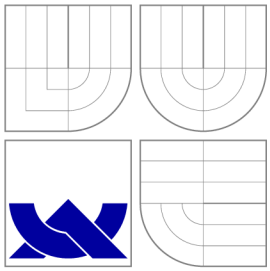
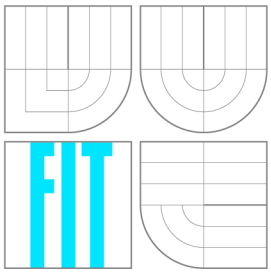


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

APLIKAČNÍ PROGRAMOVÉ ROZHRANÍ PRO ITIL SERVICE DESK SYSTÉMY

API FOR ITIL SERVICE DESK SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL ŠIRŮČEK

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2016

Zadání diplomové práce/17365/2015/xsiruc01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Širůček Pavel, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Aplikační programové rozhraní pro ITIL Service Desk systémy**
API for ITIL Service Desk Systems
Kategorie: Informační systémy

Pokyny:

1. Prozkoumejte poslední verzi IT Infrastructure Library (ITIL) pro řízení IT služeb, zejména funkci Service Desk a procesy pro správu incidentů, problémů, změn a konfigurací.
2. Seznamte se s požadavky na software podporu ITIL a analyzujte dostupný SW pro Service Desk/Help Desk (dále SD systémy).
3. Po konzultaci s vedoucím navrhněte pro vybrané SD systémy jednotné aplikační programové rozhraní (API), kterým lze do těchto systémů hlásit stav a události IT infrastruktury a vstupy ITIL procesů (např. hlášení incidentu).
4. Navrhněte a implementujte aplikaci, která bude zpřístupňovat instance SD systémů pomocí Vámi navrženého API.
5. Implementujte ukázkového klienta Vaší aplikace, který nahlásí incident nad definovanými IT komponentami a zareaguje na průběh jeho řešení v SD systémech (např. restart IT služby).
6. Výsledky zveřejněte jako open-source, zhodnoťte a navrhněte případná rozšíření.

Literatura:

- Vernon Lloyd, Colin Rudd. *ITIL Service Design*. The Stationery Office, 2007. ISBN 978-0-11-331047-0.
- Shirley Lacy, Ivor Macfarlane. *ITIL Service Transition*. The Stationery Office, 2007. ISBN 978-0-11-331048-7.
- David Cannon, David Wheeldon. *ITIL Service Operation*. The Stationery Office, 2007. ISBN 978-0-11-331046-3.
- George Spalding, Gary Case. *ITIL Continual Service Improvement*. The Stationery Office, 2007. ISBN 978-0-11-331049-4.
- *ITIL Software Scheme Mandatory Assessment Criteria 2011 Version*. [http://www.itil-officialsite.com/SoftwareScheme/MandatoryAssessmentCriteria.aspx]

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Diplomová práce se zabývá ITIL procesy v systémech service desk. Cílem práce bylo zvolit několik service desk systémů a zpracovat rozhraní, přes které bude možné se service desky komunikovat a pracovat s vybranými procesy. V práci je obecně popsána knihovna ITIL, podrobněji funkce service desku a důležité ITIL procesy. Množinu procesů, se kterými lze v rozhraní manipulovat, tvoří správa incidentů, problémů, změn a konfigurací. Pro práci s těmito procesy je navrženo a následně implementováno aplikační rozhraní fungující na bázi webových služeb. V závěru práce je prezentováno několik scénářů pro demonstraci komunikace mezi klienty a service desky přes vytvořené rozhraní.

Abstract

Main focus of this thesis is creating workaround for ITIL processes in service desk systems. As a objective with highest priority was to choose service desk systems, then create interface which can make possible communication between service boards and selected processes. I described also in generally meaning ITIL library, common service desk functions and important ITIL processes. Management of incidents, problems, changes and configurations is based on a set of processes, which are able to manipulate with interface. Web services are main tool of implemented API for working with these processes. In conclusion are also presented several scenarios to demonstrate communication between clients and service across the board created interface.

Klíčová slova

ITIL, Information Technology Infrastructure Library, service desk, help desk, IT služba, tiket, správa incidentů, správa problémů, správa změn, správa konfigurací, Java, webová služba, aplikační rozhraní

Keywords

ITIL, Information Technology Infrastructure Library, service desk, help desk, IT service, ticket, Incident Management, Problem Management, Change Management, Configuration Management, Java, web service, api

Citace

Pavel Širůček: Aplikační programové rozhraní pro ITIL
Service Desk systémy, diplomová práce, Brno, FIT VUT v Brně, 2016

Aplikační programové rozhraní pro ITIL Service Desk systémy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a pod vedením pana RNDr. Marka Rychlého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Širůček
24. května 2016

Poděkování

Děkuji panu RNDr. Marku Rychlému, Ph.D. za velmi rychlé odpovědi na mé dotazy a za všechny nápady, rady a připomínky, které mi během realizace práce poskytl. Dále bych chtěl poděkovat pánům Petru Vitouchovi a Petru Novohradskému ze společnosti ALVAO za odbornou konzultaci k systému ALVAO Service Desk. Poděkování nakonec patří také mé rodině a přítelkyni za podporu při realizaci práce.

Obsah

Úvod

S tím, jak dnes informační technologie pronikají téměř do každé firmy či společnosti, stává se stále důležitější částí následná podpora. V softwarových firmách nejde už jen o vývoj IT systémů, ale i o následné poskytování doplňkových služeb. Tímto způsobem přidávají výrobci svým produktům další hodnotu. Software je poskytováno jako služba. Kvalitní správa a řízení těchto služeb vyžaduje nemalé úsilí ze strany výrobce a přitom musí být vše perfektní. Nikdo přece nechce ztrácet zákazníky. Pro efektivní řízení služeb a komunikaci se zákazníky se využívají systémy známé jako service desk (případně help desk).

Co si lze představit pod slovním spojením provoz služeb v IT? I člověk, který se v problematice příliš neorientuje, si většinou u služby vybaví, že mu bude něco nabízeno. U provozu zase, že se bude jednat o nějakou zodpovědnost, aby služba fungovala a podobně. Znalejší osoba si pak již představí na pozadí určitou strukturu, kdy se bude jednat o systém software a lidí, kteří se starají o jeho běh. Tito lidé pro správu a řešení problémů budou jistě využívat právě service desk jako podpůrný systém a tím podporovat stabilní běh služby.

Tato práce se zajímá právě o systémy Service Desk, určené pro komplexní správu IT problémů. Implementačním cílem je navržení univerzálního API pro komunikaci s různými Service Desk systémy. Výsledné rohraní pak může posloužit při testování systémů service desk, může pomoci při výběru a může se stát základem pro testování ITIL procesů v různých systémech.

V první kapitole bude představena knihovna ITIL a její užitečnost v řízení podnikové informatiky. Následovat bude stručný popis historie a budou posány jednotlivé části knihovny, shrnující nejlepší praktiky návrhu, implementace a řízení služeb.

Druhá kapitola popisuje hlavně 4. část knihovny ITIL - provoz služeb. Největší pozornost bude věnována funkci service desk. S funkcí je spojena právě správa problémů, incidentů, změn a konfigurací. Správa těchto čtyř procesů bude tvořit hlavní náplň této práce. Stručně budou popsány další důležité ITIL procesy.

Ve třetí kapitole budou specifikovány cíle této práce a požadavky na funkce implementované aplikace. Bude představen postup, jak cílů dosáhnout.

Čtvrtá kapitola představí a popíše několik konkrétních řešení service desk systémů, které budou podporovány ve vytvářeném rozhraní. Nejprve budou definovány požadavky na vhodné systémy, podle kterých nakonec proběhl výběr. U vybraných systémů budou rozebrány funkce, jimiž systémy disponují a jejich přínos v implementaci rozhraní.

Pátá kapitola se zabývá návrhem aplikačního rozhraní. V první části této kapitoly je představen návrh struktury rozhraní a dat, která budou přes rozhraní k dispozici. Následovat bude důležitá část popisující způsoby komunikace mezi rozhraním a service desky. Poslední částí kapitoly je návrh struktur a formátů pro komunikaci mezi rozhraním a klienty.

Šestá kapitola popisuje vlastní implementaci aplikace. Představeny jsou použité nástroje a rozebrány implementační detaily.

Sedmá kapitola se zabývá testováním a tvorbou testovacích scénářů. Součástí kapitoly je popis klientů, pracujících podle připraveného scénáře.

Zhodnocení celé práce a dosažených výsledků je uvedeno v závěru práce. Kapitola shrnuje dosažené výsledky a navrhuje možná budoucí rozšíření práce.

Součástí práce je také několik příloh. V příloze A je uveden obsah přiloženého disku. V příloze B je uveden přehled webových služeb, které jsou rozhraním zpřístupněny. V příloze C jsou uvedeny příklady požadavků pro komunikaci s vytvořeným rozhraním.

Kapitola 1

ITIL - IT Infrastructure Library

Jedním z hlavních cílů v podnikovém řízení (nejen) informatiky, je především vědět co dělat, aby podnik neztrácel zákazníky. Vedle standardní správy informačních technologií jde především o to vydefinovat a popsat služby, které podniková informatika poskytuje zaměstnancům. Naučit zaměstnance s těmito službami pracovat a následně tyto služby nepřetržitě a efektivně spravovat a řídit. A to jak na operativní, tak na taktické a strategické úrovni. Disciplína, která se této oblasti řízení podnikové informatiky věnuje, se nazývá IT Service Management (ITSM), přeloženo do češtiny - Řízení služeb informačních technologií. Popis nejlepších zkušeností a postupů, jak takového efektivního řízení služeb dosáhnout je uveden v sadě publikací, souhrnně nazývaných ITIL - IT Infrastructure Library, v češtině Knihovna infrastruktury informačních technologií. [13]

ITIL je možné považovat za rámec, ve kterém lze najít techniky, upozornění, návody a další rady ohledně toho, jak věci dělat či nedělat. Hlavním účelem je zlepšení kvality služeb IT a jejich optimalizace v rámci podniku a jeho obchodních cílů. Tento proces je měřitelný k obchodnímu úspěchu podniku. Zkušenosti a doporučení se postupně staly defacto standardem. Jelikož se jedná o rámec, poskytuje ITIL dostatečnou flexibilitu k přizpůsobení vlastním požadavkům a potřebám. [6]

1.1 Historie

První verze knihovny ITIL vznikla koncem 80. let minulého století, kdy britská vládní agentura CCTA začala postupně vydávat publikace shrnující nejlepší zkušenosti z oblasti řízení služeb IT. Již tehdy bylo klíčovým prvkem to, že se nejednalo o pravidla vymyšlená úředníky či teoretiky, ale o skutečnou a ověřenou praxi. První svazek ITIL vyšel v roce 1989 a celá sbírka V1 zahrnovala celkem 46 svazků.

Vývoj druhé verze knihovny ITIL započal koncem 90. let. První publikace (Service Support) vyšla v roce 1999, poslední publikace pak v roce 2006. Verze V2 měla celkem 10 titulů. V praxi se ovšem používaly víceméně pouze dva tituly - Service Support a Service Delivery. Tyto dvě publikace jsou mnohými ITSM specialisty pokládány za bibli ITSM, a to i po vydání ITIL V3 [6].

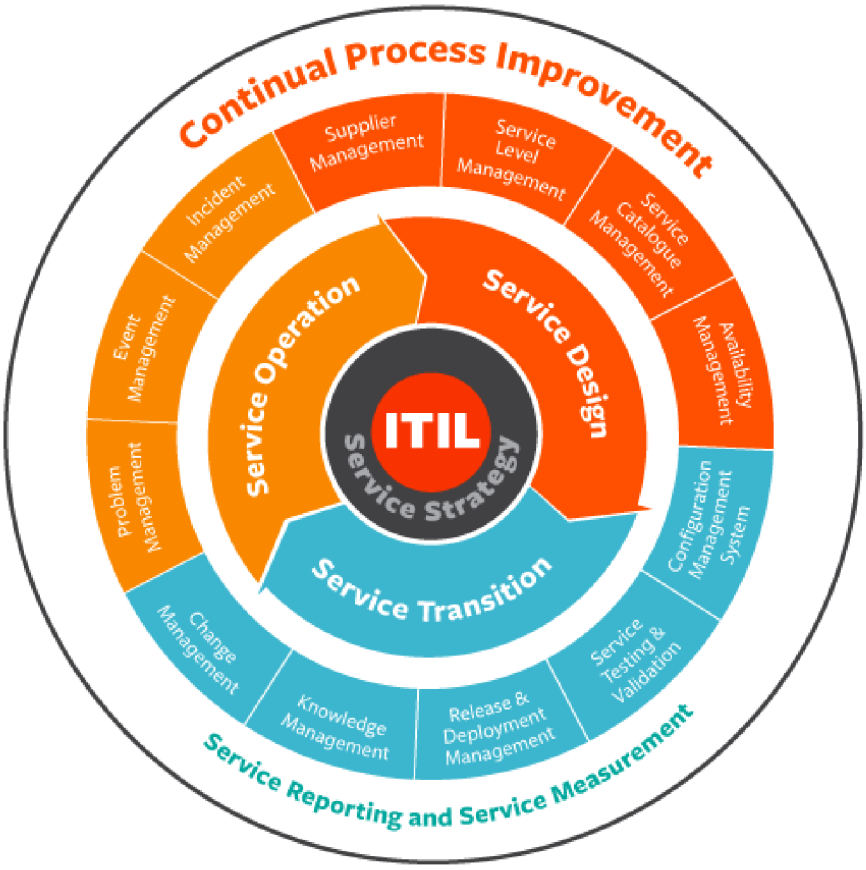
Práce na třetí, dosud poslední verzi, započaly v roce 2004 a byly završeny v květnu roku 2007 vydáním pěti ústředních publikací ITIL. Rozvoj třetí verze však zdaleka neskončil, postupně byly a stále jsou vydávány další rozšiřující a doplňující publikace (v současnosti čítá knihovna okolo 20 svazků). Pět ústředních publikací bylo v roce 2011 aktualizováno a tato poslední verze je označována jako ITIL 2011 Edition. Základní princip V3 je postaven

na řízení životního cyklu služby (hodnoty), kterou informační technologie poskytují svým zákazníkům (odběratelům). [6]

Pět fází ITIL spolu s nejdůležitějšími procesy je zobrazeno na obrázku 1.1.

K historickému přehledu je užitečné zmínit normu ISO/IEC 20000 z roku 2005 [11]. Norma se jako první celosvětový standard vztahuje k managementu IT služeb a zaměřuje se na zlepšování kvality, zvyšování efektivity a snižování nákladů u IT služeb a procesů. Poslední aktualizace normy, jedenáctá v pořadí, proběhla v roce 2015.

Základní principy řízení služeb IT jsou od první verze ITIL neměnné, tj. incidenty a události, problémy a známé chyby, releasy apod., jsou definovány stále stejně. Rozdíly mezi jednotlivými verzemi spočívají zejména ve způsobu zpracování a v rozsahu (počtu) prvků best practice. [13]



Obrázek 1.1: Schéma zobrazující jednotlivé fáze ITIL a nejdůležitější procesy. Převzato z [5].

1.2 Strategie služeb

První kniha ITIL se zabývá fází nazvanou *Strategie služeb* (*Service Strategy*). Tato fáze se prolíná všemi ostatními fázemi a ovlivňuje je. Funguje jako osa životního cyklu služby. Návrh služby převádí strategii do fáze návrhu. Převod a provoz služby za pomoci návrhu tuto strategii implementují. Fáze neustálého zlepšování služeb se opírá o strategii při dalším vylepšování služby z hlediska zvyšování kvality. [6]

Obecně lze strategii služeb chápat jako návod jak navrhovat, vyvíjet a implementovat správu služeb.

1.3 Návrh služeb

Fáze následující po strategii služeb se nazývá *Návrh služeb* (*Service Design*) a zabývá se jí druhá knihy ITIL. V souladu se strategií služeb se navrhují a tvoří hodnotové služby. Již v této fázi se navrhují a implementují zlepšovací mechanismy pro použití v pozdějších fázích životního cyklu služby. [6]

Obecně lze říci, že návrh služby nabízí návody pro návrh a vývoj služeb a procesů.

1.4 Přechod služeb

Po návrhu následuje fáze nazvaná *Přechod služeb* (*Service Transition*), která shrnuje zavedení služby do provozu. Tato fáze je zodpovědná za převedení strategie, obsažené v návrhu, z teorie do praxe. Řízeně a organizovaně převádí IT službu a všechny její části do provozu. [6]

Přechod služby poskytuje zásady a rady pro vývoj, zlepšování a předávku nových nebo upravených služeb do provozu.

1.5 Provoz služeb

Čtvrtá kniha ITIL, *Provoz služeb* (*Service Operation*), se věnuje fázi po přechodu služby do provozu, tedy vlastním provozem služby. Zajišťuje, aby zákazník dostal odpovídající hodnoty. Cílem fáze je stabilita IT služeb a zvyšování účinnosti a efektivnosti při poskytování služeb. Provoz služeb je také zodpovědný za správu technologií potřebných pro podporu a poskytování služeb. [6] Důležitým procesům v této fázi, stejně jako funkci service desku se bude věnovat následující kapitola.

1.6 Neustálé zlepšování služeb

Poslední fáze ITIL, *Neustálé zlepšování služeb* (*Continual Service Improvement*), poskytuje nástroje a návody pro nepřetržité zefektivňování služeb a dříve zmiňovaných aspektů jako je návrh, zavádění a provoz služeb. [6]

Kapitola 2

Service desk a důležité procesy ITIL

Service desk (případně SD systém) plní funkci hlavního kontaktního místa pro uživatele při hlášení poruch a problémů se službami (incidenty, problémy) a žádostech o službu. SD systém představuje také jediné kontaktní místo (SPoC) a jde o první linii IT podpory v organizaci. V principu se pro SD systémy používají čtyři základní struktury, uvedené v následujícím seznamu:

- centrální – existuje jediný service desk pro všechny organizační jednotky,
- lokální – každé pracoviště (oddělení) má svůj vlastní service desk,
- virtuální – části service desku jsou na různých místech, ale jsou dosažitelné přes jeden komunikační bod (např. přes jedno telefonní číslo),
- service desk „následující slunce“ – speciální forma virtuálního service desku, která zajišťuje dostupnost napříč různými časovými pásmy.

Service desky je možné rozdělovat i podle úrovně dovedností zaměstnanců. Mohou existovat silně specializované service desky (vysoká úroveň dovedností) nebo service desky nižní úrovně (základní úroveň dovedností).

Při vyhodnocování efektivního fungování SD systémů je možným ukazatelem procento vyřešených poruch a dosažitelnost. Dalším ukazatelem je např. průměrná doba řešení problému. Velmi důležitým parametrem, možná tím nejdůležitějším, je také spokojenost zákazníka. [6]

2.1 Role v rámci service desku

Při provozu service desk se uplatňují následující role:

- manažer service desku – provozní zodopovědnost ve velkých podnicích,
- vedoucí týmu service desku – zajišťuje aby byly vždy k dispozici potřebné úrovně dovedností, vede a stará se o tým,
- analylik service desku – podpora první linie provádí aktivity service desku,

- superuživatelé – zaměstnanci z jednotlivých oddělení se speciálním know-how, kteří pomáhají „IT méně zdatným“ kolegům s menšími problémy.

Význam service desku spočívá v jeho roli rozhraní mezi IT a koncovými uživateli. Dá se říci, že service desk představuje pro zákazníka organizaci IT. Proto jsou důležitým faktorem lidské zdroje kolem service desku. Důležité je zejména správné provázání personálního obsazení, úrovně dovedností a tréninku [6].

2.2 Service desk versus help desk

Oba systémy jsou určeny pro podporu zákazníka (koncového uživatele). V obou případech se jedná o SPoC. Každý ze systémů však přistupuje k této problematice odlišně.

Help desk je zaměřen primárně na pomoc koncovému uživateli, řešení jeho problémů. Neřeší interní procesy. Ve většině případů jde pouze o poskytování podpory zákazníkům.

Service desk je zaměřen více strategicky a kromě pomoci jde více do „hloubky“. Cílem je nejen řešení problémů uživatelů, ale také další zlepšování interních procesů, monitorování, vyhodnocování a zlepšování chodu organizace. [15] [5]

Role help desku a service desku je názorně demonstrována na následujícím příkladu. Přestala fungovat kopírka a uživatel vytvořil požadavek na vyřešení tohoto problému. Help desk se snaží co nejrychleji vyřešit problém a obnovit chod zařízení (restart, aktualizace ovladačů). Service desk tuto činnost může rozšířit, a tím jednoduše zjistit, že zařízení vykazuje chyby již delší dobu a problém reportovalo více uživatelů. Tím pádem může být vyhodnocena potřeba detailnějšího zkoumání zařízení, které by v budoucnu dalším podobným problémům zamezilo. Service desk tedy nejdříve zajistí roli help desku pro okamžité zprovoznění zařízení. Následně zajistí systémové řešení opakujícího se problému. Service desk podporující metodiku ITIL může dále ujasnit terminologii a způsoby řešení problémů. Z požadavku koncového uživatele, tak vznikne incident (neplánovaný výpadek služby), který je třeba rychle řešit například restartem. Z incidentu je možné dále vytvořit problém (příčina jednoho či více incidentů). V ukázkovém případě může problém díky většímu počtu incidentů odhalit chybné zařízení, zajistit jeho výměnu pomocí požadavku na změnu a zabránit vzniku dalších podobných incidentů. [15]

2.3 Důležité pojmy v procesech provozu služeb

Před popisem důležitých procesů service desku je nutné vymezit některé pojmy, se kterými se bude pracovat v následujících sekcích a kapitolách.

Tiket

Tiket (ticket) je instance procesu v rámci service desku. V závislosti na typu procesu se může jednat o incident, problém, změnu apod. [12]

SLA

SLA neboli *Service Level Agreement - Dohoda o úrovni služeb* popisuje služby IT a dohody o poskytovaných službách mezi zákazníkem a organizací. Používá se v ní netechnických výrazů (kvůli srozumitelnosti pro zákazníka). [3]

Service desk	**Help desk**
Service desk je zaměřen na *zákazníky*, *zaměstnance* a interní procesy.	Help desk je zaměřen hlavně na *zákazníky*
Strategický nástroj, smyslem je požadavky efektivně řešit a dále vyhodnocovat.	*Taktický nástroj*, smyslem je co nejrychleji řešit požadavky zákazníků.
Cílem je navrhovat zlepšení procesů pro efektivnější chod organizace.	*Cílem* je poskytovat podporu zákazníkům.
Podpora interního zaměření, každé oddělení (nejen IT) uvnitř organizace může být součástí a mít své *interní* zákazníky.	*Pouze externí* zaměření na externí zákazníky.
Málo organizací provozuje service desk bez help desku.	*Mnoho* organizací provozuje help desk bez service desku.
Obsahuje *pokročilou funkcionalitu* (Incident Management, Request Fulfilment, Problem Management, Change Management atd.)	Obsahuje *základní funkcionalitu* pro správu požadavků.

Tabulka 2.1: Tabulka shrnující základní rozdíly mezi service desk a help desk systémy. Převzato z [15].

2.4 Správa incidentů

Incident je v [3] definován jako neplánované přerušení služby IT nebo snížení její kvality. Incidentem je rovněž porucha konfigurace, která doposud neovlivnila běh služby. Dalším důležitým pojmem ve správě incidentů je tzv. *náhradní řešení*. To je využito pro omezení nebo vyloučení dopadu incidentu (problému), pokud není k dispozici úplné řešení.

Hlavním cílem procesu je obnovení normálního provozu služby. Důležité je, aby obnovení proběhlo co nejrychleji a s co nejmenšími dopady výpadku na uživatele a zákazníky. Dalším cílem je zajištění požadované kvality služby podle dohodnutých SLA.

Proces je odpovědný za včasnou detekci incidentů, jejich záznamu a řízení. Proces nezkoumá příčinu incidentu, ale hledá jakékoli řešení, které povede k obnovení služby nebo alespoň k omezení důsledků výpadku služby. Proces může ovlivnit pouze dobu trvání incidentů a obchodní ztráty (plynoucí z existence incidentu). Důležitým rysem procesu je řešení incidentů podle priorit, stanovených na základě důležitosti (zejména z obchodní stránky věci) nebo na základě SLA. To v důsledku znamená, že IT zdroje jsou prioritně přiřazovány závažným výpadkům, oproti méně významným.

Nejčastěji jsou incidenty detekovány procesem správy událostí a uživateli kontaktujícími service desk. Je žádoucí, aby co nejvíce incidentů bylo detekováno dohledovými systémy (hlavní cíl procesu správy událostí).

Důležitá je v procesu také automatizace pomocí vhodných nástrojů a znalostní databáze (informace o předchozích incidentech a jejich řešení). [13] [4]

Přínosy procesu a důsledky neexistence procesu

Mezi nejdůležitější [13] přínosy procesu správy incidentů patří:

- snížení důsledků dopadu incidentu a zkrácení doby výpadku IT služeb,
- zvýšení spokojenosti zákazníků a uživatelů,

- alokace IT zdrojů na řešení incidentů podle priorit.

V případě neexistence procesu se podnik musí připravit na problémy způsobené tím, že se při špatném řízení nebo neřízení incidentů mohou incidenty „ztrácet“ nebo se může významně prodloužit doba jejich odstranění. Tím se pak samozřejmě prodlužuje i doba výpadku služby. Z drobných incidentů se časem mohou stát incidenty závažné a mohou již významně ovlivnit kvalitu služby.

2.5 Správa problémů

Problém je v [3] označován jako příčina jednoho či více incidentů. *Známá chyba* je problém, který má popsánu příčinu a náhradní řešení.

Proces správy problémů odpovídá za správu všech problémů po celý jejich životní cyklus. Aktivně by měl proces zabraňovat výskytu nových incidentů (zamezení jejich příčinám). Opět by měla být k dispozici znalostní databáze, a informace v ní by měly být přístupné specialistům podpory a pracovníkům service desku. V databázi by měly být zaznamenané identifikované chyby a další postupy, jak na chyby a incidenty jimi vyvolané reagovat. Rovněž se v databázi zaznamenává náhradní řešení k incidentům.

Proces je dále odpovědný za včasnou detekci problémů, jejich záznamu a řízení. Cílem je nacházet příčiny incidentů a jejich efektivní odstraňování. Proces má zodpovědnost za rozhodování, zda bude chyba odstraněna nebo se dočasně ponechá. Důležitou funkcí procesu je ověřování, zda byly původní příčiny chyby skutečně odstraněny a nedochází k dalším incidentům.

Přínosy procesu a důsledky neexistence procesu

Největším přínosem je jistě snížení počtu incidentů a tím zvýšení dostupnosti IT služeb a zajištění stabilní infrastruktury.

Bez procesu správy problémů se může stát skutečností každodenní „hašení požárů“. Ztráta důvěry zákazníků může být následkem neustálého opakování stejných incidentů. Neexistence řešení a opakované výskyty stejných incidentů, to jsou další věci, které demotivují zaměstnance a mohou být příčinou např. velké fluktuace. [13] [4]

2.6 Správa změn

Změnou je v [3] nazýváno přidání, modifikace nebo odstranění čehokoliv, co má vliv na služby IT. *Požadavek na změnu* je formálním návrhem obsahujícím detaily změny. *Návrh změny* dokumentuje popis zavedení změny s odpovídající případovou studií a harmonogramem implementace.

Proces řídí životní cyklus všech změn (od prvotního návrhu po nasazení a zkušební provoz) a umožňuje realizaci požadovaných změn při minimálním narušení služeb IT. Změna jako taková se v procesu nevytváří, pouze se v něm řídí. Proces neřídí malé změny, mající spíše status incidentů (jako příklad se uvádí výměna porouchané tiskárny za jiný kus stejného typu). Rozsáhlé změny s celopodnikovým dopadem také nejsou řízeny procesem správy změn.

Přínosy procesu a důsledky neexistence procesu

Podle statistik analytických společností je až 70% incidentů způsobeno nezdokumentovanými chybami v nově nasazených změnách a konflikty při nasazování těchto změn [3]. Proces správy změn se snaží tyto problémy eliminovat a tím zvyšovat dostupnost služeb IT. Existence procesu snižuje negativní dopad změn na produkční prostředí a zlepšuje výkonnost procesu správy problémů.

Následkem neexistence procesu je např. neefektivní využívání zdrojů (neexistuje plánování změn) nebo omezená schopnost zvládnout zpracovat požadovaný objem změn. Může se stát že díky chybnému schvalovacímu procesu budou implementovány změny, které nejsou potřebné. Spolu se změnami mohou být implementovány zdroje dalších problémů a s tím související zvýšení počtu možných výpadků. [13] [4]

2.7 Správa konfigurací

K podpoře procesu slouží *Systém správy konfigurací* (CMS – Configuration Management System), což je soubor dat, nástrojů a informací. Komponenta (nebo jiné aktivum) služby, která by měla být spravována za účelem dodání služby se nazývá *konfigurační položkou*. Konfigurační položky zahrnují služby IT, hardware, software, lidské zdroje a dokumentaci. Detailní informace o všech konfiguračních položkách jsou zaznamenávána v *konfiguračním záznamu*. Uložiště konfiguračních položek se nazývá konfigurační databází (CMDB). Podle [13] tvoří CMDB srdce ITIL. Proces má za úkol udržovat CMS systém a zpřístupňovat informace rolím a funkcím, které je potřebují. Životní cyklus konfiguračních položek je dokumentován prostřednictvím konfiguračních záznamů. Proces dále vede evidenci softwarových licencí a spravuje softwarovou dokumentaci.

Proces zodpovídá za správné řízení aktiv služeb a za správné a přesné informace o těchto aktivech a jejich dostupnost kdykoliv a kdekoliv. Nejdůležitější odpovědností procesu je vlastnictví CMS, a z toho vyplývající zodpovědnost za aktuálnost dat v systému uložených.

Přínosy procesu a důsledky neexistence procesu

Mezi největší přínosy patří zejména existence spolehlivých informací o konfiguračních položkách, jejich vazbách a dokumentacích. Významně se zvyšuje efektivita téměř všech ostatních ITSM procesů. Dalším přínosem je redukce používání neautorizovaného softwaru a plnění legislativních podmínek.

Bez správy konfigurací nejsou některé procesy téměř schopny své existence a efektivita mnoha ostatních je významně snížena (např. správa incidentů a problémů). Velmi obtížně se také kontroluje dodržování licenčních smluv a obecně správa licencí. [13] [4]

2.8 Další procesy

Čtyři výše popsané procesy patří mezi 14 nejdůležitějších z celkových 26 procesů, které popisuje ITIL [13]. Následující seznam stručně popisuje další důležité procesy.

- *Event management* - zodpovídá za správu událostí (změna stavu významná z hlediska řízení konfigurační položky nebo služby IT).
- *Request fulfilment* - řízení životního cyklu všech požadavků na službu (požadavek uživatele na něco, co má být poskytnuto, např. rada, instalace programu, . . .).

- *Access management* - zodpovědnost za umožnění uživatelům používat služby a data
- *Release and deployment management* - zodpovídá za plánování, sestavení, testování, distribuci a nasazení release. Dále zajišťuje post-implementační podporu uživatelům IT služeb.
- *IT service continuity management* - zajišťuje obnovu funkčnosti kritických IT služeb po vážných výpadcích. Minimalizuje pravděpodobnost vzniku rozsáhlých výpadků.
- *Capacity management* - zodpovídá za to, aby IT služby a infrastruktura byly schopny dosáhnout požadované kapacity a výkonu
- *Availability management* - zodpovídá za to, aby IT služby a infrastruktura byly schopny dosáhnout požadované dostupnosti
- *Service level agreement* - zodpovídá za sjednání SLA s dosažitelnými parametry a jejich plnění.
- *Service catalogue management* - zajišťuje údržbu a aktualizace katalogu služeb. Zpřístupňuje obsah katalogu určeným rolím a funkcím.
- *Financial management for IT services* - poskytuje efektivní správu majetku a zdrojů používaných při poskytování služeb IT. Zajišťuje příslušnou úroveň financování IT služeb.

Kapitola 3

Specifikace požadavků

Třetí kapitola se obecně věnuje specifikaci požadavků na výslednou práci. Úvodní část zahrnuje popis cílů práce. Dále budou popsány možnosti pro výsledné použití implementovaného rozhraní a požadavky na funkčnost.

3.1 Cíle práce

Hlavním cílem práce je vytvoření aplikačního rozhraní pro přístup k určitým procesům service desku. Tvorba rozhraní zahrnuje návrh, implementaci a testování. Rozhraní by mělo být na konci vývoje volně dostupné jako open source. Dalším, velmi důležitým cílem, je vytvoření ukázkových klientů pro vyvíjené rozhraní. Úkolem klientů je nejen otestovat funkčnost rozhraní, ale také předvést jeho funkčnost podle připravených scénářů. Scénáře by měly reprezentovat reálné použití service desku uživatelem. Klient tak bude vystupovat v roli uživatele service desku.

První částí práce, ještě před návrhem samotného rozhraní, je průzkum existujících řešení pro SD systémy. V této části budou zvoleny vhodné service desky, pro které bude rozhraní navrženo. Je nutné specifikovat, podle čeho bude výběr SD systémů probíhat (licence, vlastní api pro přístup k procesům apod.).

Dalším krokem je návrh aplikace. Zde bude nutné specifikovat strukturu dat a informací, pomocí kterých bude rozhraní komunikovat, a to jak s klientem, tak na druhou stranu se SD systémem. Samotné rozhraní by mělo být pro klienta (uživatele) černou skříňkou. Proto je důležitým faktorem jednotný formát dat, který bude klient používat pro komunikaci s různými service desky přes implementované rozhraní. Nemělo by být složité přidávat další funkcionalitu, jako např. podporu dalšího systému service desk. Stejně tak by měly jít bezproblému rozšiřovat struktury rozhraní nesoucí informace o procesech a datech SD systému (tikety a konfigurační položky).

Podle návrhu bude následně rozhraní implementováno. Protože bude rozhraní pracovat s více SD systémy, ověří se rovnou správnost návrhu. Postupně bude implementována podpora jednotlivých SD systémů.

Poslední částí práce je příprava scénářů pro testování procesů podporovaných rozhraním. To zahrnuje implementaci několika ukázkových klientů, kteří podle daného scénáře vykonávají akce pomocí rozhraní v konkrétním v service desku.

Použití výsledného rozhraní

Rozhraní by mělo být možné použít pro testování procesů podporovaných SD systémů. Bude možné porovnávat vlastností jednoho systému s jinými. Při výukových účelech by mělo být možné rozhraní použít pro simulaci implementovaných procesů.

V praxi by rozhraní mohlo být použito v případě, kdy by bylo potřeba převést nějaká data z jednoho service desk systému do druhého. Jako příklad poslouží situace, kdy jedna společnost využívá jeden service desk a druhá společnost, úzce propojena s první (např. dodavatel), používá jiný. Potom bude pomocí rozhraní možné „zkopírovat“ data z jednoho systému (např. incident nad určitou konfigurační položkou) do druhého. Bude samozřejmě nutné, aby byly oba systémy v rozhraní podporovány. To je další důvod proč bude kladen při návrhu a implementaci důraz na jednoduchou rozšiřitelnost rozhraní o další SD systémy.

Funkce výsledného rozhraní

Pomocí rozhraní bude možné hlásit do konkrétního SD systému stav a události infrastruktury. Ohledně stavu se jedná o stav konfiguračních položek - evidovaného hardware a software. Ohledně událostí se jedná o vytváření a sledování procesů incidentů, problémů a změn. Tyto procesy se většinou budou týkat konkrétní konfigurační položky (hlášení incidentu kvůli nedostupnosti tisku apod.), ale nemusí to být pravidlem. Přes rozhraní bude možné sledovat a upravovat stav řešení těchto tiketů.

Příklad, jak může komunikace přes rozhraní vypadat:

- Počáteční stav - nelze tisknout.
- Klient si zobrazí informace o tiskárně tak, že zašle požadavek na rozhraní, to zašle požadavek na service desk, service desk zašle odpověď zpět rozhraní a to zašle nakonec odpověď klientu.
- Nad tiskárnou klient založí incident (stejný postup komunikace požadavek-odpověď jako při zobrazení informací o tiskárně).
- Klient sleduje průběžně stav řešení incidentu pomocí rozhraní.
- V závislosti na postupu řešení klient vykoná nějakou akci, např. po vyřešení incidentu jej uzavře.

Kapitola 4

Analýza dostupných service desk systémů

Pro systémy service desk a help desk existuje mnoho řešení. Nemá smysl vybírat a popisovat konkrétní systémy, proto budou v této kapitole popsány existující systémy obecně a pozornost bude zaměřena až přímo na systémy vybrané pro tuto práci.

Při výběru vhodného systému, se může potencionální zákazník řídit mimo jiné podle hodnocení stávajících zákazníků. Lze si tak udělat přehled o tom, co který systém podporuje a nabízí. Případně jak je na tom z hlediska uživatelské přívětivosti. Na stránkách [7] lze nalézt mnoho aplikací, doplněných právě o uživatelská hodnocení. Nevýhodou může být fakt, že stránka sdružuje informace nejen o help desk nebo service desk systémech, ale obecně o všech aplikacích, které se nějakým způsobem dotýkají IT podpory (např. aplikace pro hlasovou komunikaci s IT podporou apod.). Stránka může být užitečná jako výchozí bod při výběru service desk systému.

Dalším zajímavým zdrojem jsou stránky certifikací ITIL PinkVERIFY mezinárodní autority Pink Elephant [14]. Zde je možné nalézt service desk systémy spolu s přehledem procesů, které tyto systémy podporují a které jsou v systémech certifikovány. Největší výhodou je tak právě informace o SD systémech a podporovaných procesech na jednom místě a v přehledné formě. Podle požadovaných funkcí se tak lze zaměřit na konkrétní SD systémy.

Na stránce [10] je možné nalézt přehled oblíbených systémů za konkrétní časové období, v tomto případě pro rok 2016. Mimo jiné zde jsou informace o cenách jednotlivých produktů a podporované funkce. Z přehledu lze například zjistit, zda systém disponuje rozhraním webových služeb. Nevýhodou přehledu je fakt, že prezentuje systémy jako volně stažitelné (free verze), i pokud se jedná o časově omezenou trial verzi. Výčet je stručný, na druhou stranu si lze udělat obrázek o cenách systémů.

4.1 Service desk systémy podporované v rozhraní

Při výběru vhodných systémů, které budou podporovány v aplikaci, byly brány v potaz tři důležité vlastnosti.

První je cena. V potaz byla brána pouze řešení, která disponují volně dostupnou verzí, jedná se o open source nebo je dostupná univerzitní licence. Service desky často nabízí právě trial verze, v drtivé většině případů jde však o 14-ti denní plnou verzi. Po uplynutí této doby si musí uživatel systém zakoupit nebo smazat.

Druhou vlastností je podpora požadovaných ITIL procesů. Minimum je podpora ale-

spoň správy incidentů (incident management). Nemusí se jednat přímo o proces tak, jak jej definuje ITIL, ale měl by být zřejmý rozdíl mezi incidentem a klasickým požadavkem (user request). Další procesy, které by vhodný service desk měl podporovat je správa problémů a změn (problem a change management). Proces správy problémů bývá ještě občas v některých volně dostupných verzích systémů podporován (souvisí přímo s procesem správy incidentů, který je v základu ve většině SD systémů), většinou je ale dostupný až v placených verzích. Stejně tak proces správy změn a konfigurací (change management a configuration management), které jsou většinou podporovány v pokročilejších (placených) verzích systémů.

Třetí vlastností je přístup k datům service desku, či možnosti komunikace externí aplikace se service deskem. Pokud by byla jedinou možností, jak se dostat k datům, nutnost přistupovat přímo k databázi a operovat nad ní, nejednalo by se o příliš dobré řešení. Nevýhodou tohoto přístupu je hlavně to, že na některé operace nad daty mohou být navázané další akce, které se pak neprovedou, protože se operuje s daty systému na nízké úrovni, aniž by o tom systém sám věděl. Tento způsob přístupu navíc není ve většině případů ani možný. Výjimkou mohou být systémy, které databázi zpřístupňují implementátorům systému v organizaci a umožňují databázi doplňovat a v jistých mezích upravovat. Ideální tedy je, pokud systém disponuje rozhraním, které umožňuje přístup externí aplikace k vlastním datům a procesům. Nejčastějším typem takového rozhraní je přístup pomocí webových služeb (ve většině případů REST api). Při výběru service desk systémů se proto jedná o velmi důležitou vlastnost.

V následující části bude představeno pět vybraných service desk systémů.

ALVAO Service Desk

ALVAO Service Desk pochází od českých vývojářů a je k němu dostupná akademická licence. To umožňuje využít v service desku více funkcionality než ve free verzích jiných komerčních řešení. Podle [14] je v service desku certifikováno 6 ITIL procesů:

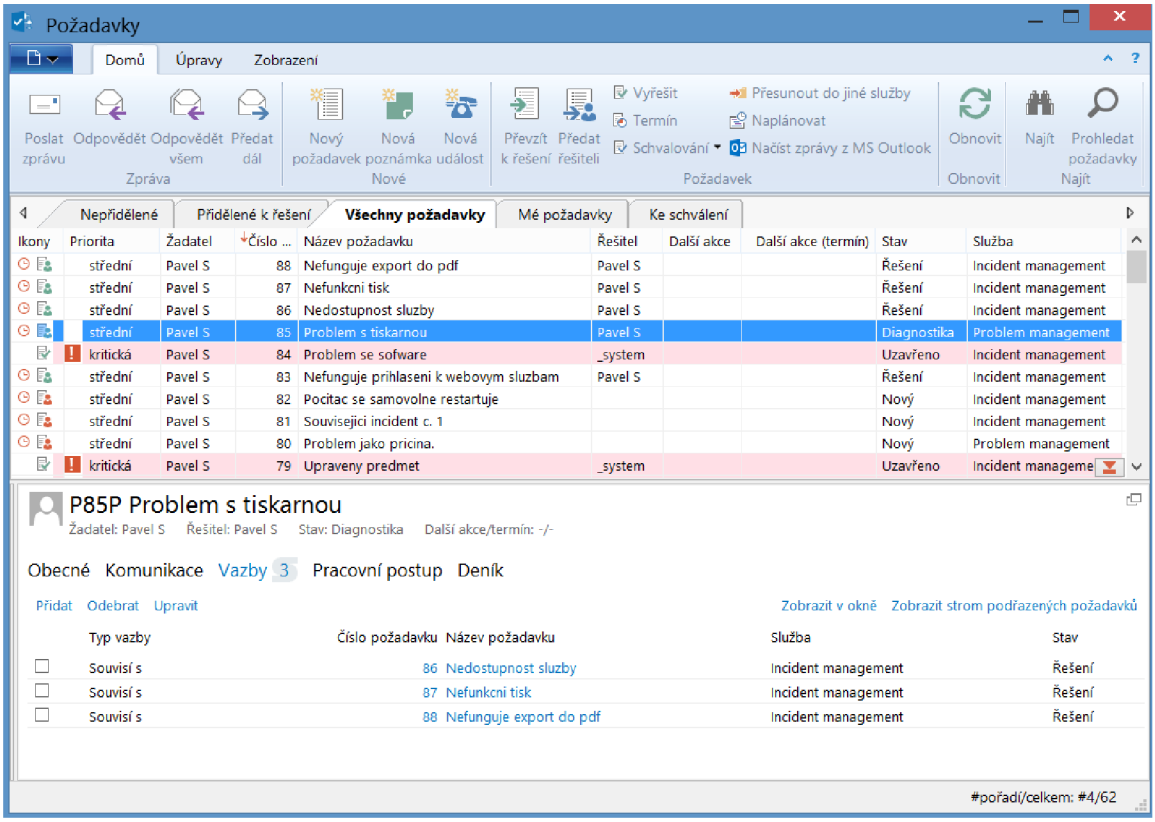
- Incident Management,
- Problem Management,
- Request Fulfilment (běžné žádosti),
- Service Asset & Configuration Management,
- Change Management,
- Service Catalog Management (správa katalogu služeb).

Service desk běží pouze na systémech Windows s databází MSSQL. Nevýhodou může být pro některé uživatele složitější instalace, kdy může být potřeba upravovat i konfigurační soubory aplikace.

Koncepce je postavena architektuře stromu služeb, která umožňuje konfigurovat služby podle potřeb uživatele nebo zákazníka. Díky univerzálnímu přístupu je možné si vytvořit libovolné vlastní služby, s vlastními pravidly. [2]

Service desk disponuje webovým rozhraním, které je dostupné koncovým uživatelům. Dále disponuje aplikací Service Desk Console. Ačkoliv by to mohlo být patrné z názvu, nejedná se o konzolovou aplikaci. Jde v podstatě o desktop verzi webového rozhraní, které umožňuje provádět pokročilejší úkony a umožňuje hromadné zpracování tiketů. Služby,

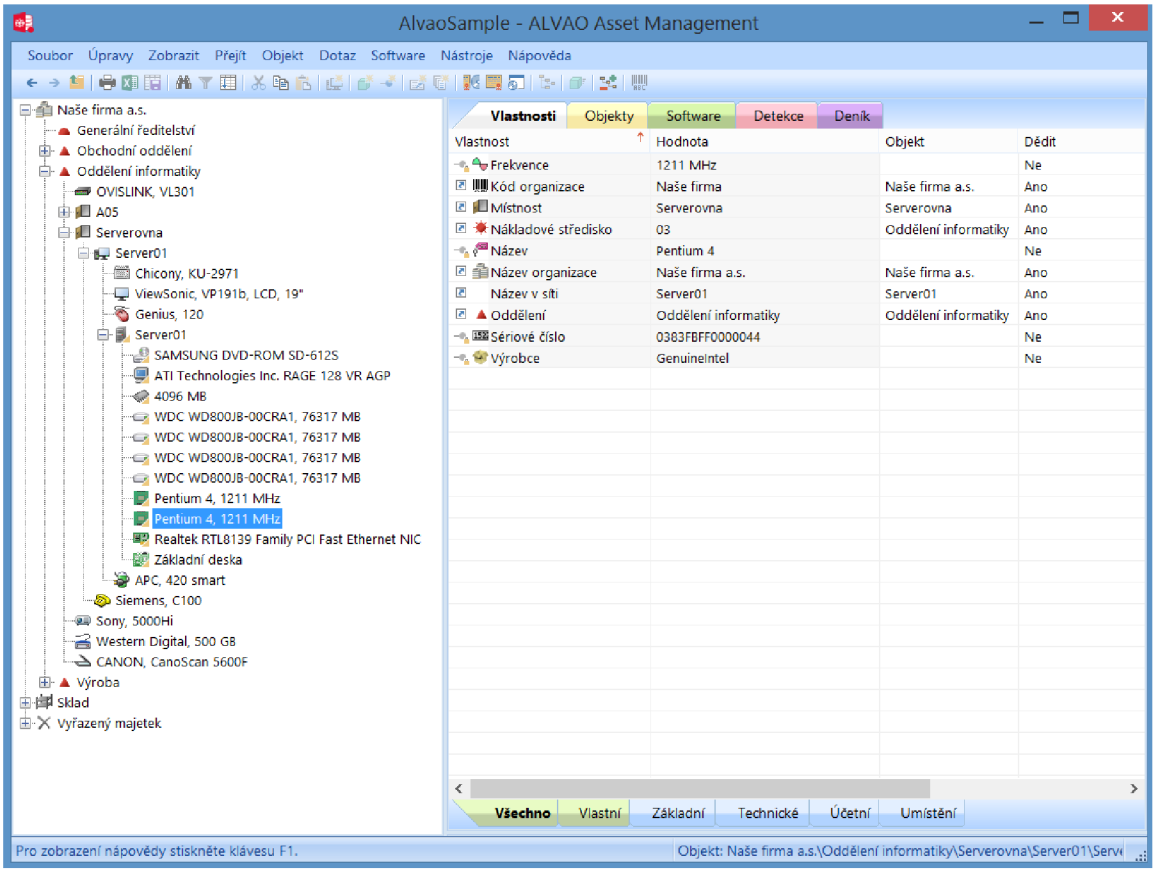
uživatele, databáze a další věci je nutné konfigurovat v aplikaci ALVAO Admin, což je hlavní administrační prostředí ALVAO aplikací.



Obrázek 4.1: ALVAO Service Desk Console.

ALVAO Service Desk disponuje SOAP webovými službami, avšak v omezené formě. Lze touto cestou vytvářet tikety, ale již není možné je rozumně spravovat. Webové služby slouží hlavně k notifikacím (zasílání emailů) podle různých podmínek. Pro správu tiketů je nutné přistupovat přímo do databáze. Webové služby umožňují číst jednotlivé sloupce z tabulky, která obsahuje informace o tiketech. K přečtení kompletních informací by tedy bylo nutné službu volat mnohokrát, což není úplně optimální. Jednodušším řešením je tak přistoupit přímo k tabulce do databáze, pokud je to nutné, tak připojit další tabulky, a data hromadně přečíst. Stejně tak úpravu tiketů je nutné řešit přímo v databázi. Dokumentace k těmto úkonům poskytuje dobrý návod, protože se počítá s tím, že organizace si bude systém přizpůsobovat svým potřebám.

Ke správě konfiguračních položek, assetů a obecně cmdb databáze slouží *ALVAO Asset Management*. Jde o samostatnou aplikaci, která není závislá na service desku. Pomocí aplikace lze kompletně spravovat majetek společnosti. Konfigurační položky lze libovolně rozšiřovat o jakékoliv parametry a vlastnosti. Lze využít také automatickou detekci hardware a skenování software. Systém pak udržuje sám aktuální informace o hardware a software. I Asset Management disponuje SOAP webovými službami, které jsou ale ještě omezenější než v případě service desku a pro tvorbu rozhraní nejsou využitelné. [1]



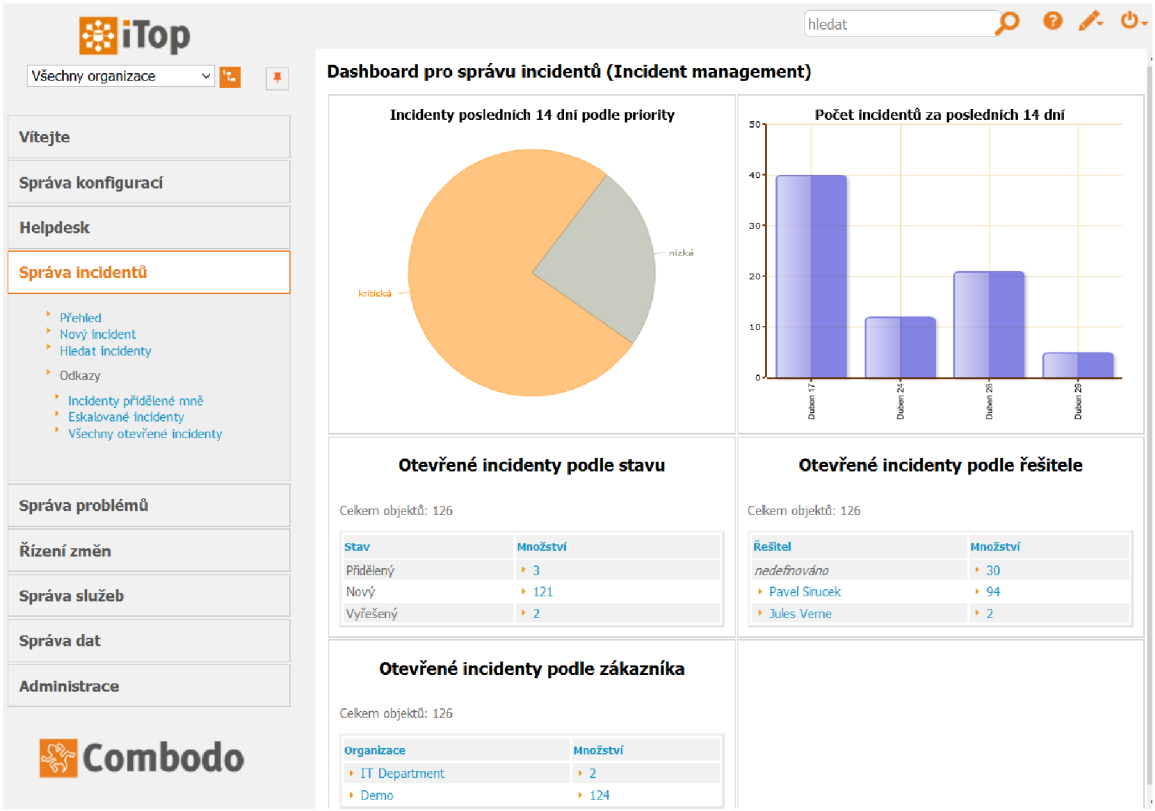
Obrázek 4.2: ALVAO Asset Management.

Combodo iTop

Combodo iTop Community [8] je open source ITSM řešení naprogramované v PHP. Využívá databázi MySQL. Pro provoz je tedy nutné mít k dispozici databázi MySQL a příslušný webový server. Nicméně instalace systému je velmi jednoduchá. Komunitní verze je zcela zdarma a podporuje mimo jiné následující ITIL procesy:

- Helpdesk (Request Fulfilment),
- Incident Management,
- Problem Management,
- Configuration Management,
- Change Management.

Vedle systému ALVAO, bude tedy i iTop podporovat všechny procesy vytvářeného rozhraní.

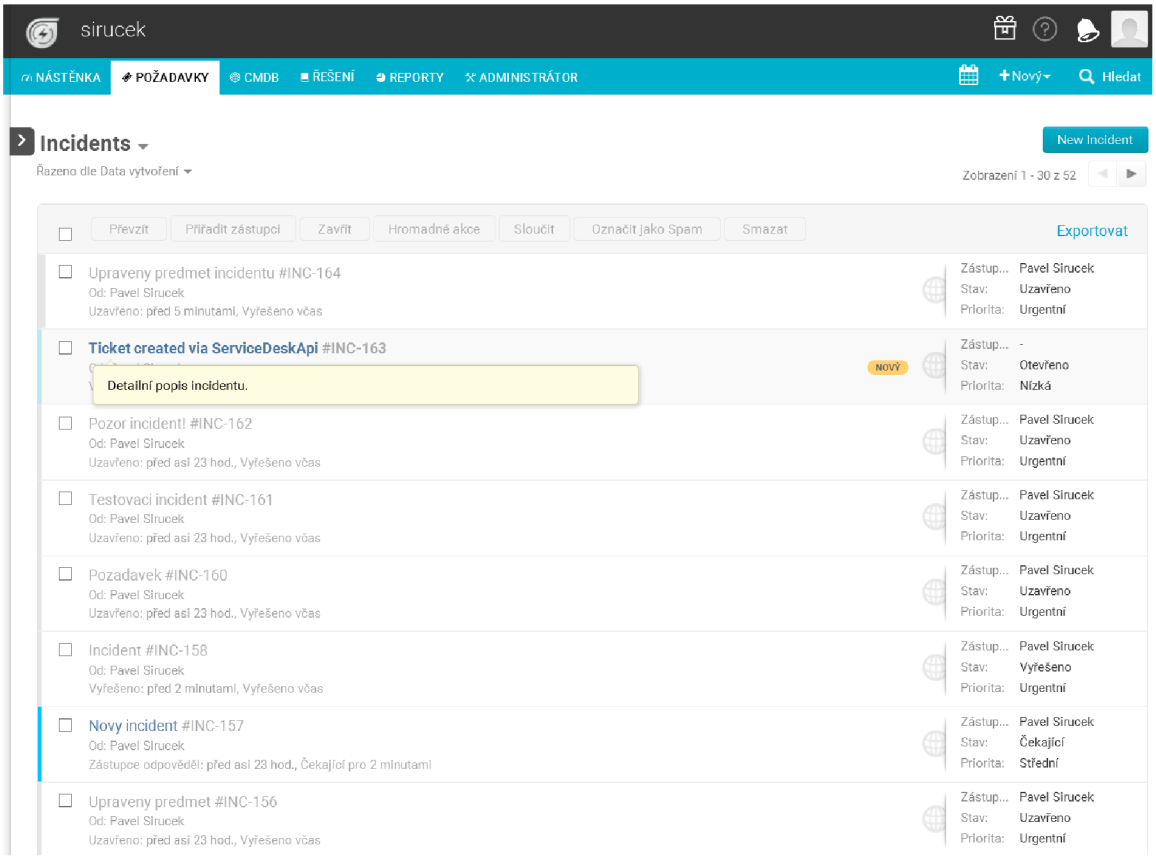


Obrázek 4.3: Webové rozhraní iTop.

K dispozici jsou REST webové služby, pracující s formátem JSON, pomocí nichž je možné kompletně spravovat jak tikety, tak konfigurace a další funkcionalitu. Formát dotazů a přímo přes webové rozhraní dostupná dokumentace celého systému s popisem tříd používaných objektů umožňuje i pomocí webových služeb komplexní správu systému. Dále je umožněno dotazování se na objekty service desku pomocí vlastního dotazovacího jazyka (velmi podobného SQL). Dotazy je užitečné a někdy i nutné v některých případech použít při volání webových služeb.

Freshservice

Freshservice Service Desk je v základu cloudová aplikace, není tedy třeba instalace žádných aplikací. Nabízí čtyři verze service desku. Ve volně dostupné verzi (Sprout) nabízí Incident Management a kompletní správu CMDB (další procesy jsou až v placených verzích). Ve volně dostupné verzi jsou k dispozici maximálně tři agenti (technici). Výhodou Freshservice je jednoduchost a přehlednost. [9]

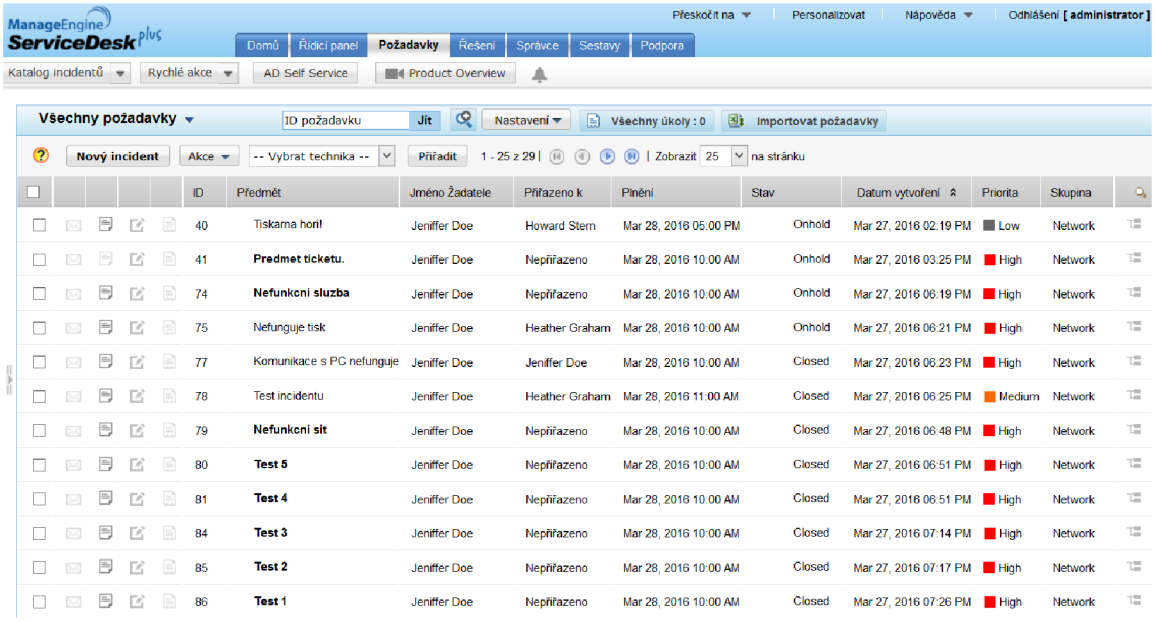


Obrázek 4.4: Webové rozhraní Freshservice.

Freshservice nabízí REST webové služby a pracuje také s formátem JSON. K dispozici jsou metody pro kompletní správu tiketů a konfiguračních položek v CMDB.

ManageEngine ServiceDesk Plus

Service desk od ManageEngine lze provozovat opět jako cloudovou aplikaci nebo lze nainstalovat na vlastní server. Volně dostupná verze obsahuje IT Help Desk s Incident Managementem a jedním technikem. Tato verze je tedy vhodná jen pro zkušební, případně akademické účely. CMDB databáze není ve volně stažitelné verzi dostupná. [17]

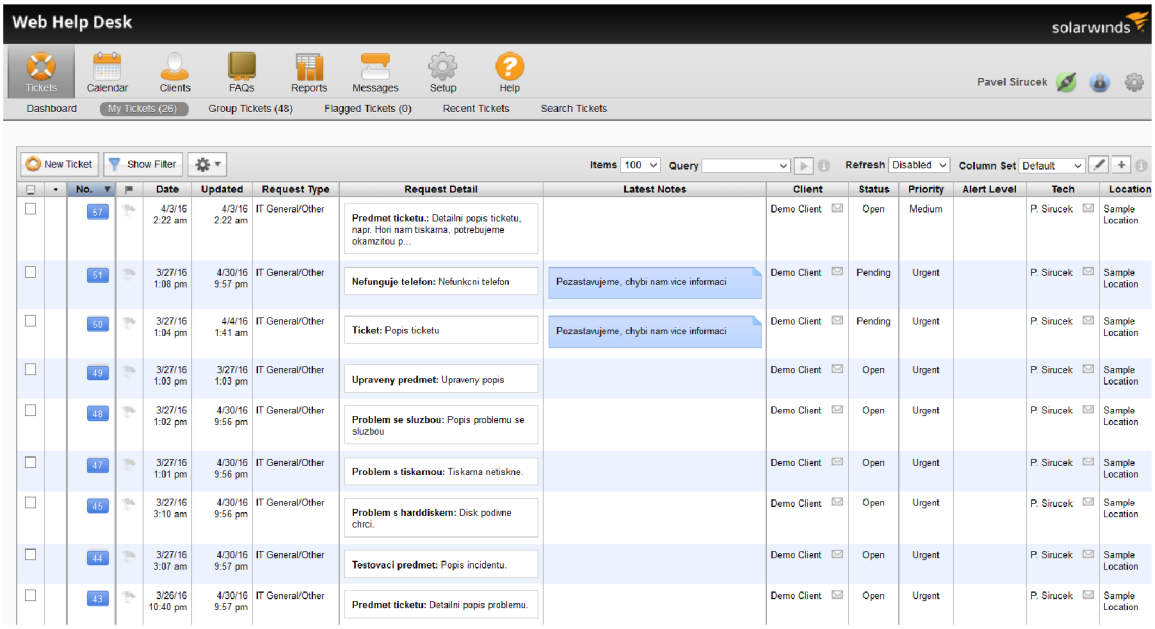


Obrázek 4.5: Webové rozhraní ManageEngine ServiceDesk plus.

Service desk nabízí REST webové služby, pracující tentokrát s formátem XML. Opět jsou k dispozici operace pro kompletní správu ticketů.

Solarwinds Web Help Desk

Help desk od Solarwinds ve volně dostupné verzi nabízí Incident Management. Aplikaci je možné nainstalovat na vlastní server nebo využít cloud. K dispozici je jeden technik. CMDB databáze není ve volně stažitelné verzi dostupná. [16]



Obrázek 4.6: Webové rozhraní Web Help Desk.

Help desk disponuje REST webovými službami pro kompletní správu tiketů.

Shrnutí

Vybrané service desky umožňují spravovat data (tikety, CMDB) pomocí webových služeb, pouze u ALVAO Service Desk je nutné přímo přistupovat k databázi. Všechny čtyři hlavní procesy podporují pouze ALVAO a iTop, u ostatních je k dispozici pouze Incident Management.

Další existující systémy většinou neumožňovaly provozovat bezplatnou verzi. U volně dostupných systémů zase často nebylo možné z externí aplikace provádět operace nad daty. Z dalších zajímávých volně dostupných nebo open source řešení je možné zmínit Project Open, který obsahuje mnoho modulů pro kompletní správu IT služeb. Nebo systém OTRS, fungující na podobné bázi jako Project Open, kdy je možné instalovat nezávisle na sobě různé moduly pro různou správu služeb IT. U obou těchto systémů je však problém s webovými službami, kdy nebylo možné dostatečně spravovat požadované procesy.

Kapitola 5

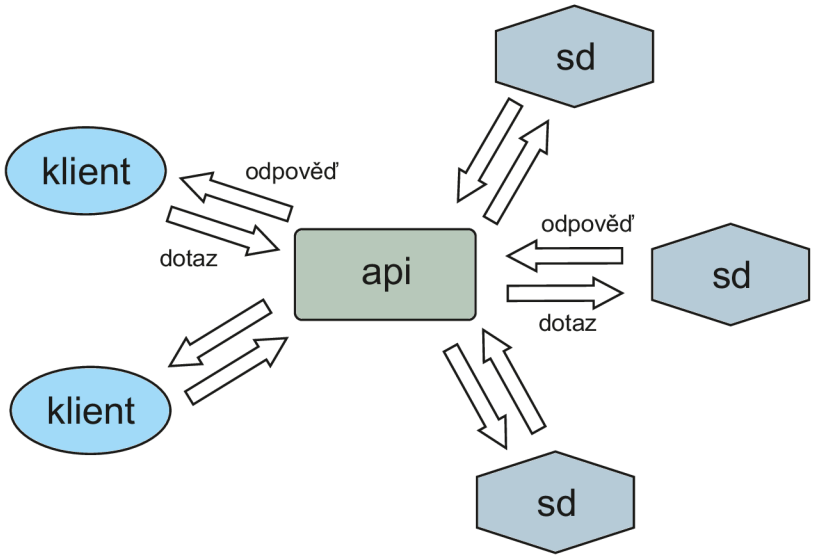
Návrh aplikačního rozhraní

V této kapitole bude představen návrh aplikačního rozhraní. Budou navrženy struktury dat, kdy je potřeba specifikovat požadované informace, jak o tiketech, tak o konfiguračních položkách. V další části budou popsány formáty požadavků a odpovědí, za pomoci kterých bude rozhraní komunikovat s příslušnými service desky. Poslední částí bude návrh komunikace mezi klienty a rozhraním, kdy bude třeba navrhnout vlastní formát požadavků a odpovědí.

Rozhraní, které má být výsledkem této práce, slouží jako mezibod při komunikaci klienta s konkrétním service deskem. Klient komunikuje s rozhraním, posílá dotaz a očekává odpověď. Rozhraní na požadavek klienta zformuluje dotaz, případně přistoupí přímo do databáze service desku, a získá odpověď. Odpověď je zformulována do obecné podoby a zaslána klientu. Následující příklad komunikace rozšiřuje příklad uvedený v 3.1:

- Klient komunikuje s rozhraním - zasílá požadavek na vytvoření tiketu - incidentu.
- V zaslaném požadavku bude specifikován konkrétní service desk, se kterým chce klient komunikovat.
- Požadavek od klienta na service desk je stejný, ať si klient vybere jakýkoliv service desk (jednotný formát pro všechny systémy).
- Rozhraní přijme požadavek a vytvoří požadovaného vnitřního klienta pro přístup ke konkrétnímu service desku.
- Rozhraní přeformuluje požadavek do formátu, kterému rozumí konkrétní service desk.
- Požadavek je odeslán a je přijata odpověď (úspěch či neúspěch).
- V dalším kroku zasílá klient požadavek na získání informací o vytvořeném incidentu (stejným způsobem).
- Rozhraní přeformuluje odpověď service desku s informacemi o incidentu do obecné formy, kterou zašle jako odpověď klientu.

Základní koncept komunikace dotaz-odpověď (request-response) je vidět na obrázku 5.1.

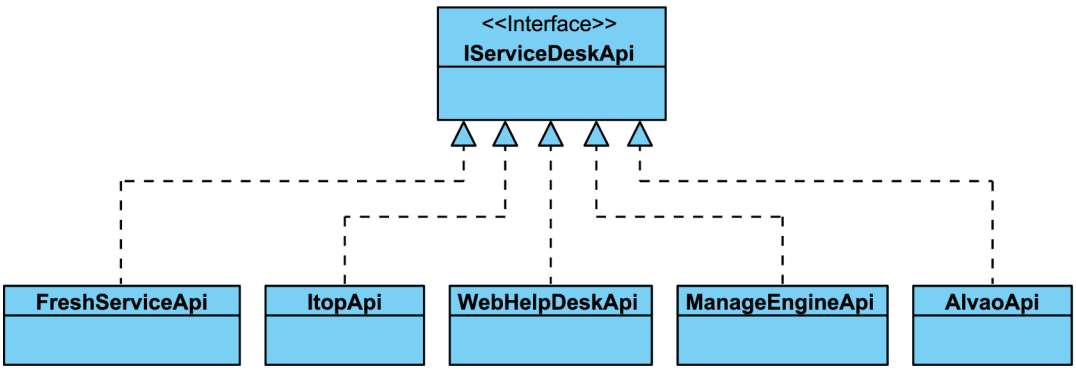


Obrázek 5.1: Princip komunikace klienta, rozhraní a service desk systémů.

Rozhraní by mělo být konzolovou aplikací, fungující bez závislosti na běhu další aplikace. Jako nejvhodnější způsob komunikace se jeví komunikace pomocí webových služeb. Komunikace je jednoduchá, na bázi dotaz-odpověď. Webové služby nejsou závislé na platformě a implementačním jazyku.

5.1 Základní struktura aplikačního rozhraní

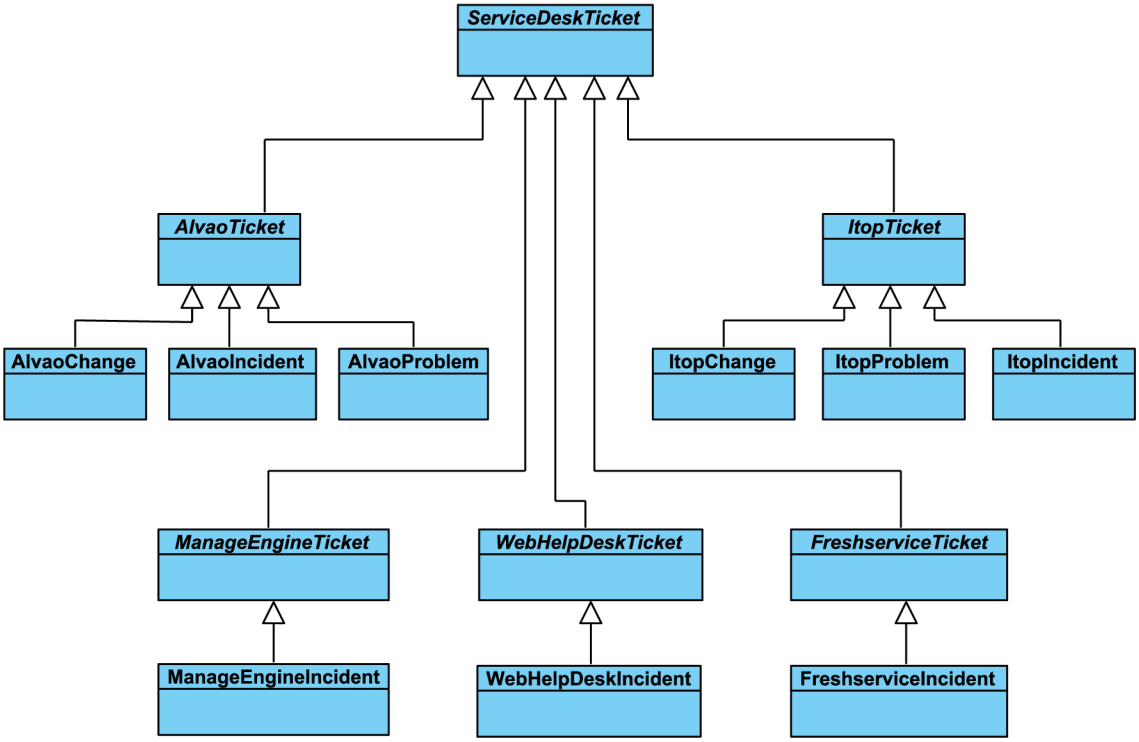
Čtyři z vybraných service desků umožňují komunikaci pomocí webových služeb. Rozhraní bude proto s těmito service desky komunikovat pomocí vlastního klienta webových služeb. Ve všech případech se jedná o REST webové služby. Požadavky a odpovědi budou implementovány v příslušných třídách, se kterými pak bude pracovat příslušný vnitřní klient rozhraní. Základní návrh je možné vidět na obrázku 5.2.



Obrázek 5.2: Model rozhraní. Zobrazuje konkrétní vnitřní klienty rozhraní (klienti webových služeb konkrétních SD systémů).

Všechny konkrétní rozhraní budou podporovat obecné operace s tikety. Tiket je v pod-

statě abstraktní třída, kdy konkrétními implementacemi jsou incident, problém a změna. Na obrázku 5.3 je znázorněno, jak by měla v konečném výsledku vypadat hierarchie tiketů. Na nejvyšší úrovni je již zmíněná obecná třída reprezentující tiket, nazvaná *ServiceDeskTicket*. Z ní jsou pak odvozeny tikety patřící k jednotlivým SD systémům a nakonec konkrétní typy tiketů.



Obrázek 5.3: Hierarchie tiketů v rozhraní.

Operace s jednotlivými tikety pak budou nsáledující:

- vytvoření tiketu,
- získání informací o konkrétním tiketu,
- úprava tiketu,
- změna stavu tiketu (specifická operace úpravy tiketu),
  - pozastavení řešení tiketu,
  - znovuotevření tiketu,
  - vyřešení tiketu,
  - uzavření tiketu,
- přidání poznámky/logu/zprávy.

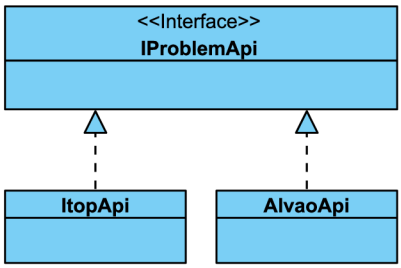
Přidání poznámky (zprávy, logu) bude důležité pro navázání dalších akcí, které bude provádět klient vytvářeného rozhraní. Pomocí poznámek lze specifikovat příkazy či operace, podle kterých pak klient provede určitou akci. V případě testovacího scénáře např. vloží jeden klient do zprávy příkaz k restartu služby. Druhý klient si zprávu přečte a pokusí se

restartovat uvedenou službu. Tento způsob komunikace má však jednu nevýhodu. Tou je skutečnost, že kromě jednoho ze SD systémů není možné určit tvůrce poznámky. Výsledkem je, že poznámky jsou vkládány vždy pod jedním uživatelem (podle přihlašovacích údajů k webovým službým service desku). To může následně komplikovat simulaci ITIL scénářů, protože výsledná komunikace v service desku vypadá jako od jednoho uživatele. Možností, jak tento problém částečně řešit, je například doplnění informace o uživateli přímo do těla zprávy, a tak odlišit uživatele vystupující v rámci scénáře.

U konkrétních typů tiketů nastává problém se skutečností, že všechny typy nejsou dostupné ve všech systémech. Kromě incidentů, které jsou dostupné vždy. Problémy a změny jsou dostupné jen ve dvou systémech. U problémů nastává potřeba mít dostupné další operace, týkající se pouze problémů. Bude proto nutné vytvořit samostatné rozhraní, které budou implementovat příslušní vnitřní klienti service desků, kde je správa problémů dostupná. Rozhraní pro práci s problémy by mělo poskytovat následující operace:

- propojení incidentu s problémem (v případě, že je problém zjištěn a určen jako příčina incidentu),
- získání informací o tom, které incidenty jsou k problému přidružené.

Rozhraní pro práci s problémy a příslušní vnitřní klienti, kteří jej budou implementovat jsou vidět na obrázku 5.4.



Obrázek 5.4: Rozhraní pro práci s problémy. Zobrazuje konkrétní vnitřní klienty implementující toto rozhraní.

Pro správu změn není třeba vytvářet další rozhraní, protože nejsou vyžadovány žádné specifické operace.

Nutné je ovšem vytvořit specifické rozhraní pro práci s konfiguračními položkami. Opět je to kvůli tomu, že správa konfigurací je dostupná pouze ve dvou service desk systémech. Operace které by mělo poskytovat rozhraní jsou následující:

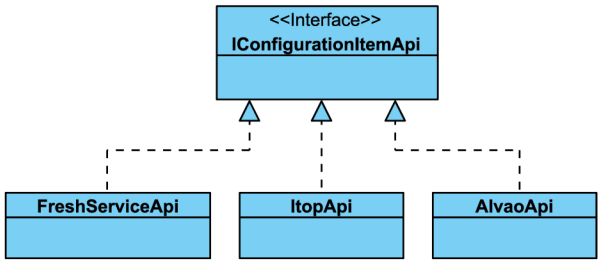
- získání informací o konkrétní konfigurační položce,
- úprava konfigurační položky,
- propojení konfigurační položky s konkrétním tiketem.

U rozhraní chybí operace pro vytvoření konfigurační položky. Je to z důvodu, že konfigurační položky obsahují informace v závislosti na tom, o jaký typ položky jde. Pro představu, u procesoru je možné nalézt parametr frekvence, který nebude u pevného disku. U disku bude zase kapacita, kterou nenalezneme u paměti apod. Vytváření položek všech dostupných typů by tedy znamenalo vytvořit komplexní právu položek stejně, jako je v příslušném service desku. Tohle se týká service desků iTop a Freshservice. ALVAO je specifické tím,

že konfigurační položky mohou obsahovat uživatelem definované parametry a tudíž každá položka, i když bude stejného typu, může mít naprosto odlišné parametry. Pro vysvětlení situace je možné uvést další příklad. Ve Freshservice obsahuje konfigurační položka typu pevný disk parametr *zbývající místo na disku*. V systému ALVAO je možné takový parametr manuálně přidat. Nakonec v systému iTop takový parametr není a nelze přidat, tudíž s ním není ani možné pracovat.

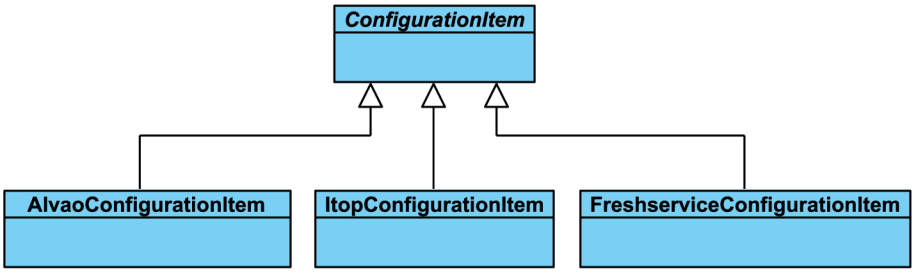
Kvůli výše zmíněným důvodům je jasné, že ani úprava konfiguračních položek nemůže být kompletní. Základní informace o položkách bude nutné vyčlenit a umožňovat je upravovat, detailní informace vázané přímo k typům položek pak bude možné upravovat pouze se znalostí formátu dat (klient rozhraní musí vědět přesný název parametru, který chce upravit).

Rozhraní pro práci s konfiguracemi a příslušní vnitřní klienti jsou vidět na obrázku 5.5.



Obrázek 5.5: Rozhraní pro práci s konfiguračními položkami. Zobrazuje konkrétní vnitřní klienty implementující toto rozhraní.

Na dalším obrázku 5.6 je vidět hierarchie konfiguračních položek. Na nejvyšší úrovni je obecná třída *ConfigurationItem*, ze které jsou odvozeny konfigurační položky jednotlivých service desků.



Obrázek 5.6: Hierarchie konfiguračních položek v rozhraní.

5.2 Struktura dat

Pro reprezentaci obecného tiketu v rozhraní je nutné specifikovat základní množinu informací, které bude rozhraní zpřístupňovat. Jde o informace, které budou dostupné v případě čtení tiketu ze service desku. Informace získáné ze SD systémů pomocí rozhraní (případně přečtené z databáze u ALVAO) se různí, základ však zůstává všude stejný nebo velmi podobný. Mnoho informací je pro tuto práci nepotřebných, proto budou ignorovány informace například o emailových adresách. Informace získatelné z rozhraní tedy určitě nebudou úplné, ale bezproblému dostačující pro další práci s procesy SD systémů. Následující výčet

uvádí informace, které budou k tiketům dostupné.

- typ tiketu (incident, problém nebo změna),
- identifikátor tiketu,
- předmět tiketu a popis,
- stav,
- priorita,
- uživatelé service desku,
  - žadatel (tvůrce tiketu),
  - řešitel (technik/agent).
- datumy (vytvoření tiketu, datum úpravy, deadline),
- související tikety - v případě problémů obsahuje informace o incidentech,
- související konfigurační položka - identifikátor konfigurační položky, ke které je tiket vztažen.

Priorita

Parametr priorita určuje přednost v řešení tiketů. Každý z podporovaných systémů má odlišné označení priority. Některé společné znaky je však možné najít. Většinou jsou dostupné 4 stupně priority, často doplněné o čas, který určuje, do kolika hodin je potřeba tiket vyřešit (může se různit podle SLA). Pro potřeby rozhraní je nutné označení priorit sjednotit, a to tak, aby byl co nejlépe zachován původní smysl priority. Jednotlivé service desky mají následující označení priorit:

- ALVAO - 5 stupňů: Planning, Low, Medium, High, Critical.
- iTop - 4 stupně: Low, Medium, High, Urgent. Prioritu ovšem nelze v systému měnit, lze však měnit parametr naléhavost (urgency), který prioritu ovlivňuje. Stupně naléhavosti jsou stejné jako stupně priorit, ale změny nejsou 1:1. Většinou je při nastavení naléhavosti nastavena priorita o stupeň vyšší (nastavení na high nastaví prioritu na critical, nastavení naléhavosti na critical už prioritu dále neovlivní).
- Freshservice - 4 stupně: Low, Medium, High, Urgent.
- ManageEngine - 4 stupně: Low, Normal, Medium, High.
- Web Help Desk - 4 stupně: Low, Medium, High, Urgent.

Po sjednocení do jednoho označení budou využity následující priority:

- kritická (critical) - pokrývá stupně critical a urgent (u ManageEngine high),
- vysoká (high) - pokrývá stupně high (u ManageEngine medium),
- normální (normal) - pokrývá stupně medium (u ManageEngine normal),
- nízká (low).

Stav

Velmi důležitý parametr označující stav řešení tiketu. V případě incidentů a problémů se stavy v systémech příliš neliší. Samostatnou kapitolou je správa změn, která má stavy odlišné od ostatních tiketů. Stavy tiketů (incidentů a problémů) budou následující:

- otevřený (open) - otevřený tiket, zahrnuje v sobě stavy new (nový), assigned (přiřazený) a reopened (znovu otevřený),
- pozastavený (on hold) - tiket, na kterém se z nějakého důvodu nepracuje,
- vyřešený (resolved) - řešitel skončil s řešením tiketu,
- uzavřený (closed) - řešení bylo schváleno a tiket byl uzavřen.

V případě tiketu typu změna je nutné doplnit další stavy, kterých tiket může nabývat. Systémy ALVAO a iTop mají odlišné označení stavů, proto je nutné stavy sjednotit do jednoho používaného formátu. Stavy použité u změn:

- analýza (analysis) - zahrnuje stav assigned (přidělený) v systému iTop,
- plánování (planning),
- implementace (implementation),
- monitorování (monitoring) - zahrnuje stav testing (testování změny) v systému ALVAO,
- obecné stavy tiketu vyřešený a uzavřený.

Další struktury je potřeba navrhnout pro operace vytvoření a úpravy tiketu. Při vytvoření je nutné zadat do každého ze service desků, kromě základních, ještě dodatečné informace, které se liší systém od systému. ALVAO tak například vyžaduje při vytváření tiketu zadání služby, které se tiket týká a SLA. Freshservice zase namísto jména uživatele vyžaduje buď jeho identifikátor nebo email. V iTop service desku je zase nutné při vytváření zadat název organizace, ve které je tiket vytvářen (systém může zahrnovat více organizací). Při úpravě dat je nutné zvolit pouze vlastnosti, u kterých je potřeba je nějakým způsobem měnit. Jedná se hlavně o stav tiketu, případně prioritu. Na druhou stranu třeba datum vytvoření není potřeba měnit (a nemusí to být ani žádoucí).

Konfigurační položky

Podobná situace jako u dat tiketů je i v případě konfiguračních položek. Opět je třeba vytvořit formát reprezentující konfigurační položku v rozhraní. Situace je o něco složitější kvůli různým parametrům jednotlivých typů položek (nebo parametrům vytvářených za běhu, viz ALVAO). Návrh je takový, že bude vybrána množina několika informací, které lze získat vždy a doplňující informace budou získatelné pomocí tabulky (mapy), kde bude vždy pod klíčem reprezentujícím parametr položky uvedena hodnota tohoto parametru. Struktura informací o konfigurační položce by tedy měla vypadat následovně:

- identifikátor konfigurační položky,
- typ konfigurační položky (hardware - tiskárna, procesor, . . . , software - aplikace, licence, . . .),

- název, výrobce, umístění,
- datum přidání do systému,
- stav v systému - stav užívání (rezerva, v produkci, zastaralé, ...),
- další informace, pod označením detaily (details) vázané přímo ke konkrétní položce nebo typu (velikost u pevného disku, frekvence u pamětí, ...) ve formě mapy (klíč - hodnota).

U systému ALVAO není zaručeno že budou i zákaldní informace dostupné. Pokud budou jinak pojmenované, tak bude nutné získat z výše zmíněné struktury s podrobnějšími informacemi (details). Struktura pro vytváření konfiguračních položek není nutná, protože přes rozhraní nebude možné položky vytvářet. Bude možné upravovat základní informace a se znalostí i detailní parametry. Idea je taková, že klient při požadavku na zobrazení informací o položce dostane kromě základních informací také detailní tabulku (mapu). Pokud bude chtít některý z parametrů v mapě upravit, bude nutné zadat do požadavku na úpravu strukturu s upravovanými parametry.

5.3 Komunikace mezi rozhraním a service desky

Pro umožnění komunikace mezi rozhraním a service desky pomocí webových služeb je nutné ke každému service desku vytvořit příslušnou strukturu pro požadavek a vrácenou odpověď. Pro komunikaci se systémy přes databázi (pouze ALVAO) je třeba vytvořit třídy, které se budou starat o komunikaci s databází a vykonávat SQL dotazy. V této sekci bude proto popsána komunikace s jednotlivými systémy.

Komunikace pomocí webových služeb

Pro každý jednotlivý požadavek a každou odpověď bude nutné vytvořit třídu. Třídy by měly požadavky a odpovědi realizovat 1:1. S využitím dalších nástrojů pak bude možné požadavky a odpovědi ve formátu JSON nebo XML převádět přímo do objektů a naopak. Obecný seznam požadavků a odpovědí, které budou muset být naimplementovány, je uveden v následujícím výčtu.

- Požadavek na vytvoření tiketu a příslušná odpověď. V případě odpovědi není nutné specifikovat formát, protože ten se nebude dále využívat. Je potřebné pouze z odpovědi získat identifikátor nově vytvořeného tiketu nebo informace, že tiket nebyl vytvořen.
- Požadavek na vrácení informací o tiketu a příslušná odpověď. Zde je nutné specifikovat hlavně vrácenou odpověď, protože z ní budou získány informace, kterými pak rozhraní disponuje a vrací je klientům.
- Požadavek na úpravu tiketu a příslušná odpověď. V tomto případě je nutné specifikovat požadavek. Z odpovědi stačí pouze zjistit, zda byla úprava provedena a s ostatními daty není potřeba dále pracovat.
- Požadavek na vložení poznámky a příslušná odpověď. Nutné je specifikovat požadavek na vytvoření poznámky, z odpovědi stačí opět zjistit, zda bylo vytvoření úspěšné či nikoliv.

- Požadavek na vrácení informací o konfigurační položce a příslušná odpověď. Týká se pouze systémů iTop a ALVAO. Je nutné specifikovat požadavek i odpověď, ze které budou získány informace o konfigurační položce.
- Požadavek na úpravu konfigurační položky a příslušná odpověď. Stačí specifikovat požadavek a z odpovědi získat pouze informaci o tom, zda byla úprava provedena.

Komunikace přes databázi

Pro komunikaci s databází bude potřeba vytvořit DAO rozhraní a ta následně implementovat. Metody, které bude DAO zpřístupňovat by měly funkcí odpovídat webovým službám. Pro vytvoření tiketu se tak zavolá metoda DAO pro vytvoření tiketu, která vrátí buď strukturu dat, ze které bude vytvořen tiket nebo přímo tiket a podobně.

Komunikace se systémem ALVAO

Při návrhu rozhraní pro komunikaci se systémem ALVAO je nejprve třeba zjistit, kde jsou uložena požadovaná data. Informace o tiketu jsou v ALVAO databázi uloženy ve více tabulkách, takže je třeba nejlépe tabulky spojit pomocí SQL JOIN a získat všechna potřebná data. Další tabulky je potřeba připojit pro získání informací o stavu, prioritě, uživatelích a poznámkách. Samotná tabulka s tiketem obsahuje většinou cizí klíče (identifikátory) do tabulek s příslušnými informacemi.

DAO rozhraní by měly být ve výsledku dvě. Jedno pro práci s tikety a druhé pro práci s konfiguračními položkami.

Při získávání tiketu bude vhodné přečíst všechny sloupce k hledanému tiketu. I když se nebudou všechny využívat, může to posloužit v případných rozšířeních.

Pro operace s konfiguračními položkami se využívá předpřipravený databázový pohled. Ten shrnuje většinu požadovaných informací, není proto třeba spojovat mnoho tabulek. Pro detailnější informace o vlastnostech konfigurační položky stačí přečíst už jen jednu další tabulku.

Komunikace se systémem iTop

Se systémem iTop se komunikuje pomocí webových služeb. Ve všech případech je použita metoda `POST`. Struktura dat je ve formátu JSON. V případě požadavků se ale nejedná o validní JSON, protože iTop vyžaduje před JSON data přidat řetězec `json_data=`. Pokud by se požadavek odesílal jako JSON, volání webové služby by selhalo. Požadavek je třeba odeslat jako data formuláře, tedy `x-www-form-urlencoded`.

Požadavky mají vždy stejný tvar. Je třeba specifikovat několik položek. Operace určuje požadovanou akci iTop rozhraní (`core/create` pro vytvoření objektu, `core/get` pro vrácení objektu nebo `core/update` pro úpravu objektu). Třída určuje typ objektu, ke kterému se operace vztahuje (incident jako `Incident`, problém jako `Problem`, změnu jako `RoutineChange` a konfigurační položku obecně jako `FunctionalCI` nebo konkrétní typ. Povinný je ještě komentář, který se ale poté nikde neobjevuje. Protože jsou požadavky pro všechny objekty stejné, je možné vytvořit třídu předka, která bude definovat hlavní strukturu a v konkrétních třídách pak specifikovat formáty pro jednotlivé typy objektů.

Při vytváření tiketů je nutné základní informace, potřebné pro vytvoření, zadat do dalšího objektu, který je na stejné úrovni jako výše uvedené. Pokud má nově vytvářený tiket obsahovat informaci o konfigurační položce, tak je nutné identifikátor uvést do pole.

Z toho je zřejmé, že iTop umožňuje k tiketů přiřazovat více konfiguračních položek. Bohužel je jediný ze service desků s touto vlastností. Příklad toho, jak může požadavek na vytvoření tiketu vypadat je uveden v následující ukázce kódu 5.1.

```
{
   "operation": "core/create",
   "comment": "Vytvoreni problemu skrze rest api",
   "class": "Incident",
   "fields":
   {
      "org_id": "SELECT Organization WHERE name = 'Demo'",
      "caller_id" : "SELECT Person WHERE id = 1",
      "agent_id" : "SELECT Person WHERE id = 1",
      "priority": "4",
      "urgency": "4",
      "title": "Predmet incidentu",
      "description": "Popis incidentu"
   }
}
```

Ukázka kódu 5.1: Příklad požadavku na vytvoření nového tiketu v systému iTop.

Při získávání tiketů nebo konfiguračních položek je nutné specifikovat, které informace o konkrétním objektu jsou vyžadovány. Pro všechny informace stačí uvést znak *. Tikety obsahují také jednotlivé zprávy z komunikace. Zprávy jsou buď privátní nebo veřejné. U incidentů budou používány veřejné zprávy, u ostatních typů privátní, protože veřejný záznam není k dispozici. Příklad odpovědi na požadavek získání tiketu je zobrazen v ukázce 5.2.

```
{
   "objects": {"Incident::100":  {
      "code": 0,
      "message": "",
      "key": "100",
      "fields":       {
         "ref": "I-000100",
         "org_id": "3",
         "org_name": "Demo",
         "caller_id": "1",
         "caller_name": "Sirucek",
         "agent_id": "1",
         "agent_name": "Sirucek",
         "title": "Testovaci incident",
         "description": "Testovaci incident c. 100",
         "start_date": "2016-04-17 17:54:41",
         "end_date": "",
         "last_update": "2016-04-17 17:54:48",
         "close_date": "",
         "private_log": {"entries": []},
         "functionalcis_list": [],
         "status": "new",
         "impact": "1",
         "priority": "1",
         "urgency": "2",
         "assignment_date": "",
         "resolution_date": "",
         "last_pending_date": "",
         "pending_reason": "",
         "parent_problem_id": "0",
         "parent_problem_ref": "",
         "public_log": {"entries": []},
         "finalclass": "Incident",
      }
   }},
   "code": 0,
   "message": "Found: 1"
}
```


Ukázka kódu 5.2: Příklad odpovědi (tiketu) ze systému iTop.

Tiket typu problém navíc obsahuje pole identifikátorů souvisejících incidentů (problém je identifikován jako příčina těchto incidentů).

Konfigurační položku je možné získat za použití buď obecného názvu typu nebo konkrétní třídy položky. Pokud je použit obecný název, tak jsou vráceny pouze obecné informace, které mají všechny konfigurační položky. Pro získání konkrétních informací je tedy nutné uvést konkrétní třídu objektu (v odpovědi označena jako `finalclass`). Pro tiskárnu je třída `Printer`, pro počítač `PC` a podobně. Třídy typu počítač, laptop a podobné mohou navíc obsahovat pole dalších konfiguračních položek - nainstalovaného software. V SD

systému iTop je zajímavě vyřešené vytvoření vazby mezi tiketem a konfigurační položkou. Nestačí totiž provést úpravu tak, že by se k datům tiketu přidal identifikátor konfigurační položky. Pro tuto vazbu je nutné vytvořit nový objekt třídy `lnkFunctionalCIToTicket`. Protože se vytváří nový objekt, jde o operaci `core/create`.

Komunikace se systémem Freshservice

Komunikace se systémem Freshservice probíhá pomocí webových služeb. Požadavky a odpovědi jsou ve validním JSON formátu. Při volání webových služeb jsou využívány HTTP metody POST (vytváření tiketů), PUT (úpravy tiketů a konfiguračních položek) a GET (získání tiketů nebo konfiguračních položek). HTTP metoda DELETE je využita při mazání tiketů ze systému. Freshservice pracuje pouze s tikety typu incident.

Při vytváření tiketu je možné místo identifikátoru žadatele zadat email. Podle emailové adresy je pak přiřazen k tiketu správný uživatel.

První zvláštností Freshservice je nutnost uvedení jména konfigurační položky při její úpravě. Zadání identifikátoru nestačí (navíc je identifikátor konfigurační položky ignorován), webové služby vrací v tomto případě v odpovědi informaci o tom, že nebylo zadáno jméno konfigurační položky. Jména ale nejsou ani jedinečná, systém může obsahovat několik stejných a stejně pojmenovaných položek. Při vyhledávání podle jména pak webové služby vrací vždy první z položek a stejně tak při úpravě. Pokud se konfigurační položka přidává k tiketu při jeho vytváření, je možné ji uvést jak jménem, tak identifikátorem.

Další odlišností je fakt, že při úpravě tiketu nejde změnit jeho popis. Ten tak zůstává pořád stejný. Předmět tiketu však měnit lze.

Ukázka požadavku pro vytvoření tiketu a odpovědi na žádost o získání tiketu jsou vidět v ukázkách 5.3 a 5.4.

```
{
  "helpdesk_ticket": {
    "description": "Nefunguje tisk",
    "subject": "Tiskarna netiskne",
    "email": "sirucek@sdapi.freshservice.com",
    "requester_id": 4000215144,
    "priority": 2,
    "status": 2,
    "source": 2,
    "ticket_type": "Incident"
  }
}
```

Ukázka kódu 5.3: Příklad požadavku na vytvoření nového tiketu v systému Freshservice.

```
{"helpdesk_ticket": {
    "created_at": "2016-05-22T21:01:22+02:00",
    "deleted": false,
    "description": "Nefunguje tisk",
    "display_id": 10,
    "due_by": "2016-05-23T17:00:00+02:00",
    "frDueBy": "2016-05-23T16:00:00+02:00",
    "id": 4000261872,
    "isescalated": false,
    "notes": [],
    "owner_id": null,
    "priority": 2,
    "requester_id": 4000215144,
    "source": 2,
    "status": 2,
    "subject": "Tiskarna netiskne",
    "ticket_type": "Incident",
    "updated_at": "2016-05-22T21:01:22+02:00",
    "urgent": false,
    "status_name": "Otevreno",
    "requester_status_name": "Probiha zpracovani",
    "priority_name": "Stredni",
    "source_name": "Portal",
    "requester_name": "Pavel Sirucek",
    "responder_name": "No Agent",
    "department_name": "IT",
}}
```

Ukázka kódu 5.4: Příklad odpovědi (tiketu) ze systému Freshservice.

Komunikace se systémem ManageEngine

ManageEngine využívá ke komunikaci za pomocí webových služeb formát XML. Jde o jediný podporovaný service desk, který XML využívá. Požadavky nejsou validním XML, protože podobně jako iTop, vyžaduje ManageEngine uvedení řetězce `INPUT_DATA` před obsah XML. Opět jsou tedy požadavky odesílány jako formulářová data (`x-www-form-urlencoded`). Webové služby využívají HTTP metody POST, PUT a GET a DELETE. Nicméně je nutné ještě přidat do URL webových služeb parametr `OPERATION_NAME`, který definuje typ operace. Pro získání tiketu je to `GET_REQUEST`, pro vytvoření `ADD_REQUEST`, pro úpravu `EDIT_REQUEST` a nakonec pro vložení zprávy nebo poznámky k tiketu `ADD_NOTE` .

Při vytváření tiketu se nezadává uživatel, který tiket vytváří. Ten je identifikován podle api klíče. Kromě požadavku na vytvoření tiketu, který obsahuje tagy pojmenované podle názvů vlastností (ukázka 5.5), se využívá struktury parametrů, které vždy obsahují dvě hodnoty. První je název parametru a druhou je hodnota parametru. Tento formát lze vidět v ukázce 5.6, která představuje odpověd na žádost o tiket. Odpověď je zkrácená a obwsahuje pouze informace v elementu `Details`. V odpovědi také není přímo komunikace k tiketu (zprávy nebo poznámky). Namísto toho obsahuje odpověď URL, odkud je možné komunikaci získat.

```
<Operation>
    <Details>
        <description>Nefunguje tiskarna</description>
        <group>Printer Problems</group>
        <priority>Normal</priority>
        <status>Open</status>
        <subject>Nelze tisknout</subject>
        <technician>Jeniffer Doe</technician>
    </Details>
</Operation>
```

Ukázka kódu 5.5: Příklad požadavku na vytvoření tiketu v systému ManageEngine.

```
<Details>
 <parameter><name>workorderid</name><value>40</value></parameter>
 <parameter><name>requester</name><value>Jeniffer Doe</value></parameter>
 <parameter><name>createdby</name><value>Jeniffer Doe</value></parameter>
 <parameter><name>createdtime</name><value>145908114</value></parameter>
 <parameter><name>duebytime</name><value>1463990400</value></parameter>
 <parameter><name>responseduebytime</name><value>-1</value></parameter>
 <parameter><name>fr_duetime</name><value>-1</value></parameter>
 <parameter><name>respondedtime</name><value>0</value></parameter>
 <parameter><name>resolvedtime</name><value>0</value></parameter>
 <parameter><name>completedtime</name><value>0</value></parameter>
 <parameter><name>subject</name><value>Problem se siti</value></parameter>
 <parameter><name>sla</name><value>High SLA</value></parameter>
 <parameter><name>isvipuser</name><value>No</value></parameter>
 <parameter><name>category</name><value>Network</value></parameter>
 <parameter><name>status</name><value>Onhold</value></parameter>
 <parameter><name>priority</name><value>High</value></parameter>
 <parameter><name>level</name><value>Tier 3</value></parameter>
 <parameter><name>group</name><value>Network</value> </parameter>
 <parameter><name>description</name><value>Popis tiketu</value></parameter>
 <Notes URI="http://localhost:8080/sdpapi/request/40/notes/"/>
</Details>
```

Ukázka kódu 5.6: Příklad odpovědi (tiketu) ze systému ManageEngine.

Komunikace se systémem Web Help Desk

Web Help Desk využívá validní JSON formát. Webové služby opět využívají metody POST, PUT, GET a DELETE. Formát požadavků a odpovědí je pravděpodobně nejkomplikovanější že všech service desků. Namísto jednoduchých parametrů s hodnotou, jako u iTop nebo Freshservice, je často nutné uvádět komplexní typ, ve kterém jsou informace uzavřeny.

Při vytváření tiketu je možné uvést parametr, který vypne automatické rozesílání emailů (defaultně je zapnuté). Některé informace vyžadují použití právě komplexního typu jako obalujícího objektu. Jde například o stav nebo prioritu. Stav vypadá tak, že obalující objekt `statustype` obsahuje objekt `type` s hodnotou `StatusType` a teprve pak objekt `id` s hodnotou stavu. V požadavku je také nutné zadat lokaci, ke které se tiket váže a typ problému. Tento parametr nemá nic společného se správou problémů, ale označuje typ IT služby, ke které tiket patří.

Formát požadavku na vytvoření tiketu je vidět v ukázce 5.7 a formát odpovědi na získání tiketu je v ukázce kódu 5.8. Odpověd je opět zkrácená.

```
{
  "subject": "Nefunguje program Dia",
  "detail": "Program prestal fungovat",
  "assignToCreatingTech": false,
  "sendEmail": false,
  "clientReporter": {
        "type": "Client",
        "id": 1
  },
  "clientTech": {
    "type": "Tech",
    "id": 1
  },
  "statustype": {
    "type": "StatusType",
    "id": 1
  },
  "prioritytype": {
    "type": "PriorityType",
    "id": 3
  },
  "problemtype": {
    "type": "ProblemType",
    "id": 3
  }
}
```

Ukázka kódu 5.7: Požadavek na vytvoření nového tiketu v systému Web Help Desk.

```
{
    "id": 43,
    "type": "Ticket",
    "lastUpdated": "2016-04-30T21:57:34Z",
    "locationId": 1,
    "priorityTypeId": 1,
    "statusTypeId": 1,
    "subject": "Predmet ticketu",
    "clientTech":    {
       "id": 1,
       "displayName": "Pavel Sirucek"
    },
    "prioritytype":    {
       "id": 1,
       "type": "PriorityType",
       "priorityTypeName": "Urgent"
    },
    "problemtype":    {
       "id": 3,
       "type": "RequestType",
       "detailDisplayName": "IT General/Other"
    },
    "statustype":    {
       "id": 1,
       "type": "StatusType",
       "statusTypeName": "Open"
    },
    "detail": "Detailni popis problemu.",
    "reportDateUtc": "2016-03-26T21:40:03Z",
    "displayDueDate": "2016-03-28T17:00:00Z",
    "displayClient": "Demo Client",
    "ticketEditable": true,
    "techId": 1,
    "clientId": 1,
    "canEscalate": true,
    "isDeleted": false,
    "notes": [],
}
```


Ukázka kódu 5.8: Příklad odpovědi (tiketu) ze systému Web Help Desk.

Zajímavostí u SD systému Web Help Desk je operace pro přidání zprávy k tiketu. Přes tuto operaci lze totiž měnit přímo i stav tiketu. Lze tak tuto operaci použít například pro vyřešení tiketu a přidání zprávy o vyřešení v jednom volání webové služby.

5.4 Komunikace mezi klientem a rozhraním

Komunikace bude probíhat za pomoci webových služeb. Aplikace bude fungovat jako server odpovídající na požadavky. V této části bude specifikován formát požadavků a odpovědí,

které budou použity v komunikaci mezi klientem a rozhraním. Formát bude jednotný, ať už chce klient komunikovat s kterýmkoliv service deskem. Samozřejmě budou existovat výjimky, kdy pro komunikaci s jedním service deskem bude nutné zadat další parametr, který se v jiné komunikaci nevyskytuje (typicky komunikace s ALVAO a zadání SLA nebo služby). Zadání těchto parametrů (např. omylem) při komunikaci s jiným service deskem by však nemělo komunikaci vůbec ovlivnit (rozhraní bude pracovat pouze s parametry požadovanými konkrétním SD systémem).

Nejprve bude nutné vybrat metody webových služeb, pomocí kterých se bude s rozhraním komunikovat. Při tvoření objektů by to měla být HTTP metoda `POST`. Pří požadavku na získání dat je možné využít metodu `GET`, kdy budou požadované parametry zadány jako parametry URL nebo budou součástí resource webové služby. Lepší možností je však pravděpodobně využití metody `POST`. Parametrů může být totiž víc a požadavek bude přehlednější než URL s parametry. K úpravám bude využita metoda `PUT`.

Vytvoření tiketu

Při vytváření (tedy vkládání nového tiketu do service desku) musí formát dat odpovídat datům nutným pro vytvoření tiketu v service desku. Seznam informací, které budou požadované, bude vypadat následovně:

- název service desku (`system`) - jedinečný název identifikující service desk
- typ tiketu (`ticketType`) - typ tiketu, povolené hodnoty budou `Incident` pro incident, `Problem` pro problém a `Change` pro změnu
- předmět tiketu (`subject`)
- popis tiketu (`description`) - informace o incidentu, problému apod.
- priorita tiketu (`priority`) - priorita je ignorována v případě tiketu typu *změna*
- stav tiketu (`status`) - v případě, že chce klient nastavit hned při vytvoření status na nějaký jiný, než defaultní
- id konfigurační položky (`configItemId`) - ignorováno v případě, že service desk nepodporuje správu konfigurací
- zadavatel tiketu - uvedení buď jména (`requesterName`) nebo identifikátoru (`requesterId`), ignorováno v případě ManageEngine, různé systémy mohou brát v úvahu pouze jeden z parametrů
- řešitel tiketu - uvedení buď jména (`technicianName`) nebo identifikátoru (`technicianId`) řešitele tiketu, různé systémy mohou brát v úvahu pouze jeden z parametrů

Kromě těchto položek je nutné pro specifické systémy zadat doplňující informace. Jak bylo již řečeno, pro komunikaci se systémem ALVAO je nutné zadat SLA (`sla`) a název služby (`service`). Pro iTop systém je nutné zadat název organizace (`organization`). V následující části budou prezentovány příklady formátů požadavků a odpovědí. Další příklady (požadavků) je možné nalézt v příloze C.

V ukázce 5.9 je uveden příklad požadavku na vytvoření tiketu.

```
{
  "configItemId": 64,
  "system": "itop",
  "requesterId": 1,
  "subject": "Pomaly PC",
  "organization": "Demo",
  "description": "PC je velmi pomale a stale zamrzava.",
  "ticketType": "Incident",
  "priority": "normal",
  "technicianId": 1,
  "status": "open"
}
```

Ukázka kódu 5.9: Příklad požadavku na vytvoření tiketu přes rozhraní.

Odpověď na požadavek obsahuje jediný parametr `success`, který bude nabývat hodnoty `true` nebo `false`, podle toho zda se tiket povedlo vytvořit.

Získání tiketu

Pro získání dat byla nakonec byla zvolena metoda `POST`, protože parametry je nutné zadat minimálně tři a strukturovaný požadavek je přehlednější. Formát zobrazuje ukázka 5.10. Požadované parametry by měly být název systému, typ tiketu a identifikátor požadovaného tiketu.

```
{
  "system": "Alvao",
  "ticketType": "Incident",
  "ticketId": 372
}
```

Ukázka kódu 5.10: Příklad požadavku na získání tiketu přes rozhraní.

Odpověď na požadavek zahrnuje informace popsané v sekci o struktuře dat na straně 29. Formát odpovědi zahrnuje informace zadané při vytváření tiketu a další informace, které rozhraní doplní z informací získaných od service desku, mimo jiné seznam poznámek (zpráv) k tiketu uvedený v poli objektu `notes`.

Úprava tiketu

Úprava tiketu bude probíhat za pomoci metody `PUT`. Formát požadavku pro úpravu tiketu je stejný jako formát dat pro vytvoření tiketu. Je nutné zadat parametry, které mají být změněny, s novými hodnotami.

Odpověď bude obsahovat opět pouze jediný parametr, informující o úspěšné úpravě dat.

Přidání poznámky

Metoda webové služby bude v tomto případě `PUT`, protože jde v podstatě o úpravu tiketu. Přidání poznámky bude velmi jednoduché, požadavek bude vypadat stejně jako požadavek na získání tiketu, pouze přibude parametr `note` s textem poznámky.

Odpověď bude opět pouze parametr `success` s hodnotou `true` nebo `false`.

Získání konfigurační položky

Požadavek na získání konfigurační položky bude obsahovat kromě názvu service desku, pouze identifikátor konfigurační položky `configurationItemId`.

Formát odpovědi je složitější. Základní informace budou pro konfigurační položky stejné. Půjde o název (`name`), detailní popis (`description`), identifikátor, status používání (`status`) a dopad na obchod (`impact`). Další informace, které už ale nemusí být přítomny jsou název typu konfigurační položky (`typeName`), identifikátor typu konfigurační položky (`typeId`), datum přidání do systému (`createdDate`) a název produktu (`productName`). Poslední položkou bude komplexní objekt (`details`), který bude obsahovat detailní informace o konfigurační položce, které jsou vázané buď na typ položky nebo jsou manuálně přidané (např. uživatelem v SD systému ALVAO). Ukázka odpovědi je vidět v ukázce 5.11. Zároveň je v ukázce vidět nevýhodu ALVAO systému, a to skutečnost, že konfigurační položka se v tomto případě jmenuje stejně jako velikost paměti. Pod tímto názvem je totiž uvedena v databázi.

```
{
  "details": {
    "Maximalni velikost": "1024 MB",
    "Slotu": "2",
    "Obsazenych slotu": "1",
    "Velikost": "1024 MB"
  },
  "name": "1024 MB",
  "id": 152
}
```

Ukázka kódu 5.11: Příklad odpovědi (konfigurační položky) ze systému ALVAO.

Úprava konfigurační položky

V požadavku bude nutné specifikovat nové hodnoty, a to u základních parametrů (viz popis odpovědi na získání konfigurační položky). Pokud bude požadována úprava některého z parametrů, které jsou v odpovědi v objektu `details`, je nutné použít podobný formát. Ten nebude mít tvar komplexního objektu s parametry, ale půjde o pole `details`. Pole musí mít sudý počet prvků. Prvek na sudé pozici určuje název parametru a prvek na liché pozici pak hodnotu tohoto parametru (uvedeného jako předchozí prvek pole). Pole bylo využito proto, že v implementaci webové služby se pak jednoduše převede na mapu. Použití přímo mapy v požadavku přináší problémy s nerozpoznáním komplexního objektu. Příklad, jak může takový požadavek vypadat je uveden v ukázce kódu 5.12.

```
{
  "configItemId": 152,
  "system": "Alvao",
  "details": [
    "Velikost",
    "2 GB"
  ]
}
```

Ukázka kódu 5.12: Požadavek na úpravu konfigurační položky.

Odpověď bude pouze parametr `success` s hodnotou `true` nebo `false`.

Kopírování tiketu mezi service desky

Zvláštní požadavek je třeba pro zkopírování tiketu z jednoho systému do druhého. Nejedná se přímo o kopii. Z jednoho systému se získá tiket, jehož data jsou použita při vytvoření tiketu v druhém systému. V požadavku je nutné zadat oba systémy (`srcSystem` a `destSystem`), identifikátor tiketu ve zdrojovém systému (`ticketId`) a volitelně konfigurační položku v cílovém systému. Při „kopírování“ se nepřenáší konfigurační položka, protože cílový systém může mít jinou konfigurační databázi a položku nemusí obsahovat nebo může mít jiné identifikační číslo. Příklad požadavku je vidět v ukázce 5.13. Odpovědí pak bude příznak, zda se podařilo tiket „zkopírovat“.

```
{
  "requesterId": 1,
  "organization": "Demo",
  "destSystem": "itop",
  "ticketType": "Incident",
  "srcSystem": "Alvao",
  "technicianId": 1,
  "ticketId": 369
}
```

Ukázka kódu 5.13: Požadavek na zkopírování tiketu.

Kapitola 6

Popis implementace

Tato kapitola se zabývá vlastní implementací rozhraní. Popisuje použité technologie a fungování rozhraní.

6.1 Aplikační rozhraní jako open source

Výsledné rozhraní je zveřejněno jako open source připravené pro další případný vývoj a rozšíření. Repozitář je dostupný na adrese `https://github.com/xsiruc01/sdapi`, odkud je možné jej stáhnout (naklonovat).

6.2 Zvolené technologie

Jako implementační jazyk byla zvolena Java. Aplikace bude tímto přenositelná na různé operační systémy. Požadavkem je pouze nainstalované běhové prostředí pro Javu (jre). Samotný vývoj aplikace probíhal ve vývojovém prostředí Netbeans.

Projekt využívá Maven jako nástroj pro správu buildů aplikace. Výhoda Mavenu spočívá v tom, že lze projekt velmi lehce naimportovat do jiného vývojového prostředí (ať už do Netbeans na jiné stanici nebo jiného IDE, kde lze naimportovat Maven projekt). Maven se postará o všechny závislosti, které stáhne a projekt by měl jít bezproblému sestavit.

Pro implementaci komunikace mezi rozhraním a service desky byl použit Jersey RESTful Web Services framework. Jersey je open source, využitelný k tvorbě REST webových služeb.

Při odesílání požadavků do service desku je nutné požadavek převést do správného formátu. V případě JSON formátu byl využit framework GSON, který umožňuje jednoduchou práci při převodu objektů na JSON formát a naopak.

Pro komunikaci se systémem ALVAO, kdy se komunikuje s databází byl využit open source JDBC ovladač jménem jTDS, určený pro komunikaci s Microsoft SQL databázemi.

Jelikož má rozhraní fungovat také jako webová služba, přijímající požadavky od klientů, bylo potřeba zvolit vhodný nástroj, který by umožnil provozovat webové služby a přitom nepotřeboval instalaci nebo provoz dalších aplikací (typicky některý z webových kontejnerů nebo aplikační server). Důležité je, aby výsledná aplikace zůstala samostatným spustitelným souborem (doplněným pouze o konfigurační soubory). Vhodným řešením se proto stalo použití Grizzly Web Serveru. Jde o webovou službu v Javě, která používá komponenty GlassFish serveru. Největší předností je právě fakt, že výsledkem je stále samotný .jar soubor, spustitelný kdekoliv.

Při implementaci byl dále využit framework lombok, který přináší hlavní výhodu v zpřehlednění kódu. Díky anotacím, kterými stačí anotovat třídu, se nemusí psát nebo generovat metody pro nastavování a čtení atributů objektů (gettery a settery). Výhoda je zřejmá u tříd, které reprezentují odpovědi ze service desků. Třídy obsahují několik desítek attributů a díky anotaci je jasné, že každý atribut bude mít vygenerován get a set metodu a přitom kód zůstane velmi krátký a přehledný.

Při testování komunikace s webovými službami, ať už mezi klienty a rozhraním nebo mezi rozhraním a SD systémy se velmi osvědčila aplikace SoapUI.

6.3 Popis komunikace uvnitř rozhraní

Následující část popisuje podrobněji akce prováděné při komunikaci s rozhraním. Komunikaci zahajuje vždy klient zasláním požadavku na rozhraní. Podle typu požadavku se rozhraní zachová následujícími způsoby:

- požadavek na vytvoření tiketu
  - Z požadavku je zjištěno, zda obsahuje povinné informace. Pokud ne, rozhraní vrací odpověď s příznakem neúspěchu. Podle zadaného service desku se vybere patřičný vnitřní klient, který poté komunikuje s vybraným service deskem. Zadaná data jsou zkopírována do struktury pro vytvoření tiketu a struktura je předána konkrétnímu vnitřnímu klientu. Tento klient pak strukturu zpracuje do formy požadavku, který už je známý konkrétnímu service desku. Při tomto zpracování se využívá tzv. konvertorů, které převádí „obecné“ informace do požadované formy. Jedná se například o konverzi stavů a priorit z formátu používaném v rozhraní do formátu používaném v service desku. Požadavek je dále převeden do formátu JSON nebo XML a odeslán webové službě service desku. V případě ALVAO systému je struktura s daty potřebnými pro vytvoření předána přímo DAO metodě, která data pomocí konvertorů zpracuje a vytvoří příslušné SQL příkazy. Ty jsou následně provedeny nad databází. Úspěšné vykonání SQL vrací identifikátor nově vloženého tiketu. Webové služby SD systémů vrací v případě úspěchu různé odpovědi v závislosti na typu SD systému. Vždy je však možné z odpovědi získat informaci o nově vytvořeném tiketu. Zjištěný identifikátor nového tiketu je poslán klientu. Při odesílání odpovědi rozhraní zformuluje odpověď, kam vloží jediný parametr, a to identifikátor nového tiketu nebo null v případě, že se stala chyba.
- požadavek na získání tiketu
  - Podle informací v požadavku je vybrán konkrétní vnitřní klient a zavolána metoda pro získání tiketu, které je předán identifikátor. Vnitřní klient volá příslušnou metodu webových služeb a přijímá odpověď. Tou jsou v případě úspěchu kompletní informace o požadovaném tiketu. Informace jsou z JSON nebo XML formátu převedeny do třídy reprezentující příslušnou odpověď. V případě vnitřního klienta ALVAO, je vytvořen SQL SELECT, který získá informace z databáze. Následuje zpracování dat pomocí konvertorů, které převedou odpověď na formát používaný v rozhraní. Tento formát se dále nijak nepřevádí, protože je již jednotný pro všechny systémy. Je pouze převeden do formátu JSON a odeslán klientu. V případě nenalezení tiketu v service desku je vrácen příznak neúspěchu.

- úprava tiketu
  - Úprava probíhá stejným způsobem jako vytvoření tiketu. Pouze požadavky odesílané až z rozhraní do service desku mohou mít jiný formát. U ALVAO systému probíhá úprava pomocí SQL UPDATE, kdy jsou upraveny příslušné sloupce v databázové tabulce týkající se tiketu nebo souvisejících dat. V případě, že webové služby vrátí informaci o úspěchu nebo neselže update databáze, vrací rozhraní klientu informaci o úspěchu.
- vložení poznámky
  - Vložení poznámky probíhá jednoduchým způsobem. Podle informace v požadavku je vybrán příslušný service desk. Následuje volání metody, která poznámku převede do správného formátu požadavku. Spolu s textem poznámky se zpracuje identifikátor a typ tiketu. Požadavek je odeslán webovým službám a je přijata odpověď, která je zpracována a přeposlána klientu. V případě ALVAO systému je poznámka vložena přes DAO metodu, která vytvoří příslušný SQL INSERT a vloží poznámku do databáze.
- změna stavu tiketu
  - Změna stavu tiketu není nic jiného než úprava tiketu, která je volána uvnitř rozhraní. Upraven je pouze stav a volitelně je přidána zpráva informující o změně stavu.
- získání konfigurační položky
  - Získání konfigurační položky probíhá podobně jako získání tiketu. Rozhraní požadavek zpracuje a zavolá metodu příslušného vnitřního klienta, které předá identifikátor konfigurační položky. Metoda zavolá webovou službu, které předá identifikátor a přijme odpověď. Pro ALVAO je využit SQL SELECT, který čte data z příslušné tabulky pomocí předpřipraveného databázového pohledu a jedné další související tabulky. Odpověd je pomocí konvertoru pro konfigurační položky zpracována požadovaného tvaru a následně odeslána klientu.
- úprava konfigurační položky
  - Úprava konfigurační položky probíhá stejným způsobem jako úprava tiketu. Jedinou odlišností je zpracování objektu s detailními informacemi. Tyto informace jsou ve formě pole, proto jsou nejdříve převedeny do mapy (klíč-hodnota). Mapa je spolu s ostatními informacemi převedena do JSON požadavku. V případě ALVAO systému je vytvořen z informací příslušný SQL UPDATE, který se pokusí data upravit (hledá sloupce pojmenované stejně jako zadané parametry). Ze získané odpovědi je zjištěno, zda byla úprava úspěšná a tato informace je nakonec odeslána klientu.
- kopírování tiketu mezi service desky
  - Operace kopírování tiketu je rozdělena na dvě části. První částí je získání tiketu ze zdrojového service desku. Z tohoto tiketu je následně následně vytvořen objekt s informacemi pro vytvoření nového tiketu. V cílovém service desku je pak tento tiket vytvořen. Až na výjimky (není zaručeno například, že pokud se

tiket kopíruje do systému iTop, tak bude mít stejnou prioritu) by si měly tikety odpovídat. Odpověď rozhraní obsahuje příznak o úspěšnosti „kopírování“.

6.4 Konfigurace aplikace

Aby mohlo rozhraní komunikovat se service desky, musí znát URL jejich webových služeb. Pokud jde o ALVAO, tak musí být znám SQL server a databáze (dohromady ve formě connection stringu). Aplikace pro nastavení těchto údajů používá konfigurační soubor ve formátu JSON. V souboru jsou specifikovány dostupné service desky. Konfigurační soubor je pojmenovaný `sd_config.json`. Požadované parametry jsou:

- pro všechny systémy je vyžadován parametr `type`, který určuje typ service desku a parametr `name`, který jednoznačně service desk identifikuje v rámci aplikačního rozhraní
- pro ALVAO
  - connection string (`dbString`)
- pro iTop, Freshservice a Web Help Desk
  - URL webových služeb (`endpoint`)
  - přihlašovací jméno (`login`)
  - heslo (`password`)
- pro ManageEngine
  - URL webových služeb (`endpoint`)
  - API-KEY konkrétního technika (`apiKey`)

Formát JSON byl zvolen hlavně z důvodu jednoduchosti a možnosti přehledně zapsat pole (array) konfigurací. Díky tomu je možné jednoduše rozhraní použít pro více instancí jednoho typu service desku. Například je možné nakonfigurovat dva systémy ALVAO, které se tak budou lišit pouze v databázi a oba používat. Systémy se budou lišit pouze jménem (parametr `name`) a connection stringem.

Konfigurační soubor musí být umístěný v adresáři spolu s .jar souborem aplikace. Po spuštění aplikace se načtou konfigurace jednotlivých service desků. Vnitřní klienti jednotlivých service desků jsou pak k dispozici pomocí třídy `ServiceDeskApiProvider`, která vrací příslušného vnitřního klienta rozhraní podle zadaného jména service desku. Pokud je v konfiguracích chyba, je oznámena výpisem do konzole a program je ukončen.

Pro konfiguraci aplikace je vyžadován pouze parametr pro určení URL, na kterém budou dostupné webové služby. Důležité je také nastavení portu, který je nutné zvolit tak, aby se aplikace neblokovala s ostatními aplikacemi v PC. Pro tuto informaci byl zvolen konfigurační soubor s příponou .properties. Soubor obsahuje jediný řádek s url, například lze použít: `url=http://localhost:8088/ServiceDeskApi/`. Tento soubor musí být umístěn také v adresáři se spustitelnou aplikací. Název souboru je `api_config.properties`.

6.5 Testování aplikace

Kromě testování pomocí implementovaných klientů, které bude popsáno v následující kapitole, bylo v průběhu vývoje napsáno několik jednotkových testů (unit testy). V podstatě jde o předchůdce implementovaných klientů pracujících podle připravených scénářů. Narozdíl od klientů však nevyužívají implementované webové služby, ale používají přímo vnitřní klienty rozhraní. Vždy po implementaci nějaké změny nebo úpravě stávající funkčnosti následovalo spuštění testů a kontrola správných výsledků. Testy využívaly reálné service desk systémy, proto pro správné fungování bylo potřeba mít všechny service desky spuštěné a správně uvedené v konfiguračním souboru aplikace.

6.6 Rozšíření aplikačního rozhraní

Rozhraní bylo vyvíjeno s důrazem na možná budoucí rozšíření o další funkčnost a hlavně o další service desk systémy. Pro přidání podpory dalšího service desku, je nutné implementovat existující rozhraní (a tedy vytvořit nového vnitřního klienta pro nový service desk). Jde o rozhraní pro práci s tikety (`IServiceDeskApi`), s problémy (`IServiceDeskProblemApi`) a konfiguracemi (`IConfigurationItemApi`). Nové rozhraní nemusí podporovat vše, pouze požadovanou nebo dostupnou funkčnost (nutným minimem je samozřejmě implementace rozhraní pro práci s tikety). Dále je nutné pouze doplnit třídy s načítáním konfigurací service desků a případně doplnit nový formát pro konfiguraci. Poslední krok je doplnění providera jednotlivých vnitřních klientů, aby podporoval i nově přidané.

Kapitola 7

Implementace klientů a testovací scénáře

Kapitola popisuje implementaci klientů výsledného aplikačního rozhraní. Bylo napsáno pět ukázkových scénářů, podle kterých byli klienti programováni.

7.1 Implementace klientů

Klienti byli programování taktéž v jazyce Java. V základu jde o klienta webových služeb, proto byly využity stejné technologie, jako při implementaci rozhraní. Jde hlavně o framework Jersey pro implementaci volání webových služeb a framework Gson, pro jednoduchou práci s JSON formátem. Každý klient je samostatný Maven projekt, stejně jako výsledné rozhraní.

Každý z klientů očekává ve stejném adresáři konfigurační soubor, který obsahuje url webových služeb aplikačního rozhraní. Mimo tuto informaci je potřeba v několika scénářů doplnit url a přístupové údaje k příslušnému service desku. Tyto informace jsou nutné k inicializaci CMDB příslušného service desku, kdy je vytvořena příslušná konfigurační položka, se kterou klient poté pracuje. V případě ALVAO systému není nutné konfigurační databázi inicializovat, protože k ALVAO systému je přiložena databáze obsahující všechna data potřebná pro běh klienta.

Testovací klienti jsou jednoduché aplikace, které fungují jako skripty. Provádějí postupně operace a simulují kroky testovacích scénářů (popsaných v následující kapitole). Požadavky na komunikaci s rozhraním jsou vytvořeny přímo v kódu klienta ve formě mapy. Jednotlivé kroky jsou pomocí komentářů v kódu odděleny a popsány.

Jeden klient komunikuje přes rozhraní vždy s jedním service deskem. Cílový service desk (jeho jméno/identifikátor) je zadán klientu jako parametr při spuštění (netýká se posledního scénáře, který má použité service desky nastaveny přímo v kódu). Klient provede všechny kroky scénáře a poté se ukončí. Mezi jednotlivými kroky je krátká pauza, kdy je vykonávání programu pozastaveno. Pauzu je možné nastavit na vyšší čas (defaultně je to jedna sekunda). Pokud je klient spuštěn s parametrem `-d`, tak je pauza ignorována a namísto ní čeká klient po každém kroku na stisk klávesy enter. Tímto způsobem lze kontrolovat a ověřovat jednotlivé kroky v příslušném service desku.

7.2 Testovací scénáře

Cílem testovacích scénářů je otestování a předvedení funkčnosti vyvíjeného rozhraní. Scénáře by měly reflektovat realitu, tedy postup by měl být podobný tomu, jak by probíhal ve skutečnosti reálnými uživateli. U každého scénáře jsou uvedeny kroky, které odpovídají volání webových služeb.

Konfigurační databáze

Před spuštěním klientů je potřeba vytvořit základní databázi konfiguračních položek (CMDB), se kterou pak budou klienti pracovat. Pro účely scénářů byla jsou potřeba pouze dvě konfigurační položky. První je počítač využitý ve scénáři č. 2. U počítače se pracuje se zadanou pamětí RAM. Druhou položkou je tiskárna (scénář č. 4). U tiskárny se pracuje s parametrem určujícím stav používání.

Scénář č. 1

První testovací scénář lze spustit ve všech service desk systémech. Prvním krokem je založení incidentu, kdy klient popisuje problém s nedostupností programu. Technik přečte popis tiketu a doplní informaci o tom, že potřebuje více informací. Následně tiket pozastaví. Žadatel reaguje tak, že doplní informaci o názvu programu a změní tiketu prioritu na vysokou. Technik tiket znovu otevře a pustí se do řešení. Následně dodá k tiketu informaci o nutnosti restartovat službu a tiket nastaví jako vyřešený. Žadatel službu restartuje a ověří, že je vše v pořádku. Nakonec tiket uzavře.

Kroky scénáře:

1. Vytvoření incidentu.
2. Pozastavení incidentu s přidáním zprávy (technik potřebuje více informací).
3. Přidání zprávy (žadatel doplňuje informaci o programu).
4. Úprava incidentu - nastavení priority na vysokou.
5. Znovuotevření incidentu s přidáním zprávy (technik pokračuje v řešení).
6. Vyřešení incidentu s přidáním zprávy (technik oznamuje nutnost restartu služby).
7. Získání informací o incidentu a z nich získání poslední zprávy (žadatel zjišťuje co je třeba udělat).
8. Uzavření incidentu s přidáním zprávy (žadatel oznamuje úspěšný restart).

Scénář prezentuje práci s incidenty a operace s tikety. Prezentovány jsou operace pro úpravu tiketů, změnu stavů a přidávání zpráv (komunikace mezi účastníky).

Scénář č. 2

Druhý scénář využívá service desky, které mají dostupnou správu konfigurací, tedy ALVAO, iTop a Freshservice. Klient zakládá v systému incident kvůli celkové pomalosti a zamrzávání svého počítače. K incidentu připojuje informaci o svém počítači (což je konfigurační položka CMDB databáze). Přidělený technik po prozkoumání problému zjistí, že počítač disponuje

nedostatečnou operační pamětí. Tuto skutečnost oznámí ve formě zprávy a připojí k tiketu. Následně přidá do počítače další pamět a upraví informaci o paměti v CMDB. Tiket označí jako vyřešený a klient řešení akceptuje uzavřením tiketu.

Kroky scénáře:

1. Vytvoření incidentu.
2. Získání informací o incidentu (technik získá id konfigurační položky).
3. Získání informací o konfigurační položce (technik zjistí, že operační pamět je nedostačující).
4. Přidání zprávy (technik informuje o přidání paměti).
5. Úprava konfigurační položky (zvýšení kapacity operační paměti).
6. Vyřešení incidentu.
7. Uzavření incidentu s přidáním zprávy (žadatel oznamuje zlepšení běhu počítače).

Scénář prezentuje práci s konfiguračními položkami napříč systémy. Prezentovány jsou operace pro získnáí informací o konfiguračních položkách a jejich úpravy.

Scénář č. 3

Třetí scénář využívá service desky, které podporují správů problémů, tedy ALVAO a iTop. V systému je založen incident s informací, že není možné tisknout. Technik začne s řešením tiketu a přitom zjistí, že byly nahlášeny už dva podobné incidenty. Dalším bádáním je zjištěno, že tisk neprobíhal kvůli spuštěné naplánované úloze, která blokovala odesílání dokumentů na tiskárnu. Technik proto založí tiket typu problém s příslušným popisem. Následně vytvoří vazbu incidentů, které se týkaly tisku s problémem. Problém je vyřešen tak, že je naplánované úloha přesunuta na jiný vhodný čas. Všechny tikety jsou nakonec uzavřeny.

Kroky scénáře:

1. Vytvoření incidentu č. 1.
2. Vytvoření incidentu č. 2.
3. Vytvoření incidentu č. 3.
4. Přidání zprávy k poslednímu incidentu (technik informuje o příčině).
5. Vytvoření problému.
6. Vytvoření vazby mezi incidentem č. 1 a problémem.
7. Vytvoření vazby mezi incidentem č. 2 a problémem.
8. Vytvoření vazby mezi incidentem č. 2 a problémem.
9. Vyřešení problému.
10. Uzavření problému.

11. Uzavření incidentu č. 1.
12. Uzavření incidentu č. 2.
13. Uzavření incidentu č. 3.

Scénář prezentuje práci s problémy, incidenty a jejich vazbami.

Scénář č. 4

Čtvrtý scénář využívá opět service desky ALVAO a iTop, protože pracuje s procesem správy změn. V systému je založen incident s informací o vadném tisku. K incidentu je přidána informace o tiskárně (konfigurační položce) Technik po prozkoumání problému založí do systému tiket typu změna, který se týká výměny tiskárny za novou, protože stávající tiskárna je zastaralá. Tiket je nejprve ve stavu analýza, kdy se rozhoduje, která tiskárna bude pořízena. Následuje výběr a objednávka tiskárny, tiket je ve stavu plánování. Při výměně tiskárny je tiket nastaven na stav implementace. Poté je tiketu nastaven stav na monitorování a v tomto stavu je několik dní. Tiskárna je testována a když je zjištěno, že je vše v pořádku, je tiketu nastaven na stav vyřešeno a posléze uzavřeno.

Kroky scénáře:

1. Vytvoření incidentu.
2. Získání informací o incidentu (informace o konfigurační položce).
3. Vytvoření změny.
4. Změna stavu změny na plánování.
5. Změna stavu změny na implementaci.
6. Změna stavu změny na monitorování.
7. Vyřešení změny.
8. Uzavření změny.
9. Uzavření incidentu.

Scénář prezentuje možnosti správy změn, hlavně změnu stavů změny v závislosti na prováděných úkonech ve skutečném světě.

Scénář č. 5

Poslední scénář lze simulovat všech systémech. Klient však využívá z důvodů jednoduchosti pouze systémy ALVAO a iTop. V systému ALVAO, který používá odběratel IT služeb, je vytvořen incident. Následně je třeba stejný incident vytvořit v systému dodavatele. Dodavatel využívá systém iTop. Tiket v systému ALVAO je opatřen poznámkou o kopírování a je zkopírován do systému iTop.

Kroky scénáře:

1. Vytvoření incidentu v systému ALVAO.
2. Přidání zprávy k incidentu (informace o kopírování).

3. Získání informací o tiketu.
4. Zkopírování tiketu do systému iTop.
5. Přidání zprávy k novému incidentu v systému iTop.

Scénář prezentuje možnost „kopírovat“ tikety z jednoho systému do druhého.

Závěr

Jednotlivé kapitoly této práce ilustrovaly vývoj aplikačního rozhraní pro service desk systémy. Od nutných teoretických základů popisujících knihovnu ITIL, funkci service desku a procesy, se kterými lze za pomoci rozhraní pracovat. Přes výběr vhodných SD systémů, návrh rozhraní až po vlastní vývoj aplikace. V závěru byly implementovány testovací a ukázkové scénáře ve formě klientů, kteří výsledné rozhraní využívají. Výsledná aplikace byla zveřejněna jako open source pro volné využití či pro možnosti budoucího rozšíření. Všechny jednotlivé cíle, specifikované v kapitole 3, tak byly úspěšně splněny.

V průběhu implementace byla nejzajímavější práce s různými technologiemi. To se týká hlavně používání webových služeb service desků. Každý service desk totiž webové služby implementuje jinak, a tak byla práce velmi různorodá. Mimo jiné i proto, že pro stejné operace bylo kromě používání webových služeb nutné pracovat i s databází. Stejně tak návrh vlastních webových služeb, od formátu dat přes výběr webového serveru a jeho používání. To všechno přineslo nové poznatky a zkušenosti.

Největší problémy působily v průběhu práce nedostatečné dokumentace k některým service desk systémům. Často bylo nutné experimentovat a hledat správná řešení stylem pokus-omyl. Stejně tomu bylo v případě práce s databází SD systému ALVAO. Ta je sice dobře dokumentovaná, ale pro potřeby této práce bylo nutné databázi prozkoumat a hledat v ní zdroje pro další informace.

Vyvinuté aplikační rozhraní lze využít v případě potřeby přístupu do různých SD systémů. Může posloužit při vývoji dalších projektů z oblasti simulování ITIL procesů, jako je např. herní engine pro ITIL scénáře. V SD systémech lze využít rozhraní při kopírování dat (tiketů) z jednoho systému do druhého. Tato funkcinalita by mohla být přínosná pro provozovatele více service desků. Jde o příklad, který ukazuje jeden ze scénářů, kdy spolupracující společnosti používají různé systémy a mohou mít potřebu přenášet data (tikety) z jednoho systému do druhého. Rozhraní by v praxi mohla využít i osoba, která se teprve rozhoduje, který systém si pořídí. Mimo jiné zde může rozhraní posloužit jako testovací nástroj webových služeb, kterými service desk disponuje.

Rozhraní lze rozšiřovat v mnoha směrech. Hned na začátku se nabízí rozšíření o podporu dalších SD systémů, na což byl při návrhu a implementaci kladen důraz. Jednoduše lze rozšířit struktury s informacemi o tiketech nebo konfiguracích IT komponent. Dalším krokem by mohlo být vytvoření spolupracující aplikace, která by se starala o rozesílání emailových notifikací. Notifikace by byly generovány na základě událostí v SD systémech (změna konfigurací, hlášení incidentů apod.) a rozesílány uživatelům. Podobná funkce není v rozhraní implementována (o to se starají až případně SD systémy), proto by tohle rozšíření mohlo být užitečné. Jako další rozšíření se nabízí komplexní správa konfiguračních položek. Tato část je ale velmi rozsáhlá a pro úplné pokrytí by vyžadovala správu konfiguračních položek na úrovni samotné CMDB databáze.

Literatura

[1] ALVAO s.r.o.: *ALVAO Asset Management*. [Online; navštíveno 24.2.2016].
URL http://www.alvao.com/products/asset-management-easy-to-use/

[2] ALVAO s.r.o.: *ALVAO Service Desk*. [Online; navštíveno 29.1.2016].
URL http://www.alvao.com/products/service-desk-microsoft-platform/

[3] AXELOS: *AXELOS Global Best Practice*. [Online; navštíveno 13.12.2015].
URL https://www.axelos.com/best-practice-solutions/itil

[4] AXELOS: *ITIL Service Operation*. Norwich: TSO, 2011, ISBN 978-0-11-331307-5, 370 s.

[5] BMC SOFTWARE: *ITIL Processes & Best Practices*. [Online; navštíveno 15.3.2016].
URL http://www.bmc.com/guides/itil-introduction.html

[6] Bucksteeg, M.; Ebel, N.; Eggert, F.; aj.: *ITIL 2011 Stručný a srozumitelný výklad*. Brno: Computer Press, 2012, ISBN 978-80-251-3732-1, 216 s.

[7] Capterra Inc.: *Top Help Desk Software Products*. [Online; navštíveno 4.2.2016].
URL http://www.capterra.com/help-desk-software/

[8] Combodo: *iTop, ITIL for all*. [Online; navštíveno 8.2.2016].
URL http://www.combodo.com/itop-193

[9] Freshdesk Inc.: *Freshservice Service Desk*. [Online; navštíveno 8.2.2016].
URL https://freshservice.com/online-it-service-desk

[10] Haslam, O.: *The Best Help Desk Software of 2016*. [Online; navštíveno 4.3.2016].
URL http://www.pcmag.com/article2/0,2817,2489457,00.asp

[11] ISO: *ISO/IEC 20000, Information technology - Service management*. [Online; navštíveno 15.12.2015].
URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=5013818

[12] mITSM GmbH: *Ticket, Ticket Owner and Ticket Agent*. [Online; navštíveno 22.4.2016].
URL https://www.mitsm.de/ticket-ticket-owner-and-ticket-agent-en

[13] OMNICOM s.r.o.: *BESTPRACTICE.CZ, IT and Management Knowledge Base*. [Online; navštíveno 13.12.2015].
URL http://www.bestpractice.cz

[14] Pink Elephant company: *ITSM & Business Management Training and Consulting - Pink Elephant*. [Online; navštíveno 6.2.2016].
URL http://www.pinkelephant.com/

[15] Requestor: *Help Desk nebo Service Desk: základní rozdíly*. [Online; navštíveno 17.2.2016].
URL http://www.requestor.com/cs/blog-cs/clanky/help-desk-nebo-service-desk-zakladni-rozdily/

[16] SolarWinds Worldwide, LLC: *Affordable help desk ticketing & IT asset management software*. [Online; navštíveno 10.2.2016].
URL http://www.webhelpdesk.com/

[17] Zoho Corp.: *ManageEngine ServiceDesk Plus*. [Online; navštíveno 7.2.2016].
URL https://ondemand.manageengine.com/service-desk/index.html?index

Přílohy

Seznam příloh

Příloha A

Obsah CD

- `ServiceDeskApi/` - aplikační rozhraní
  - `src/` - zdrojové kódy aplikace
- `ServiceDeskApiClients/` - ukázkoví klienti rozhraní
- `Report/` - text práce ve formátu PDF a zdrojové texty v LaTeXu
- `README.txt` - informace o aplikaci a klientech

Příloha B

Přehled webových služeb rozhraní

Operace s tikety

- `tickets/getTicket` (POST)
- `tickets/createTicket` (POST)
- `tickets/addMessage` (POST)
- `tickets/updateTicket` (PUT)
- `tickets/resolveTicket` (PUT)
- `tickets/closeTicket` (PUT)
- `tickets/suspendTicket` (PUT)
- `tickets/reopenTicket` (PUT)
- `tickets/copyTicket` (POST)

Operace s problémy

- `problems/linkIncident` (POST)
- `problems/unlinkIncident` (POST)
- `problems/getRelatedIncidents` (POST)

Operace s konfiguračními položkami

- `cmdb/getCi` (POST)
- `cmdb/linkCi` (POST)
- `cmdb/updateCi` (PUT)

Operace s rozhraním

- `systems/getSystem` (POST)

Příloha C

Příklady požadavků pro komunikaci s rozhraním

Požadavek na vytvoření tiketu


```
{
  "requesterId": "4",
  "subject": "Predmet incidentu",
  "technicianName": "Pavel S",
  "description": "Popis incidentu",
  "sla": "Incident SLA",
  "ticketType": "Incident",
  "priority": "normal",
  "system": "Alvao",
  "service": "Incident management",
  "requesterName": "Pavel S",
  "technicianId": "4",
  "status": "open"
}
```


Požadavek na získání tiketu


```
{
  "system": "Alvao",
  "ticketType": "Incident",
  "ticketId": 373
}
```


Požadavek na pozastavení tiketu


```
{
  "note": "Duvod pozastaveni",
  "system": "Alvao",
  "ticketType": "Incident",
  "ticketId": 373,
  "status": "On hold"
}
```

Požadavek na úpravu tiketu (priority)

```
{
  "system": "Alvao",
  "ticketType": "Incident",
  "priority": "high",
  "ticketId": 373
}
```

Požadavek na otevření tiketu

```
{
  "note": "Pokracujeme v reseni",
  "system": "Alvao",
  "ticketType": "Incident",
  "ticketId": 373,
  "status": "Open"
}
```

Požadavek na přidání zprávy

```
{
  "system": "Alvao",
  "ticketType": "Incident",
  "message": "Obsah zpravy",
  "ticketId": 374
}
```

Požadavek na získání informací o konfigurační položce

```
{
  "configItemId": 65,
  "system": "itop"
}
```

Požadavek na úpravu konfigurační položky

```
{
  "configItemId": 65,
  "system": "itop",
  "details": [
    "ram",
    "2048 MB"
  ]
}
```

Požadavek na propojení incidentu a problému

```
{
  "system": "itop",
  "incidentId": 395,
  "problemId": 397
}
```

Požadavek na zkopírování tiketu

```
{
  "requesterId": 1,
  "organization": "Demo",
  "destSystem": "itop",
  "ticketType": "Incident",
  "srcSystem": "Alvao",
  "technicianId": 1,
  "ticketId": 375
}
```

Požadavek na získání informací o SD systému

```
{
  "system": "Web help desk"
}
```