Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Cloudová platforma Cordys, vývoj cloudové aplikace.

Plzeň, 2013 Jan Bubela

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 2. dubna 2013

Jan Bubela

Abstract

This work is an overview of Cloud Computing, especially its component Platform as a Service. It describes several cloud platforms, show them from business and technical point of view.

The second part is a specification, design and an implementation of a web based cloud application created for real usage. This work also evaluates a suitability of using cloud application for this type of business process and comes with developer´s and user´s view on this application.

Abstrakt

Tato práce je věnována Cloud Computingu, zvláště pak jedné z jeho forem - PaaS (Platforma jako služba). V práci je popsáno několik cloudových platforem a to z obchodního a technického hlediska.

V druhé části práce je uveden popis, vzhled a implementace cloudové aplikace, která byla následně používána uživateli ve skutečném firemním prostředí. Práce také nabízí hodnocení, zda je cloudové řešení vhodné pro takovou aplikaci a také hodnocení vývojáře aplikace i konečných uživatelů.

Obsah

1 Úvod a cíl práce

Cílem této práce je seznámit čtenáře s přístupem „Cloud Computingu“, zvláště pak jeho variantou "Platforma jako služba" (Platform as a service - PaaS) a několika různými společnostmi, které Cloud Computing nabízí. V teoretické části budou tyto cloudové platformy představeny z obchodního (licenční podmínky, cena) a technického hlediska (programovací jazyky, API, škálování, apod.).

V realizační části práce budou čtenáři představeny požadavky na vývoj konkrétní aplikace, analýza těchto požadavků a implementace této aplikace, včetně vývojářských a uživatelských dojmů z provozování aplikace. Aplikace je určena ke správě uživatelských účtů a přidělených IT prostředků ve firemním prostředí, umožňuje vytváření, aktualizaci a archivaci záznamů a obsahuje schvalovací workflow.

Mým cílem je tedy jednak čtenáři představit vývoj a používání cloudové aplikace na konkrétní platformě a zároveň také předat informace pro vytvoření vlastního názoru ke Cloud Computingu a pro zhodnocení jeho výhod i rizik.

2 Cloud Computing

Definice:
Název Cloud Computing byl převzat jako metafora pro internet. Internet je totiž obvykle v síťových diagramech zobrazován jako obláček. Ikona obláčku představuje "všechno to ostatní", co umožňuje síti fungovat, oblast, která je starostí někoho jiného [1].

Myšlenku, ze které vychází technologie Cloud Computingu dnes, představil v roce 1961 profesor americké univerzity MIT John McCarthy. Budoucí technologii sdílení počítačových technologií přirovnal k modelu využívání elektrické energie mnoha domácností z jedné elektrárny (datové úložiště a výpočetní centrum cloudu), ke které se za pomoci elektrorozvodné sítě (internet) budou jednotlivé domácnosti připojovat. Myšlenka přišla už v době, kdy ještě softwarová a hardwarová virtualizace neexistovala a přitom počítala s modelem, že neexistuje pouze jedna elektrárna, ale mnoho vzájemně propojených elektráren, poskytujících si pomoc v případě výpadku jedné z nich.

Cloud Computing je souhrn různých technologií, které společně mění přístup společností k vytváření jejich vlastní IT infrastruktury. Principem Cloudu je užití konkrétní technologie, když jí potřebujeme a ani minutu navíc. Není třeba nic instalovat, neplatíme za službu, když ji nepoužíváme [2].

G. Reese uvádí jednoduchý test: "Jestliže můžete vejít do knihovny nebo internetové kavárny, posadit se k jakémukoli počítači bez ohledu na zvolený operační systém nebo prohlížeč a používat konkrétní službu, pak se jedná o cloudovou službu". Dále definuje tři základní podmínky pro rozhodnutí o tom, zda se jedná o cloudovou službu:

- jestliže je služba dostupná přes webový prohlížeč nebo pomocí webových služeb,
- nulové počáteční náklady,
- platby pouze za to, co skutečně spotřebujeme [2].

Výraz „Cloud Computing“ popisuje přesun lokálních výpočetních zdrojů a datových úložišť k „outsourcingu“ těchto služeb, s okamžitým přístupem k těmto službám, přičemž jejichž kvalita a rozsah jsou smluvně stanoveny. Lokální IT firmy tak už nezajišťuje vlastní chod všech systémů, ale pouze dohled nad dodáváním sjednaných služeb. Cloud Computing tak představuje alternativu k tradičnímu přístupu k investicím do lokální infrastruktury a lidských zdrojů.

Rozdělení:
Cloud Computing je možné rozdělit podle dvou hledisek a to podle toho, jak je poskytován a podle služby, kterou poskytuje.
První hledisko (jak je poskytován) rozděluje Cloud Computing na veřejný (pro širokou veřejnost), soukromý (pro určitou organizaci), hybridní (kombinace

veřejného a soukromého) a komunitní (pro několik organizací spojených např. stejným oborem podnikání).

Pro nás bude následně důležitější rozdělení Cloud Computingu podle toho, co poskytuje, tj. co vše je v rámci služby nabízeno:

IaaS - infrastruktura jako služba (Infrastructure as a Service)
Poskytovatel služeb poskytuje infrastrukturu, hardware, virtualizaci. Služba je zaměřena jako outsourcing výpočetních zdrojů, jako úložiště a výpočetní kapacita. Virtualizace umožňuje elastickou alokaci a poskytování těchto zdrojů po síti. Známí poskytovatelé jsou např. Amazon Web Services a Windows Azure.

SaaS - software jako služba (Software as a service)
Poskytovaná služba je aplikace, můžeme k ní přistupovat odkudkoli a kdykoli. SaaS aplikace jsou namísto běhu na lokálním PC poskytovány přes webový prohlížeč. Příkladem poskytovatele je Google se sadou aplikací Google Apps.

V další kapitole se budu podrobněji věnovat třetímu z hlavních typů podle poskytované služby - Platforma jako služba - PaaS (Platform as a service).

Mezi dalšími z typů, kombinujících různé prostředky, bych mohl jmenovat Software plus Services, poskytovaný společností Microsoft. Tento typ je rozšířením SaaS o lokální zdroje, kde běží část softwaru nebo jsou uložena data [1]. Dále pak je to Storage as a Service, což je poskytování datového úložiště (např. Google Drive, Picasa).

Na následující tabulce převzaté z knihy Cloud Application Architectures [2] je zobrazen seznam tradičních systémů používaných v malém nebo středním podniku a k nim přiřazených odpovídajících cloudových řešení.

Tradiční řešení	**Cloud**
Souborový systém	Google Drive
MS Outlook, Apple Mail	Gmail, Yahoo!, MSN
SAP CRM/Oracle CRM/Siebel	SaleForce.com
Quicken/Oracle Financials	Intacct/NetSuite
Microsoft Office/Lotus Notes	Google Apps
Stellent	Valtira
Off-site zálohování	Amazon S3
Servery, firewall	Amazon EC2, GoGrid, Mosso

Tabulka 2.1 - tradiční IT infrastruktura versus cloud [2]

2.1 Platform as a service

Platformou rozumíme infrastrukturu pro vývoj softwarových aplikací. As a Service (jako služba) znamená, že platforma provozovaná poskytovatelem je umístěna v cloudu dostupná zákazníkovi prostřednictvím internetu a je možné ji používat bez instalaci jakéhokoli doplňkového softwaru (mimo webového prohlížeče).

Platformy jsou velmi často soustředěny na jeden programovací jazyk a jeden postup. Poskytují tak efektivní programování pomocí automatizace opakovaných úloh, předpřipraveného uživatelského rozhraní, napojení databází a vyzkoušené infrastruktury aplikace. To významně vývoj aplikací urychluje oproti tradičním Java nebo .Net metodám. Přístup Platform as a Service také umožňuje vyvíjet aplikace pomocí stejných standardů a technologií, stejných jako jsou použity v produkční fázi. Tento postup odstraňuje problémy s migrací aplikace od vývoje do cílového prostředí.

2.2 Výhody, rizika

2.2.1 Výhody

Dynamická alokace: moderní IT týmy musí být schopny poskytnout rychlý náběh a spuštění nové služby nebo funkcionality, včetně testu nové služby na malém okruhu zákazníků. Toto bylo před Cloud Computingem velmi těžko obtížné. Nyní mohou IT používat přístup neustálého vývoje, testování a zlepšování služeb (podobně jako věčná beta známá z Google platformy) [3].

Testování při vývoji: Cloud Computing nabízí vývojářům testy na více strojích, různých nastaveních, různých lokací, zátěžové testy, testy kompatibility, testy výkonu a odezvy, což v takto rozsáhlém spektru je v lokálním prostředí těžko proveditelné [3].

Rychlost vývoje: poskytovatelé nabízí často vizuálně jednotné prostředí pro vytvářené webové aplikace, specifické postupy, širokou nabídku knihoven a webových služeb.. Využívání těchto prostředků, vývojářům umožňuje svou práci značně zefektivnit, protože je pro každou aplikaci nemusí psát znovu od začátku.

Mezi dalšími výhodami uvedu několik bodů společných pro Cloud Computing obecně:
Garantovaná dostupnost: ve smlouvě s poskytovatelem má zákazník definovánu jasnou procentuální dostupnost nakupované služby včetně sankcí za nedodržení. Úroveň infrastruktury, lidských zdrojů a zabezpečení velkých cloudových poskytovatelů je velmi vysoká (garantovaná dostupnost např. u Amazon EC2 je 99,95%).

Aktualizace softwaru: zákazník už nadále nemusí řešit komplikace spojené s novými verzemi softwaru - zálohování, instalace, atd. Vše zajišťuje poskytovatel a zákazník se ani o probíhajících aktualizacích nemusí dozvědět.

Údržba infrastruktury: zákazníkům odpadá starost a náklady spojené s napájením, chlazením, ochranou proti hackerským útokům, požárním zabepečením, atd.

Cena, škálovatelnost: při použití lokální infrastruktury musel dosud zákazník mít takovou kapacitu zdrojů, aby byl schopen pokrýt špičky, kdy požadavky na výkon byly maximální. Ty ale byly např. jen několikrát za měsíc, po zbylou dobu velká část výkonu zůstávala nevyužita. Cloud Computing zákazníkovi umožňuje buďto manuálně, programově nebo dynamicky (což je jedna z největších výhod Cloud Computingu) alokovat potřebné zdroje v závislosti na požadované kapacitě v průběhu měsíce nebo i dne. A stále platí jen to, co opravdu spotřeboval. Ovšem neznamená to, že se cenou vyplatí Cloud Computing vždy, závisí na cenové kalkulaci, kterou si musí zákazník před vstupem na Cloud provést - kolik by ho stál nákup infrastruktury a její údržba, softwarového vybavení a mzdy IT zaměstnancům v porovnání s poplatky poskytovateli cloudu. "Jestliže například provádíte v cloudu streaming videa ve vysokém rozlišení na 100 zdrojích, vaše náklady porostou velmi rychle." [1].

2.2.2 Rizika

Zabezpečení: poskytovatelé nabízí velmi různorodé a rozsáhlé metody ochrany, je samozřejmě v jejich zájmu nabízet služby s vysokou úrovní. Ale přes všechny metody zabezpečení utajení dat a jejich ztráty, ten kdo by nakonec byl poškozen ztrátou dat, by byl uživatel, ne poskytovatel [3]. Toto nebezpečí neznamená pouze napadení a ztrátu dat např. hackerským útokem nebo živelnou pohromou, ale i další rizika spojená s umístěním dat u vzdáleného poskytovatele - bankrot poskytovatele, soudní nebo vládní zásah. T. Velte uvádí zajímavé srovnání: „V roce 2008 vláda USA převzala kontrolu nad bankou Washington Mutual. To bylo vnímáno jako největší bankovní pád do té doby a připomínkou, že na velikosti společnosti nemusí záležet. Vezměme např. společnost Google, jejíž tržní hodnota byla nedávno 250 mld. $ (1.10.2012). Taková velikost a hodnota se zdá být neprůstřelná. Ale Washington Mutual měl hodnotu 307 mld. $, když došlo k jeho pádu. Díky vládnímu pojištění, každý dostal zpět své peníze. Ale u cloudových dat není žádné pojištění třetí stranou, jestliže poskytovatel z jakéhokoli důvodu skončí, data mohou být ztracena.“ [1].

Technická nezralost: poměrně nezralá povaha PaaS přístupu sází vše na jednu kartu: jakékoli změny cen, výpadky služby, zhoršující se kvalita, atd., může zákazníka velmi poškodit. Týká se zejména kritických business aplikací [3].

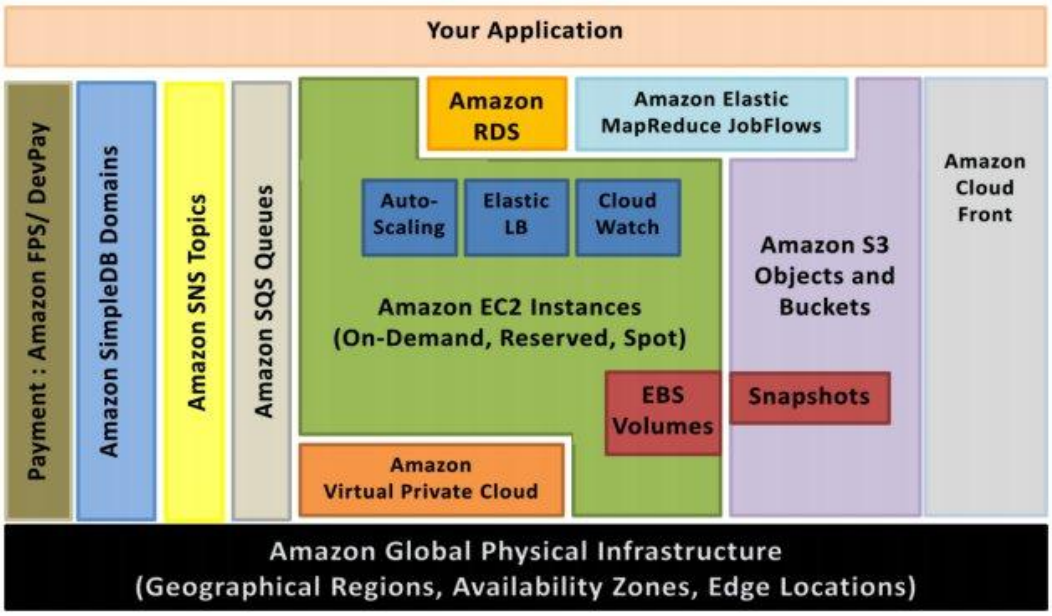
Jeden poskytovatel: většina poskytovatelů se snaží nabídnout komplexní služby svým zákazníkům, kteří se přechodem k nabízeným službám svým způsobem spoléhají na životaschopnost poskytovatele a jeho schopnost růstu a reakce na

požadavky svých uživatelů [3]. Framework webových aplikací uvedený ve výhodách PaaS může mít na druhou stranu negativní efekt, tzv. vendor lock-in (proprietární uzamčení) a to pro případ nucené nebo dobrovolné migrace k jinému poskytovateli. Používání specifických jazyků a knihoven může mít za následek až nemožnost migrace, nebo velké náklady k ní potřebné.

Cloudové platformy
Cílem následujícícho textu je představit několik společností poskytující cloudové služby. Společností poskytující infrastrukturu nejenom přímým uživatelům, ale i dalším cloudovým společnostem, je Amazon - v této práci zástupce IaaS. Dále budou představeny společnosti Salesforce.com (hlavně její služba PaaS Force.com) a Google App Engine, která nabízí SaaS i PaaS. Poslední popisovanou PaaS bude Cordys Process Factory společnosti Cordys.

2.3 Amazon Web Services

Na přelomu 20. a 21. století zástupci společnosti Amazon, do té doby převážně internetového prodejce, přemýšleli, jak lépe využít své kapacity výpočetních zdrojů po dobu celého roku a nejen v průběhu špiček využívání internetového obchodu, tj. před Vánocemi a dalšími velkými svátky. Přišli tedy s myšlenkou nabízet své volné zdroje zákazníkům, start Amazon Web Services (AWS) se datuje k červenci 2002. Postupně tak vznikl jeden z nejrozsáhlejších a nejrozvinutějších cloudů, s komplexní nabídkou cloudových služeb, co se týká jejich počtu a šíře.



Obr. 2.1 - Amazon webové služby [11]

Tyto služby svým rozsahem pokrývají prakticky veškeré potřeby pro vytvoření velké a robustní webové aplikace. Obrázek 2.1 představuje, jak vytvořená aplikace interaguje s různými webovými službami a jak jednotlivé služby spolupracují mezi sebou.

Kde není v následujícím popisu platformy přímo uvedeno jinak, je čerpáno z oficiálních webových stránek společnosti Amazon - http://aws.amazon.com [11].

2.3.1 Cena

Po vytvoření nového účtu je možné využívat po dobu jednoho roku zdarma tyto služby každý měsíc:

- 750 hodin EC2 (Elastic Compute Cloud - výpočetní služba) na Linux/Unix
- 750 hodin Elastic Load Balancing (síťová služba) + 15GB přenesených dat
- 10GB EBS (Elastic Block Storage - služba úložiště)
- 15GB šířku pásma přes všechny AWS služby
- 1GB lokálního přenosu dat

Při potřebě většího množství jsou služby zpoplatněny následovně:

On-Demand
= výpočetní kapacita okamžitě k dispozici, bez dlouhodobých závazků. Umožňuje tak používat nízké variabilní náklady namísto komplexního plánování, nákupu, údržby, atd..., tj. namísto vysokých fixních nákladů.

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.060 per Hour	$0.091 per Hour
Medium	$0.120 per Hour	$0.182 per Hour
Large	$0.240 per Hour	$0.364 per Hour
Extra Large	$0.480 per Hour	$0.728 per Hour

Tabulka 2.2 - Amazon webové služby, hodinové sazby za služby „On-demand“ [11]

Rezervovaná kvóta
= jednorázovou platbu za každou službu, kterou si chce zákazník rezervovat, díky čemuž je dosaženo výrazných slev v hodinových sazbách za danou službu. Rezervované kvóty je možné zakoupit v ročních nebo tříletých termínech. V případě nespotřebování rezervovaných služeb je platba nevratná.

Sazby se dělí na "light", "medium" a "heavy" s možností rezervace na 1 rok nebo 3 roky. V tabulce 2.3 si ukážeme typ "medium" uzavřený na 1 rok pro platformu Windows.

Standard Reserved Instances	Upfront	Hourly
Small (Default)	$160	$0.044 per Hour
Medium	$320	$0.088 per Hour
Large	$640	$0.175 per Hour
Extra Large	$1280	$0.35 per Hour

Tabulka 2.3 - Amazon webové služby, platby za rezervovanou kvótu [11]

Transfer dat
Zdarma přenos dat mezi službami ve stejném regionu, data přenesena mezi AWS službami v různých regionech budou zpoplatněna viz tabulka 2.4. Ceny za přenos nad 350 TB za měsíc lze dohodnout individuálně.

Data Transfer IN	Pricing
All data transfer in	$0.000 per GB
Data Transfer OUT	**Pricing**
First 1 GB / month	$0.000 per GB
Up to 10 TB / month	$0.120 per GB
Next 40 TB / month	$0.090 per GB
Next 100 TB / month	$0.070 per GB
Next 350 TB / month	$0.050 per GB

Tabulka 2.4 - Amazon webové služby, platby za přenos dat [11]

2.3.2 Licenční podmínky

Z velmi rozsáhlých a podrobných licenčních podmínek (uvedených na http://aws.amazon.com/agreement/) vybírám to, co se týká vlastnických práv:

- Amazon si nenárokuje jakákoli práva nad duševním vlastnictvím objednatele.
- Objednatel služeb (ani jeho koncový zákazník) nesmí měnit žádný software začleněný do nabízených služeb.
- Některý obsah AWS může být poskytován pod vlastními licenčními podmínkami (např. Apache Software License).

2.3.3 Pro vývojáře

API

- AWS poskytuje standardní SOAP a REST rozhraní pro komunikaci s jednotlivými službami
- Jazykově-specifické API, viz programovací jazyky.

Programovací jazyky
Pro vývojáře jsou dostupné knihovny, ukázky kódů, tutoriály a další zdroje v těchto jazycích:

- Java
- PHP
- Ruby
- Windows a .NET

Verze aplikací
Vývojář může vytvářet různé verze aplikací. Každá aplikace se skládá z jedinečného WAR souboru a informace o verzi. Implementace nové verze je možné udělat z AWS Management Console, příkazové řádky nebo přímo z API dané aplikace.

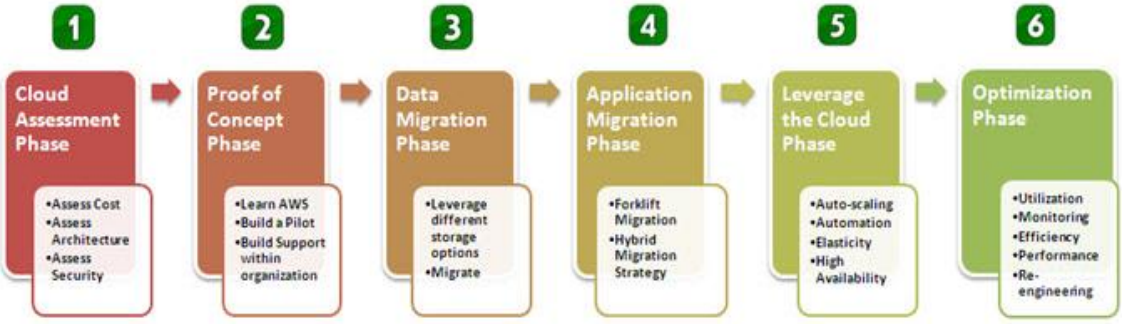
Škálování
AWS umožňuje automatické škálování vhodné zvláště pro aplikace, u kterých se mění jejich využití v řádech hodin, dnů nebo týdnů. Automatické škálování je poskytováno službou Amazon CloudWatch, bez dalších poplatků navíc.

Škálování je plynulé podle rostoucí poptávky, po odeznění se snižuje. Je možné škálovat dynamicky podle aktuálních potřeb i podle plánu, který si uživatel služby určí.

Automatické notifikace (za použití služby Amazon Simple Notification Service) při vzniku změny a ukončení akce škálování.

Škálování je automaticky registrováno pro uživatele hlavní výpočetní služby Amazon EC2.

Vytváření aplikací
Migrování hotové aplikace do AWS: jedním z klíčových znaků infrastruktury AWS je její flexibilita. Poskytuje svobodu volby programovacích modelů, jazyků, operačních systémů a databází se kterými vývojář již pracuje. AWS tak poskytuje virtuální prostředí umožňující nahrát software a služby, které aplikace potřebuje. Obrazy operačních systémů se nazývají Amazon Machine Image (AMI), lze zvolit již připravenou instanci aplikace.



Obr. 2.2 - schéma migrace existující aplikace do AWS [11]

Vytvoření nové aplikace:

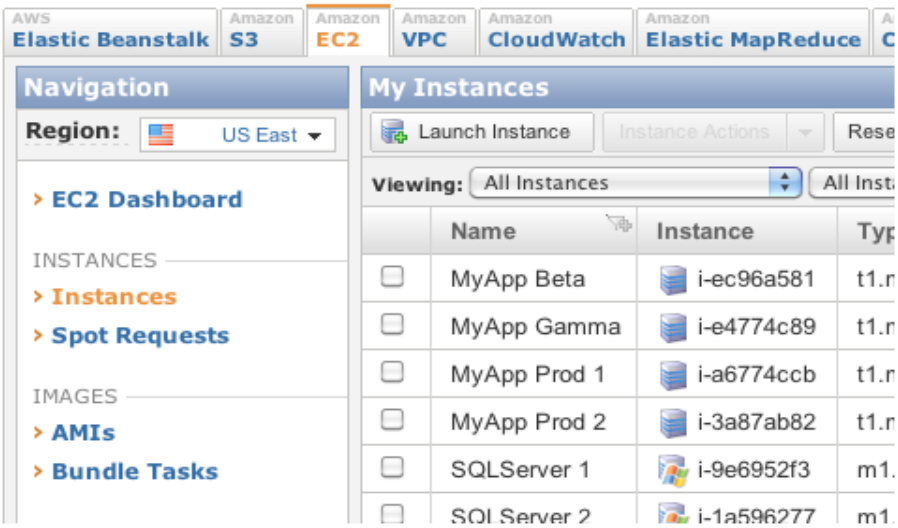
- AWS Toolkit for Eclipse - je plug-in pro Eclipse Java IDE, který umožňuje vyvíjet Java aplikace pomocí AWS. Toolkit také umožňuje interagovat s mnohými AWS službami přímo z Eclipse IDE.
- AWS Toolkit for Microsoft Visual Studio - je plug-in pro Visual Studio IDE umožňující vyvíjení .NET aplikací.

Možnosti

Jak již bylo zmíněno AWS poskytuje široké spektrum služeb zajišťující nejrůznější potřeby objednavatele. Zde jsou některé hlavní kategorie webových služeb s příkladem konkrétní služby:

- Výpočetní služby: Amazon Elastic Compute Cloud (Amazon EC2) - služba poskytující variabilní výpočetní kapacitu pro snadnější škálování.
- Dodávání obsahu: Amazon CloudFront - služba umožňující distribuci uloženého obsahu s nízkou ztrátovostí a vysokou přenosovou rychlostí.
- Databáze: Amazon SimpleDB - vysoce flexibilní, dostupné a škálovatelné úložiště, poskytující základní databázové funkce indexování a dotazů.
- Zprávy a synchronizace: Amazon Simple Queue Service - ukládání zpráv při komunikaci mezi počítači. Umožňuje vytvářet automatizovaná workflow při úzké spolupráci s Amazon EC2.
- Monitorování: Amazon CloudWatch - poskytuje monitoring nad cloudovými zdroji a spuštěnými aplikacemi na AWS.
- Sítě: Amazon Route 53 - je vysoce dostupná a škálovatelná DNS služba poskytující vývojářům vysoce spolehlivou cestu přístupu koncových uživatelů k webovým aplikacím.
- Platby a cena: Amazon Flexible Payments Service - služba nabízí možnosti, jak prakticky účtovat koncovým uživatelům cenu za dodané služby - prodej zboží, zasílání darů, opakovaných platby, atd...
- Úložiště: Amazon S3 (více v 3.8 Ukládání dat)

Správa a monitorování aplikací
AWS Management Console - konzole pro administrátory, která umožňuje praktickou správu výpočetních zdrojů, úložišť a dalších zdrojů - viz obrázek 2.3.



Obr. 2.3 - AWS Management Console [11]

Ukládání dat
Simple Storage Service (S3) je služba s webovým rozhraním k ukládání a získávání dat. Možnost ukládání jakéhokoli typu dat, „neomezeného“ množství objektů s maximální velikostí 5TB. Vysoce škálovatelná služba, zpoplatněná jen za to, co je skutečně spotřebováno, takže vývojáři mohou začít s malou aplikací a postupně ji rozšiřovat bez vlivu na výkon a spolehlivost.

Data je možné ukládat v jednom z několika regionů (US Standard, US West, EU, atd...) Uložené objekty region nikdy neopouštějí, pokud nejsou vývojářem transferovány. Pro vysokou dostupnost a odolnost, jsou objekty ve zvoleném regionu zálohovány na více zařízeních.

Základní protokol pro download je HTTP. Možné je také protokolové rozhraní BitTorrent, pro levnější vysoce škálovatelnou distribuci dat.

Reduced Redundancy Storage (RRS) je jedna z možností, kterou S3 nabízí a to jako možné snížení ceny za úložiště, tím že ukládaná data jsou označena jako nekritická, reprodukovatelná. Pomocí RRS jsou pak uložena s nižším stupněm zálohování (i když i tímto způsobem jsou data uložena se 400 násobnou trvanlivostí než na typickém HD).

	Standard Storage	Reduced Redundancy Storage
First 1 TB / month	$0.095 per GB	$0.076 per GB
Next 49 TB / month	$0.080 per GB	$0.064 per GB
Next 450 TB / month	$0.070 per GB	$0.056 per GB
Next 500 TB / month	$0.065 per GB	$0.052 per GB
Next 4000 TB / month	$0.060 per GB	$0.048 per GB
Over 5000 TB / month	$0.055 per GB	$0.037 per GB

Tabulka 2.5 - Amazon web services, ceny S3 a RRS [11]

Přenositelnost

- Přenositelnost aplikace (zda aplikace může být přemístěna do jiného prostředí) - dobrá přenositelnost, jestliže aplikace nepoužívá API specifické pro tuto platformu. Např. jestliže aplikace využívá Amazon DevPay, je třeba zůstat u AWS, protože při migraci na jinou platformu tato služba nebude funkční.

- Přenositelnost dat (zda data aplikace mohou být přesunuta na jiný server) - dobrá přenositelnost, jestliže jsou data na standardním databázovém serveru (např. MySQL). Při uložení data na specifických úložištích typu S3, SimpleDB není přenositelnost možná.

2.4 Google App Engine

Kde není v následujícím popisu platformy přímo uvedeno jinak, je čerpáno z oficiálních webových stránek společnosti Google: http://code.google.com/intl/cs-CZ/appengine [10].

V současné době jsou základem platformy Google tři hlavní služby:

- Google apps - Gmail, Kalendář, Dokumenty, Skupiny, Weby, Video.
- Google Apps Marketplace - otevřené prostředí pro směnu specializovaných aplikací vyvíjených třetími stranami. Marketplace nabízí cloudové aplikace, které využívají propojení aplikačních rozhraní s daty a prvky nacházejícími se v Google Apps.
- Google App Engine (GAE) - je cloudová platforma umožňující vývoj a implementaci webových aplikací na infrastruktuře společnosti Google. Této službě se budeme nadále věnovat.

Pro přístup k datům a práci s aplikacemi jsou dostupné tyto vstupní body:

- webový prohlížeč Chrome,
- mobilní operační systém Android,
- cloudově orientovaný operační systém Chrome OS [4].

2.4.1 Cena

Zdarma pro vytvoření účtu a aplikace, kterou mohou používat ostatní uživatelé, při splnění těchto podmínek: 1GB velikost úložiště a max. 5 miliónů prohlédnutí stránky měsíčně. Maximální počet aplikací u účtu zdarma je 10. Při potřebě většího množství je nutno zakoupit placený účet (od $150 za měsíc) a služba je zpoplatněna následovně:

Resource	**Unit**	**Pricing**
On-demand Frontend Instances	28 free instance hours	$0.08 / hour
Reserved Frontend Instances		$0.05 / hour
High Replication Datastore	1G	$0.18 / G / month
Outgoing Bandwidth	1G	$0.12 / G
Incoming Bandwidth	1G	Free
Datastore API	50k free read/write/small	$0.09/100k operations $0.06/100k operations $0.01/100k operations
Blobstore API	5G	$0.13 / G / month
Email API	100 recipients	$0.01 / 100 recipients
XMPP API	10k stanzas	$0.10 / 100k stanza
Channel API	100 channels opened	$0.01 / 100 channels opened
Total Storage (Documents and Indexes)	GB per month	$0.18
Simple Queries	10K queries	$0.13
Complex Queries	10K queries	$0.60

Tabulka 2.6 - Google App Engine, ceny výpočetních zdrojů [10]

Poznámky k uvedeným cenám:

- Namísto standardní hodinové sazby (On-demand Frontend Instances) se může vývojář rozhodnout pro zakoupení fixního množství hodin za týden za sníženou cenu (Reserved Instances). Služba je ale zpoplatněna bez ohledu na to, zda jsou rezervované hodiny využity, v případě že některé hodiny nesou spotřebovány, platba za ně se promítne do posledního dne v rezervovaném týdnu.
- Mezi small operace v datacentru řadíme například procházení indexů, count(), query s identifikátory.

Při uzavření smlouvy vývojář musí specifikovat svůj denní rozpočet. To je maximální suma, kterou si přeje za spotřebované prostředky platit. Tento denní rozpočet by měl počítat se špičkami používání aplikace, protože aplikace může za den spotřebovat pouze zdroje do výše denního rozpočtu. Všechny operace nad tuto výši se neprovedou.

Cena za spotřebované zdroje jsou vyhodnocovány na denní, týdenní (souhrn denních faktur + dodatečné platby, např. za "Reserved Instance" - viz výše) a měsíční (souhrn denních faktur + případné daně) bázi.

2.4.2 Licenční podmínky

Z velmi rozsáhlých a podrobných licenčních podmínek (uvedených na http://code.google.com/intl/cs-CZ/appengine/terms.html) vybírám pouze to, co bude vývojáře pravděpodobně zajímat nejvíce:

- Google přenechává vlastnictví a kontrolu nad vyvinutými aplikacemi a jakémukoli dalšímu uloženému obsahu druhé smluvní straně, která je sama zodpovědná za ochranu těchto práv. Zároveň je společnosti Google dáno všeobecné právo k adaptaci, editaci, překladu, publikování, apod... uloženého obsahu k jedinému účelu a to umožnění společnosti Google poskytovat smluvní straně smluvené služby v souladu s ochranou osobních údajů.
- Uživatel nesmí jakkoli kopírovat, editovat, nebo jinak extrahovat zdrojový kód App Engine.
- Další různé licenční podmínky se týkají softwaru třetích stran dodávaných s App Engine pro Java SDK (Jetty, Appache Ant a Geronimo, Jakarta http Client, Jasper a další).

2.4.3 Pro vývojáře

Programovací jazyky

Podpora aplikací psaných v několika programovacích jazycích

- Java runtime environment: JVM, Java servlets, Java, Javascript, Ruby
- Python runtime environment: Python 2.5.2, Python 2.7.2
- Go runtime evironment: Go r60.3

API

- Go API nabízí rozhraní pro tyto služby (popis služeb níže): Blobstore, Capabilities, Channel, Datastore, Mail, Memcache, Task Queues, URL Fetch, User
- Python API: Datastore, Google Accounts, URL Fetch, Mail
- Java API: Java Data Objects, Java Persistence, JavaMail, URL Fetch, Datastore, Memcache, Mail, Images, Google Accounts

Popis uvedených služeb:

- Datastore: robustní, škálovatelné objektové datové úložiště,
- Blobstore: umožňuje obsluhu datových objektů (zvaných "bloby"), které mají mnohem větší velikost než objekty v Datastore. Velké datové soubory, videa...,
- Capabilities: služba k detekci výpadků a plánovaných odstávek,
- Channel: služba pro zajištění trvalého spojení mezi aplikací a Google servery,
- Memcache: náročné škálovatelné webové aplikace často pro některé úkoly používají paměťovou cache pro doplnění nebo nahrazení trvalého úložiště,
- Task Queues: s touto službou mohou aplikace spouštět některé akce na pozadí, organizovat je do malých jednotek: tasks,
- URL Fetch: aplikace může komunikovat s jinými aplikace nebo jinými zdroji na webu pomocí načítání URL. Aplikace tak URL Fetch používá ke zpracování HTTP nebo HTTPS požadavků a přijímání odpovědí,
- Java Persistence: je standardní rozhraní pro ukládání, čtení a interakci s objekty v relační databázi,
- Java Data Objects: standardizované aplikační rozhraní pro ukládání objektů v Javě. Je možné ho použít pro relační a objektové databáze i pro souborový systém,
- Images: umožňuje aplikacím transformaci a manipulaci s obrazovými daty v několika formátech,
- Google Accounts: služba pro autentizaci uživatelů.

Verze aplikací

Je možné nahrávat nové verze aplikace, která již na App Engine běží. „Stará“ verze bude stále dostupná pro uživatele, dokud admin nepřepne na nově nahranou verzi. Je také možné testovat novou verzi na App Engine, zatímco „stará“ verze je stále používána.

Škálování

Pro aplikace jsou používány stejné škálovací technologie jako pro samotné Google aplikace (např. GFS a BigTable). S App Engine je škálování automatické, nezáleží na tom, kolik přistupuje uživatelů nebo kolik dat aplikace ukládá, App Engine automaticky škáluje podle aktuálních potřeb.

Možnosti

App Engine poskytuje široké spektrum pro vývoj aplikací, od základních nástrojů a webových rozhraní až po náročné aplikace pro spolupráci uživatelů, herní aplikace a vysoce škálovatelné business procesy:

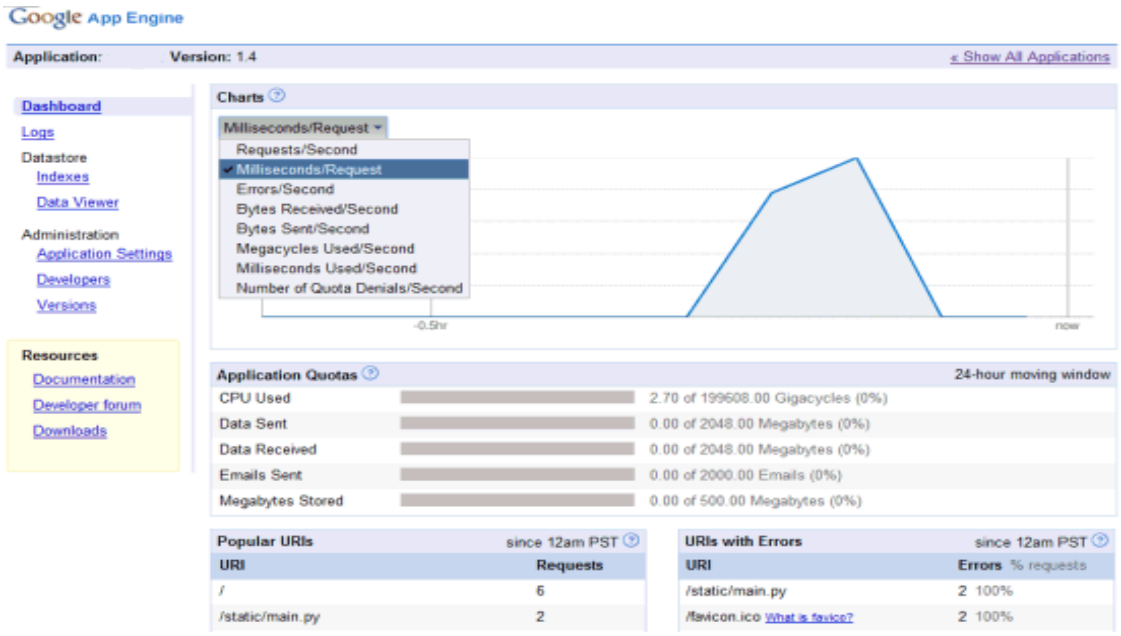
- Převodník skriptů (Script transliteration),
- Aplikace zprostředkující průzkumy (Survey App),
- Hry (Multiplayerová online strategie Neptune's Pride),
- Blogy (Wordpress, Bogger),
- Sledování novinek, návštěvnosti článků na sociálních sítích (Social Feed),
- Analyzování finančních dat společnosti (Financial Analyzer).

Vytváření aplikací

Konzole pro administrátora (viz obrázek 2.4) je webové rozhraní pro správu aplikací běžících na App Engine. Lze ji použít pro vytváření nových aplikací, konfiguraci názvů domén, změnu aktivní verze aplikace, správu přístupů, atd...

Každý SDK (pro Javu, Python a Go) obsahuje webovou aplikaci, která simuluje všechny služby App Engine na lokálním počítači. Obsahuje také lokální verzi datového úložiště, Google účty a možnost posílání emailů z počítače pomocí API App Engine [1].

Každý SDK také obsahuje nástroj pro upload aplikace na App Engine přímo z lokálního počítače. Po vytvoření kódu aplikace, souborů nastavení a souborů s daty je nástroj spuštěn a aplikace je nahrána.



Obr. 2.4 - Google App Engine, konzole administrátora [10]

Monitorování aplikací

System Status Dashboard umožňuje vývojářům vyhodnocovat a monitorovat celý systém App Engine měřením historie běhu systému, monitorováním frekvence chyb (i skrytých) pro každou hlavní komponentu App Engine, tj. úložiště, paměť (memcache), identifikaci uživatelů, infrastrukturu mailu, infrastrukturu služby aplikace.

Monitorování je nepřetržité, aktualizované každých několik minut a zobrazuje tak současný stav systému, zobrazovaných v desetiminutových průměrech. Dashboard také obsahuje historii dat za poslední týden.

Stav každé sondy může nabývat hodnot „Normal“, žlutý „Anomaly“ (např. při detekovaném zpomalení přenosů) a červený „Anomaly“ (když už chyba nebo zpoždění přenosu může znamenat hrozbu). Po kliknutí na tuto hodnotu, je zobrazen graf zobrazující výkon v průběhu daného dne. Tak může vývojář porovnávat incidenty, které se v aplikaci mohly ve sledované době odehrát, s výkonem odpovídajícího App Engine systému během sledované doby.

Ukládání dat

HRD - High Replication datastore je základní úložiště pro nové aplikace. Je to vysoce dostupné a spolehlivé řešení. Zůstává dostupné pro čtení/zápis během plánovaných odstávek. Je dražší než úložiště Master/Slave.

Data jsou replikována napříč datovým centrům pomocí systému založeném na algoritmu Paxos. Výhoda: nejvyšší úroveň dostupnosti, nevýhoda: přibližně 3x větší náročnost na velikost úložiště a čas CPU než Master/Slave.

Master/Slave využívá asynchronní replikaci dat do jiných datových center, tak jak jsou data zapisována. Je pouze jeden master pro zápis v daném čase, takže všechna čtení jsou velmi konzistentní, ale toto řešení je náchylné k nedostupnosti při výpadcích center, kde jsou data uložena.

Při vytváření nové aplikace je nastaveno HRD. Jestliže vývojář preferuje Master/Data, může to při vytváření nové aplikace provést, ale tento proces je nezvratný, tj. pro opětovný přechod na HRD je třeba vytvořit novou aplikaci a překopírovat úložiště do této nové aplikace.

	High Replication	**Master/Slave**
Performance		
Put/delete latency	1/2x-1x	1x
Get latency	1x	1x
Query latency	1x	1x
Consistency		
Put/get/delete	Strong	Strong
Most queries	Eventual	Strong
Ancestor queries	Strong	Strong
Occasional planned read-only period	No	Yes
Unplanned downtime	Extremely rare; no data loss	Rare; possible to lose a small % of writes occurring near downtime
Python 2.7 support	Only supported Datastore	Not supported

Tabulka 2.7 - Google App Engine, srovnání HRD a Master/Slave [10]

Přenositelnost

Na Google AppEngine mohou být nasazeny a používány nástroje pro přenositelnost aplikací:

- Web2spy umožňuje migraci mezi AppEngine a SQL databázemi,
- Django web framework a aplikace na něm běžící, mohou být používány po úpravě i na App Engine,
- Grails web application framework: aplikace mohou být po úpravě nasazeny na App Engine pomocí App Engine Plugin,
- AppScale je open-source framework, který umožňuje běh Java, Go a Python aplikací vyvinuté na AppEngine také na EC2 (Amazon), Eucalyptus a dalších,
- Bez těchto nástrojů je migrace na jinou platformu prakticky nemožná. "Aplikace jsou vytvořeny v Pythonu pro specifické API GAE. Dokonce i když je zákazník zkušený programátor v Pythonu, musí počítat s tím, že jeho GAE aplikace nemusí mimo infrastrukturu Google fungovat správně." [2]

2.5 Salesforce.com

Salesforce.com je cloudová společnost poskytující business software pro řízení vztahů se zákazníky. Řešení řízení vztahů se zákazníky se dělí na několik kategorií:

- Sales cloud
- Service cloud
- Data cloud
- Collaboration cloud
- Custom cloud

Nadále se budu v popisu věnovat poslední jmenované kategorii a to konkrétně platformě Force.com, umožňující vývojářům vytvářet aplikace, které jsou integrovány do Salesforce.com infrastruktury.

Force.com

Kde není v následujícím popisu platformy přímo uvedeno jinak, je čerpáno z oficiálních webových stránek společnosti Salesforce.com: http://www.salesforce.com/eu/platform/overview/ [9].

Tato služba se vytvořena ze dvou hlavních částí: AppExchange, online trh pro sdílení aplikací a samotná Force.com platforma, na které jsou tyto aplikace vyvíjeny.

2.5.1 Cena

Force.com nabízí dva základní typy licencí:

- Light: v ceně 10 dolarů za jednu licenci k jedné aplikaci nabízí všechny funkcionality potřebné k nasazení vlastní aplikace na Force.com - automatizaci workflow a business procesů, náhledy pomocí reportů a dashboardu, vývoj pomocí Drag&drop, nástroje a API pro profesionální vývojáře a mobilní služby,
- Enterprise: v ceně 25 dolarů zahrnuje všechny funkcionality dostupné pro verzi licence Light, navíc přidává zlepšenou správu přístupů a pokročilé API pro integraci třetích stran.

	Light	Enterprise
Cost per month	$10/app/user	$25/app/user
Features		
Accounts & contacts	Read-only access	Full read and write access
Activities, Tasks, Calendar, Events, Content, Documents, Ideas, Q&A	Full read and write access	Full read and write access
Enterprise Analytics • Data bucketing • Joined reports • Cross filters	Yes	Yes
Profiles	Yes	Yes
Permission Sets	Yes	Yes
Force.com Canvas - allow any app, in any language to run securely inside Salesforce	Yes	Yes
SOAP & REST APIs	Yes	Yes
Bulk API	No	Yes
Streaming API	No	Yes
Sharing mode	No	Yes
Capacity		
Custom objects	up to 10 /app	up to 10 /app
Custom tabs	up to 10 /app	up to 10 /app
Data Storage	20 MB /subscription	20 MB /subscription
File Storage	612 MB /subscription	612 MB /subscription
API calls per day	200 /subscription	200 /subscription

Tabulka 2.8 - Force.com, typy licencí [9]

2.5.2 Licenční podmínky

Hlavní znaky licenčních podmínek u Force.com

- Autoregulace: služby nemůže využívat více uživatelů, než je nakoupený počet licencí, to odstraňuje problémy s dostupností známých u služeb s neomezenou licencí.
- Synchronizace termínů: při přidání nových licencí v průběhu běžících (hlavních) licencí, tyto nové licence budou končit ve stejný termín jako hlavní licence.
- Synchronizace plateb: stejně tak bude účtována cena nových licencí, tedy jako cena již běžících, i když budou podepsány třeba v polovině nasmlouvaného období.
- Unikátnost uživatelských účtů: přístupové údaje uživatele jsou

jedinečné, nemohou být používány nikým jiným, k účtu nemůže přistupovat více jak jeden uživatel. Zajištěno monitoringem IP adres.
- Licence může být po odchodu uživatele ze společnosti převedena na nového uživatele.
- Jediná cesta je zvyšování: v průběhu nasmlouvaného období lze licence pouze přidávat, jejich snížení je možné pouze na výročí smluvního období.

2.5.3 Pro vývojáře

Force.com k nabízeným přednastaveným aplikacím umožňuje možnost jejich vlastních úprav (úprava objektů, layoutu stránek). Pro případ, že zákazník požaduje implementovat nějaký komplexní proces, který není podporován existujícími prostředky, nabízí Force.com pokročilým vývojářům několik způsobů pro implementaci vlastních funkcí. Tyto způsoby zahrnují SOAP, REST, Bulk a Metadata API, Apex a Visualforce.

API

- SOAP API: vytváření, čtení, změny a mazání dat (jako např. účty, vlastní objekty), udržování přístupových hesel, provádění vyhledávání a další.
- REST API: výkonné, praktické řešení webového rozhraní pro interakci s Force.com, vhodné zvláště pro využití mobilní aplikací a projektů Web 2.0.
- Bulk API: API na základě principů REST, optimalizováno pro nahrávání, aktualizaci a mazání velkých množství dat, akce jsou zpracovávány na pozadí Force.com (oproti tomu SOAP API je optimalizováno pro real-time aplikace, které v jednom okamžiku pracují s malým počtem záznamů).
- Metadata API: používané pro čtení, aktualizaci, vytváření a mazání vlastního nastavení, tj. nepracuje přímo s daty. Nejpoužívanější je pro migraci změn z testovací platformy na produktivní. Přístup k funkcím v Metadata API je umožněn pomocí Force.com IDE nebo Force.com Migration Tool.

Force.com IDE

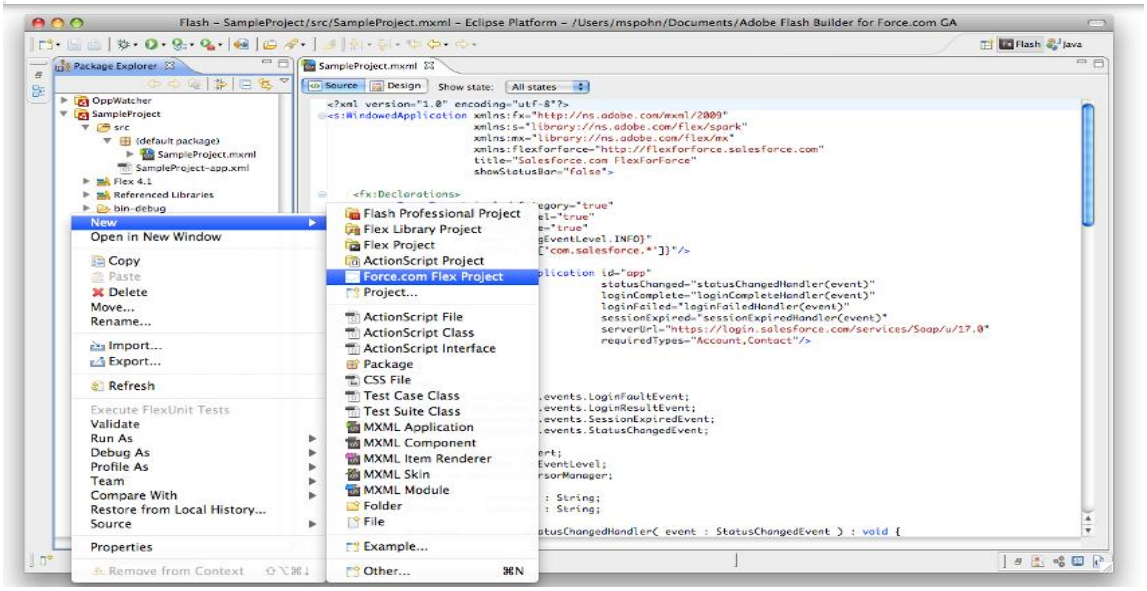
Force.com IDE je klientská aplikace pro vytváření, změny, testování a nasazování aplikací ve Force.com. Force.com IDE je založeno na platformě Eclipse a vývojářům tak nabízí pohodlné a známé prostředí pro kódování, kompilaci, testování a nasazování aplikací.

Hlavní prvky Force.com IDE:

- Kódování Apex: viz níže.
- Visualforce: viz níže.
- Komponenty aplikace: používají se pro stahování a editaci komponentů metadat z Metadata API. Komponenty se stáhnou do IDE, tam se provedou změny a po uložení jsou aktivní na produktivní platformě.
- Deploy to Server wizard: průvodce pro nasazení aplikace na produktivní

platformě.

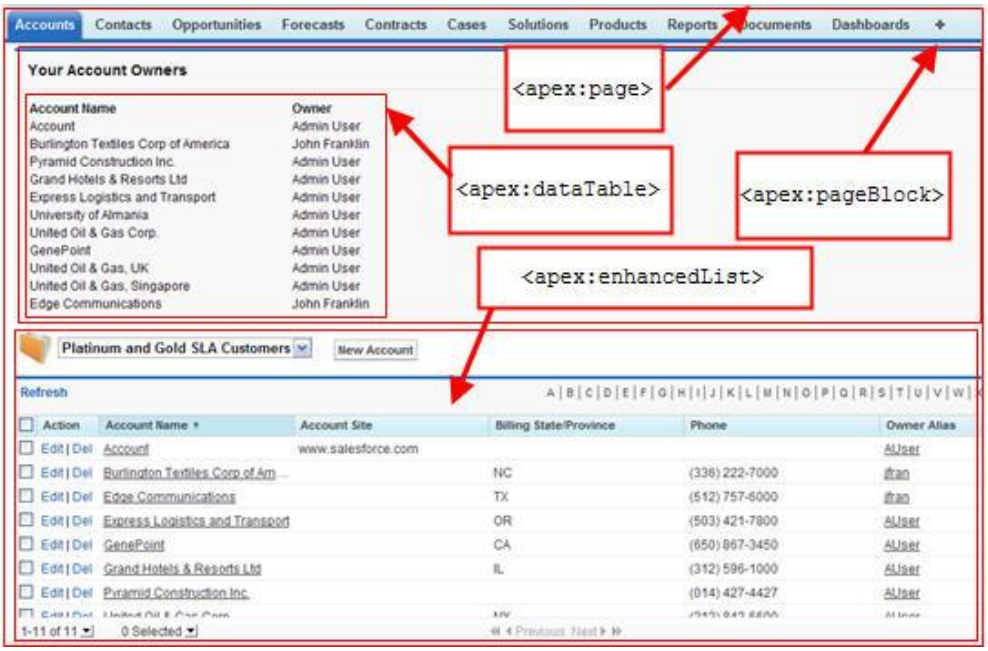
- Online Project Mode: pro udržování lokálních metadat aktualizovaných podle změn ve Force.com (plus detekce a náprava konfliktů).



Obr. 2.5 - Force.com IDE, platforma Eclipse [9]

Visualforce

„Visualforce je vlastně framework pro vytváření nových vzhledů rozhraní, umožňující interakci s uživatelem, které může být vytvořeno a dodáno bez jakýchkoli softwarových nebo hardwarových požadavků.“ [1].



Obr. 2.6 Ukázka komponentů Visualforce [13]

Visualforce nabízí následující vlastnosti a schopnosti [1]:

- Pages: tato funkce umožňuje definovat design uživatelského rozhraní aplikace, vytvářet nové stránky použitím standardních webových technologií (HTML, AJAX, Flex).
- Components: možnost vytvářet nové aplikace, které mají automaticky vzhled ostatních Saleforce aplikací, použitím preddefinovaných Salesforce standardů.
- Logic controllers: standardní kontrolery nabízí možnost využít předdefinovaných chování UI, jako je uložit, změnit, nový (new, edit and save). Kontrolery také umožňují zákazníkům vytvořit libovolné chování uživatelského rozhraní pomocí kódu v Apex.

Programovací jazyky

- HTML, Ajax a Flex: pro Visualforce
- Apex: vlastní programovací jazyk podobný Java. Ve Force.com používaný ve Force.com IDE pro:
  - vytváření webových služeb,
  - vytváření emailových služeb,
  - zpracování komplexních validací přes několik objektů,
  - vytváření komplexních procesů,
  - vytváření vlastní transakční logiky (přes několik záznamů nebo objektů).

Verze aplikací

Použitím Metadata API je možné vytvářet, implementovat a měnit definice aplikačních objektů, layout stránek, atd... Nejčastější využití je migrace změn z testovacího prostředí na produkční platformu.

Služba migrace je možná přes Force.com Migration Tool. Tento nástroj je nástavba Metadata API a používá standardní Eclipse a Ant nástroje. Force.com Migration Tool je ideální pro použití skriptu nebo příkazové řádky k migraci metadat mezi lokálním adresářem a infrastrukturou Force.com.

Škálování

Force.com v současné době zprostředkovává více než denně cca 170 miliónů transakcí, více jak polovinu z nich přes API, při současné rychlosti odezvy pod 300ms. Výkon je neustále sledován „výkonnostním“ týmem Force.com, který případné problémy vyhodnocuje a okamžitě odesílá vývojovému týmu. Dost často je problém vyřešen, než ho zákazník vůbec zaznamená. Protože platforma klade na vývojáře už při vývoji aplikací jistá omezení, co mohou dělat (např. není možné najednou nahrát obrovské množství řádek, tak že každá bude individuálně zpracovávána ve smyčce), takže při designové fázi vývoje aplikace nemusí být přímo škálovatelnost zvažována.

Vytváření aplikací
Force.com pomocí point-and-click zpřístupňuje vývoj aplikací nové skupině uživatelů bez znalostí programování, ale také urychluje vývoj zkušeným programátorům. Vývojář nemůže měnit kód jádra aplikace (např. způsob renderování layoutu stránky), namísto toho musí pracovat v rozmezí omezení definovaných metadaty. To ale na druhou stranu umožňuje bezpečné modifikace aplikací, bez velkých nároků na cenu a námahu.

Postup pro vytvoření jednoduché aplikace pomocí modelu point-and-click [1]:

- Vytvoření objektu: v objektu budou uložena data. Force.com nabízí řadu předdefinovaných objektů, ale je možné vytvořit vlastní. Stačí zvolit vytvoření nového objektu a otevře se obrazovka s nabídkou popisu a vlastností objektu, není třeba nic programovat.
- Vytvoření polí: do polí budou uživateli zadávána data, která budou následně ukládána v objektu. Podobně jako u objektů, je otevřena nabídka charakteristiky pole, je možné vybírat z předdefinovaného seznamu typů polí - Auto Number, Formula, Lookup, Checkbox, Date, Text, atd...
- Vytvoření záložky: aby bylo možné aplikaci na webu Salesforce vyhledat je nutné vytvořit záložku a zařadit jí mezi už existující záložku.
- Vytvoření aplikace: služba New Custom App Wizard umožní aplikaci pojmenovat, popsat, připojit logo, nastavit viditelnost a další vlastnosti. Po ukončení průvodce je aplikace nabízena v seznamu dostupných aplikací.

Vytvořením tří základních prvků (objekt, pole, záložka) máme hotovu jednoduchou spustitelnou aplikaci, která umožňuje základní funkčnost - otevřít nový záznam, zapsat do polí data, která jsou následně uložena do objektů, odkud mohou být kdykoli přečtena.

AppExchange
Jak bylo zmíněno výše, pro tvorbu a používání aplikací je možné využít on-line trh hotových aplikací AppExchange (úložiště aplikací vytvořených pro Salesforce.com). AppExchange v současné době obsahuje přes 1700 aplikací (při spuštění v roce 2005 to bylo 70 aplikací, v roce 2008 už 750 [1]). Využitím AppExchange mohou zákazníci snadno přidávat nové aplikace k jejich už existujícím na Saleforce.com architektuře. To je umožněno vývojem aplikací v širokém rozsahu business požadavků:

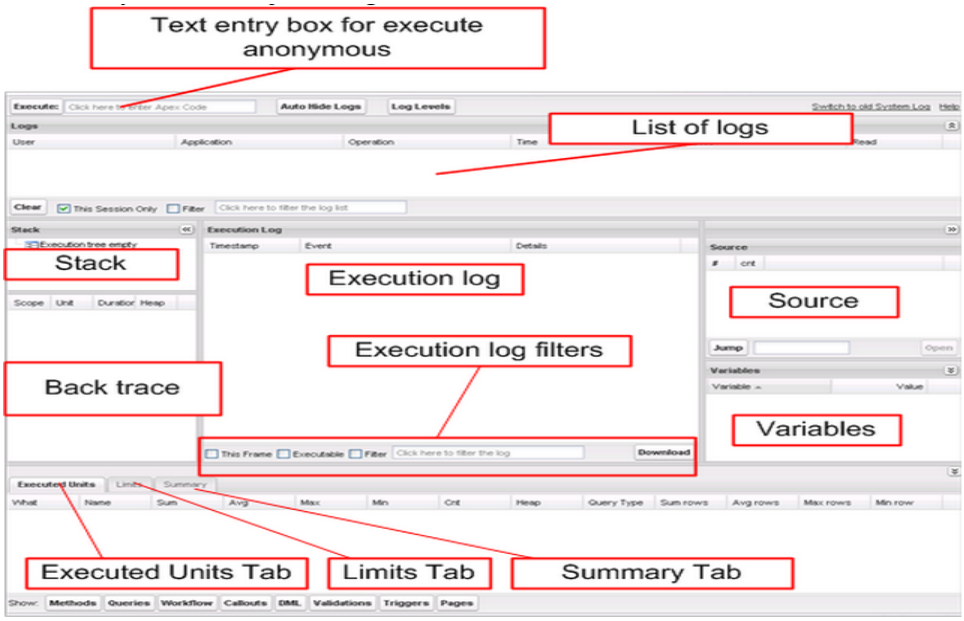
- Finance,
- Elektronické podpisy,
- Management dokumentace,
- Projektový management,
- Platby a sbírky,
- Mobilní řízení pracovních sil,
- Čištění dat,
- Profesionální služby v oblasti řízení,
- Lidské zdroje [1].

Možnosti
Pro vývoj a běh aplikace není třeba používat všechny funkcionality Force.com. Podle toho jaká aplikace je vyvíjena, tak je nutná jen jedna nebo dvě funkce. Jsou tři základní typy Force.com aplikací: původní, kombinované a klientové.

- Původní aplikace jsou vyvinuty výhradně konfigurací metadat bez kódování. Běh je zcela na platformě Apex bez nutnosti používat externí služby nebo infrastrukturu,
- Kombinované aplikace jsou vytvořeny spojením použití metadat s nízkoúrovňovým kódováním a voláním API. Umožňují větší flexibilitu a integraci dalších služeb, ale mohou vyžadovat běh a správu externího kódu,
- Klientové aplikace běží mimo uživatelské rozhraní Force.com, používají pouze Apex API, obvykle běží na mobilních zařízeních (BlackBerry) nebo stolních počítačích (spojení s Microsooft Outlook). Tyto aplikace používají platformu Apex pouze jako datový zdroj, neběží přímo na Force.com platformě, takže nemohou být automaticky instalovány pomocí AppExchange.

Správa a monitorování aplikací
Force.com Developer Console - je samostatné okno zobrazující informace o ladění programu, použitých zdrojích v porovnání s limity systému, o spuštěném zdrojovém kódu. Je to souhrnný prohlížeč zdrojů sledované operace, co operaci spouští a co se s ní po spuštění děje. Informace o ladění zahrnuje změny v databázi, zpracovávání kódu Apexu, workflow a logice schvalování.

Další typické možnosti konzole vývojáře mohou být např. vyhodnocování stránek Visualforce, sledování DML v transakcích nebo monitorování výkonu.



Obr. 2.7 - Force.com, konzole pro vývojáře [9]

Ukládání dat
Force.com má data uložena ve tří hlavních datových centrech, používajících šířku pásma s vysokou dostupností, poskytujících služby 3 miliónům registrovaným předplatitelům. Zařízení jsou propojena odolnými multi-carrier úložišti, zálohováním MPLS (Multiprotocol label switching) / VPLS (Virtual Private LAN service), poskytující „failover" spínání v reálném čase, „point to multipoint" replikaci dat a také „disk to disk to tape" zálohování.

- všechny síťové komponenty jsou nastaveny duplicitně,
- všechna uživatelská data jsou uložena na primárním databázovém serveru, který je kvůli duplicitě spojen se záložním databázovým serverem,
- všechna data jsou uložena na diskovém úložišti, která jsou replikována napříč různými dalšími datovými úložišti,
- všechna data až k poslední transakci jsou automaticky zálohována na páskách každou noc,
- pásky jsou okamžitě zálohovány na sekundární pásky pro zajištění jejich integrity, tyto nové pásky jsou pak pravidelně přemisťovány na bezpečná, proti požáru zajištěná místa, umístěná mimo hlavní úložiště (zajištění proti přírodním katastrofám).

Přenositelnost
Force.com je velmi výkonná a vysoce vnitřně propojená platforma, vyvinuté aplikace využívají vlastních technologií. Je to konečné, špatně přenositelné řešení, využívající enterprise aplikace, které jsou integrované v Salesforce.com. Tato platforma je tedy nejlépe využitelná pro organizace již využívající některé Salesforce.com aplikace. V porovnání s výše představenými platformami (AWS, GAE) je to asi nejvíce samostatná platforma s jedinečnými vlastnickými právy → velmi špatná přenositelnost.

- Přenositelnost aplikace (zda aplikace může být přemístěna do jiného prostředí) - Apex je vlastní jazyk, aplikace tedy není přenositelná, pokud tedy není kód zcela přepsán.

- Přenositelnost dat (zda data aplikace mohou být přesunuta na jiný server) - špatná přenositelnost kvůli používání vlastní jedinečné databáze, metadat a multi-tenantní architektury.

2.6 Cordys

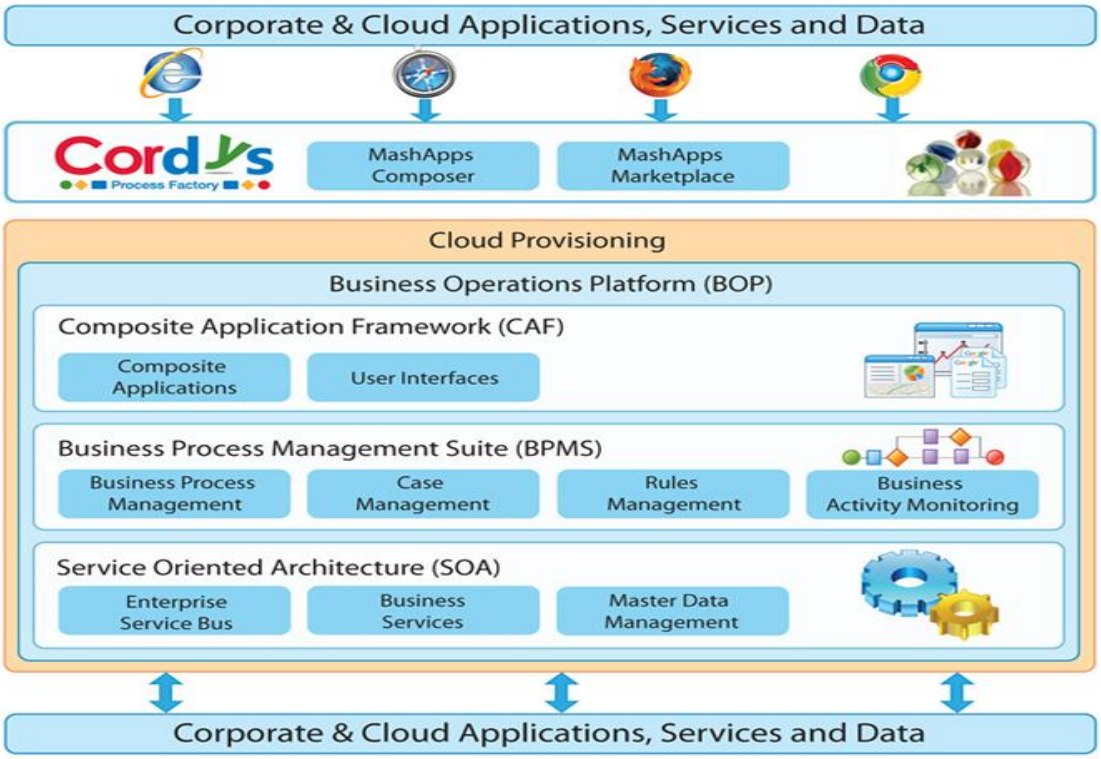
Nebudeli v textu uvedeno jinak, je čerpáno z oficiálních webových stránek společnosti Cordys http://www.cordys.com/business_operations_platform [8], v části "2.6.3 Pro vývojáře" také z vlastní zkušenosti.

Cordys je holandská společnost nabízející koncepci Business Process Management, tedy řízení komplikovaných procesů, které zároveň probíhají v různých informačních systémech a skládajících se mnoha kroků.

Platforma Cordys je umístěna na infrastruktuře Amazon Web Services (EC2), firma je jedním z Independent Software Vendors (ISVs) platformy AWS.

Pro uvedené řízení procesů platforma Cordys nabízí Cordys Business Operations Platform, což je kompletní nabídka služeb pro management business procesů a jejich monitoring. Tato nabídka má pomoci přímo sladit business procesy s obchodními cíli a současně zajistit zlepšení těchto procesů.

Ústředním bodem Cordys Business Operations Platform je Cordys Business Process Management Suite - služba, která je navržena k podpoře neustálé procesní optimalizace. Služba zahrnuje Identifikaci, Modelování, Realizaci, Sledování a Vylepšování procesu, tak aby proces co nejlépe odpovídal kladeným požadavkům dle aktuální situace ve firmě.



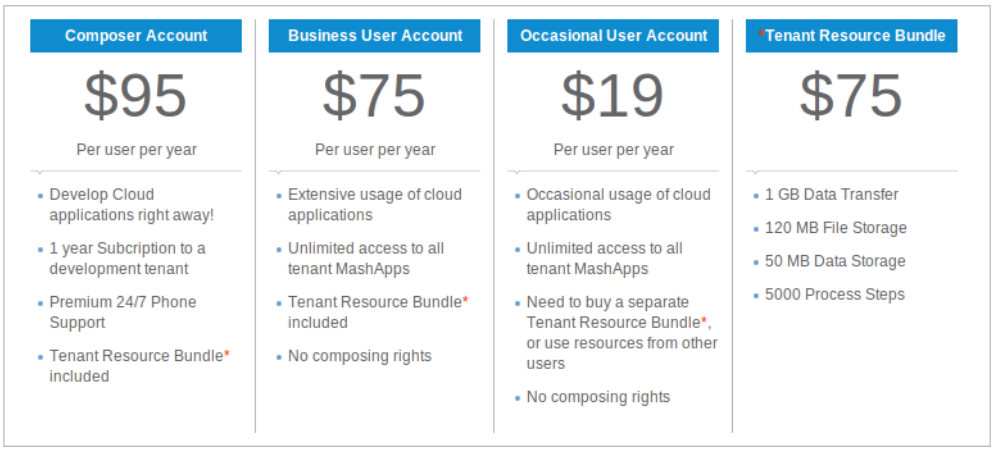
Obr. 2.8 - architektura platformy Cordys [8]

Pro vývoj a běh firemních procesů ve webovým rozhraním nabízí společnost Cordys svou PaaS - Cordys Process Factory. V následujícím textu se budu zabývat popisu této platformy, tj. CPF - Cordys Process Factory. Tato služba nabízí vytváření business procesů a cloudových aplikací zvaných MashApps (od „mashup“ - webová aplikace používající prostředky ze dvou a více zdrojů). Lze kombinovat standardní business aplikace (např. Google aplikace) a další komerčně dostupné služby s vlastními či předdefinovanými webovými službami.

Cordys Process Factory

2.6.1 Cena

Na prvních 30 dní používání služby je možné zvolit zkušební bezplatnou verzi. Další zpoplatnění je počítáno podle počtu licencí, od účtu pro vývojáře až po účet běžného uživatele (viz obrázek 2.9). (Poznámka: "Tenant" v tomto případě v řeči CPF znamená typ platformy, tj. vývojová, testovací, produktivní. Máme tedy *Development Tenant, Acceptance Tenant* a *Production tenant*).



Obr. 2.9 - ceny licencí Cordys Process Factory [8]

2.6.2 Licenční podmínky

Detailně o licenčním ujednání se lze dočíst na webu Cordys: http://www.cordysprocessfactory.com/node/add/enterpriseuser?acctype=trial#, kde např. Cordys přímo uvádí, že není designován ani zamýšlen pro vysoce rizikové aktivity.

2.6.3 Pro vývojáře

Programovací jazyky

CPF podporuje AJAX. Ve skript editoru formulářů používá Javascript, pro design jednotlivých komponent uživatelského rozhraní je to HTML. Výsledná podoba aplikace pro nasazení na CPF platformu je ve formátu XML souborů.

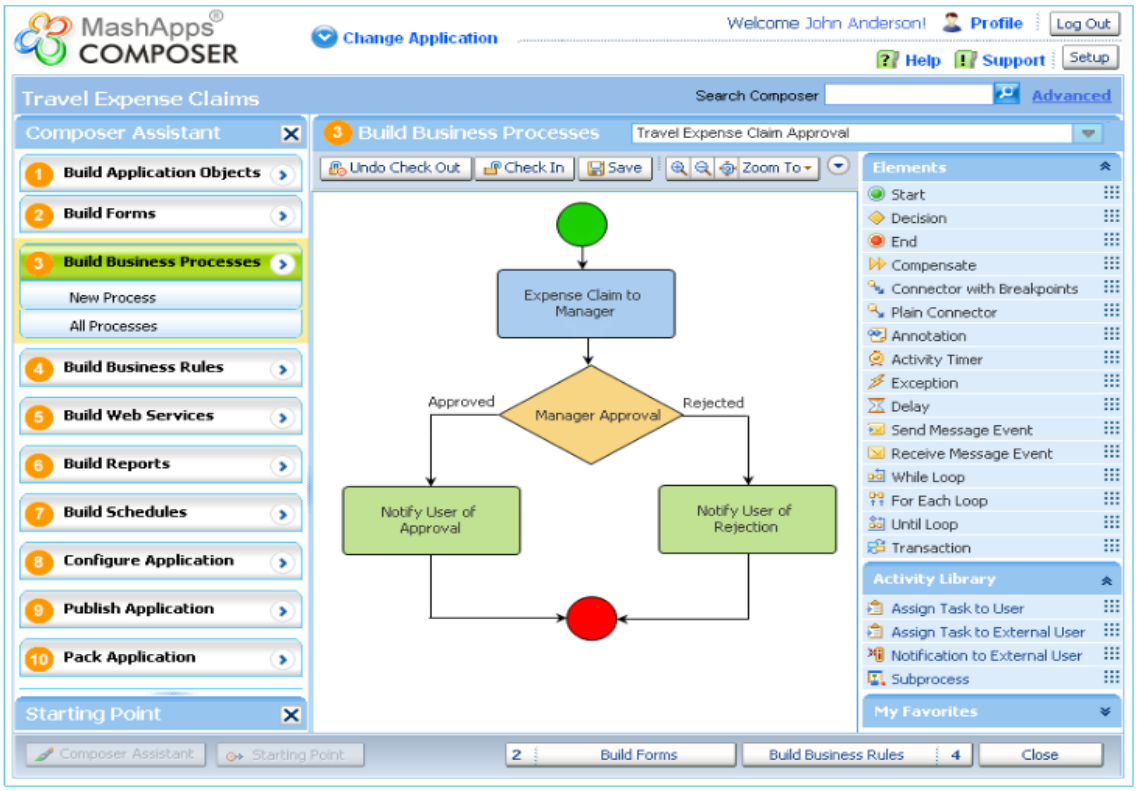
Verze aplikací

Vývojář může vyvíjet a testovat novou verzi aplikace, zatímco je předchozí verze aplikace stále používána. Po transferu aplikace na production platformu je nová verze připravena ihned k použití bez ztráty dat uložených v aplikačních objektech.

Vytváření aplikací

CPF obsahuje „MashApps composer“, online vývojové prostředí, které používá vývojář pro vývoj své cloudové aplikace.

- „Application Objects“: objekty aplikace jsou databázové tabulky pro ukládání dat.
- „Forms“: formuláře lze designovat pomocí drag-and-drop, jejich akce a chování ovlivňovat pomocí „Form rules“ a script editoru.
- „Business processes“: umožňují vývoj procesů pomocí vizuálního online nástroje (viz obrázek 2.10) s řadou předdefinovaných funkcí a webových služeb, které slouží k interakci s uživatelem a úpravám dat v aplikačních objektech. Lze také používat vlastní služby, které byly vytvořeny vývojářem.



Obr. 2.10 - MashApps composer, business procesy [12]

- „Business Rules“: pravidla nad jednotlivými aplikačními objekty, sledující vytváření nových záznamů nebo jejich aktualizace. Tato pravidla většinou slouží ke spouštění nebo naopak zastavování procesů nebo akcí (např. kontrola povinných polí při ukládání záznamů).

- „Reports“: reporty je možné vytvářet pomocí „Report Wizard“, různé typy reportů: standardní, skupinové, tabulky, grafy, atd... Každý report čerpá data z jednoho aplikačnícho objektu (v případě že objekt je spojen relací s dalším objektem, tak i z tohoto druhého objektu), které pak může exportovat do xls, pfd a dalších formátů.
- „Webové služby“: generování a vývoj vnitřních webových služeb a registrace webových služeb třetích stran (SOAP a REST).

Možnosti
Integrace Google Apps do platformy, v aplikacích je možná přímá interakce např. s Google dokumenty, Google kalendářem a dalšími Google službami.

Po vytvoření aplikace v „MashApps composer“ je možné nabízet a prodávat své aplikace v „Cloud Marketplace“. Uživatelé, kteří potřebují nějakou CPF aplikaci, si ji mohou samozřejmě vytvořit sami, nebo využít Cloud Marketplace, které nabízí různé kategorie již vytvořených aplikací, které byly vytvořeny přímo CPF nebo vybranými partnery. Některé základní aplikace jsou zdarma, jiné jsou zpoplatněny.

Monitorování aplikací
BAM (Business Activity Monitoring) je služba umožňující monitorování klíčových business událostí pro sledování změn nebo trendů. Řešení BAM poskytuje dynamické prostředí obsahující KPI (Key Performance Indicators), analýzy problémů a varování na hrozící problémy.

BAM umožňuje uživatelům kontrolu jejich business operací prostřednictvím:
- transparentního realtime pohledu na aktuální stav jejich procesů, umožňující lépe informovaná rozhodnutí,
- nepřetržitým monitorováním operací umožňující odhalení hrozících událostí,
- poskytováním možnosti pro vytvoření smyček, které pravidelně provádějí nastavené kontroly a vyrovnávání, také okamžité odpovědi při změnách v business procesech.

Přenositelnost
- Přenositelnost aplikace (zda aplikace může být přemístěna do jiného prostředí) - CPF nabízí vlastní vývojářské rozhraní, jednotný vzhled aplikací, předdefinované knihovny a funkce, je to vlastní, jedinečné a nepřenositelné řešení.

- Přenositelnost dat (zda data aplikace mohou být přesunuta na jiný server) - data z objektů aplikace jsou velmi dobře exportovatelná např. do MS Excel, kde názvy polí jsou pak zobrazeny jako hlavička souboru. Data jsou tedy přenositelná velmi dobře. Existují i API třetích stran pro download a upload souborů - příloh.

2.7 Zhodnocení

Jednou z největších výhod Cloud Computingu proto je, že platím to, co opravdu spotřebuji. Firemní zákazník tedy nepotřebuje udržovat rezervní lokální výpočetní prostředky, aby je využil jednou měsíčně např. v době špiček finančních uzávěrek (o snadné navýšení alokovaných zdrojů v určitých hodinách nebo dnech se stará škálovatelnost), nepotřebuje platit navíc personál, který se bude o lokální zdroje starat, nepotřebuje se starat o napájení, nepotřebuje chladit, odpadá starost za udržování aktuálního softwaru, antivirové ochrany, apod.

Další výhodou Cloud Computingu je možnost přístupu ke svým datům a aplikacím odkudkoli, na jakémkoli operačním systému, pomocí libovolného webového prohlížeče. Zde ovšem je ke svážení personální charakteristika konkrétní firmy. Rozdílný přístup určitě bude u firmy, kde většina zaměstnanců cestuje a používá přístup k aplikacím z různých stanic, opakem je centralizovaná firma, kde 90% zaměstnanců přistupuje k firemním datům ze stejné stanice.

Jedním z důležitých hledisek pro rozhodnutí zda Cloud Computing ano nebo ne, je určitě to, zda se svou firmou teprve začínáme a vytváříme nové systém pro práci, nebo budeme migrovat stávající systém na novou platformu. Pro nové firmy je Cloud Computing určitě zajímavou možností, u firmy se stávajícím zavedeným systémem to bude záležet na finančním zhodnocení, kolik už bylo investováno do stávajícího systému, kolik bude potřeba investovat do vývoje aplikací v novém prostředí, počet zaměstnanců, kteří byli schopni udržovat systém před migrací a kolik jich bude potřeba poté, případně jak naložíme s uvolněnými kapacitami lidských zdrojů.

Na finanční rozvaze zákazníka také bude záležet výběr konkrétní cloudové služby, jejichž nabídka je u uvedených platforem veliká. Např. AWS a GAE nabízí různé druhy úložišť s různou úrovní zabezpečení a dostupností, samozřejmě s rozdílnou cenou.

Jednou z dalších velkých výhod Cloud Computingu jsou pravidelné platby za poskytované zdroje, tj. každý měsíc odvedeme poskytovateli přibližně stejnou částku, zatímco nákup lokálních výpočetních zdrojů je vysoká jednorázová položka, kterou žádný finanční ředitel neuvidí rád, zatímco s rozepsanou do měsíčních plateb by s ní neměl problém.

Ohledně názorů na rizika využívání Cloud Computingu bych chtěl zmínit dva faktory - lokalizace a „vendor lock in“:

Lokalizace:
Zákazník většinou nezná přesnou lokalizaci, kde jsou jeho data umístěna (např. jen zákazníci z EU si mohou být jisti, že jejich data nejsou ukládána mimo EU, nebo že zákazníci z Kanady nemají ukládány data v USA, obojí ze zákonných požadavků daných států), proto se nemusí cítit jistě, ohledně uložení svých dat.

Ale na druhou stranu to, že zákazník nezná přesné umístění, může naopak pomoci k ochraně těchto dat. Představme si hackera, který chce zaútočit na data konkrétní firmy. Jakmile by znal přesné umístění dat, bylo by pro něj jednodušší k těmto datům přistoupit, než když jsou tato data umístěna i pro konkrétní firmu na neznámém místě.

Vendor lock in:
Druhým rizikovým faktorem je pro mne vázání dat a aplikací na konkrétního poskytovatele a obtížný přenos dat k jinému poskytovateli. Kdo zákazníkovi zaručí, že za několik let bude jeho současný cloudový poskytovatel existovat? Že nezbankrotuje, že mu kvůli nějakému porušení zákona nebudou zablokovány servery? Např. některé charakteristiky průmyslových výrobků musí být archivovány 15 let, některé účetní a personální záznamy dle zákona dokonce 45 let. V případě, že mezitím poskytovatel zanikne, jakou bude mít zákazník šanci svá data exportovat a nepřijít o ně?

V možnosti přenositelnosti proto vidím nejdůležitější hledisko srovnání uvedených platforem. Amazon Web Services, v případě že nepoužíváme specifické služby, které migraci znemožňují (DevPay), nabízí snadnou přenositelnost aplikací, Google k přenositelnosti poskytuje několik speciálních nástrojů. Oproti tomu úzce zaměřené platformy Force.com (při použití Force.com IDE) a Cordys Process Factory migraci aplikací, kvůli vlastním specifickým vývojovým rozhraním, prakticky znemožňují. Pokud navíc společnost nabízející PaaS nemá platformu umístěnou na vlastní infrastruktuře (podobně jako CPF umístěná na Amazon, oproti tomu Force.com na infrastruktuře Salesforce.com), tak mezi zákazníka a vlastníka infrastruktury vstupuje ještě další poskytovatel, takže se ve výsledku spoléháme na dvě společnosti, pravděpodobnost ztráty dat se tak násobí.

Z výše uvedeného mi jako nejvýhodnější forma Cloud Computingu vychází IaaS, tj. využívám pouze infrastrukturu poskytovatele, na které běží mé aplikace, které v případě potřeby mohu přenést k jinému poskytovateli.

3 Cloudová aplikace

Tato kapitola přiblíží implementaci cloudové aplikace na platformě Cordys. V první části bude popsáno zadání aplikace, ve druhé architektura a vývoj aplikace a ve třetí rozšíření aplikace, ke kterým bylo přistoupeno po nějaké době používání, ať už z důvodu nápravy objevených chyb nebo požadovaných vylepšení.

3.1 Zadání

Na platformě Cordys bude vytvořena aplikace pro správu a archivaci uživatelských přístupů a přidělených IT zdrojů, která bude sloužit pro interní potřeby stejně jako důkazní materiál pro externí audity.

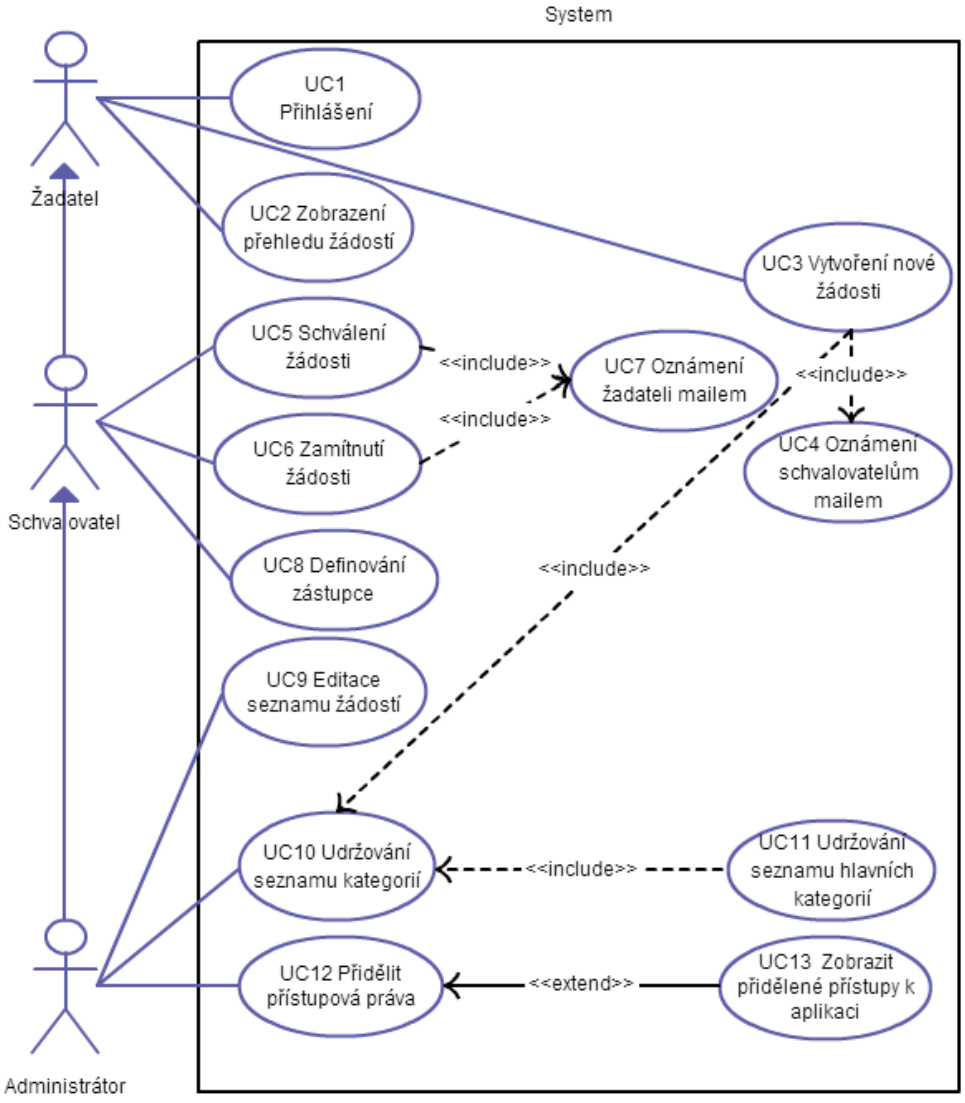
Aplikace umožní žadateli podat elektronickou žádost o přidělení IT zdroje nebo získání přístupu, která následně projde schvalovacím workflow.

Aplikace umožní žadateli podat žádost nejen pro sebe, ale i pro svého kolegu (např. pro případ nového uživatele, který ještě nemá přidělen PC nebo přístup ke Cordys).

Název aplikace „User Management“.

Use Case diagram
Na následujícím obrázku 3.1 Use Case diagram - můžeme vidět hlavní funkcionality systému:

- Běžný uživatel se může do aplikace přihlásit (UC1), zobrazit přehled všech vytvořených žádostí (UC2) a novou žádost vytvořit (UC3) - k tomu bude potřebovat zobrazit seznam všech kategorií požadavků (spravuje administrátor - UC 10).
- Schvalovatelé mohou žádost schválit nebo zamítnout (UC5 nebo UC6), mohou také za sebe definovat zástupce (UC8 - tato funkcionalita v první verzi aplikace neexistovala, byla přidána později, viz část 3.3 Rozšíření).
- Administrátor má možnost editace a mazání všech žádostí (UC9), může přidělovat přístupová práva k aplikaci (UC12), případně zobrazit přehled všech aktivních přístupů a udržovat seznam kategorií požadavků (UC10), ze kterých si žadatel při vytváření své žádosti vybírá. Správa kategorií je navíc rozšířena o správu seznamu hlavních kategorií (UC13 - tato funkcionalita v první verzi aplikace neexistovala, byla také přidána později, viz část 3.3 Rozšíření).



Obr. 3.1 - Use Case diagram

3.1.1 Vyhledávání

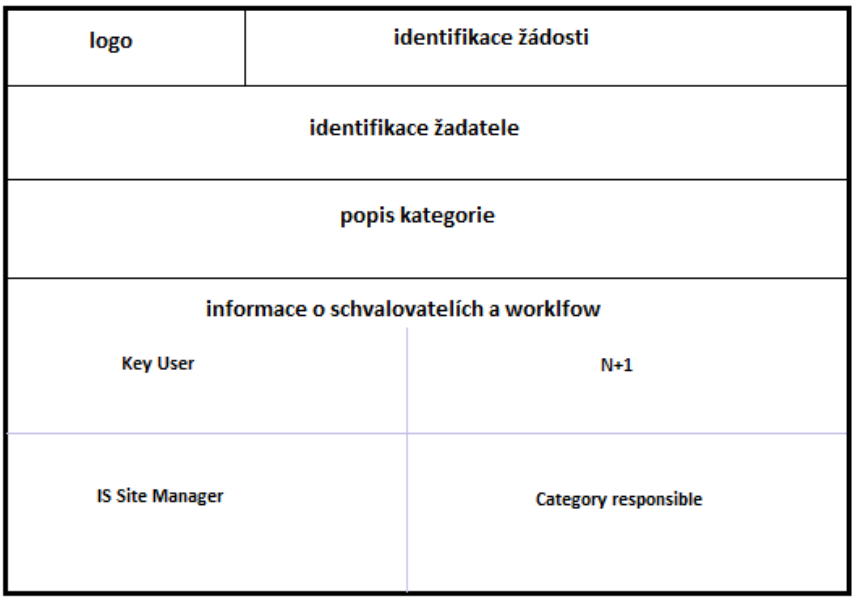
Všechny schválené uživatelské přístupy a přidělené zdroje budou archivovány a bude je možno vyhledávat alespoň podle kategorie požadavku a jména žadatele, tak abychom např. po ukončení zaměstnaneckého poměru konkrétního uživatele mohli vyhledat a poté zrušit všechny jeho aktivní přístupy a neplatili zbytečně poplatky za licence (SAP, BW, atd…)

Administrátorům bude umožněno vyhledávat podané žádosti a editovat je v jakémkoli stavu workflow.

3.1.2 Formulář elektronické žádosti

Formulář požadavku nechť obsahuje:

- identifikace žádosti: identifikační číslo požadavku, jeho datum vytvoření a aktuální stav workflow,
- identifikace žadatele: jméno žadatele (nadále *Created by*), jméno uživatele, pro kterého je žádost podávána (nadále *Request for*). Patří sem také *Orgaloc* (organizační jednotka) uživatele *Request for*, jeho zaměstnanecký stav, v případě že je znám, tak i předpokládaný datum konce zaměstnaneckého poměru,
- popis kategorie: název kategorie, detaily kategorie (nepovinné), komentář uživatele,
- informace o schvalovatelích a workflow: historie schvalování požadavku tak, aby bylo možné poznat kdo, kdy a s jakým komentářem (povinné pouze pro případ zamítnutí požadavku) schválil nebo zamítl konkrétní požadavek.



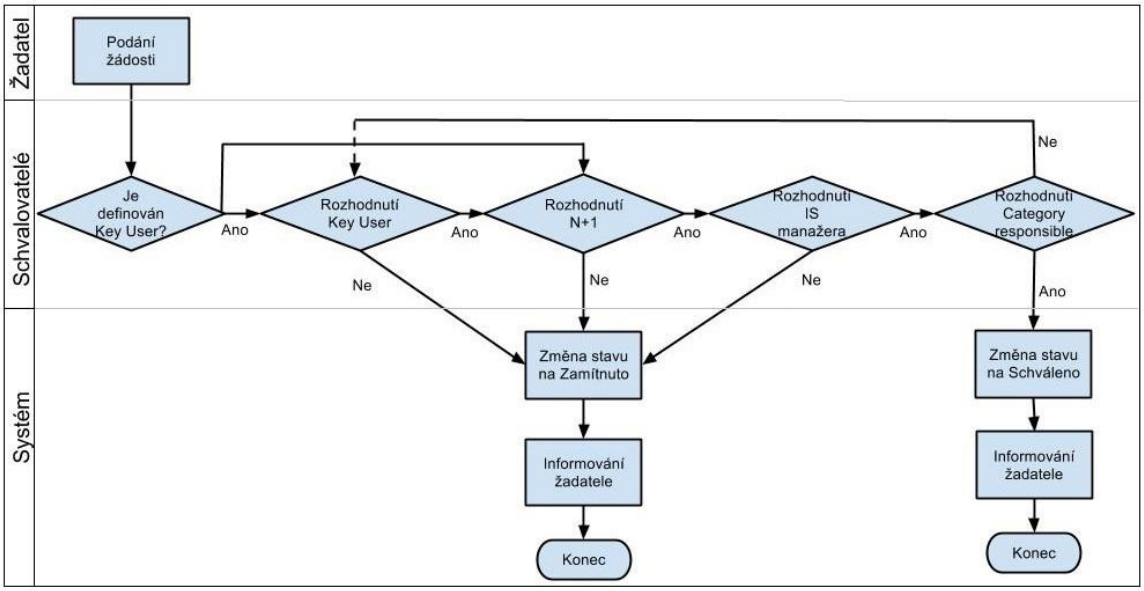
Obr. 3.2 - prototyp hlavního formuláře

3.1.3 Schvalovací workflow

Aplikace bude obsahovat pevně definované schvalovací workflow (klíčový uživatel (nadále Key User), nadřízený žadatele (nadále N+1), manažer informačních systémů (dále IS Site Manager) a uživatel odpovědný za konkrétní kategori požadavku (dále Category responsible). Žadatelům nebude povoleno workflow měnit.

Key User není povinný účastník workflow, je zahrnut pouze pro v případě, že je pro danou kategorii definován. Všichni účastníci workflow mají možnost požadavek schválit nebo zamítnout (tím vrátit žadateli), Category responsible nemůže požadavek zamítnout, pouze vrátit pro doplnění uživateli Key User.

Žadatel bude informován mailem o podání a schválení/zamítnutí žádosti, v případě, že žadatel podává žádost pro svého kolegu, budou informováni oba.



Obr. 3.3 Vizualizace workflow

3.1.4 Kategorie z uživatelského hlediska

Uživateli je při podávání žádosti nabídnut seznam kategorií, z nich může pro každou žádost vybrat pouze jednu. Kategorie budou tříděny dle uživatelova *Orgaloc* (organizační jednotka), protože pro každý závod budou předdefinováni jiní účastníci workflow.

3.1.5 Kategorie z administrátorského hlediska

Administrátorům aplikace bude umožněno snadné vyhledávání stávajících kategorií, jejich kompletní přehled, editace a přidávání nových kategorií. Ke každé kategorii je možné uvést i podrobnější popis.

3.1.6 Ergonomie

Uživatelské prostředí nechť je přehledné, uživateli bude umožněno vytvořit nový požadavek s minimem akcí, tj. zvolit vytvoření nové žádosti, vybrat kategorii, doplnit komentář a odeslat ke schválení. Vyhledání všech ostatních informací o uživateli a nastavení workflow bude automatické.

3.2 Vypracování

Vývoj aplikace na platformě Cordys Process Factory se provádí v modelovacím prostředí založeném na webovém prohlížeči, které umožňuje definovat různé druhy modelů. Mezi základními modely to jsou business procesy, uživatelské rozhraní a aplikační objekty. XML dokumenty reprezentující tyto modely jsou ukládány v relační databázi.
Aplikační objekty slouží k ukládání dat ve formě tabulky, kde jednotlivé řádky představují jednotlivé záznamy. Hlavním aplikačním objektem bude ten pro uchovávání záznamů jednotlivých žádostí, konkrétně *aoRequest*. Jako další doplňující aplikační objekt bude vytvořen *aoCategory* pro uchovávání seznamu předdefinovaných kategorií.

Dalším modelem je uživatelské rozhraní, které je v CPF vytvářeno za pomocí formulářů. Ty jsou postaveny nad aplikačními objekty a slouží k přímé interakci s uživatelem - k zobrazování nebo zadávání nových dat do aplikačních objektů. V této aplikaci bude vytvořen pro vytváření a editaci jednotlivých požadavků formulář *fRequest*, pro vyhledávání *fSearchRequest* a pro editaci *fEditRequests,* všechny nad aplikačním objektem *aoRequest*, pro editaci a zobrazování seznamu kategorií v objektu *aoCategory* to bude *fCategory*, resp. *fCategoryDisplay*.

Dalším druhem modelů jsou business procesy. Business procesy slouží k provádění změn v aplikačních objektech, získávání dat z objektů a interakci s uživateli. Jako hlavní business proces bude vytvořen *bpValidation*, který bude obsahovat funkce pro účastníky workflow: emailová oznámení, aktualizaci dat v aplikačním objektu *aoRequest*, čtení dat z *aoRequest* a další. Vedle tohoto procesu budou vytvořeny další procesy, jako např. *bpGetUserInfos* nebo *bpGetISManager*, které pomocí webových služeb s přístupem do Master Dat společnosti vrací informace o aktuálně přihlášeném uživateli.

Mezi doplňující funkční celky můžeme zařadit *Business Rules* (databázové spouště), *Advanced Web Services, Schedules* (databázové události) nebo *Reports* (sestavy).

Byly vyjmenovány pouze nejdůležitější komponenty, celkem bude v aplikaci vytvořeno:

- 5 aplikačních objektů
- 14 formulářů
- 7 business procesů
- 6 business rules
- 5 advanced web services
- 2 události

Samotný vývoj aplikace probíhá z části podobně jako u popsaného postupu vývoje aplikací na platformě Force.com IDE, tedy že vývojář u aplikačních objektů, formulářů a procesů využívá předpřipravených rozhraní, které mu umožňují velkou část aplikace vyvinout bez nutnosti kódování. Ke složitějším

funkcionalitám (např. mapování webových služeb, interakci s uživatelem, pokročilý design formulářů, práci s přílohami) je nutné využít javascriptu nebo html ve skript editoru formulářů a dalších prostředků, kam je možný zápis vlastního kódu.

Jmenné konvence
Názvy formulářů začínají předsazením *f*, aplikačních objektů *ao*, business procesů *bp*, business rules *br*, vlastní webové služby *aws* (advanced web services).

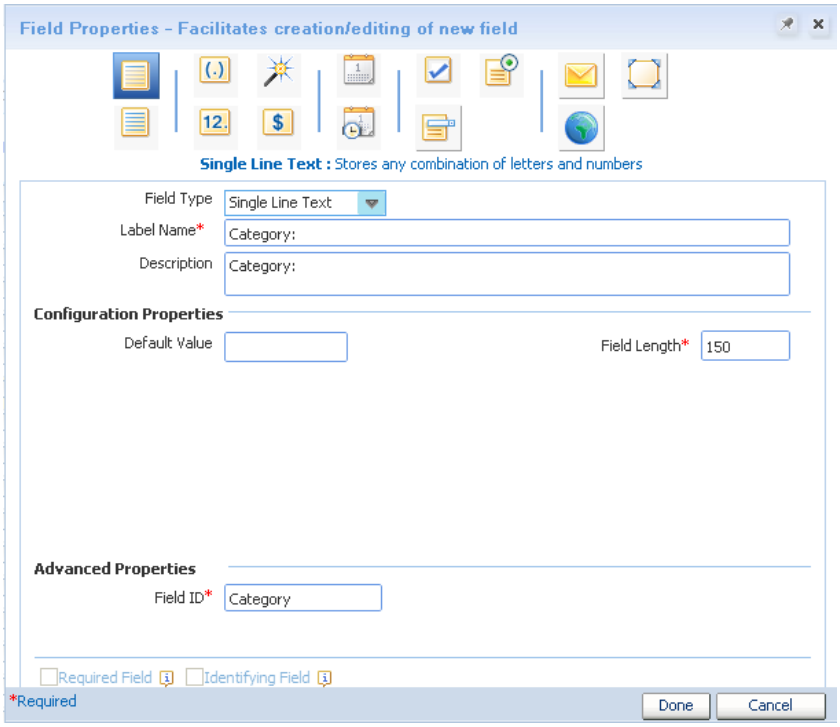
Ukládání dat

- Data jsou ukládána do aplikačních objektů, vytvoření těchto objektů je prvním krokem při vývoji nové aplikace. Aplikační objekt je relační databáze, řádky tabulky představují jednotlivá pole. Každé pole je identifikováno primárním klíčem („field id“) a dalšími atributy (size, description, label a další). Každý aplikační objekt musí mít definováno nejméně jedno pole jako identifikátor (hodnotu *isprimary* má nastavenu na "Y"). Pomocí těchto identifikátorů je možné vytvářet vazby s ostatními aplikačními objekty.

Ukázka vytvořeného pole *Category*:

```
<field id="Category"
elementtype="SINGLELINE"
nullable="Y"
isprimary="N"
objectdefault=""
datatype="VARCHAR"
size="150"
formlabel="Category:"
objectlabel="Category:"
relatedcolumn="STRINGCOLUMN6"
description="Category:"
fieldmodel="aoRequestModel"
_xgrid_rowDataId="5"/>
```

- V Cordys Process Factory je toto kódování objektů nahrazeno tabulkou, do které stačí hodnoty vepsat, což výrazně urychluje práci - viz obrázek 3.4.



Obr. 3.4 - rozhraní pro vytváření polí, pole Category

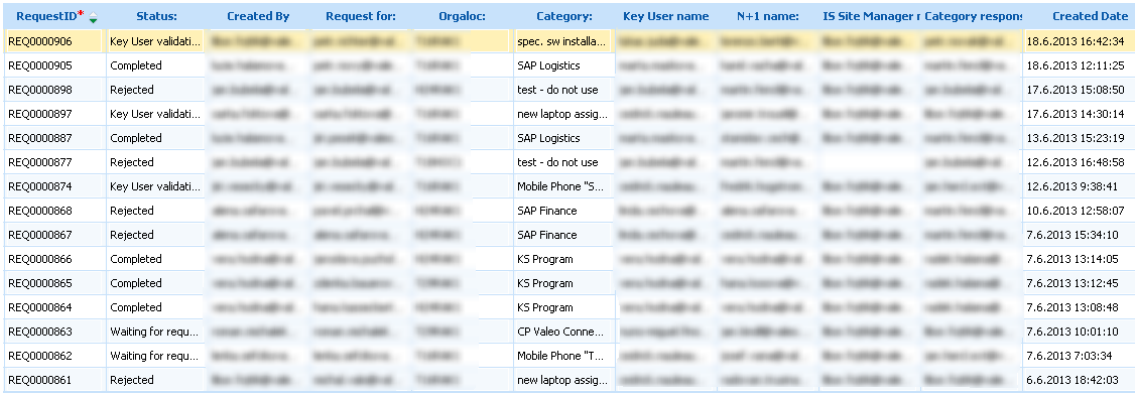
- Nad takto vytvořenými objekty poté můžeme vystavět dva typy formulářů.
- *Detail* je základní formulář uživatelského rozhraní, sloužící k zobrazování záznamů z aplikačního objektu, interakci s uživatelem, ukládání nových hodnot do objektů, apod. Jednotlivá pole můžeme volně přesouvat, skrývat, aktivovat, vytvářet k nim různé podmínky, zjednodušeně řečeno, vytváříme to, co uživatel uvidí, odkud bude informace číst a kam bude data zapisovat.
- Existuje několik předdefinovaných typů polí (*Single Line Text*, *Select Box*, *Radio Button* a další). Tato pole poté využíváme přímo ve formulářích typu *Detail*. Při prvním otevřením formuláře jsou většinou prázdná (může jim být ale přiřazena implicitní hodnota), jestliže je do konkrétního pole vepsána nějaká hodnota a formulář je uložen, pak se hodnota uloží v aplikačním objektu, odkud je při dalším otevření formuláře přečtena a v tomto poli zobrazena.
- *Grid* je tabulka, která slouží k zobrazení dat, která byla do objektu vložena pomocí formuláře typu *Detail*. *Grid* může být typu *Edit* - umožňuje záznamy přímo v tabulce editovat, *Display* - slouží pouze k náhledu a výběru hodnoty jednoho ze záznamů a *Navigation* - který slouží k otevření zvoleného záznamu (link na formulář typu *Detail*).

3.2.1 Vyhledávání

Pro vyhledávání požadavků jsou vytvořeny čtyři formuláře, základním je formulář *fSearchRequest*.

fSearchRequest

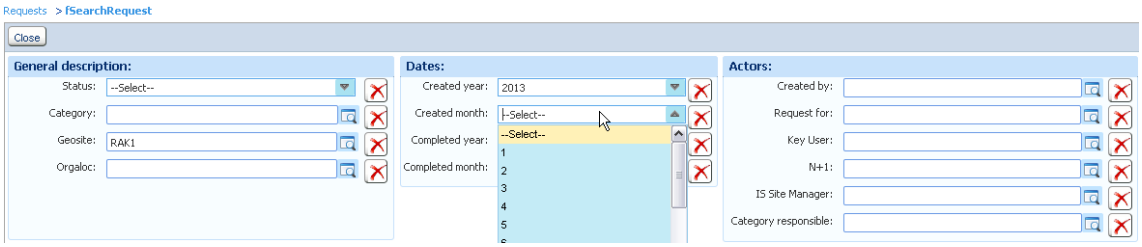
Tento formulář je vytvořen nad aplikačním objektem *aoRequest*, obsahuje tabulku (tedy typ *Grid*) se všemi dosud vytvořenými záznamy.



RequestID*	Status:	Created By	Request for:	Orgaloc:	Category:	Key User name	N+1 name:	IS Site Manager r	Category respons	Created Date
REQ0000906	Key User validati...				spec. sw installa...					18.6.2013 16:42:34
REQ0000905	Completed				SAP Logistics					18.6.2013 12:11:25
REQ0000898	Rejected				test - do not use					17.6.2013 15:08:50
REQ0000897	Key User validati...				new laptop assig...					17.6.2013 14:30:14
REQ0000887	Completed				SAP Logistics					13.6.2013 15:23:19
REQ0000877	Rejected				test - do not use					12.6.2013 16:48:58
REQ0000874	Key User validati...				Mobile Phone "S...					12.6.2013 9:38:41
REQ0000868	Rejected				SAP Finance					10.6.2013 12:58:07
REQ0000867	Rejected				SAP Finance					7.6.2013 15:34:10
REQ0000866	Completed				KS Program					7.6.2013 13:14:05
REQ0000865	Completed				KS Program					7.6.2013 13:12:45
REQ0000864	Completed				KS Program					7.6.2013 13:08:48
REQ0000863	Waiting for requ...				CP Valeo Conne...					7.6.2013 10:01:10
REQ0000862	Waiting for requ...				Mobile Phone "T...					7.6.2013 7:03:34
REQ0000861	Rejected				new laptop assig...					6.6.2013 18:42:03

Obr. 3.5 - formulář *fSearchRequest* se záznamy z produktivní platformy

Filtrovat požadavky je možné podle řady kritérií (stav žádosti, datum vytvoření, žadatel a další). Pole *Status* a všechna pole v sekci *Dates* jsou typu *Select box*, která nabízí všechny možné hodnoty, které se mohou v seznamu požadavků objevit. Pole *Category, Geosite, Orgaloc* a všechna pole v sekci *Actors* jsou náhledy do Master Dat, např. u sekce *Actors* je to náhled do formuláře *fUserGrid* s návratovou hodnotou *EmailAddress.*



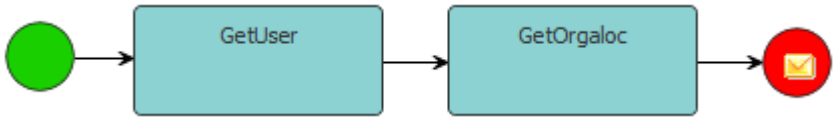
Obr. 3.6 - filtry ve formuláři *fSearchRequest*

Všechna tato pole určená k filtrování jsou porovnávána s odpovídajícími poli v objektu *aoRequest*, např. *aoRequest/fld_status Equal formelements/f_Status*. Nulové hodnoty jsou ignorovány, tj. v případě že v nějakém poli není zadána žádná hodnota, tak odpovídající filtr není aktivní.

Při otevření formuláře jsou automaticky vyplněna dvě pole: *Created year* a *Geosite*. Do pole *Created year* je automaticky vyplněn aktuální rok, starší záznamy tak nejsou implicitně zobrazovány.

```
function c_Form_Init(eventObject)
{
    var date = new Date();
    var year = date.getFullYear();
    fld_Createdyear.setValue(year);
}
```

Pro získání hodnoty *Geosite* aktuálního uživatele je potřeba při otevření formuláře spustit business proces *bpGetUserInfos*, který byl vytvořen za pomocí dvou webových služeb (*GetUser* a *GetOrgaloc*) nahlížejících do Master Dat:



Obr. 3.7 - proces bpGetUserInfos

URI	Data
/SystemVariables/CurrentUser	Emailová adresa aktuálního uživatele

Tabulka 3.1 - vstup do webové služby *GetUser*

URI	Data
/GetUserOutput/OrgalocCode	Kód organizační jednotky

Tabulka 3.2 - výstup z webové služby *GetUser*

URI	Data
/ GetUserOutput/OrgalocCode	Kód organizační jednotky

Tabulka 3.3 - vstup do webové služby *GetOrgaloc*

URI	Data
/GetUserOutput/ Geosite	Kód závodu

Tabulka 3.4 - výstup z webové služby *GetOrgaloc*

Oba tyto automatické filtry umožňují většinu záznamů při otvírání formuláře vůbec nenačíst, což má pozitivní vliv na rychlost otvírání formuláře. Filtrování je co nejvíce automatizované, tj. jakmile uživatel doplní/změní jakékoli filtrovací pole, je ihned aktualizován seznam požadavků. Příklad při změně pole *Status*:

```
function c_fld_Status_Change(eventObject)
{
	aoRequest1_navRefresh.click();
}
```

Podrobnější ukázka podmínek pro filtrování záznamů je v části *7. Příloha A - obsah CD*, soubor *RelationConditions.* Pro zrušení filtru je možné záznam manuálně smazat, nebo kliknout na odpovídající červený křížek vedle konkrétního pole, což je vlastně tlačítko, které nuluje hodnotu v odpovídajícím poli, např. opět pro pole *Status*:

```
function c_fld_Clearstatus_Click(eventObject)
{
	fld_Status.setValue("");
}
```

Formulář *fSearchRequest* je typu *Navigace*, tj. slouží k otevření formuláře (dvojklikem), na který odkazuje, v tomto případě *fRequest* (podrobněji o tomto formuláři dále). Proto, aby byl správně otevřen zvolený formulář, slouží mapování identifikačního čísla zvoleného záznamu */aoRequest/RequestID* s */forminput/RequestID.*

fSearchMyRequests, fSearchRequestsForMe

Tyto formuláře jsou také vytvořeny nad objektem *aoRequest*, obsahují tabulku s vytvořenými záznamy, jsou si velmi podobné a slouží k co nejrychlejšímu vyhledání žádostí, které buď aktuální uživatel vytvořil, nebo byly pro něj vytvořeny. Ve formuláři *fSearchMyRequests* je proměnná */SystemVariables/CurrentUser* porovnávána s hodnotou *aoRequest/CreatedBy*, ve formuláři *fSearchRequestsForMe* s hodnotou *aoRequest/RequestFor.* Oba formuláře jsou opět typu *Navigace*.

fEditRequests

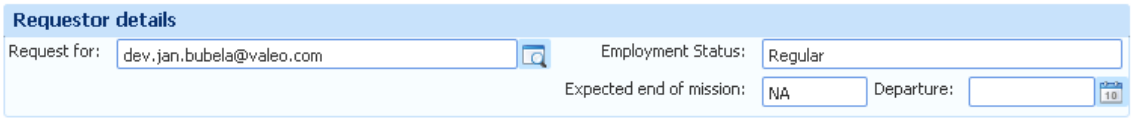
Tento formulář je vytvořen nad objektem *aoRequest*, obsahuje tabulku s vytvořenými záznamy a je přístupný pouze administrátorům. Je typu *Edit*, takže je možné editovat nebo i mazat žádosti v jakémkoli stavu.

3.2.2 Formulář elektronické žádosti

Nastavení pro žadatele

Pro vytvoření nové žádosti slouží formulář *fRequest,* typu *Detail.* V závislosti na požadavek co nejvyšší ergonomie, bude po otevření formuláře většina polí vyplněna automaticky:

Sekce *Requestor details*

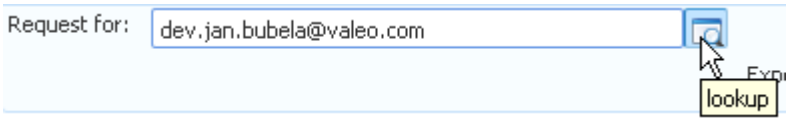


Obr. 3.8 - část *Requestor details* ve formuláři *fRequest*

Pole *Request for* je doplněno automaticky dle aktuálního uživatele pomocí *Form Rule*:

```
If String([aoRequestModel.Requestor])==""
{
	[aoRequestModel.RequestorFor] = [aoRequestModel.CreatedBy]
}
```

Pole zůstává nadále editovatelné, pro případ že chce uživatel podat žádost např. pro svého kolegu. K tomu slouží *lookup* do Master Dat.



Obr. 3.9 - *Lookup* do Master Dat

Ukázka mapování *lookup*:

```
Section Name: Requestor details
Field Name: Request for:
Lookup To: Master Data Application
Lookup Form: fUserGrid
Return Value: EmailAddress
```

Lookup je tedy v tomto případě náhled do formuláře *fUserGrid* nad objektem *aoUserGrid*, který je jednou denně synchronizován s firemním Enterprise Directory.

Na změnu pole *Request for* je navázáno spuštění již zmiňovaného business procesu *bpGetUserInfos*, který nám ale tentokrát vrací více hodnot z Master Dat:

URI	Data
/SystemVariables/CurrentUser	Emailová adresa aktuálního uživatele

Tabulka 3.5 - vstup do webové služby *GetUser*

URI	Data
/GetUserOutput/OrgalocCode	Kód organizační jednotky
/GetUserOutput/ManagerCode	Emailová adresa nadřízeného
/GetUserOutput/EmploymentStatus	"AT", "IR" (hodnoty budou vysvětleny)
/GetUserOutput/Departure	"0", "1" (hodnoty budou vysvětleny)
/GetUserOutput/EndOfMission	Datum ukončení pracovního poměru

Tabulka 3.6 - výstup z webové služby *GetUser*

URI	Data
/GetUserOutput/OrgalocCode	Kód organizační jednotky

Tabulka 3.7 - vstup do webové služby *GetOrgaloc*

URI	Data
/GetOrgalocOutput/Geosite	Kód závodu

Tabulka 3.8 - výstup z webové služby *GetOrgaloc*

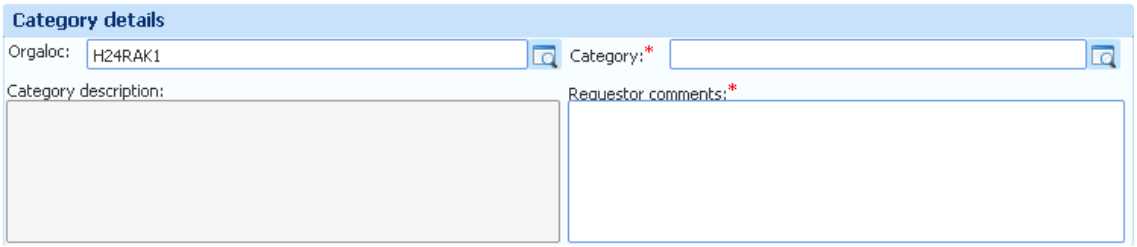
Protože *EmploymentStatus* vrací pouze zkratky a *Departure* pouze hodnotu true/false (tedy "1" a "0"), jsou na změny těchto polí navázány následující funkce, které hodnoty upraví do srozumitelné podoby:

```
function c_fld_employmentstatus_Change(eventObject)
{
    if (fld_employmentstatus.getValue() == "AT")
        fld_employmentstatus.setValue("Trainee");
    if (fld_employmentstatus.getValue() == "IR")
        fld_employmentstatus.setValue("Regular");
}


function c_fld_expectedendofmission_Change(eventObject)
{
    if (fld_expectedendofmission.getValue() == "0")
        fld_expectedendofmission.setValue("NA");
    if (fld_expectedendofmission.getValue() == "1")
        fld_expectedendofmission.setValue("Known");
}
```

Sekce *Category details*

Pole *Orgaloc* je doplněno automaticky pomocí zmiňovaného procesu *bpGetUserInfos.* Pole *Category* je *lookup* do formuláře *fCategoryDisplay* (typ formuláře *Display*).



Obr. 3.10 - sekce *Category details*

Formulář *fCategoryDisplay*
Tento formulář zobrazuje hodnoty aplikačního objektu aoCategory. Definovaná pole v tomto objetku: *Category, CategoryDescription, ResponsibleUser, Key User name a Orgaloc* (poslední pole skryto, viz dále).

Protože každý pro každý *Orgaloc* jsou definovány jiné kategorie, je třeba záznamy třídit podle uživatele v poli *Request for*, respektive podle jeho *Orgaloc*. Proto v *lookup* z formuláře *fRequest* mapujeme hodnotu v poli *Orgaloc* jako vstupní hodnotu formuláře *fCategoryDisplay*, tj.: */aoOrgaloc* mapujeme s */formOrgalocInput*, které je v tomto formuláři uživatelům skryto a slouží k filtrování výsledků. Takže při otevření formuláře *fCategoryDisplay* jsou uživatelům rovnou zobrazeny filtrované výsledky kategorií jeho organizační jednotky.

```
function c_fld_Orgaloc_Change(eventObject)
{
	aoCategory1_navRefresh.click();
}
```

Lookup z formuláře *fRequest* nevrací pouze hodnotu do pole *Category*, ale zároveň dalšími návratovými hodnotami jsou hodnoty polí *CategoryDescription* (mapováno s polem *Category description*), *ResponsibleUser* (mapováno s *Category responsible*) a *Key User name* (mapováno s *Key User name*).

Sekce *Validation details*
Tato sekce je rozdělena do několika záložek: *Key User*, *N+1*, *IS Manager* a *Category responsible*. Všechny sekce obsahují podobný set polí - jméno uživatele, jeho rozhodnutí, pole pro komentář a datum validace žádosti.

Záložka *Key User*
Tento krok není povinný, tj. pokud je pro konkrétní kategorii definován, je hodnota Key User z objektu aoCategory vrácena pomocí výše popsaného *lookup*. Jakmile hodnota není prázdná a je nějaká hodnota vrácena, je aktivována následující podmínka, která odkryje záložku Key User (která je při otevření formuláře skrytá):

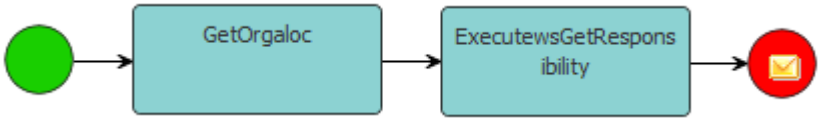
```
If (fld_keyusername.getValue() != "")
{
	fld_keyusername.unhide();
	fld_keyuserdecision.unhide();
	fld_keyusercomments.unhide();
	fld_keyuservalidationdate.unhide();
}
```

Záložka *N+1*

Jak bylo zmíněno výše, tato hodnota je návratovou hodnotou procesu *bpGetUserInfos* (buďto *ManagerCode* nebo *Responsible*). Hodnota je automaticky doplněna při otevření formuláře.

Záložka *IS Manager*

Při otevření formuláře je také spuštěn proces *bpGetISManager*, který automaticky za pomocí dvou webových služeb vrací uživatelské jméno IS manažera a doplní ho do pole *IS Site Manager name*.



Obr. 3.11 - proces *bpGetISManager*

URI	Data
/aoRequest/Orgaloc	Kód organizační jednotky

Tabulka 3.9 - vstup do webové služby *GetOrgaloc*

URI	Data
/GetOrgalocOutput/GeositeCode	Kód závodu
/GetOrgalocOutput/orgalocBGCode	Další organizační členění nutné pro korektní vyhledání uživatele v roli *IS Site Manager*

Tabulka 3.10 - výstup z webové služby *GetOrgaloc*

Nejdůležitější částí tohoto procesu je webová služba *ExecutewsGetResponsibility*, která na vstupu přijímá 6 hodnot a na základě tzv. role vrací uživatelské jméno.

URI	Data
/aoRequest/Functional network	Neměnná hodnota zadaná ze seznamu rolí při vytváření aplikace. Pole je v základním formuláři pro uživatele skryto, potřebujeme ho pouze pro tuto webovou službu.
/GetOrgalocOutput/GeositeCode	Obě hodnoty jsou návratové hodnoty z webové služby *GetOrgaloc* - viz výše.
/GetOrgalocOutput/orgalocBGCode	
/aoRequest/Levels	Podobně jako *FunctionalNetwork*, i pole *Levels* má pevně zadanou hodnotu.
/aoRequest/AnswerScope	Stejně jako *FunctionalNetwork* a *Levels*, i pole *AnswerScope* je skryté pole s pevně zadanou hodnotou, která ho definuje (v konkrétním případě hodnota *Actor*).
/aoRequest/metaRoleId	Jako *FunctionalNetwork*, *Levels* a *AnswerScope*, toto pole je skryté s pevně zadanou hodnotou, která ho definuje (v konkrétním případě 82aed5f4-f60d-c8d9-b11a-0f78fcbba4e9).

Tabulka 3.11 - vstup do webové služby *ExecutewsGetResponsibility*

URI	Data
/ExecutewsGetResponsibility Output/userID	Konkrétní emailová adresa, kterou mapujeme do formuláře *fRequest* do pole *IS Manager name.*

Tabulka 3.12 - výstup z webové služby *ExecutewsGetResponsibility*

Ve formuláři jsou tedy při zadávání nové žádosti za pomoci zmíněných webových služeb automaticky vyplněna pole nadřízeného žadatele (*N+1*) a IS manažera (*IS Site Manager name*).

Záložka *Category responsible*

Podobně jako Key User je tato hodnota vrácena pomocí lookup do *fCategoryDisplay*. Na rozdíl od Key User je tato hodnota vrácena pokaždé.

Jediné co v tuto chvíli musí uživatel vyplnit, je pole *Requestor comments*. Pak už stačí jen kliknout na tlačítko *Submit*, které spustí schvalovací proces. Na podání žádosti mu tedy stačí 4 akce - kliknutí na vytvoření nové žádosti, výběr kategorie, doplnění komentáře a odeslání ke schválení.

Všeobecné nastavení polí je ještě doplněno tím, že některá pole jsou pro žadatele neaktivní (*RequestID*, *Status*, *Category description*, všechna pole

určená pro schvalovatele, tj. záložky *Key User*, *N+1*, *IS Manager* a *Category responsible*) nebo přímo skryta (záložka *Hidden* určená pro technická pole).

Skrývání a deaktivace polí je řešeno ve skript editoru formuláře, pomocí funkcí

- *disable()* - zcela znemožní editaci pole,
- *readOnly()* - do pole není možný přímý zápis, ale je ho možné změnit pomocí *lookup,* vhodné když potřebujeme vstup v určitém formátu,
- *hide()* - k úplnému skrytí pole.

Např.

```
function c_Form_Init(eventObject)            /*while opening the form*/
{
      if (fld_status1.getValue() == "0")     /*draft status*/
      {
            fld_requestor1.readOnly = true;
            fld_orgaloc1.readOnly = true;
            TabGroup.getTab("Hidden").hide();
            fld_status1.disable();
            .........
      }
}
```

Nastavení pro schvalovatele

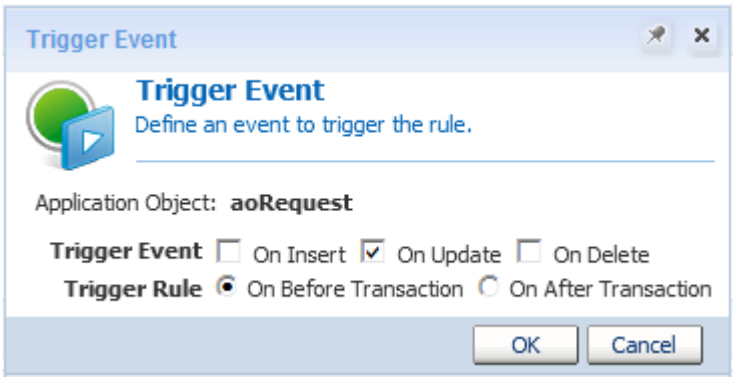
Při otevření formuláře konkrétním schvalovatelem je aktivována odpovídající záložka, tj. schvalovatel nemusí překlikávat na tu svou. Důležité je také pro něj aktivovat původně neaktivní pole (*Decision, Comments*), o což se stará dočasná proměnná *isTheCurrentUserCurrentActor*, která nabývá hodnoty "TRUE" v případě, že uživatelské jméno aktuálního uživatele je shodné s konkrétním schvalovatelem. Poté je volána funkce, která pro dosud neaktivní pole volá funkce *enable()* nebo *readOnly = false*, což pole aktivuje. Např. pro *N+1*:

```
function c_Form_Init(eventObject)            /*while opening the form*/
{
      if (fld_status1.getValue() == "3")     /*status is N+1 Validation*/
      {
            if (contains(fld_n1name1.getValue(),GlobalVariables.currentuser)
            == true)     /*check if current user is current actor*/
            {
                  fld_isthecurrentusercurrentactor.setValue("TRUE");
            }
            10TabGroup.getTab("N11TabPage").select(); /*corresponding tab is
            active*/
      }
}
```

K uložení změn provedených schvalovatelem (*Decision*, *Comments*) slouží standardně tlačítko *Complete task*, které uloží provedené změny do objektu *aoRequest* a posune workflow do dalšího kroku.

K tomu, aby nemohl schvalovatel svojí akci dokončit bez doplnění rozhodnutí (pole *Decision* typu *Select box*), slouží tzv. *Business rules*. Pro tento případ jsou rules aktivována s dvěma podmínkami:

- při pokusu o uložení změn
- jejich akce (v tomto případě *Abort transaction*) je provedena před provedením (*On Before* Transaction) samotného *Complete task*, viz obrázek 3.12:



Obr. 3.12 - definování podmínky spuštění *business rule*

Následně jsou tedy vyhodnoceny tyto podmínky (v pseudokódu):

```
If (stav == "N+1Validation")    /*podobně i pro jiné kroky schvalování*/
      If (decision == "0")      /*žádné zvolené rozhodnutí*/
            Then
                  Abort Transaction Message ("Please select an action
                  before Completing the task")
      If (decision == "2") and (comments == "")
      /*rozhodnutí je "Reject" ale nebyl doplněn komentář*/
            Then
                  Abort Transaction Message ("Please put your comments when
                  rejectin the request")
```

Druhou podmínkou je ještě ošetřeno, aby nebyl požadavek zamítnut bez doplnění komentáře zdůvodňujícího zamítnutí.

3.2.3 Schvalovací workflow

Workflow se v Cordys Process Factory tvoří za pomocí modelů business procesů, které se sestavují za pomocí různých aktivit (*Assign Task to User*, *Assign Task to External User*, *Assign Value to Fields*, *Notification to User*, *Notification to External User*, *Subprocess, Web Service*) a prvků (*Decision*,

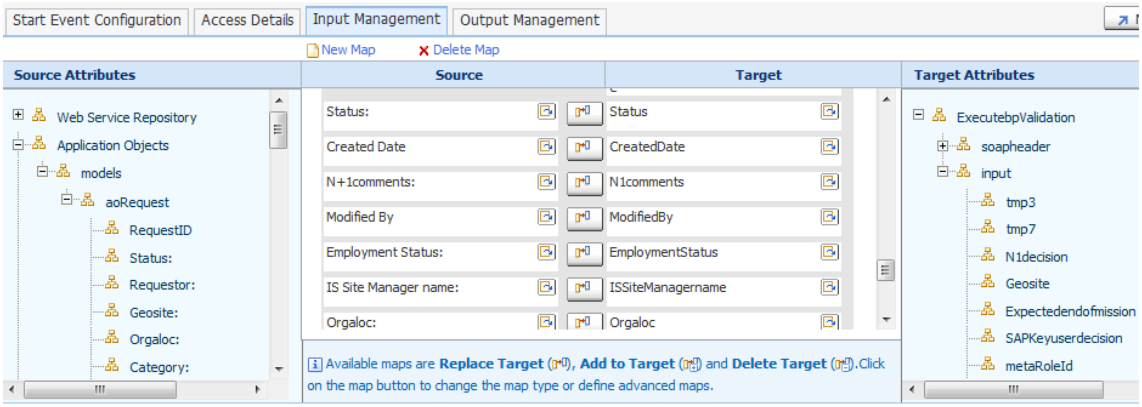
Connector, *Loop - while*, *for* nebo *until*, *Delay*, a dalších), které pomáhají k přehlednému grafickému vyjádření procesů. Business proces je vlastně jen grafické znázornění funkce, která by byla volána po stisknutí tlačítko Submit. Jednotlivé aktivity představují další funkce, které hlavní funkce volá, postupně nebo dle rozhodovacích podmínek.

Před samotným spuštěním funkce je třeba mít namapovány parametry funkce, tj. vstupní hodnoty. Cordys Process Factory opět nabízí jednoduché uživatelské rozhraní, kde můžeme výstupy z aplikačního objektu mapovat k vstupům funkce tj. např. *aoRequest/Status --> ExecutebpValidation/inputStatus.*

V pseudokódu by mohla deklarace a volání této funkce vypadat asi takto:

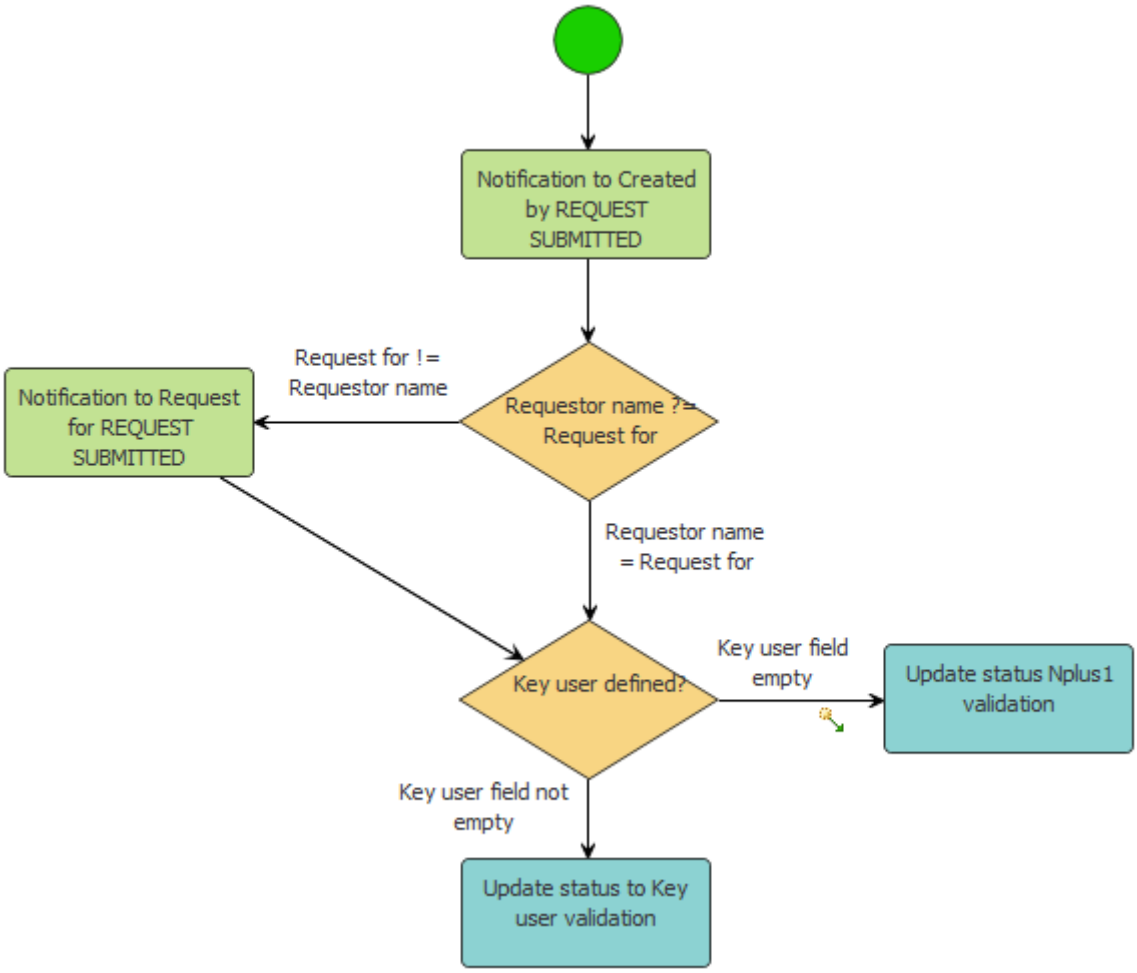
```
function bpValidation(inputID, inputStatus, inputRequestor, .....)
{
	do something;
}

function c_fld_Submit_Click(eventObject)
{
	if (check if all mandatory fields are filled)
		than
			bpValidation(ID, Status, Requestor, .....);
}
```



Obr. 3.13 - mapování parametrů funkce/procesu *bpValidation*

Ukážeme si jednoduchý příklad - část workflow - která se odehraje ihned po startu procesu (uživatel volá funkci *bpValidation*).



Obr. 3.14 - začátek procesu *bpValidation*

Aktivita *Notification to Created by* umí na základě vstupních hodnot poslat email. Ukázka definice aktivity:

```
var ActivityName = {
                   Subject: "Request submitted.",
                   Notification: "Your request has been submitted",
                   Sent to: "User id specified in a field",  /*select box*/
                   Field Name: aoRequest.Createdby,
                   From Email: "no-reply@gmail.com",
                   Display Name: "User Management database",
                   Send form link: fRequest(RequestID),
                   }
```

Do *Subject* a *Notification* můžeme vkládat jakékoli vstupní hodnoty, tj. *například Notification* může vypadat např. takto:
"Your request *[aoRequest.RequestID]* has been submitted" a výsledek:
"Your request REQ000053 has been submitted".

Uživateli v poli *Created by* přijde email dle uvedené konfigurace (předmět, zpráva) včetně odkazu na formulář *fRequest*. Aby bylo zajištěno, že se nám otevře správný záznam z aplikačního objektu *aoRequest*, musíme jako parametr uvést hodnotu *RequestID*.

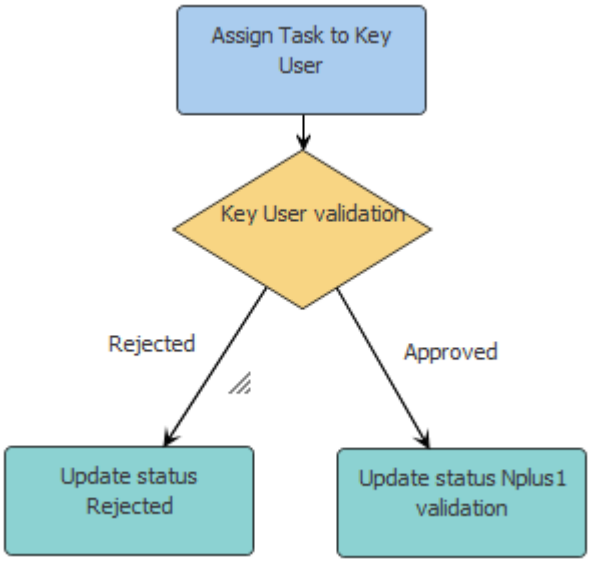
Dalším prvkem je jednoduchá podmínka, která porovnává pole *Created by* a *Request for*, jestliže jsou to různé hodnoty, aktivita *Notification to Request for REQUEST SUBMITTED* pošle ještě email uživateli v poli *Request for*.

Další podmínka zjišťuje, zda je definován *Key User* (to je nepovinný účastník workflow), podle toho je určen další krok workflow.

Aktivita *Update status* využívá webové služby *updateaoRequest*, která přistoupí k aplikačnímu objektu *aoRequest,* vyhledá konkrétní záznam (podle (*RequestID*) a provede požadovanou akci, v tomto případě změnu stavu - hodnotu "0" (*Draft*) přepíše na "2" nebo "3" *(Key User Valiation* nebo *N+1 validation*).

Následující obrázek ukazuje příklad dalších prvků workflow. Každý ze schvalovatelů má stejný set aktivity, tj. *Assign Task to User*, rozhodovací podmínku a dvě *updateAoRequest*.

Assign Task to User je velmi podobná aktivita jako *Notification to user*, tj. uživatel je informován emailem, dostane odkaz na konkrétní žádost a navíc jenom jemu bude ve formuláři *fRequest* zobrazeno tlačítko *Complete Task*, které slouží k uložení žádost a posun do dalšího stavu workflow.



Obr. 3.15 - proces *bpValidation*, přidělení úkolu schvalovateli a jeho rozhodnutí

Pro další tři schvalovatele je to velmi podobné, na základě hodnoty v poli *Decision* je buď workflow posunuto do dalšího kroku nebo je objekt *Status* změněn na *Rejected*. Jak bylo požadováno v zadání, *Category responsible* má

navíc možnost i vrátit požadavek ke *Key User*, tj. k aktivitě *Update Status to Key User validation*. Žádost tak není definitivně zamítnuta, *Key User* může upřesnit svůj komentář a workflow může pokračovat.

Na konci workflow - tj. pokud hodnota v poli *Status* přejde do stavu *Completed* nebo *Rejected*, je žadatel, tj. *Created by* (případně i *Request for*), informován o schválení nebo zamítnutí požadavku.

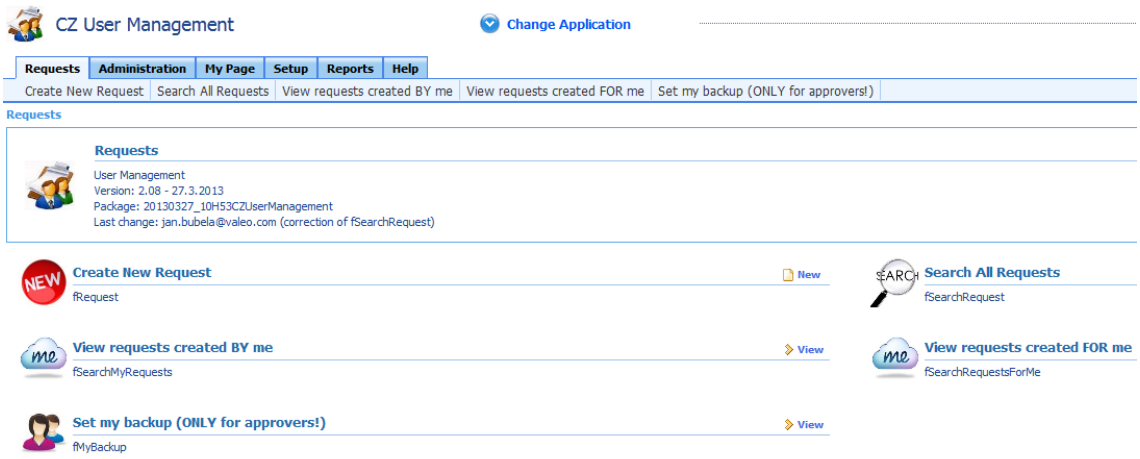
3.2.4 Ergonomie

Uživatelská ergonomie, v tomto případě minimální počet kliknutí při vytvoření žádosti, je popsána v části *3.2.2 Formulář elektronické žádosti*.
Ergonomie pro administrátory je dále popsána v části 3.3.1 *Synchronizace s Google spreadsheet*.

3.2.5 Přístupová práva

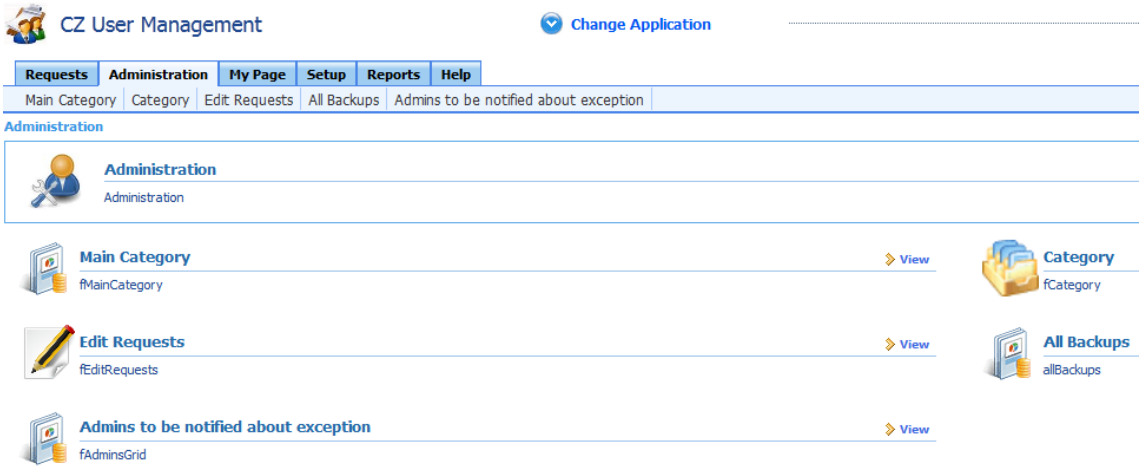
Jsou definovány dva druhy přístupových práv: *User* a *Admin*

- *User* může vytvářet, podávat a schvalovat žádosti. Může také žádosti vyhledávat (má přístup k formulářům *fSearchRequest, fSearchMyRequests, fSearchRequestsForMe*), může nastavovat svého zástupce. Jakmile není žadatelem nebo přímým účastníkem schvalovacího workflow konkrétního žádosti, nemůže žádost nijak editovat a mazat.



Obr. 3.16 - záložka *Requests* přístupná všem uživatelům

- *Admin* má přístup na záložku *Administration*, kde může prohlížet a editovat seznamy kategorií (*fMainCategory*), podkategorií (*fCategory*), editovat nebo mazat jakoukoli žádost (*fEditRequests*), prohlížet a editovat jakéhokoli zástupce (*allBackups*) a vytvářet seznam administrátorů, kteří mají být informováni o výjimkách v procesech - *fAdminsGrid* (o výjimkách a podkategoriích níže).



Obr. 3.17 - záložka *Administration* přístupná pouze administrátorům

3.3 Zkušenosti z používání aplikace, objevené chyby, rozšíření

Aplikace je v ostrém provozu používána od listopadu 2012 a to ve čtyřech různých závodech společnosti. Předtím byla jeden měsíc testována ve dvou odděleních jednoho ze závodů (cca 20 uživatelů), poté další měsíc byla používána v jednom celém závodě a od uvedeného měsíce bylo nasazení rozšířeno na všechny závody v ČR.

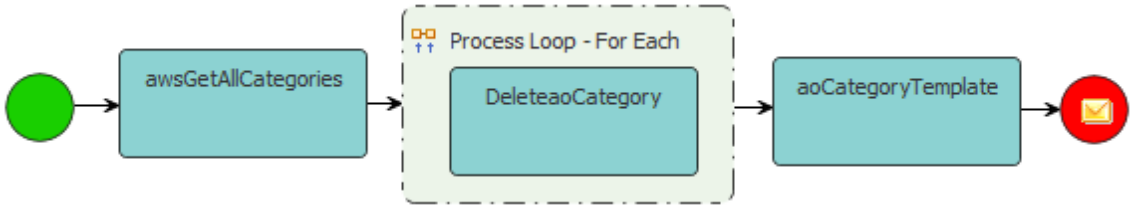
Z prvních dvou měsíců testování aplikace vzešlo několik podnětů k vylepšení i odstranění chyb:

- mezi implementovaná zlepšení můžeme zařadit synchronizaci seznamu kategorií s Google spreadsheet (část 3.3.1),
- dalším vylepšením je možnost zástupce schvalovatelů (3.3.7),
- zlepšení ergonomie pro uživatele bylo dosaženo tříděním kategorií podle hlavní kategorie (3.3.4),
- jednou z chyb byl možnost odesílání prázdné žádosti ke schválení - chyba "tupple is changed by other user" (část 3.3.3),
- dalším nedostatkem bylo nemožnost automatického sledování přerušených procesů - timeout error (3.3.5).

3.3.1 Synchronizace s Google spreadsheet

Ergonomie pro administrátory byla zlepšena pomocí synchronizace seznamů kategorií s Google spreadsheet. Protože je aplikace používána v několika závodech, vznikl poměrně rozsáhlý seznam kategorií (udržovaný v tabulce Google spreadsheet), který je neustále doplňován nebo měněn. To co se dá lehce přepsat ve spreadsheetu, je časově více náročné udělat přímo v aplikaci.

Řešení:
Byl implementován nový business proces *bpSyncAoCategories*, který změny provedené ve spreadsheetu automaticky implementuje také přímo v aplikaci. Proces se skládá že 3 hlavních aktivit viz obrázek 3.18. Nejdříve je nutné všechny záznamy v aplikačním objektu *aoRequests* smazat, aby nedocházelo k chybám při opětovném nahrávání již existujícího primárního klíče, tj. stejných hodnot v objektu *CategoryID*.



Obr. 3.18 - proces *bpSyncAoCategories*

- *awsGetAllCategories*: je nově vytvořená webová služba nad aplikačním objektem *aoCategories*, která vrací všechny záznamy v tomto aplikačním objektu,
- *DeleteaoCategory*: je standardní webová služba, která ve smyčce postupně smaže všechny záznamy v aplikačním objektu *aoCategories*,
- *WriteSpreadsheetToApplicationObject*: je webová služba typu *Google Spreadsheet Service*, na vstupu je jí předáno URL konkrétního spreadsheetu. Služba je definována pro šablonu *aoCategoryTemplate*, která slouží přímo k mapování polí ve spreadsheetu s poli v aplikačním objektu *aoCategories*. Jednotlivé sloupce ve spreadsheetu jsou mapovány s jednotlivými poli v *aoCategories*.

Aby byl synchronizační proces ještě více automatizovaný, byl vytvořen časovač, který každý den v noci (kdy je minimální vytížení aplikace), tento proces spustí. Synchronizace tak proběhne automaticky, aniž by do toho musel kdokoli manuálně zasahovat. Proces jde stále spustit kdykoli manuálně, v případě že by bylo třeba změnu ve spreadsheetu provést okamžitě.

3.3.2 Žádná hodnota v poli N+1

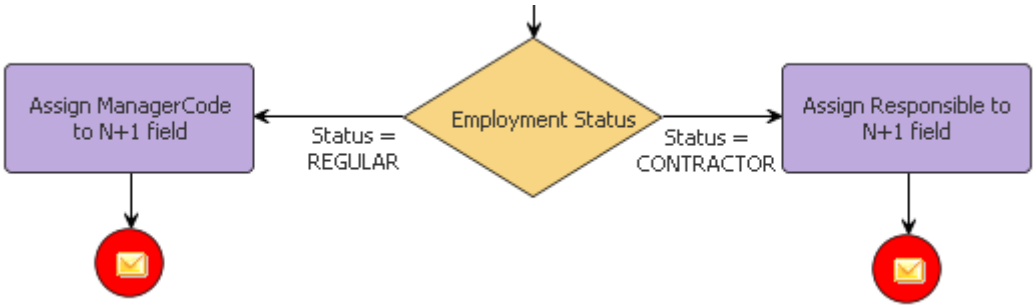
Několik schvalovacích procesů skončilo ve stavu *Aborted* z důvodu chybějící hodnoty v poli *N+1*. Při analýze bylo zjištěno, že u uživatelů ve stavu *Contractor* není tato hodnota uvedena vůbec nebo je vrácena hodnota *None*.

Řešení:
Proces *bpGetUserInfos* byl rozšířen o nové výstupy z webové služby *GetUser* a rozhodovací podmínku.

URI	Data
/GetUserOutput/EmploymentStatus	"Regular", "Trainee", "Contractor", apod...
/GetUserOutput/Responsible	Emailová adresa

Tabulka 3.13 - výstup z webové služby *GetUser*

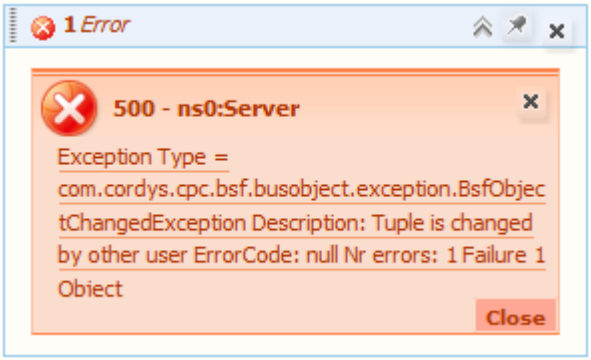
Přidaná podmínka jednoduše vyhodnotí stav uživatele a je-li to *Contractor* doplní do pole *N+1* hodnotu *Responsible* (což je nadřízený žadatele typu *Contractor*).



Obr. 3.19 - rozhodnutí dle *EmploymentStatus*

3.3.3 Neuložená data - „tuple is changed by other user error“

Představme si situaci, že dva různí uživatelé v jednu chvíli otevřou nějakou žádost. První uživatel provede nějakou změnu, kterou uloží. Druhý uživatel o této změně není informován a stále vidí původní údaje, které validuje a uloží. Data by tak byla chybně schválena změněná. Pro tyto případy CPF disponuje chybovou hláškou, která se zobrazí druhému uživateli, aby byl varován, že došlo ke změnám, které on nevidí. Druhému uživateli se tak nepodaří žádost schválit ani nic uložit.



Obr. 3.20 - chybová hláška *Tuple is changed by other user*

To je tedy správná funkce zachytávání chyby. Při testování aplikace bylo v našem helpdesku některými ze schvalovatelů otevřeno několik ticketů s tím, že

dostali ke schválení prázdnou žádost, po kliknutí na odkaz v mailu se jim sice formulář otevřel, ale bez uložených dat. Vyjádřeno procentuálně se jednalo o 1-2% celkového počtu žádostí.

Po analýze zápisů do logu u těchto případů bylo zjištěno, že data nebyla uložena, ale přitom byl spuštěn schvalovací proces. Tlačítku *Submit* byla totiž zároveň přiřazena funkce *Save&Close* a také spuštění funkce (procesu) *bpValidation*. Problém byl tedy v tom, že žadatel kliknutím na *Submit* spustil tento proces, ale formulář se nestačil korektně uložit, pravděpodobně z důvodu výkonových problémů platformy, odezvy serveru.

A protože proces *bpValidation* po několika vteřinách mění stav žádosti (z *Draft* na *Key User validation* nebo *N+1 validation*), žadatel dostane chybovou hlášku "tuple is changed..." a formulář se mu už nepodaří uložit, právě proto že proces spuštěný na pozadí už provádí v aplikačním objektu *aoCategory* změny.

Řešení:

- Implementována nová pomocná proměnná *SubmitCondition* s implicitně nastavenou hodnotou *"FALSE".* Tato proměnná nabývá hodnoty *"TRUE"* po kliknutí na tlačítko *Submit,*
- Tlačítko *Submit* nově už nespouští proces bpValidation, ale jen funkci *Save&Close*,
- Bylo implementováno nové "Business rule" *brRunBpValidation*, což je vlastně databázová spoušť, která se vyvolá po změně v aplikačním objektu *aoRequest*. Tomuto pravidlu je tedy přiřazena podmínka, za které se spustí - vložení nového záznamu, s tím že bude aktivováno až po uložení tohoto záznamu,
- Poté je zkontrolována podmínka:

```
[ aoRequest.Status ] == "0" && [ aoRequest.SubmitCondition ] == "TRUE"
```

- Jestliže je podmínka platná, spouští se proces *bpValidation*. Tedy až poté co byl korektně uložen formulář se žádostí, to nám zajišťuje proměnná *SubmitCondition.* Od doby kdy bylo *business rule brRunBpValidation* implementováno, nedošlo k žádným případům odeslání prázdných žádostí ke schválení.

3.3.4 MainCategory

S rozrůstajícím se seznamem kategorií bylo pro uživatele stále zdlouhavější vyhledat potřebnou kategorii (např. jenom pro Business Warehouse i Commerce One bylo definováno 11 kategorií, pro SAP dokonce 33). Proto bylo implementováno třídění kategorií podle "nadkategorie" (dále "hlavní kategorie" - *MainCategory*).

Řešení:

- Byl vytvořen nový aplikační objekt *aoMainCategory*, s jedním polem *MainCategory.*
- Do objektu *aoCategory* bylo také přidáno pole *MainCategory.*
- Do formuláře pro výběr kategorií uživatelem *fCategoryDisplay* byl přidán nový *lookup* do objektu *aoMainCategory* s návratovou hodnotou *MainCategory*. Podle této návratové hodnoty jsou filtrovány záznamy z objektu *aoCategory*. Když tedy žadatel chce vybrat nějakou SAP kategorii, tak jsou mu vyfiltrovány pouze SAP kategorie a nemusí se proklikávat kategoriemi, které nepotřebuje.
- Objekt *aoMainCategory* je podobně jako objekt *aoCategory* synchronizován s Google spreadsheetem se seznamem hlavních kategorií procesem *bpSyncAoMainCategory* s podobným časovačem nastaveným na noční hodiny.

3.3.5 Přerušené processy - např. „timeout error“

Několik uživatelů si v průběhu prvního měsíce používání stěžovalo na to, že jejich žádost jsou dlouho dobu v neschváleném stavu a přitom schvalovatelé ani nedostali informaci o této žádosti. Po analýze bylo zjištěno, že několik schvalovacích workflow (opět cca 1-2%) bylo přerušeno ve svém běhu s výsledkem "Timeout of length 30000...". Je to pravděpodobně pojistka proti tomu, aby některá aktivita v procesu nezpůsobila zahlcení výpočetních zdrojů, v případě že by běžela v nějaké smyčce apod. déle jak 30 vteřin. V procesu *bpValidation* není ale žádná aktivita, která by mohla trvat takovou dobu, pravděpodobně jde opět o problém výpočetního výkonu platformy nebo zpomalení síťového přenosu.

Ukázka XML chybového výstupu:

```
error message: <executeWebServiceOutput
xmlns="http://schemas.cordys.com/bsf/uc/wa/1.0/bp/sendboactivity">
  <SOAP:Fault xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <faultcode
xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/">ns0:Server</faultcode>
    <faultstring>Timeout of length 30000 for a message with id 8bdc142d-650a-
11e2-f051-1eebe9165f99 expired</faultstring>
    <details />
  </SOAP:Fault>
</executeWebServiceOutput>
```

Řešení:

Ještě doplním, že proces ve stavu *Aborted* lze restartovat, to znamená že je znovu volána poslední funkční aktivita. Po tomto restartu běží už proces v pořádku. Jen výjimečně se stane to, že je stejný proces přerušen znovu, ale další restart to vyřeší. Teď už jen tedy zbývalo najít způsob jak přerušený

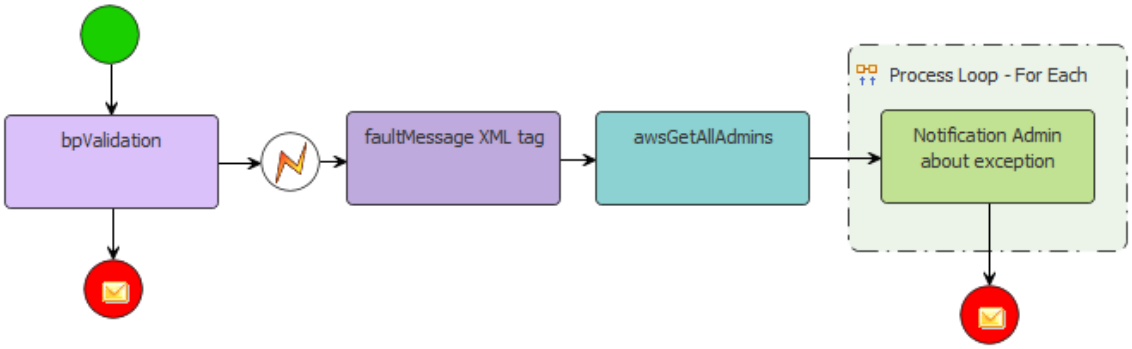
proces co nejdříve identifikovat a moci ho restartovat, aniž by došlo k nějakému většímu zpoždění ve schvalovacím workflow:

- Byl vytvořen nový aplikační objekt *aoAdmins*, kam bude možno zapsat seznam administrátorů, kterým má být přerušený proces oznámen.
- Nová webová služba *awsGetAllAdmins*, která umí z objektu *aoAdmins* číst a vracet zapsaná uživatelská jména administrátorů.
- Byl vytvořen nový proces *bpExceptionHandling*, který je spouštěn pomocí business rule *brRunBpValidation* namísto přímo schvalovacího workflow *bpValidation*. Proces *bpValidation* je vložen do *bpExceptionHandling* jako subproces, který případě svého neúspěšného běhu vyhodí výjimku.
- Aktivita *faultMessage XML tag* přiřazuje dočasné proměnné *customFaultMessage* z výstupu přerušeného procesu *bpValidation* chybovou hlášku (*faultString*).

```
customFaultMessage = string(*[local-name()='OutputMessage']/*[local-
name()='Fault']/*[local-name()='faults']/*[local-
name()='fault']/*[local-name()='faultString'])
```

- Aktivita *Notification Admin about exception* posílá administrátorům, které vrací webová služba *awsGetAllAdmins*, email s upřesňujícími informacemi, tj. kdo přerušený proces spustil, kdy by přerušen a s jakou chybou:

```
Triggered user: [systemvariables.triggereduser]
Current date: [systemvariables.currentdate]
Current time: [systemvariables.currenttime]
Fault message: [CustomMessage.customFaultMessage]
```



Obr. 3.21 - zachytávání výjimky, process *bpExceptionHandling*

Ukázka příchozího mailu:

Subject:
bpValidation process ABORTED

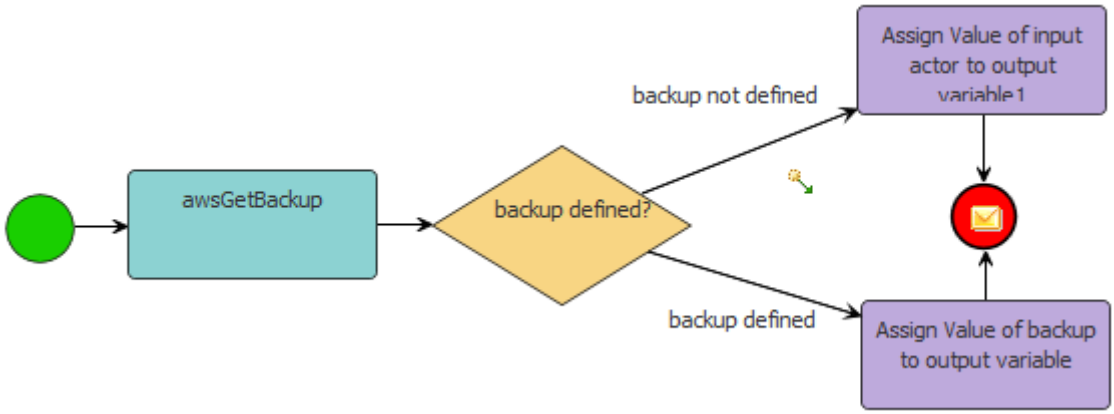
Message:
Triggered user: jan.bubela@valeo.com
Current date: 31.3.2013
Current time: 04:37:44.0
Fault message: Timeout of length 30000 for a message with id 8bdc142d-650a-11e2-f051-1eebe9165f99 expired.

3.3.6 Zástupce

Aplikace umožní členům workflow za sebe nastavit zástupce. V tomto případě bude o novém požadavku informován pouze tento zástupce a jenom on může posunout workflow do dalšího kroku.

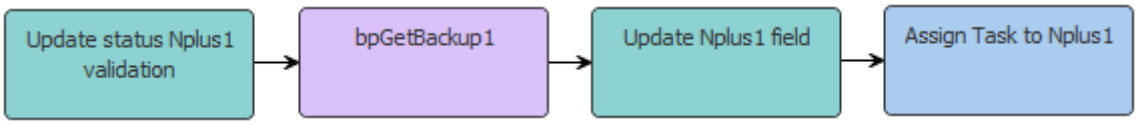
Řešení:

- Vytvoření nového aplikačního objektu *aoBackup* s dvěma poli: *Actor*, který měl vlastnost *Identifier* nastavenu na *true* (což v praxi znamená, že v tomto aplikačním objektu nemůžou existovat dva záznamy se stejným uživatelským jménem v objektu *Actor*, tj. účastník workflow nemůže za sebe definovat dva a více různých zástupců) a pole *Backup*.
- Implementace nové webové služby *awsGetBackup*, která umí z *aoBackup* číst a na svém výstupu vracet hodnotu *Backup*. Tato webová služba obsahuje podmínku, že vrací pouze hodnotu *Backup* se stejným indexem jako je její vstup, tj. následující schvalovatel - *Actor*. Což ve skutečnosti znamená, že vrací prázdný řetězec v případě, že pro tohoto schvalovatele není definován *Backup*, nebo vrací uživatelské jméno, když *Backup* definovaný je.
- Vytvoření nového business procesu *bpGetBackup*, na vstupu přijímající hodnotu následujícího schvalovatele, tj. *Actor*. Proces dále využije webovou službu *awsGetBackup*, která mu vrátí buď uživatelské jméno zástupce nebo prázdný řetězec.
- Následuje jednoduchá vyhodnocovací podmínka, která určí jaká z následujících aktivit bude použita. Aktivita *Assign Value of input actor to output variable* do výstupní pomocné proměnné uloží původní vstupní hodnotu, tedy žádný zástupce pro tohoto schvalovatele nebyl definován. Aktivita *Assign Value of backup to output variable* do pomocné výstupní proměnné uloží uživatelské jméno zástupce.



Obr. 3.22 - proces *bpGetBackup*

- Tento nově vytvořený proces *bpGetBackup* je poté vkládán do procesu *bpValidation*, když je třeba přidělit žádost některému ze schvalovatelů ke schválení a to mezi aktivitu *Update status* a *Assign Task*. Nejdříve vložíme podproces *bpGetBackup* a vrácenou hodnotou tohoto procesu aktualizujeme pomocí aktivity *Update field* (webová služba *UpdateAoRequest)* hodnotu v poli následujícího schvalovatele. Na obrázku příklad zástupce pro schvalovatele *N+1*.



Obr. 3.23 - rozšíření procesu *bpValidation* o subproces *bpGetBackup*

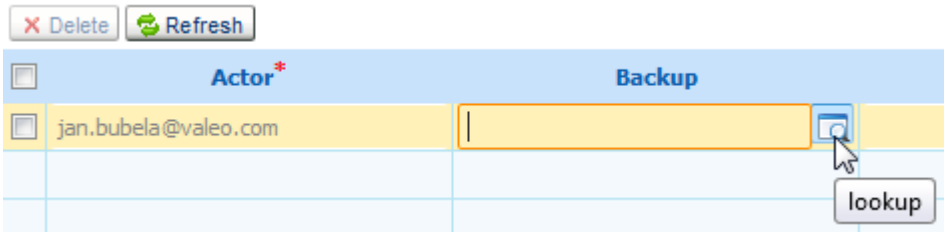
- Pro uživatele byl vytvořen formulář typu *grid fMyBackup*, kde si mohou svého zástupce nastavit.
- Při kliknutí na tlačítko *New* (pro vytvoření nového záznamu) je aktivována funkce, která v závislosti na aktuální řádce vloží jako hodnotu *Actor* uživatelské jméno aktuálního uživatele (toto pole deaktivuje, aby ho uživatel nemohl změnit), skryje tlačítko *New* (aby nemohl uživatel vkládat více záznamů) a pole *Backup* nastaví pouze ke čtení, tedy uživateli je umožněno použít pro zápis svého zástupce pouze *lookup*, aby bylo uživatelské jméno zapsáno ve správném formátu (*lookup* je do formuláře *fUserGrid* v Master Datech - seznam všech doménových uživatelů).

- Kód popisované funkce:

```
function c_aoBackup1_navInsert_Click(eventObject)
{
    aoBackup1.onrowselect = function()
    {
        var index = aoBackup1.getIndex();
        fld_actor[index].setValue(GlobalVariables.currentuser);
        aoBackup1_navInsert.hide();
        fld_actor[index].disable();
        fld_backup[index].readOnly = true;
    }
}
```

- V případě že uživatel záznam smaže, je mu znovu tlačítko *New* aktivováno.

```
function c_aoBackup1_navDelete_Click(eventObject)
{
    aoBackup1_navInsert.show();
}
```



Obr. 3.24 - formulář *fMyBackup*

Poznámka: všechny záznamy v aplikačním objektu *aoBackup* jsou při otevření tohoto formuláře filtrovány podle aktuálního uživatele, tj. *aoBackup/Actor == SystemVariables/CurrentUser.* To proto, aby uživatel viděl jen sebe a svého zástupce a nemohl smazat záznam někoho jiného.

3.4 Zhodnocení

Aplikace byla vyvíjena 1,5 měsíce z čehož jsem se vývoji mohl věnovat cca půl pracovní doby. Můžeme tedy říci, že celkově vývoj zabral 3 týdny čisté pracovní doby. Náročnější byla implementace různých rozšíření, např. možnost definování zástupce zabralo týden práce, oproti tomu v případě, že by tato funkce byla implementována rovnou při vývoji, předpokládám její náročnost na 2 dny práce. Z toho si odnáším ponaučení, že možnost zástupce budu standardně implementovat do všech nových aplikací rovnou při vývoji.

Z vývojářského hlediska bych uvítal vývoj aplikací na lokálním systému a jejich následný import na cloud. A to nejen z důvodu možné další přenositelnosti, ale navíc i rychlosti - programování ve webovém rozhraní CPF je totiž pomalé - síťová odezva, vývoj na straně serveru, při vyzkoušení provedené změny jsou výpočty prováděny na straně klienta, to vše zpomaluje práci. Oproti tomu vývoj např. na lokální instalaci Eclipse by poskytoval odezvu prakticky okamžitou.

Na druhou stranu PaaS - Cordys Process Factory - poskytuje řadu předdefinovaných funkcí a služeb, které stačí nakonfigurovat a je možné je používat, aniž by musely být pokaždé psány nanovo. Můj odhad je, že tyto předdefinované funkce urychlují práci minimálně 2x.

Má konečné volba by ale nakonec byl první způsob, tj. lokální vývoj, a to hlavně z důvodu přenositelnosti aplikací. Platforma Cordys Process Factory takový způsob neumožňuje.

Co se týká uživatelského hlediska, musím přiznat, že přechod z dosud používaného systému na Cordys Process Factory byl uživateli vnímám negativně. Hlavním negativem spatřovaným uživateli při používání aplikací na CPF je pomalá odezva, pomalá interakce uživatelského rozhraní a hlavně velmi pomalý upload souborů. Z mé zkušenosti nejvíce stížností přicházejí od uživatelů aplikací používaných technickými odděleními, kdy upload větších souborů trvá v řádech minut i více (laboratorní měření, R&D výkresy). To je samozřejmě logické, je to upload na web, ale je na zváženou zda je pro potřeby práce s velkými soubory Cloud Computing vhodný.

Co se týká aplikace "User Management", ta těmito neduhy netrpí, nejsou nahrávány a archivovány žádné velké a business kritické soubory, můžeme proto říci, že je pro PaaS vhodná.

4 Závěr

Cloud Computing je podle mě výhodnou alternativou k lokálním úložištím, ale pouze k nekritickým datům, tj. těm která nejsou ze zákona nebo z interních předpisů povinně dlouhodobě archivována nebo nejsou pro běh organizace nezbytně nutná.

Dle mého názoru Cloud Computing zcela jistě zůstane velice atraktivní pro emailové účty, sociální sítě a pro sdílení nekritických dat - uživatelských fotografií, hudby a videí, s tím, že k nim můžeme přistoupit odkudkoli. U firemních zákazníků pracující na cloudu s důležitými daty, vidím největší riziko náhlé ztráty těchto dat v důsledku pádu poskytovatele. V budoucnu může stačit pád několika středně velkých poskytovatelů, aby zájem o Cloud Computing pro firemní zákazníky poklesl.

Uživatelský pohled na aplikaci "User Management" je v současné době ve stavu, kdy se uživatelé naučili postupně s aplikací pracovat, takže se už přestávají objevovat výtky na přechod k nové platformě. Uživatelé také oceňují to, že jejich návrhy na drobná vylepšení aplikace jsou analyzována a často i implementována. Oproti začátku používání této aplikace na platformě Cordys se práce s aplikací zefektivnila, hlavně co se týká rychlosti odezvy, protože platforma je postupně neustále aktualizována (naposledy např. upgrade "Load balanceru" 5/2013).

Co se týká náročnosti údržby aplikace "User Management", tak ta je, po implementaci procesu hlídající přerušení workflow a synchronizace seznamu kategorií s Google spreadsheet, zcela minimální.

Jak již bylo zmíněno v popisu rizik, "vendor lock in" vidím jako nejslabší místo používání PaaS. Aplikace sice mohou být použitím PaaS díky frameworku poskytovanému platformou, předpřipraveným knihovnám a funkcím vyvíjeny velice rychle a pohodlně, ale i přesto bezpečnější řešení vidím v lokálním vývoji aplikace a následné nasazení na cloudové infrastruktuře, tedy IaaS (např. Amazon Web Services). Proto jako možné rozšíření této práce by mohl být vývoj podobné aplikace (jako je v této práci aplikace "User Management") v IaaS a následné srovnání náročnosti vývoje a zkušeností z používání aplikace.

5 Přehled zkratek a pojmů

PAAS: *Platform as a Service (platforma jako služba)*

IAAS: *Infrastructure as a Service (infrastruktura jako služba)*

SAAS: *Platform as a Service (software jako služba)*

CPF: *Cordys Process Factory*

ORGALOC: *organizační jednotka (každý uživatel patří pouze do jednoho Orgaloc)*

MASHUP: *webová aplikace, která používá komponenty i z jiné aplikace*.

VENDOR LOCK IN: *proprietální uzamčení - závislost na jednom poskytovateli služby s nemožnou nebo drahou migrací k jinému*

CRM: *Customer Relationship Management - řízení vztahů se zákazníky*

BPM: *Business Process Management - řízení procesů přes oddělené informační systémy*

MPLS: *Multiprotocol label switching - mechanismus k výkonému síťovému přenosu založených na značkách (labels) s krátkou cestou*

VPLS: *Virtual Private LAN service - na Ethernetu založené sítě s technologií vícebodové koncové služby*

6 Literatura

1) VELTE, T. *Cloud Computing - praktický průvodce*. Computer Press 2011, ISBN 978-80-251-3333-0
2) REESE, G. *Cloud Application Architectures*. O´Reilly 2009. ISBN 978-0-596-1536-7
3) BABCOCK, CH. *Anatomy Of The Cloud series, část 4: Platform as a service* [online]. 2009. Dostupné z: http://www.kncomputing.com/papers/CloudComputing_Platform_as_Service.pdf
4) KUTIL, I. *Google Cloud platforma* [online]. 2011.Dostupné z: http://www.lupa.cz/clanky/google-cloud-platforma-google-apps-marketplace-android-a-app-engine
5) *Creating Application with the Force.com Platform* [online]. 2011. Dostupné z: http://wiki.developerforce.com/page/Creating_Applications_with_the_Force.com_Platform
6) *Cloud Portability: Force.com vs Google App Engine vs Amazon* [online]. 2009. Dostupné z: http://www.bitsandbuzz.com/article/cloud-portability-force-com-vs-google-app-engine-vs-amazon
7) MALÝ, M. *Představení cloudových služeb: Amazon Web Services* [online]. 2011. Dostupné z: http://www.lupa.cz/clanky/predstaveni-cloudovych-sluzeb-amazon-web-services/
8) *Cordys Process Factory* [online]. Dostupné z: http://www.cordys.com/business_operations_platform
   http://www.cordys.com/process_factory
9) *Salesforce.com* [online]. Dostupné z: http://www.salesforce.com/eu/, http://www.salesforce.com/eu/platform/overview/
10) *Google App Engine* [online]. Dostupné z: http://code.google.com/intl/cs-CZ/appengine
11) *Amazon Web Service* [online]. Dostupné z: http://aws.amazon.com
12) *Cordys Process Factory - Product Brochure* [online]. Dostupné z: http://www.cordys.com/ufc/file2/cordyscms_sites/download/eba287fa469b456080bb66702fe78ee4/pu/cordys_process_factory.pdf
13) *What is Visualforce* [online]. Dostupné z: http://www.salesforce.com/us/developer/docs/pages/Content/pages_intro_what_is_it.htm

7 Přílohy

Příloha A - obsah CD

Aplikace je pro import na Cordys platformu uložena ve formátu *xml.* Jednotlivé komponenty aplikace jsou zabaleny v souboru formátu *isvp*. Po rozbalení (např. 7-Zip) dostáváme řadu *xml* souborů a jeden adresář.

Adresář **UploadedFiles**: obsahuje několik podadresářů s ikonami použitých v aplikačních formulářích, označení záložek, atd..

Soubor **937cd0f2-c0a8-6453-01d4-cb4b05a022e2.xml**: *ApplicationDirectory* obsahuje jméno a ID aplikace.

Soubor **937cd13a-c0a8-6453-01d4-cb4b09ceb5fb.xml**: *ObjectRelations* je prázdný, žádné vztahy mezi jednotlivými aplikačními objekty nejsou definovány.

Soubor **937cd13f-c0a8-6453-01d4-cb4b4eb05c0b.xml**:
AutoNumberDefinitions definuje názvy polí v konkrétních aplikačních objektech, kam jsou ukládána identifikační čísla, včetně jejich počátečních hodnot.

Soubor **937cd120-c0a8-6453-01d4-cb4b4ff08769.xml**: *Applications* obsahuje název, stav, verzi a jazyk aplikace. Také datum vytvoření a poslední změny, včetně údajů, kdo toto provedl.

Soubor **937cd131-c0a8-6453-01d4-cb4b88fabcce.xml**: *BusinessObjects* definuje jednotlivé aplikační objekty (aoBackup, aoRequest, aoCategory, aoAdmins, aoMainCategory), včetně všech polí, které obsahují.

Soubor **937cd124-c0a8-6453-01d4-cb4b6b37c094.xml**:
BusinessObjectDirectory definuje umístění aplikačních objektů, v tomto případě je to vždy tato aplikace.

Soubor **937cdefd-c0a8-6453-01d4-cb4b34a77bcb**.xml: *Filters* je prázdný. Soubor by obsahoval přednastavené filtry, která se dají používat ve formulářích sloužících k vyhledávání. Z důvody menší praktičnosti a uživatelské ergonomie je nepoužívám a vytvářím filtry vlastní.

Soubor **937cdf05-c0a8-6453-01d4-cb4b590e6ec9.xml**: *RelationConditions* obsahuje právě tyto uživatelsky vytvořené filtry ve všech vyhledávacích formulářích. Příklad jedné z podmínek z formuláře *fSearchRequest* filtrující záznamy dle zvoleného stavu:

```
<queryinfo><whereconditions><simplecondition link='AND' >
<field boid='e09be8bc-c0a8-6453-00f3-6e59c5fa34ce'
relation='aoRequest'
id='Status' match='date' >
<expression operator='EQUALSTO' >
<value type='INPUT_VARIABLE'
>:formelementsmodel.fld_Status</value></expression></field></simplecondition>
<simplecondition link='AND'...
```

Soubor **937ce6af-c0a8-6453-01d4-cb4bb8ec4585.xml**: *Aplication - WEB_SERVICE_REGISTRY* obsahuje seznam všech použitých webových služeb s informacemi o tom, kdo a kdy kterou webovou službu vytvořil, k jaké aplikaci se vztahuje, ID této aplikace, ID webové služby, atd..

Soubor **937ce6d0-c0a8-6453-01d4-cb4b945dffae.xml**: *Reports* je prázdný, obsahoval by vytvořené reporty, které slouží k exportu (do xlxs, pdf, atd..) všech záznamů dle zvolených podmínek (např. dle Orgaloc, data vytvoření, apod..)

Soubor **937ce6d4-c0a8-6453-01d4-cb4b86b953ec.xml**: *ApplicationDocTemplate* slouží k definici mapování objektů *aoCategory* a *aoMainCategory* se spreadsheetem, ve kterém je seznam kategorií zapsán.

Soubor **937ce6da-c0a8-6453-01d4-cb4b8b16b988.xml**: *Scheduler* obsahuje definice dvou časovačů pro spouštění synchronizací *aoCategory* a *aoMainCategory* se spreadsheetem se seznamem kategorií. Ukázka spouštení procesu *bpSyncAoCategories* pro synchronizaci objektu *aoCategory*.

```
<schedule
skipstartpage="false"><timezone>GMT+01:00</timezone><type>daily</type><minute
>30,</minute><hour>2,</hour><day>-1</day><month>-1</month><year>-
1</year><dayofweek>-1</dayofweek><triggercount>-
1</triggercount><templatename/>
<target><applicationid>e07a35e3-c0a8-6452-0060-30f9f7ab642c</applicationid>
<targettype>process</targettype><name>bpSyncAoCategories</name><id>8d73a5dd-
c0a8-6453-00ae-e320a902d8be</id>
<namespace>http://schemas.cordys.com/bpm/execution/1.0</namespace>
<input xPath="bpSyncAoCategories">
<CustomInput xPath="bpSyncAoCategories/CustomInput">
<CurrentUser xPath="bpSyncAoCategories/CustomInput/CurrentUser" date="false"
datefunction="false">jan.bubela@valeo.com</CurrentUser></CustomInput></input>
<inputtype>constant</inputtype><receiver/></target><accessxml/>
<autodeploy>N</autodeploy></schedule></
```

Soubor **937ce9b1-c0a8-6453-01d4-cb4bd63ad2b5.xml** FormRepository obsahuje seznam použitých formulářů. Příklad definice jednoho formuláře (bez konkrétních hodnot):

XML elementy:
<FORMID> id tohoto formuláře
<APPLICATIONID> id aplikace
<TENANTID> id platformy (tedy produktivní, testovací nebo vývojová)
<BOID> <ENTITYID> id a název aplikačního objektu nad kterým je formulář vytvořen
<VERSION> <NAME> <DESCRIPTION> verze, jméno, popis
<ICONURL> ikona formuláře
<STATUS> např. uzavřený k editaci, publikovaný
<APPDEFINITION> tato část definuje, kde můžeme v aplikaci tento formulář najít a jak ho otevřít
<CREATEDBY> <CREATEDDATE> kým a kdy vytvořeno
<LAST_CHECKEDINBY> <LAST_CHECKEDINDATE> poslední změny

Layout formuláře:
<LAYOUTXML> tato část je nejrozsáhlejší, obsahuje následující:

- celkový vzhled a rozvržení formuláře,
- použitá pole a jejich vlastnosti,
- skript editor: souhrn funkcí pro úpravu formuláře a polí, změnu jejich vlastností a hodnot, buď v závislosti na stavu záznamu nebo akce provedené uživatelem. Skript editor můžeme také použít k interakci s uživatelem,
- form rules: také slouží k úpravě polí a jejich vlastností, umí tolik co script editor, jednotlivé funkce můžeme zapisovat jako jednotlivá pravidla (rules), tím lze dosáhnout přehlednějšího zpracování bez nutnosti vše programovat manuálně. Script editor je ale nástroj mnohem mocnější než form rules,
- webové služby: seznam webových služeb, které jsou spouštěny z tohoto formuláře, včetně podmínek pro jejich spuštění a mapování polí (tj. parametrů funkcí spouštějící konkrétní webovou službu),
- lookup: definice z kterých polí můžeme nahlížet do jiných aplikačních objektů včetně návratových hodnot,
- form parameters: seznam parametrů pro spuštění formuláře s konkrétními hodnotami polí (většinou bývá jeden - pole ID, ale může být i více),
- used objects: seznam aplikačních objektů, ze kterých zobrazujeme ve formuláři konkrétní hodnoty (většinou bývá jeden, ale může být i více)
- widget repository: grafy, google gadget, atd..

Soubor **937ce63d-c0a8-6453-01d4-cb4b3a442489.xml**: *BusinessRules* je seznam všech business rules tedy funkcí, které jsou spuštěny v závislosti na změnách provedených na aplikačním objektu, nad kterým jsou postaveny (vložení nového záznamu, aktualizace stávajícího záznamu nebo jeho smazání).
Např. business rule *bpEmptyNplus1name* kontroluje korektní hodnotu v poli *N+1* před startem schvalovacího procesu.

Spouštecí podmínky:

```
<trigger> <on_insert>false</on_insert> <on_update>true</on_update>
<on_delete>false</on_delete>
```

Vyhodnocovací podmínky:

```
 <condition>[ aoRequest.Status ] == &amp;quot;0&amp;quot;</condition> <then> <if
xmlns="http://rules/commontypes"> <condition>[ aoRequest.N1name ]
==&amp;quot;&amp;quot; or [ aoRequest.N1name ]== &amp;quot;None&amp;quot;</condition>
```

Jsou-li podmínky naplněny přeruš nadcházející transakci a informuj uživatele:

```
 <then> <action> <abort xmlns="" actionID="0.21252565851246807" name="abort-
action0" id="de1b7130-c0a8-6456-01a7-c4846d518e45">Incorrect value in the
field N+1 name, please select your N+1 name</abort> </action> </then> </if>
<action xmlns="" /> <action xmlns="" /> <action xmlns="" /> <action xmlns=""
/> </then> </if> </ruledefinition> </rule></
```

Soubor **94470904-c0a8-6453-0181-81a8322f6105.xml**: *Priviledges* je seznam typů přístupových práv. Každá aplikace má standardně nastavena přístupová práva *AnonymousUser_Priviledge*, *ExternalUser_Privilegdge* a *ApplicationNameAdministrator.* K tomu je možné (a nutné) definovat další typy práv, u této aplikace je to *User* (pro běžného uživatele) a *Admin* (pro business ownera vyššími právy - editace kategorií, zástupců, jednotlivých žádostí). Přístupová práva lze definovat k jednotlivým objektům, formulářům, položkám menu nebo konkrétních polí a to v režimech *Read, Update, Insert* a *Delete*. Ke každé položce je možné definovat podmínky za jaké má být pravidlo uplatněno. Např. uživatel s právem *User* má právo mazat jednotlivé záznamy z objektu *aoRequest* pouze tehdy, je-li záznam ve stavu *Draft* a on sám vytvořil tento záznam, tj:

```
<DELETE_FILTER_XML><deletefilterxml><entity boid="e09be8bc-c0a8-6453-00f3-
6e59c5fa34ce" name="aoRequest">
<filterxml><filter id=""><scriptstring>[Status:] = '0' and [Created By] =
'Current User'</scriptstring><exactdatevalues count="0"/>
</filter></filterxml><conditionstring> [Status] = '0' and [CreatedBy] =
:CPF_USER </conditionstring></entity></deletefilterxml></DELETE_FILTER_XML>
```

Soubor **937ce364-c0a8-6453-01d4-cb4bfe010f8a.xml**: *BusinessProcesses* definuje business procesy - jejich prvky, funkce, subprocesy včetně jejich vstupů a výstupů.

Soubor **937ce939-c0a8-6453-01d4-cb4b941f7f09.xml**: *MenuData* je soubor definující základní uživatelské rozhraní aplikace. Záložky, formuláře, vlastnosti a umístění jednotlivých prvků.

Soubor **937ce941-c0a8-6453-01d4-cb4bfe214d59.xml**: *ApplicationLanguages* definuje jazyky aplikace, u této aplikace pouze Angličtina.

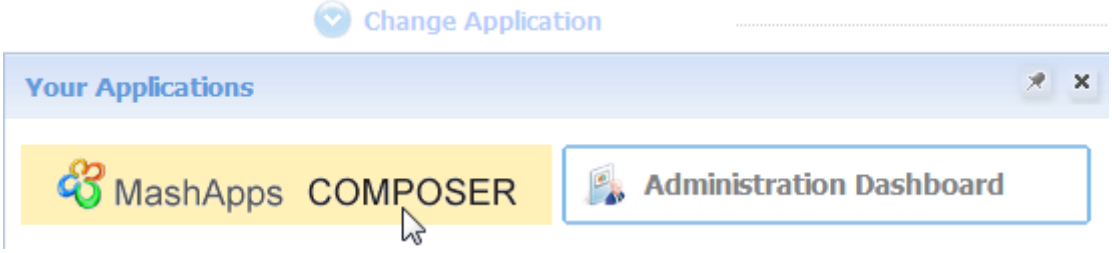
Soubor **isv.xml**: *ISVPackage* definuje, jaké nástroje budou spouštět jednotlivé komponenty *isvp* balíku:
- *ProcessFlowContent loader*: spouštění business procesů
- *RuleImporter loader*: business rules
- *menus loader*: nahrávání menu aplikace
- *busmethodsets loader*: vygenerované standardní webové služby (ke každému aplikačnímu objektu mohou být jednoduše vygenerovány webové služby jako GetObject, UpdateObject, DeleteObject a další)
- *filesystem loader:* obrázky
- *xmlstore loader*: nahrávání dat do formulářů z aplikačních objektů
- *busorganizationalroles loader*: řízení přístupových práv
- *UCImporter loader:* - seznam a cesta ke všem popsaným *xml* souborům.

Poznámka: 18.6.2013 (9 dní před termínem odevzdání této práce) jsem se dozvěděl z poznámek k vydání nové aktualizace CPF (verze 40), která je plánována na začátek července 2013, že k nově vytvořeným balíkům s aplikacemi budou ode dne aktualizace připojovány CPF certifikáty. Bez těchto podpisů nebude možné provést import aplikace z balíku *isvp* na platformu. Balík na tomto CD proto s největší pravděpodobností nepůjde po aktualizaci platformy importovat. Za toto se velmi omlouvám, neměl jsem o tom dosud žádné informace. Bude-li potřeba, tak během zpracovávání posudku této práce můžu operativně poslat aktuální balík mailem (janbubela@gmail.com) nebo CD poštou. Každopádně k obhajobě práce přinesu nové CD s certifikovaným balíkem s aplikací.

Příloha B - import aplikace na platformu

Protože se jedná o aplikaci, jejíž běh je umožněn pouze na webovém rozhraní platformy "Cordys process factory", CD neobsahuje spustitelnou verzi aplikace. Pro spuštění aplikace je proto třeba provést její import na platformu. Na stránce http://www.cordysprocessfactory.com/node/add/enterpriseuser?acctype=trial je možné si vytvořit 30 denní bezplatný účet.

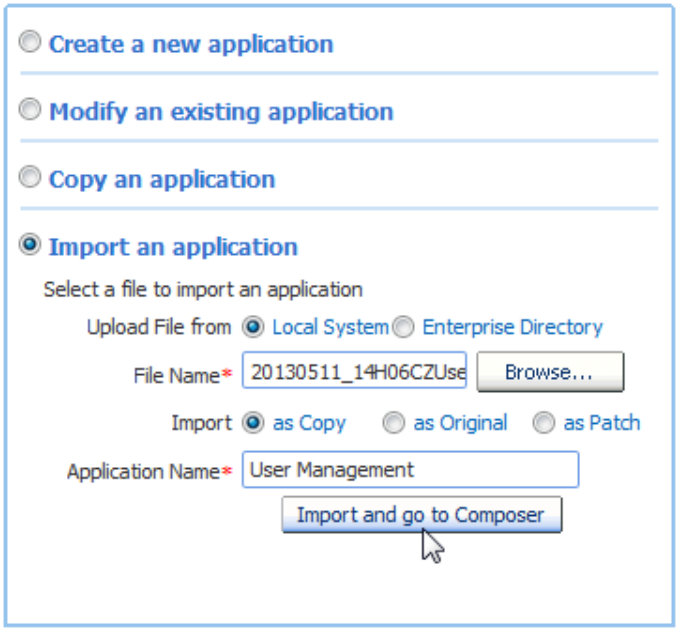
Po přihlášení zvolíme v horní části obrazovky "Change Application" a vybereme "MashApps COMPOSER".



Obr. 7.1 - menu Your Applications

Na úvodní obrazovce "MashApps Composer", je třeba provést následující:

- Import an application.
- Upload File from *Local System.*
- Vybrat požadovaný soubor.
- Zvolit Import *as Copy.*
- A zapsat požadované jméno aplikace.

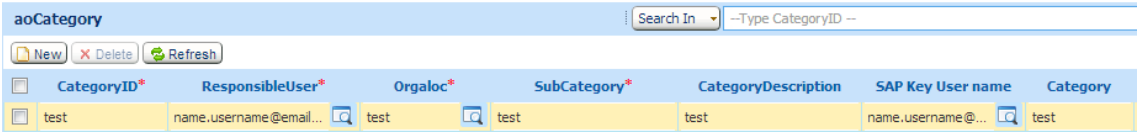


Obr. 7.2 - import aplikace

Obr. 7.3 - potvrzení o úspěšném importu aplikace

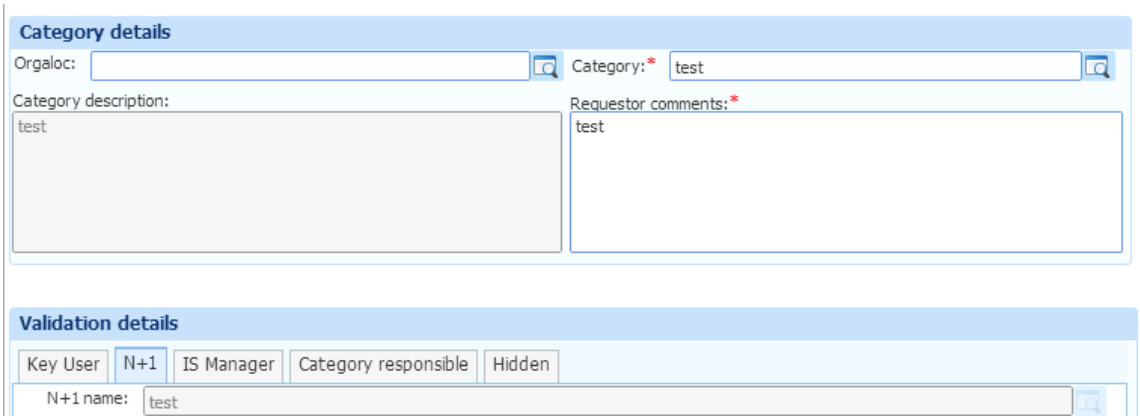
Po kliknutí na Yes (obrázek 7.3) se nám otevře COMPOSER pro změny aplikace. Chceme-li aplikaci rovnou vyzkoušet přejdeme opět do "Change application" a naší novou aplikaci vybereme.

Před vytvořením první žádosti je třeba vytvořit minimálně jednu kategorii, kterou bude moci žadatel vybrat. To se provede na záložce *Administration* kliknutím na *Category* (*fCategory*). Poté kliknutím na *New* můžeme vytvořit první záznam. Mimo povinných polí (červené hvězdičky) je nutné doplnit *LoginId* (vytvořené při registraci nového účtu) do pole "SAP Key User name" (dřívější název pro "Key User", který byl z důvodu kompatibility *fieldID* ponechán). To bude první schvalovatel. Pro uložení kliknout na *Save* nebo *Save&Close*.



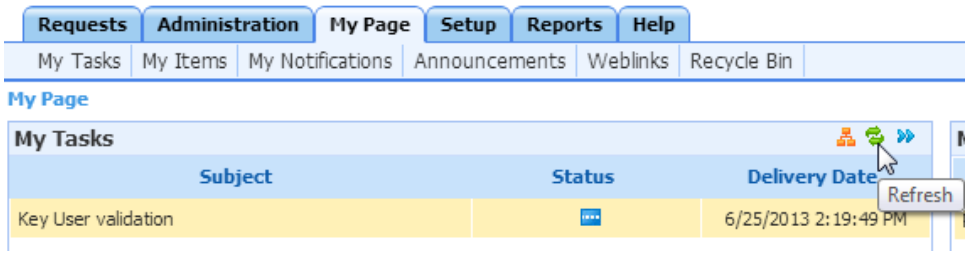
Obr. 7.4 - vytvoření kategorie

Pro vytvoření nové žádosti klikneme na záložce *Requests* na *Create New Request*. Při otevření dostaneme chybovou hlášku o neúspěšné komunikaci webových služeb s Master Daty, proto budeme muset doplnit některá pole manuálně. Vybereme naší zvolenou kategorii - lookup v poli *Category* (v případě, že jsme v předchozím kroku správně zadali "SAP Key User name", tak se nám objeví dosud zkrytá záložka "Key User") a doplníme komentář. Před odesláním je ještě třeba doplnit pole *N+1 name*. Není možné doplnit stejné *LoginId* jako je žadatel, doplníme tedy jakýkoli řetězec a klikneme na *Submit*.



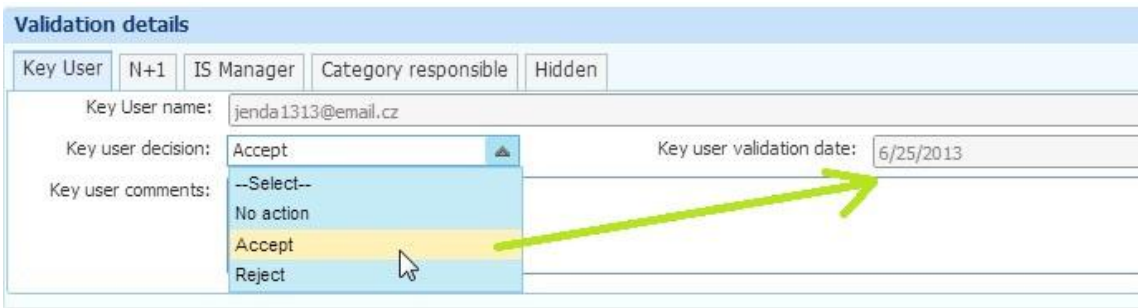
Obr. 7.5 - vyplněná žádost před podáním

Na uvedenou LoginId adresu nám přijde oznámení o nové žádosti, rychlejší cesta je pomocí záložky *My Page*, tabulka *My Tasks*, kde by se po několika vteřinách měl požadavek objevit (případně můžeme kliknout na *Refresh*).



Obr. 7.6 - tabulka My Tasks

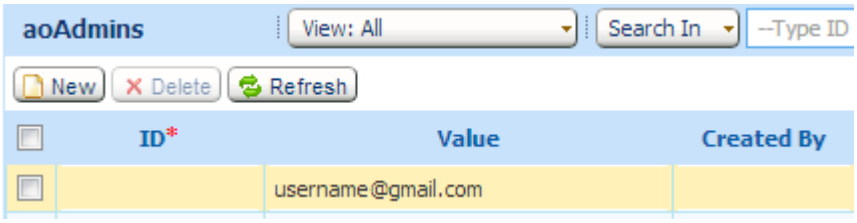
Dvojklikem na řádek s požadavkem se požadavek otevře. V dolní části formuláře fRequest doplníme rozhodnutí (pole *Validation date* se doplní automaticky) a kliknutím na *Complete Task* v horní části formuláře volbu potvrdíme.



Obr. 7.7 - doplnění rozhodnutí

Jestliže jsme zvolili rozhodnutí *Accept*, tak je nyní žádost schválena prvním schvalovatelem (*Key User*) a následně bude schvalovací workflow přerušeno, právě kvůli neplatným hodnotám v polích *N+1 Name* a *IS Site Manager Name* (chybná komunikace webových služeb s Master Daty společnosti). Pro ukázku ale myslím stačí jeden schvalovací krok, ostatní jsou podobné.

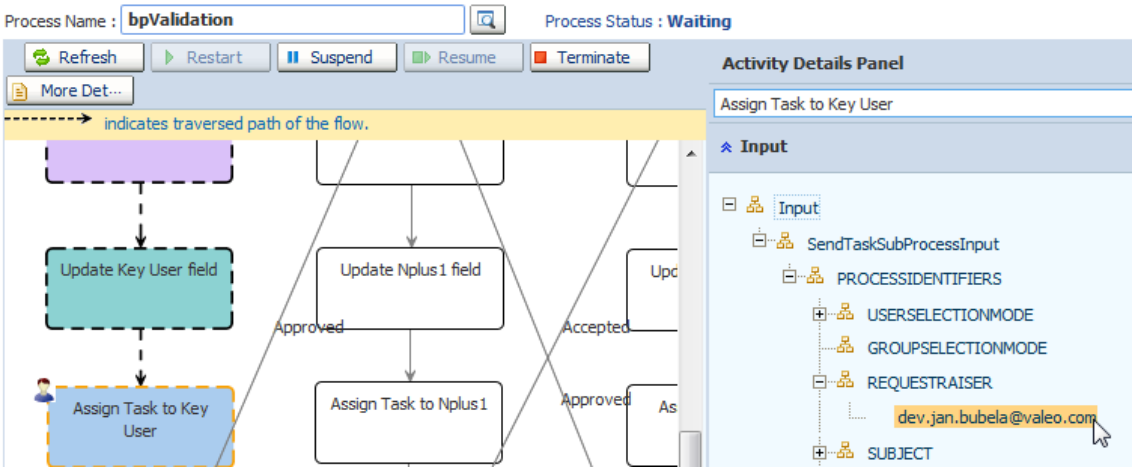
Chceme-li být automaticky informováni o takovém přerušeném procesu (viz Rozšíření část 3.3.5), tak je třeba zadat své *LoginId* do seznamu administrátorů. Na záložce *Administrations* klikneme na *Admins to be notified about exception* a po kliknutím na *New* zadáme své *LoginId* do aktivní řádky tabulky. Nezapomenout kliknout na *Save* nebo *Save&Close*.



Obr. 7.8 - tabulka se seznamem administrátorů

Pro právě provedený požadavek už oznámení administrátor nedostane, protože workflow bylo přerušeno dříve, než byl administrátor definován. O dalších přerušených požadavcích bude administrátor informován.

Jestliže se chce administrátor podívat na detaily konkrétního procesu, tak ve formuláři *fSearchRequest* vybere dvojklikem požadovaný záznam a v horní části formuláře, který se otevře, klikne na tlačítko *Process Status*. Otevře se *Process Status Viewer*, kde kliknutím na jednotlivé aktivity můžeme v pravé části obrazovky vyhledávat konkrétní hodnoty vstupů a výstupů k dané aktivitě.



Obr. 7.9 - Process Status viewer