy`, která také je součástí pole `results`. Pro přesné definice dotazů je možné nahlédnout do jejich implementace v souboru `datahub.ts` pod názvem *Advisory delivery accuracy*.

Klient komunikuje se serverem stejným způsobem jako v případě výše. Struktura souboru pro práci s daty na straně klienta je pojmenována jako `advisory-delivery-accuracy`. Získání dat a práce s daty na stráně klienta je obdobná, jelikož je také jako v předešlém řešení nutné získat procentuální hodnoty úspěchu a neúspěchu za posledních 90 dní. Pro zobrazení data grafem je vytvořena struktura odpovídající definici skládaného vertikál-

ního grafu.V souboru `advisory-delivery-accuracy.component.html` je KPI pojmenováno *Advisory Delivery Accuracy* a jeho graf je definován svg formátem se specifikovanými argumenty jako je v případě ze sekce *Úspěšnost odeslání produktů*.

7.7 Dostupnost systému

Soubor `datahub.ts` je hlavním zdrojem komunikace s databází DataHub, tudíž je nevyhnuté znovupoužití tohoto souboru pro získání dat. Struktura požadavku je vždy stejná, a to filtrování dat z indexu, v tomto případě z `pnt-devops-mars-uptime-*`, se zajištěním, že je požadavek zaslán na adresu Datahubu s indexem specifikovaným za adresou ve formě `/pnt-devops-mars-uptime-*/_search`. Upřesněním části požadavku `query_string` je možné ještě blíže specifikovat data, která jsou hledána. Součástí požadavku je také definice struktury odpovědi, která je upřesněna na obrázku 7.15. Hledaná hodnota odpovědi, která je použitá pro generování grafu, je určená názvem `uptime`. Získána data jsou předána postupem zmíněným výše v části *Uložení dat*.

```
"aggs": {
  "2": {
    "terms": {
      "field": "application_name.keyword",
      "size": 12,
      "order": {
        "1": "desc"
      }
    },
    "aggs": {
      "1": {
        "sum": {
          "field": "uptime"
        }
      },
      "3": {
        "sum": {
          "field": "downtime"
        }
```

Obrázek 7.15: Dostupnost systému - definice struktury dat

Získání dat pro toto KPI probíhá způsobem, jež byl specifikován v části *Získání dat z databáze nebo webové adresy* výše. Po získání dat ze serveru se data zpracují do podoby procent a jsou umístěny do struktury zmíněné na obrázku 7.16. Předtím než jsou data použita v grafu, je vytvořena funkce `generateColorByPercentage`, která určí jakou barvu bude mít jeden záznam grafu. Tato funkce byla vytvořena z důvodu nemožnosti specifikovat barvu částí grafu podle procentuálního rozmezí určené v návrhu 6. Výsledek funkce je uložen jako pole objektů, které podle jména servisu a procent dostupnosti servisu, obarví graf. Grafická implementace pro tuto část nástěnky je určena v souboru `system-uptime.component.html`, kde název grafu je určen jako *System Uptime*. Pro vizualizaci dat je vybrán horizontální graf a jeho definice je upřesněna v svg formátu, který je zobrazen na obrázku 7.17.

```
[
  {
    "name": "Germany",
    "value": 8940000
  },
  {
    "name": "USA",
    "value": 5000000
  }
]
```

Obrázek 7.16: Dostupnost systému - definice struktury dat[10]

```
<a class="uptime-chart col-auto">
  <ngx-charts-bar-horizontal
          [view]="uptimeColors.view"
          [animations]="true"
          [results]="uptimeStats"
          [xAxis]="true"
          [yAxis]="true"
          [scheme]="uptimeColors.colorScheme"
          [customColors]="colorServiceUptime"
          [barPadding]="12"
  ></ngx-charts-bar-horizontal>
</a>
```

Obrázek 7.17: Dostupnost systému - definice grafu

[10]Dostupné na: https://swimlane.gitbook.io/ngx-charts/examples/bar-charts/horizontal-bar-chart#data-format

Kapitola 8

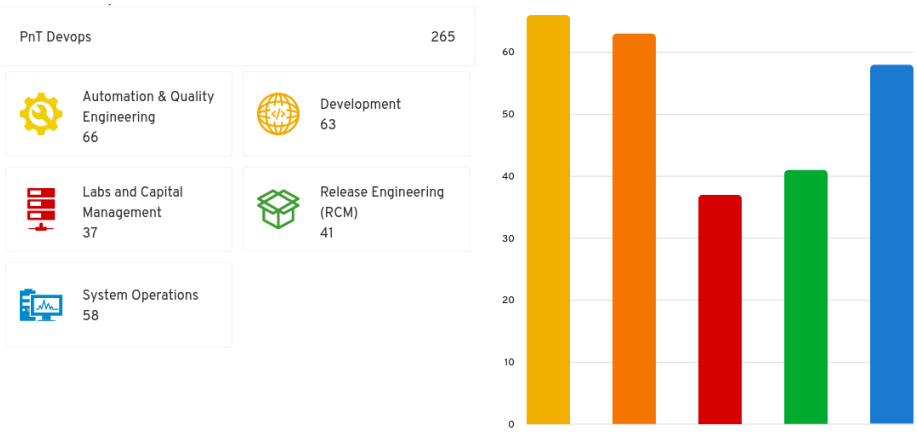
Demonstrace a zhodnocení výsledků

Tato kapitola se bude zabývat demonstrací grafických výsledků diplomové práce. Níže bude specifikován vzorek dat, na kterém budou demonstrovány metriky a klíčové ukazatele výkonnosti. Vzorek dat byl vybrán, aby nejlépe zobrazil funkcionalitu grafů. U každého pak bude přidáno zhodnocení a přínos pro management společnosti Red Hat.

8.1 Aktuální počet členů PnT DevOps týmu

První graf *Aktuální počet členů PnT DevOps týmu* na obrázku 8.1 zobrazuje celkový počet členů PnT DevOps vlevo nahoře. Pod tímto údajem se nachází číselné grafy, každého týmu, kde je zobrazena ikona týmu, název týmu a počet členů týmu. Ikona týmu je zabarvená, aby bylo zřejmé propojení číselných grafů a vertikálního grafu. Vertikální graf zobrazuje rozdíly v počtu členů mezi týmy. V této metrice jsou demonstrována data z jednoho dne, jelikož tento graf zobrazuje aktuální stav.

Pro tuto metriku bylo vybráno složení dvou typů grafů, aby bylo zprostředkováno porovnání počtu členů týmů. Důraz je kladen na vyváženost, jelikož větší rozdíl v počtu členů jednotlivých týmu může ovlivnit rychlost spolupráce mezi týmy. Tento graf nasměruje management k úvaze o najmutí více pracovníků do týmu s nižším počtem členů. Vylepšením tohoto grafu by mohlo být zobrazení trendu mezi jednotlivými měsíci, čtvrtletími nebo roky. Další možností by mohlo být zobrazení členů týmů v různých oblastech světa.

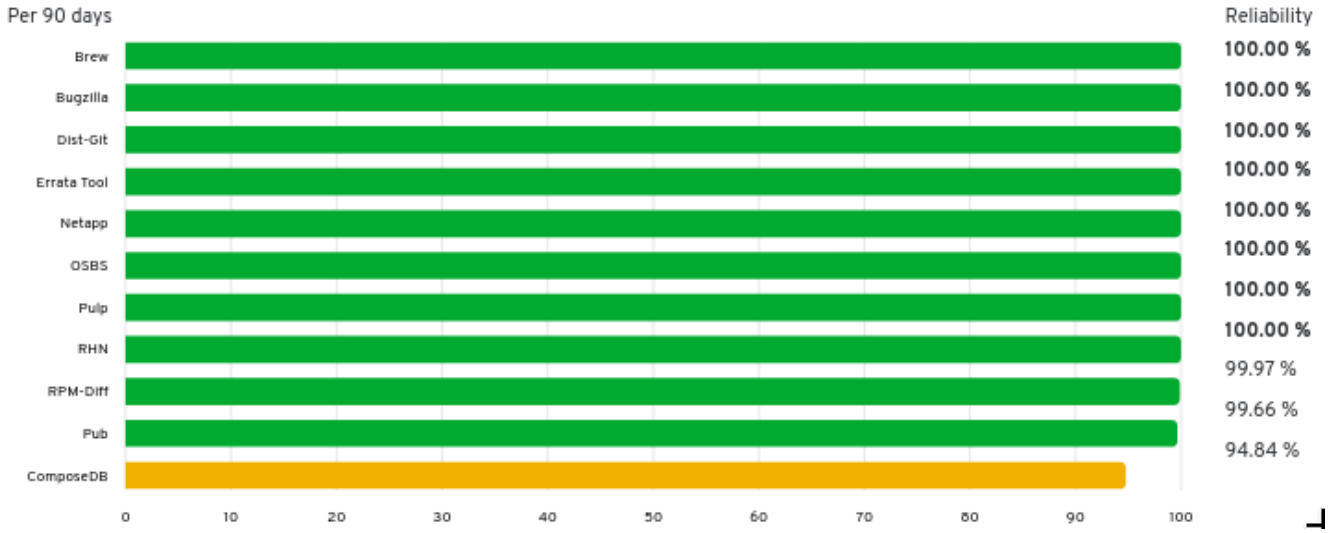


Obrázek 8.1: Aktuální počet členů PnT DevOps týmu

8.2 Dostupnost systému

Horizontální graf *Dostupnost systému* na obrázku 8.2 zobrazuje míru dostupnosti každého z jedenácti systémů. Pokud je barva grafu zelená je dostupnost systému v pořádku. Jestliže je graf zabarven barvou žlutou nebo červenou, není dostupnost systému v mezích očekávání. Pro zviditelnění přesnosti výsledků dostupnosti systému je za grafem zobrazena procentuální hodnota. Graf pro tento klíčový ukazatel výkonnosti je definován daty za 90 dní. Tohle časové rozmezí je vybráno za účelem správného demonstrování ukazatele výkonnosti.

Tento klíčový ukazatel výkonnost je hodnotným zdrojem pro zjištění efektivity práce inženýrů, jelikož zobrazuje, zda byl systém dostupný k práci. Management se podle tohoto ukazatele orientuje a plánuje nastávající kroky, pro zajištění bezproblémového běhu systémů. Vylepšení tohoto grafu by mohlo ukazovat rozdíl úspěšnosti splnění očekávání v minulých čtvrtletích. Tohle vylepšení by napomohlo porozumění, jestli se iniciativa na zlepšení statistik projevila ve výsledcích.

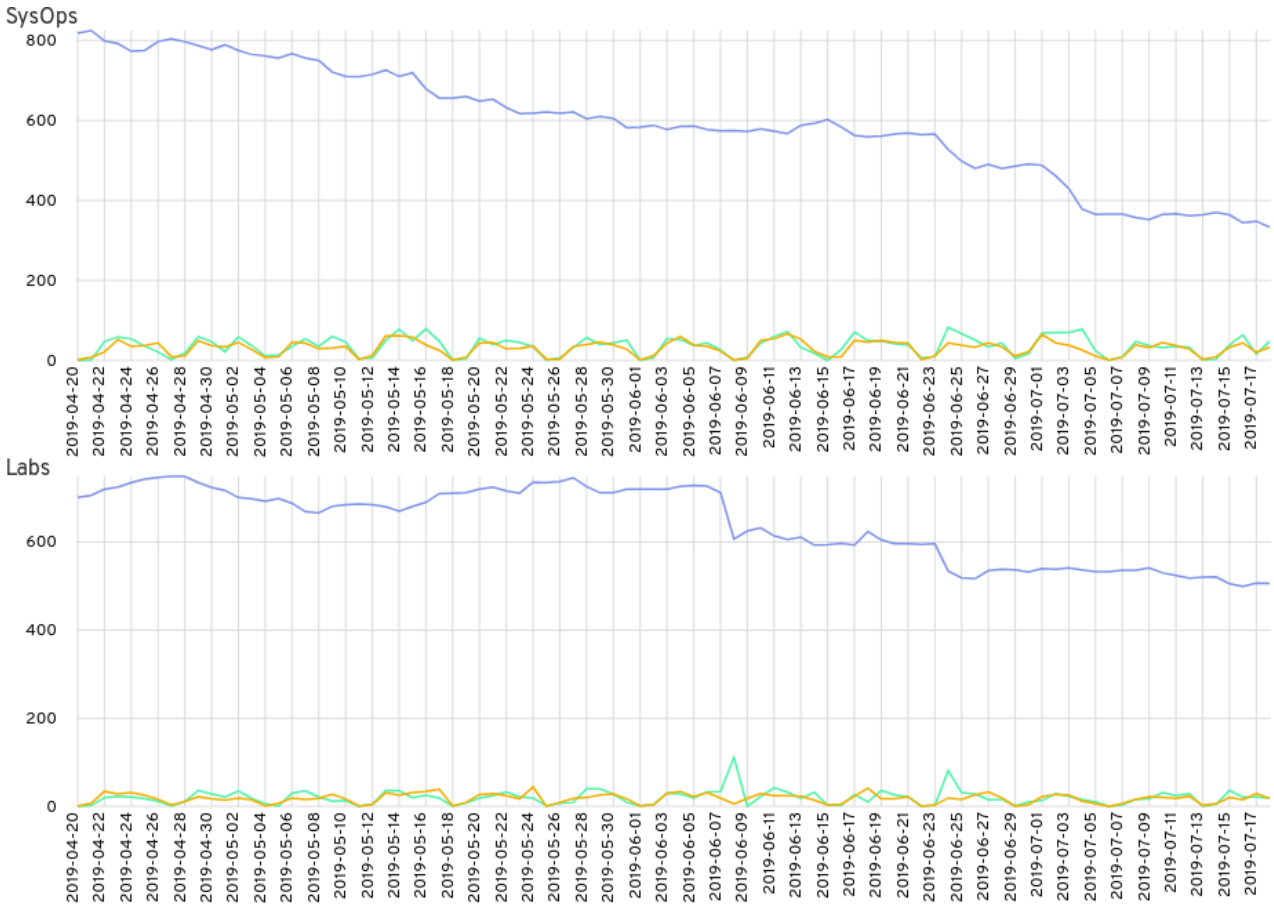


Obrázek 8.2: Dostupnost systému

8.3 Počet příchozích, otevřených a dokončených úkolů

Pro tuto metriku byly vybrána data v časovém rozmezí 90 dní viz obrázek 8.3. Na tomto rozmezí je demonstrována fluktuace práce, která je odvedena na úkolech (tickets). Vzorek dat je dostatečný pro nezkreslený pohled, jak efektivní jsou týmy ve zpracování úkolů (tickets) a jestli byly kroky pro zrychlení dokončování úkolů úspěšné, což je vidět na snižování množství otevřených úkolů. Otevřené úkoly jsou znázorněný fialovou barvou, úkoly příchozí jsou barvou žlutou a barvou zelenou dokončené úkoly. V grafu je zobrazeno, jak je snižován počet otevřených úkolů (fialová) pomocí udržování počtu dokončených úkolů (zelená) větší než je počet nově příchozích úkolů (žlutá)

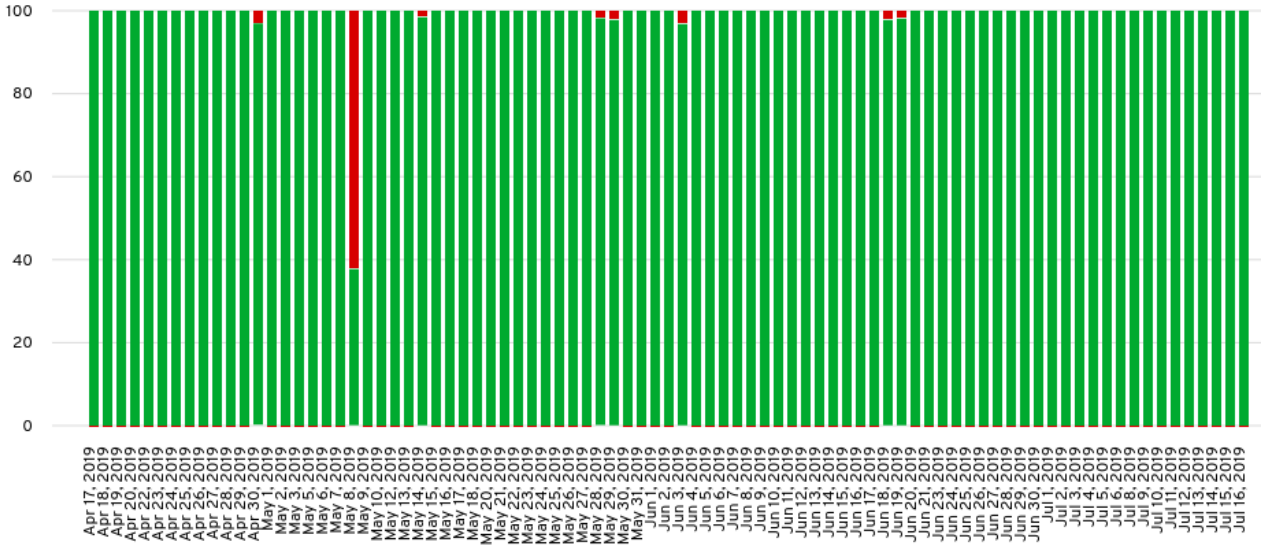
Metrika je spojována s výkonností týmů a členů týmů. Pokud je v grafu zobrazeno neobvyklé množství otevřených úkolů (tickets), naznačuje to, že by měl management začít jednat. Důvodem velkého množství otevřených úkolů (tickets) může být neefektivita pracovníků a nebo také nedostatek zaměstnanců na objem práce, který má tým vykonat. Vylepšení použitelnosti této metriky by mohla být bližší zaměření na úkoly splněné jednotlivými zaměstnanci.



Obrázek 8.3: Počet příchozích, otevřených a dokončených úkolů

8.4 Přesnost doručení softwaru na správné místo

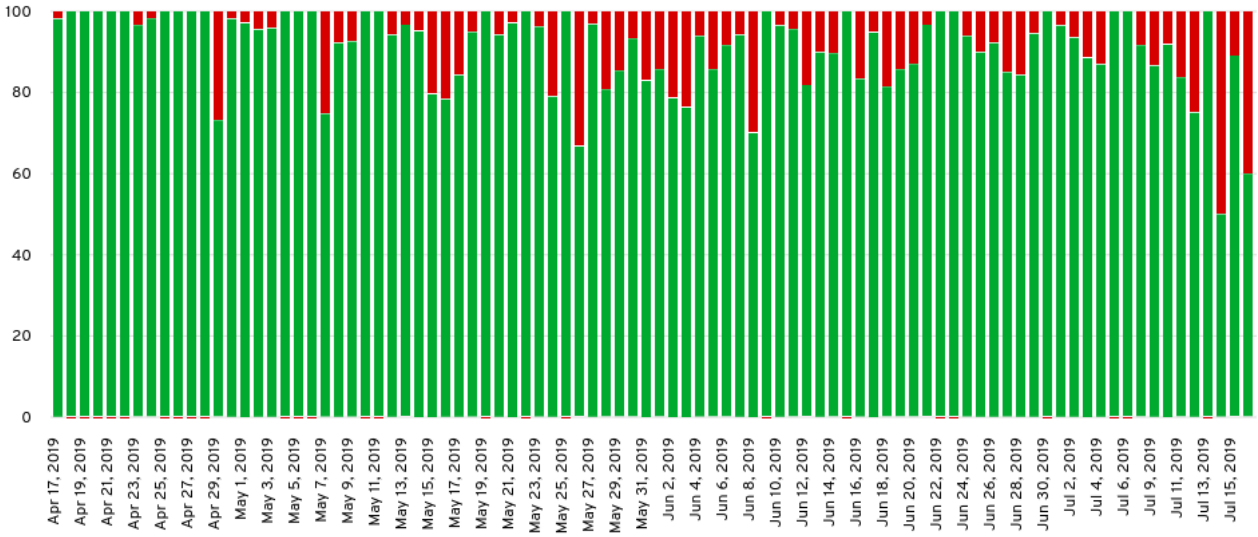
Pro tento graf byly vybrána data v časovém rozmezí 90 dní. Stejně jako u předešlých grafů je devadesáti denní rozmezí vhodné pro reprezentaci změn ve grafu. Z obrázku 8.4 je zřejmé, že přesnost doručení softwaru je stabilní i přes menší počet neúspěchů (červené části grafu). Tyto neúspěchy jsou důkazem, že systémy nejsou vždy stabilní. Klíčový ukazatel výkonnosti zobrazuje denní úspěšnost doručení softwaru. Pokud nastane neúspěch, tak je právě tento ukazatel hodnotný, aby bylo možné dohledat příčinu neúspěchu. Vylepšením tohoto ukazatele by mohlo být porovnání ukazatelů s předchozími čtvrtletími.



Obrázek 8.4: Přesnost doručení softwaru na správné místo

8.5 Úspěšnost odeslání produktů

Na obrázku 8.5 jsou zobrazena data za posledních 90 dní. Na tomto příkladu je demonstrovaná každodenní úspěšnost odesílaní produktů. Je zřejmé, že na tomto vzorku dat došlo ke kolísání úspěšnosti od konce dubna. Tento klíčový ukazatel výkonnosti je nástrojem pro upozornění managementu, že nový software není dodáván do RHN (definováno v sekci 5), kde je dostupný pro zákazníky. Pokud tedy není software dostupný, zákazník není spokojený a tudíž společnost negeneruje zisky. Tento graf slouží k znázornění výskytů chyb, které jsou pak vyhledány a opraveny. Pro vylepšení grafu by mohlo být zavedeno porovnání dat s předchozími čtvrtletími.



Obrázek 8.5: Úspěšnost odeslání produktů

Kapitola 9

Závěr

Pro diplomovou práci byla nastudována teorie o DevOps metodologii, teorie grafů a metrik. Taktéž byla zjištěna podoba nástěnky zobrazující metriky a klíčové ukazatelé výkonnosti. V požadavcích je upřesněna vizualizace grafů a nástěnky, a také kolik grafů nástěnka obsahuje. Kapitola *Návrh* odhaluje, jaký způsobem byla navržena vizuální stránka nástěnky a jednotlivých grafů. Pro vizualizaci byl použit program *AdobeXD*, který je hlavním nástrojem pro návrh vizualizace týmu *One Portal*, pro který je tento projekt vypracován. V kapitole *Implementace* je rozebráno jak byly dané grafy implementovány, odkud jsou získána data, jak jsou data zpracována, případně uložena na straně serveru. Také je popsáno, jakým způsobem byla na straně klienta zpracována data do formátu, který je přijímaný knihovnou *ngx-charts* sloužící jako hlavní nástroj pro vizualizaci grafů na webové stránce *One Portal*. V kapitole *Demonstrace a zhodnocení výsledků* byly prezentovány grafy na vybraném vzorku dat. Dále bylo zhodnoceno, jak grafy reprezentovaly daná data a následně byla vysvětlena hodnota, kterou dané grafy přináší managementu společnosti Red Hat.

Vypracováním diplomové práce, byl vytvořen nástroj pro management, který na jedné webové stránce zobrazuje nejdůležitější metriky týmu PnT DevOps. Tyto metriky byly pečlivě vybrány po konzultacích se zadavatelem tak, aby byl zobrazen viditelný rozdíl efektivity v čase. Nejen efektivita je důležitým ukazatelem v grafech, ale také růst týmů nebo dostupnost systémů. Všechny tyto metriky a klíčové ukazatele výkonnosti v budoucnosti napomohou rozhodnutí managementu, jaké kroky by měly být vykonány a proč. Data zobrazena v grafech napomohou chápání dynamiky týmu, fluktuace efektivity, jak už systému tak členů týmu.

Literatura

[1] Ansley, A.: *Data Visualization Infographic: How to Make Charts and Graphs*. [Online; navštíveno 25.07.2019].
URL https://www.tapclicks.com/guided-visualization/

[2] Blackstone, J.: *Theory of Constraints*. [Online; navštíveno 11.01.2019].
URL http://www.scholarpedia.org/article/Theory_of_Constraints

[3] Florys, M.: *The Three Ways - key principles of DevOps*. [Online; navštíveno 08.01.2019].
URL https://www.linkedin.com/pulse/three-ways-key-principles-devops-michal-florys

[4] Futong, H.; Tingting, S.: Software Project Metrics and Quality Management. In *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Oct 2013, s. 615–618, doi:10.1109/IIH-MSP.2013.158.

[5] Gilbertson, S.: *What is Red Hat Network*. [Online; navštíveno 24.07.2019].
URL https://web.archive.org/web/20121014194650/http://www.redhat.com/magazine/010aug05/features/rhn_overview/

[6] Haddad, C.: *DevOps = DevOps Principles + DevOps Practices*. [Online; navštíveno 11.01.2019].
URL https://dzone.com/articles/devops-devops-principles

[7] is an Incident Post-Mortem?, W.: *What is an Incident Post-Mortem?* [Online; navštíveno 08.01.2019].
URL
https://www.pagerduty.com/resources/learn/post-mortem-incident-report/

[8] Kavis, M. J.: *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. Wiley, 2014, ISBN 9781118617618.
URL
https://www.amazon.com/Architecting-Cloud-Decisions-Computing-Service/dp/1118617614?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1118617614

[9] Kerzner, H.: *Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance*. Wiley, 2017, ISBN 1119427282.
URL https://www.amazon.com/Project-Management-Metrics-KPIs-Dashboards/dp/1119427282?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1119427282

[10] Kim, G.; Debois, P.; Willis, J.; aj.: *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016, ISBN 1942788002.
URL https://www.amazon.com/DevOps-Handbook-World-Class-Reliability-Organizations/dp/1942788002?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1942788002

[11] Lebied, M.: *Designing Charts and Graphs: How to Choose the Right Data Visualization Types*. [Online; navštíveno 15.01.2019].
URL https://www.datapine.com/blog/how-to-choose-the-right-data-visualization-types/

[12] Meyer, M.: Continuous Integration and Its Tools. *IEEE Software*, ročník 31, č. 3, May 2014: s. 14–16, ISSN 0740-7459, doi:10.1109/MS.2014.58.

[13] Mueller, E.: *What is devops?| the agile admin*. [Online; navštíveno 07.01.2019].
URL https://theagileadmin.com/what-is-devops/

[14] org, G.: *Introduction to GraphQL*. [Online; navštíveno 27.07.2019].
URL https://graphql.org/learn/

[15] Reh, F. J.: *Understanding Metrics: A Business Management Term Definition*. [Online; navštíveno 20.01.2019].
URL https://www.thebalancecareers.com/what-are-business-metrics-2275174

[16] Rouse, M.: *ServiceNow*. [Online; navštíveno 24.07.2019].
URL https://searchitoperations.techtarget.com/definition/ServiceNow

[17] Rouse, M.: *What is infrastructure as code?| the agile admin*. [Online; navštíveno 07.01.2019].
URL https://searchitoperations.techtarget.com/definition/Infrastructure-as-Code-IAC

[18] university, K.: *Understanding the Principle of Flow in Lean Manufacturing*. [Online; navštíveno 08.01.2019].
URL https://online.kettering.edu/news/2016/07/07/understanding-principle-flow-lean-manufacturing

Příloha A

Obsah přiloženého paměťového média

- adresář server - soubory serverové částí implementace
- adresář klient - soubory klientské částí implementace