

System pro evidenci servisních zásahů

Pavel Balcárek

Bakalářská práce
2015

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel Balcárek**
Osobní číslo: **A12170**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **kombinovaná**

Téma práce: **Systém pro evidenci servisních zásahů**
Téma anglicky: **A Service Intervention Record Information System for Maintenance and Support Workers**

Zásady pro vypracování:

1. Analyzujte funkční požadavky informačního systému pro evidenci a plánování pravidelných servisních zásahů na různých strojích (výrobní stroje nebo IT – např. kopírky, tiskárny).
2. Prostudujte dostupné systémy s otevřenou licenci, které by bylo možné pro daný účel přizpůsobit.
3. Navrhněte architekturu celého řešení.
4. Implementujte a otestujte nejdůležitější z požadovaných funkcí ve formě web aplikace.
5. Analyzujte, popř. implementujte úpravy potřebné k použitelnosti aplikace na mobilních zařízeních.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BÖHMER, Marian. Návrhové vzory v PHP. 1. vyd. V Brně: Computer Press, 2012, 320 s. ISBN 978-80-251-3338-5.**
2. **GILMORE, W. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.**
3. **LECKY-THOMPSON, Ed a Steven D NOWICKI. PHP 6: programujeme profesionálně. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2010, 718 s. Programujeme profesionálně. ISBN 978-80-251-3127-5.**
4. **MITCHELL, Lorna Jane. PHP web services : [APIs for the modern web]. First edition. O'Reilly Media, 2014, x, 104 pages. ISBN 978-1449356569.**
5. **SKLAR, David a Adam TRACHTENBERG. PHP cookbook. 3rd ed. Farnham: O'Reilly, 2014, xxi, 608 p. ISBN 15-659-2681-1.**

Vedoucí bakalářské práce:

Ing. Jan Šípek

Ústav automatizace a řídicí techniky

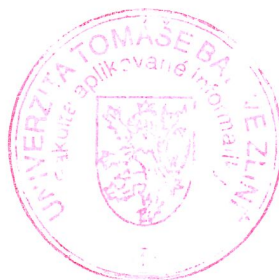
Datum zadání bakalářské práce:

6. března 2015

Termín odevzdání bakalářské práce:

22. května 2015

Ve Zlíně dne 6. března 2015



L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

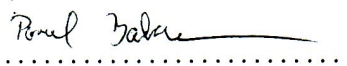
Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně



.....
podpis autora

ABSTRAKT

Tato práce se zabývá analýzou informačního systému pro firmu zaměřenou na prodej a servis kancelářské techniky. Teoretická část pojednává o informačních systémech obecně a o konceptu ERP. Praktická část je rozdělena na dvě části. V první části jsou porovnány dostupné informační systémy s otevřenou licencí. Z těchto systémů je vybrán jeden, který je přizpůsoben cílům práce. V druhé části je provedena analýza a implementace potřebných úprav a nových modulů.

Klíčová slova: informační systémy, ERP, open source, SuiteCRM

ABSTRACT

This Bachelor thesis deals with analysis of a information system for a company oriented on selling and servicing office equipment. The teoretical part describes information systems in general and the concept of ERP. The practical part is divided into two parts. The first part compares the existing information systems with an open licence. One of them is choosen to be adapted to the objectives of the work. In the second part is an analysis conducted and implementation of the necessary modification and a new modules is made.

Keywords: Information System, ERP, Open Source, SuiteCRM

Informace je název pro obsah toho, co se vymění s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním. Proces přijímání a využívání informace je procesem našeho přizpůsobování k nahodilostem vnějšího prostředí a aktivního života v tomto prostředí.

Norbert Wiener

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 SYSTÉM, INFORMACE, INFORMAČNÍ SYSTÉMY	11
1.1 DĚLENÍ INFORMAČNÍCH SYSTÉMŮ	11
1.1.1 Rozsah funkčnosti	11
1.1.2 Funkční zaměření	12
1.1.3 Technologie IS	12
1.1.4 Velikost nasazení	12
1.1.5 Způsob nasazení	13
1.2 ENTERPRISE RESOURCE PLANNING	13
1.2.1 Integrované systémy	13
1.3 ŽIVOTNÍ CYKLUS VÝVOJE IS	15
1.3.1 Zadání	16
1.3.2 Analýza	17
1.3.3 Návrh implementace	19
1.3.4 Implementace a testování	19
1.3.5 Předání do provozu, provoz a údržba	20
1.4 MODEL Y ŽIVOTNÍHO CYKLU	21
2 PŘEHLED WEBOVÝCH TECHNOLOGIÍ	23
3 SWOT ANALÝZA	24
4 ZAŘÍZENÍ TOSHIBA	25
5 SUITECRM	27
5.1 MVC	27
5.2 UKÁZKOVÝ MODUL	28
5.2.1 Model	28
5.2.2 Relationship	32
5.2.3 Mazání cenných dat	33
5.2.4 Views	34
5.2.5 Controller	35
5.3 UPGRADE-SAFE CUSTOMIZACE	35
II PRAKTICKÁ ČÁST	36
6 ANALÝZA POŽADAVKŮ	38
6.1 STÁVAJÍCÍ ŘEŠENÍ	38
6.2 ANALÝZA POŽADAVKŮ	39

6.2.1	Funkční požadavky	39
6.2.2	Nefunkční požadavky	40
6.2.3	Případy užití (UC diagram)	41
7	ANALÝZA VYBRANÝCH INFORMAČNÍCH SYSTÉMŮ	43
7.1	FUNKČNÍ ČÁST	43
7.2	NEFUNKČNÍ ČÁST, ZHODNOCENÍ	54
8	IMPLEMENTACE	56
8.1	STÁVAJÍCÍ MODULY	56
8.1.1	Uživatelé (Users)	56
8.1.2	Poznámky (Notes)	56
8.1.3	Dokumenty (Documents)	56
8.1.4	Plánovač (Scheduler)	56
8.1.5	Pošta (Emails)	56
8.1.6	Doručená pošta (Inbound Mail)	57
8.1.7	Klienti (Accounts)	57
8.1.8	Kontakty (Contacts)	57
8.2	IMPLEMENTOVANÉ MODULY	57
8.2.1	Správce souborů	59
8.2.2	Konvertory reportů	60
8.2.3	Šablony zařízení	62
8.2.4	Servisní záznamy	62
8.2.5	Správce zařízení	66
8.3	VZTAHY MEZI MODULY (ER DIAGRAM)	67
8.4	MOBILNÍ PŘÍSTUP	67
8.5	SOUČASNOST A MOŽNOSTI BUDOUCÍHO ROZŠÍŘENÍ	69
	ZÁVĚR	71
	SEZNAM POUŽITÉ LITERATURY	72
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	74
	SEZNAM OBRÁZKŮ	75
	SEZNAM TABULEK	76
	SEZNAM PŘÍLOH	77

ÚVOD

Kvalitní servis a vše pro pohodlí zákazníka. To je heslo mnoha firem bez ohledu na to, zda prodávají své výrobky či poskytují služby. Pro některé jde jen o prázdnou frázi, řádek na svých internetových stránkách. Stále více firem naštěstí bere toto heslo vážně a snaží se pro jeho splnění dělat maximum. Součástí není jen příjemná slečna na infolince, ale znamená to především mít přehled o potřebách zákazníka. Ten ale nelze získat jinak než sbíráním a následným vyhodnocováním dat. Činností, která je v dnešní informační době jednodušší než kdy jindy.

Pod pojmem *sběr dat* si lze představit cokoliv, třeba oblepování monitoru Post-it[®] štítky s telefonními čísly klientů či vedení poznámkového bloku s našimi cennými zápisky. Jestliže však chceme mluvit i o následné analýze, hovoříme téměř výlučně o digitální formě uchovávání dat. Dále si lze jen těžko představit, jak by asi ono vyhodnocování probíhalo či jak by se nám vůbec pracovalo, kdybychom používali aplikaci kalendáře od jedné firmy, adresář od druhé a místo hledali v třetí, mapové aplikaci. Mnohem snazší je mít všechny tyto informace na jednom místě, a poté se podívat do kalendáře, kde uvidíme telefon i adresu klienta se kterým máme mít schůzku. Moudrým požadavkem je také vidět jen schůzky, které jsou v tu dobu důležité - není až tak podstatné znát všechny plánované akce pro další měsíc.

Všechny tyto požadavky daly vzniknout disciplíně zvané *informační systémy*. A právě jeden takový, určený pro firmu zabývající se prodejem a servisem kancelářské techniky, by měl být výsledkem této bakalářské práce. Hlavní přínos by měl spočívat především v možnosti rychle a přehledně přistupovat k servisním záznamům konkrétního zařízení, a to nejen z počítače, ale i z chytrého mobilního telefonu přímo v terénu. Některá zařízení umí zasílat servisní záznamy i na zadaný e-mail, proto by měl systém obsáhnout i je. V současné době firma nepoužívá žádný informační systém, a je proto nutné vytvořit nový, který by odpovídal požadavkům. Protože člověk je ve své podstatě tvor líný, nemá smysl znovu objevovat kolo - existuje přešvihl informačních systémů s otevřenou licencí, které se liší v nejrůznějších aspektech, počínaje funkcionalitou a uživatelskou přívětivostí konče. Na tyto systémy se v této práci podíváme a vybereme nejvhodnější, který přizpůsobíme.

Plusů je hned několik - např. oproti tvorbě nového systému od úplného začátku, i když s použitím některého z dostupných nástrojů pro rychlý vývoj aplikace, neprogramujeme již existující části. Dalším nesporným plusem je, že mnohé systémy svým rozsahem mohou přesahovat současné požadavky, což usnadní případné budoucí rozšiřování. Nevýhodou může být neznalost použité architektury, proto je součástí i jednoduchý příklad pro seznámení se s ní.

I. TEORETICKÁ ČÁST

1 SYSTÉM, INFORMACE, INFORMAČNÍ SYSTÉMY

Hlavním tématem této bakalářské práce jsou *informační systémy*. Před samotným zavedením tohoto termínu se nejprve podíváme na jeho části jednotlivě. Slovo *system* se používá v různých souvislostech. Původně znamenal jen seskupení, sjednocení, celek. Dnes chápeme systém jako seskupení prvků doplněné o vazby mezi nimi, o jejich uspořádanost, jejich strukturu a hierarchii.[1]

Dále nás zajímá pojem *informace*. Informace se dle kybernetiky definuje jako jakákoliv zpráva, kterou lze reálně (nyní) nebo potenciálně (v budoucnosti) použít k řízení systému. Claude Elwood Shannon, považovaný za otce teorie informatiky, definoval informaci jako míru množství neurčitosti nebo nejistoty o nějakém náhodném ději, odstraněnou realizací tohoto děje.[2]

Informační systém (dále jen IS) je systém pro sběr, udržování, zpracování a poskytování informací a dat. Příkladem informačního systému může být kartotéka, telefonní seznam, účetnictví. Systém nemusí být nutně automatizovaný pomocí počítačů a může být i v papírové podobě. Informační systémy mají poskytnout správné informace pro správné lidi ve správný čas.[3]

1.1 Dělení informačních systémů

Informační systémy lze obecně dělit podle různých kritérií. Každé z těchto kritérií pohlíží na informační systém z jiného úhlu pohledu. Naším cílem v této kapitole je přiblížení těchto pohledů, díky kterému je možno námi vytvářený IS lépe kategorizovat.

1.1.1 Rozsah funkčnosti

Toto kritérium pohlíží na systém z hlediska rozsahu poskytovaných funkcí. Můžeme hovořit o:

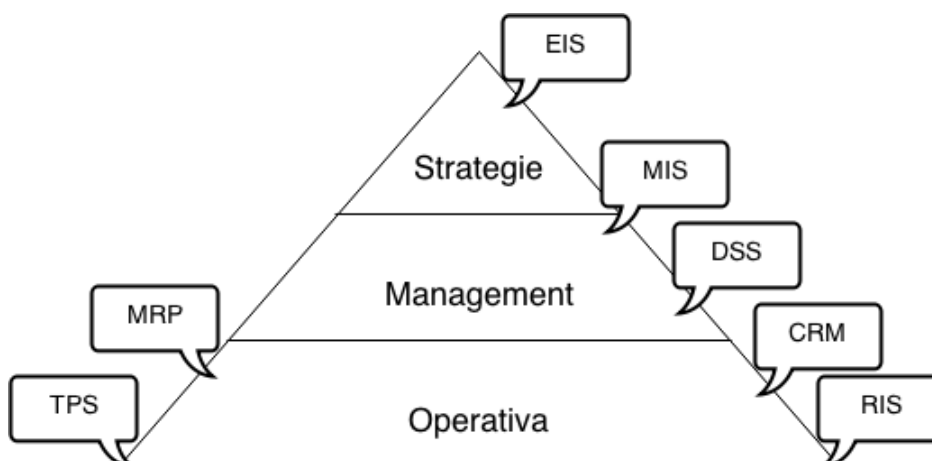
- All-in-One
Pokrývá všechny klíčové procesy a vyznačuje se vysokým stupněm integrace. Nabízí však nižší detailní funkčnost a je náročnější na upravitelnost.
- Best-of-Bread
Orientuje se jen na specifické procesy, oplývá detailnější funkcionalitou. Převážně oborové řešení.
- Lite
Jedná se o odlehčené verze systémů, které našly využití zejména u menších a středních firem.

1.1.2 Funkční zaměření

Dělení dle tohoto měřítká patří k nejdetailnějším. Systémy dělíme dle hlavního zamýšleného zaměření. Hovoříme tedy o systémech:

- manažerských (Executive IS, EIS),
- taktických (Decision Support System, DSS),
- pro vedení (Management IS, MIS),
- expertních (Knowledge Work System, KWS),
- kancelářských (Office IS, OIS),
- a operativních.

Poslední skupinu, operativní, je možno dále členit na transakční (TPS), péče o zákazníka (CRM), rezervační (RIS), konstrukční (CAM), geografické (GIS) a výrobní systémy (MRP). Rozdělení dle funkčního zaměření v hierarchii IS ve firmě je na obr. 1.1.



Obr. 1.1 Hierarchie IS ve firmě (zdroj: [6])

1.1.3 Technologie IS

Zde můžeme dále rozdělovat dle způsobu implementace klienta (tlustý, tenký), počtu vrstev aplikace (dvou a tří vrstvá architektura) a technologie použité pro vývoj a vlastní fungování systému (.NET, Java, PHP).

1.1.4 Velikost nasazení

Toto kritérium zohledňuje velikost firmy, ve kterém je IS používán. Můžeme tedy hovořit o malých (do 25ti zaměstnanců), středních (25-500 zaměstnanců) a velkých (od 500 zaměstnanců) firmách.

1.1.5 Způsob nasazení

Zjednodušeně můžeme říct, že rozhoduje o fyzickém umístění instalovaného systému. Ten může být instalován v rámci vnitřního systému nebo jako hostovaný systém u dodavatele řešení.[7]

1.2 Enterprise Resource Planning

Vyvíjený systém se řadí do kategorie systémů známé jako *Enterprise Resource Planning* (ERP). *ERP* je účinný nástroj schopný pokrýt plánování a řízení všech podnikových procesů, a to na všech úrovních (od strategické až po operativní). Umožňuje vytvářet a zpřístupňovat informace v reálném čase. Data a postupy jsou sdíleny v celé společnosti. [7]

ERP, tak jak ho známe dnes, se oproti původní verzi značně rozšířil. Na jeho počátcích, nyní označovaných jako ERP I, řešil především klíčové logistické procesy spojené s řízením výroby. Potřeba integrovat další informace vedla k rozšíření tohoto konceptu na verzi ERP II, jež je zaměřená především na zákazníky, dodavatele, životní cykly produktů, pokročilé plánování a optimalizaci. ERP III, současná verze, cílí nejen na současné zákazníky, ale také na ty potenciální. Za tímto účelem dochází k propojení se sociálními sítěmi a k analýzám velkého množství dat. Při použití ERP bez označení verze je nadále zamýšleno ERP ve verzi III. [8]

1.2.1 Integrované systémy

Dnešní ERP systémy nabízejí nepřehledné množství integrovaných systémů (příklad viz obr. 1.3). Popisovat zde všechny není účelem této práce, proto se podíváme jen na ty stěžejní, a ty jež by se svým zaměřením mohly blížit plánovanému systému.

Material Requirements Planning (MRP)

Spadá do kategorie systémů pro řízení výroby. Je určen pro *plánování výroby, nákupu materiálu* nebo *plánování zásob*. Nezohledňuje výrobní kapacitu.

Manufacturing Resource Planning (MRP II)

Rozšíření MRP o kapacitní plánování.[7]

Customer Relationship Management (CRM)

CRM bývá do češtiny překládáno jako *řízení vztahů se zákazníky*. Jak již z názvu vyplývá, v centru dění se ocitá zákazník. Jde o komplex technologií, podnikových procesů a personálních zdrojů. Jejich cílem je řízení a průběžné zajišťování oboustranně prospěšných vztahů se zákazníky firmy. Snaží se také maximálně zefektivnit každý kontakt se zákazníkem. Dotýká se oblasti podpory obchodních činností, zejména prodeje, marketingu, servisu a podpory zákazníka a zákaznických služeb. [10]

Srovnání klasického a CRM přístupu k zákazníkům je uvedeno v tabulce 1.1.

Tab. 1.1 Pojetí přístupů k zákazníkům (zdroj: [11])

transakční přístup	relační přístup (CRM)
orientace na jednorázový prodej	orientace na zákazníky a jejich udržení
diskontinuita kontaktů se zákazníky	kontinuální kontakt, dialog se zákazníkem
důraz na vlastnosti produktu	důraz na hodnoty vnímané zákazníkem
krátkodobý přístup	dlouhodobý přístup
malý důraz na zákaznický servis	vysoký důraz na zákaznický servis
malá snaha uspokojit očekávání zákazníka	vysoká snaha naplnit očekávání zákazníka
za kvalitu jsou zodpovědní pracovníci produkce	za kvalitu je zodpovědný každý zaměstnanec

Supplier Relationship Management (SRM)

Do češtiny překládané jako *řízení vztahů s dodavateli* je analogií k CRM, jen je v centru dění dodavatel.

Product Lifecycle Management (PLM)

Hlavním úkolem PLM, neboli *životního cyklu produktu*, je bezpečná správa technické dokumentace. Slouží v průběhu celého životního cyklu produktu, od vývoje po realizaci, dále servisu a opět k případné inovaci. Obvykle bývají napojeny přímo na konstrukční systémy a umožňují sledování a řízení změn nad touto dokumentací. Na dokumentaci může spolupracovat více uživatelů či projektových týmů. K dalším funkcím patří řízení projektů, zdrojů a jejich vyhodnocování či snadná simulace výrobních procesů. [5]

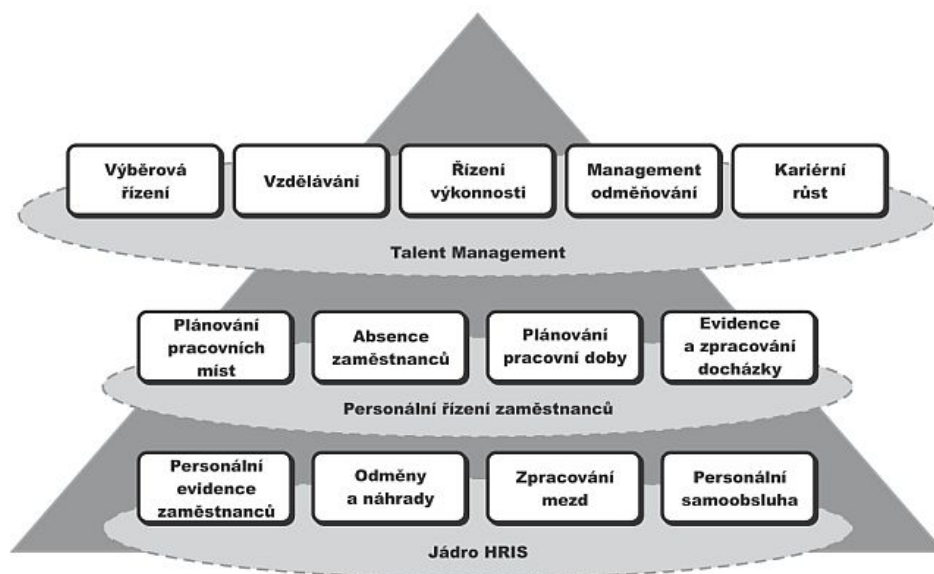
Human Resource Management (HRM)

Řízení lidských zdrojů, neboli HRM, se zaměřuje na maximalizaci výkonnosti zaměstnanců a strategických cílů zaměstnavatele. Základní IS pro řízení lidských zdrojů, využívaný především v malých a středních firmách, si vystačí s personální evidencí, zpracováním mezd, popřípadě aplikacemi určenými k řízení některých důležitých operativních činností, jako jsou výpočty a výkazy služebních cest. HRM ale také pokrývá takové činnosti jako vzdělávání, řízení výkonnosti a kariéerní růst zaměstnanců. Oblasti, které toto řízení pokrývá jsou na obr. 1.2.

Accounting

V českém jazyce známé jako *finanční účetnictví*. Poskytuje výkaz o vnějších finančních vazbách společnosti jako celku. Není určeno jen pro vlastní vedení firmy, ale i pro vnější uživatele jako jsou banky, akcionáři, finanční úřad apod. [12]

Supply Chain Management (SCM)



Obr. 1.2 HRM: Pokryté oblasti (zdroj: [4])

SCM je komplex aplikací a technologií pro *řízení dodavatelských řetězců*. Pokrývá veškeré jeho články od dodavatele surovin, přes výrobce, dopravní firmy, distribuční centra, až po obchodní firmy a konečné zákazníky. Dnes se vyskytují převážně v součinnosti s progresivními a rozvrhovacími systémy (Advanced Planning and Scheduling, APS). *SCM* ve spojení s APS umožňuje:

- vybilancování poptávky zákazníka s nabídkou na všech úrovních řetězce,
- podporu konfigurace konečného produktu zákazníkem,
- podporu řízení v dodavatelském řetězci, jako např. spolupráci partnerů při plánování dodávek,
- plánování distribuce a distribuční sítě, plánování přepravy, plánování nákupu, plánování dodávek surovin a polotovarů a plánování a řízení výroby.

[10]

1.3 Životní cyklus vývoje IS

V předchozích kapitolách jsme se dozvěděli co to jsou informační systémy a jejich rozdělení. Nyní je na čase se dozvědět něco o jejich vývoji. Stejně jako u jakékoliv jiné lidské činnosti, od které očekáváme úspěšný výsledek, nelze jen tak sednout a začít psát. I tvorba, byť jednoduššího, IS by měla projít určitými fázemi vývoje, jež nazýváme životní cyklus vývoje IS. Na jednotlivé fáze se nyní podíváme.



Obr. 1.3 Některé typické části ERP systémů (zdroj: [9])

1.3.1 Zadání

První fází, která spouští celý vývoj, je *zadání*. Obvykle je zahájena nápadem využít pro řešení nějaké evidenční úlohy počítač. Zadání, neboli specifikaci požadavků, je vhodné zpracovat v písemné formě. Ústní forma je často nejednoznačná, neúplná a nepřesná. Z těchto důvodů se jí snažíme vyhnout, neboť předělání systému po realizaci je finančně i časově náročné.

Požadavky uživatelů z věcného hlediska dělíme obvykle na *funkční* a *nefunkční*. Funkčními požadavky pak rozumíme požadavky na věcný, problémový obsah. Pro jejich co nejpřesnější formulaci je vhodné si zpracovat šablonu otázek. Šablona by měla obsahovat otázky: proč, k čemu, kdo, vstupy, výstupy, funkce a okolí systému.

Mimo věcnou náplň je nutné také uvést řadu dalších informací o okolnostech řešení. Při vývoji tuto skupinu nazýváme nefunkčními požadavky a dělíme je na tři skupiny. První skupinou jsou požadavky na priority výsledného systému zahrnující efektivitu, spolehlivost, přenositelnost do jiných softwarových prostředí a variabilitu řešení pro různé uživatele. Druhá skupina specifikuje způsoby řešení, což předepisuje použití standardů z oblasti metodiky, vývoje, dokumentace, programovací jazyk, pro-

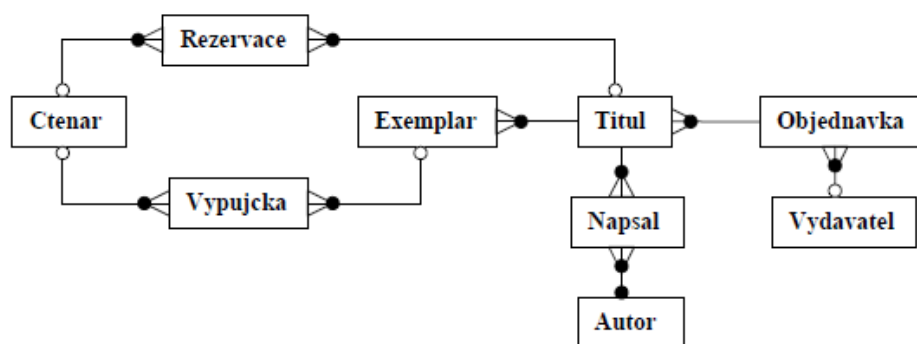
gramovací a uživatelské prostředí. Poslední skupinu tvoří vnější požadavky, která určuje např. cenové omezení, dobu řešení, harmonogram atd.

1.3.2 Analýza

V druhé fázi se provádí analýza. Úkolem analýzy je blíže specifikovat problém, který nás vede k tvorbě systému. Součástí je i analýza současného stavu řešení, ať již automatizovaného či neautomatizovaného. V tomto kroku převádíme zadání do několika modelů budoucího systému. Při tvorbě těchto modelů využíváme datovou, funkční a dynamickou (časovou) analýzu. Tyto modely jsou zmíněny v dalších odstavcích. Závěrem je proveden i návrh ovládání systému a jeho uživatelského rozhraní. Vhodné je použít i prototypového řešení systému. Výsledkem analýzy může být i nedoporučení realizace.

Datová analýza Při datové analýze jsou ze zadání vybrány potřebné evidence objektů a jejich atributů. Určí se také funkční závislosti mezi atributy a navrhne se struktura databáze. Výsledný model pak obsahuje:

- lineární zápis seznamu typů entit a jejich atributů,
- úplný grafický tvar Entity-Relationship diagramu (ERD, příklad viz obr. 1.4),
- úplné tabulky atributů (datový slovník).

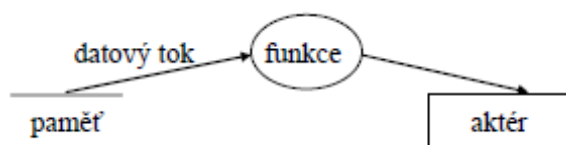


Obr. 1.4 Příklad ER diagramu při tvorbě IS pro knihovnu (zdroj: [1])

Funkční analýza Po dokončení datové analýzy je přistoupeno k funkční analýze. Ta má za úkol popsat všechny operace prováděné s daty, jinými slovy popsat všechny funkce IS. Obecně těmito akcemi jsou ukládání, modifikace, rušení dat, výpočty apod. Vytvořený funkční model vyjadřuje logický sled a podstatu transformací údajů do systému vstupujících a ze systému vystupujících. Funkční model obsahuje následující úrovně:

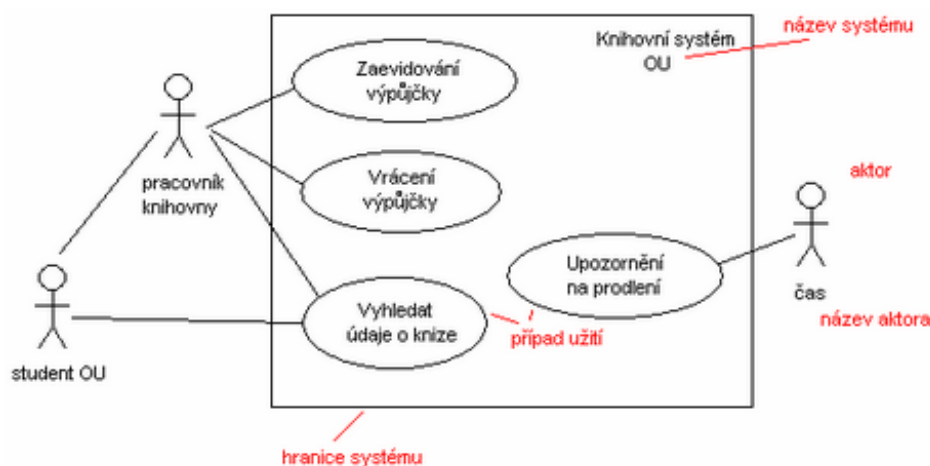
- *vnější pohled* je hrubý grafický náhled na strukturu a hierarchii funkcí systému,
- *vnitřní pohledy* jsou podrobně rozpracované algoritmy pro jednotlivé akce.

Vnější pohled bývá reprezentován pomocí *diagramu datových toků* (Date Flow Diagram, DFD). DFD zobrazuje algoritmy systému a vyjadřuje transformace dat z jedné formy do druhé. Používá k tomu čtyři základní prvky: funkce, paměť, aktér, datový tok (příklad značení na obr. 1.5).



Obr. 1.5 Diagram datových toků DFD
(zdroj: [1])

Vnější model je možno popsat i za pomoci *případů užití* (Use Case, UC). UC popisuje chování a komunikaci systému jako černou skříňku a také vymezuje a strukturalizuje okolí systému. Stejně jako u DFD, i tento model tvoří čtyři základní komponenty: účastníci, případy užití, relace a hranice systému. Příklad tohoto modelu je na obr. 1.6.

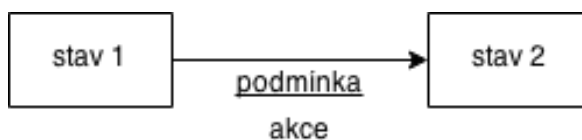


Obr. 1.6 Příklad UC modelu pro jednání knihovny (zdroj: [12])

Dynamická analýza Posledním typem analýzy IS je *dynamická analýza*. Dynamická analýza zohledňuje časové návaznosti (například *dokud student nevrátí předchozí zapůjčenou knihu, nemohu mu vydat další*). I u tohoto typu analýzy se k popisu využívá diagramu, konkrétně stavového diagramu (State Transition Diagram, STD).

STD slouží k modelování chování systému v časových návaznostech, tedy v návaznosti na čase nebo na pořadí funkcí. Modeluje chování systému v závislosti na působení

vnějších událostí nebo na základě vnitřních změn stavů. Stav definujeme jako podmnožinu hodnot atributů jednoho nebo více objektů (typů entit). Transition, neboli přechod mezi stavy, je taková změna hodnot atributů, že objekt přejde z jednoho stavu do druhého. Akcí chápeme provedení operace nad objektem. Příklad STD je na obr. 1.7.



Obr. 1.7 Příklad STD

1.3.3 Návrh implementace

Po doladění prototypu je zahájena třetí fáze, ve které se provádí návrh implementace. Zde se doladují detaily funkcí, systémové funkce a zvažuje se budoucí hardwarová (HW) a softwarová (SW) výbava. Výsledkem jsou zpřesněné datové struktury a doplněné algoritmy funkcí systému o systémové části. Dochází k rozdělení celého systému na menší celky, moduly. U větších IS se v této etapě řeší úlohy na dvou úrovních:

- systémový návrh,
- vlastní návrh implementace.

Systémový návrh vychází z výsledků analýzy a z nefunkčních požadavků zadání. Z analýzy je jasný rozsah a jeho dělení na funkční celky - subsystemy. Pod systémový návrh spadá koncepce řešení, návrh HW a SW prostředí, harmonogram, požadované prioritní vlastnosti, cena, kontext ostatních IS a použitá architektura. Můžeme využít tři základní typy architektur: centrální, file-server a klient-server. Pokud je to v daném systému vyžadováno, řeší systémový návrh také odpovídající legislativní stránku.

Vlastní návrh implementace, nazývaný též objektový návrh, doplňuje všechny modely z analýzy o implementační detaily. Často doplňuje i potřebné systémové funkce zabezpečující správné fungování systému.

1.3.4 Implementace a testování

Teprve v této etapě přichází na řadu samotné programování. Systém je implementován po dílčích částech (tak jak byl rozložen v předchozí fázi). V případě velkého IS většinou probíhá implementace paralelně několika programátory. Každý má ke zpracování své moduly. Začíná se laděním jednotlivých funkcí, přes moduly, subsystemy, až po samotný systém. Součástí výsledku této fáze je také uživatelská a programová dokumentace.

Společně s implementací se začíná provádět kontrola správnosti výsledného IS. Pod kontrolou správnosti chápeme:

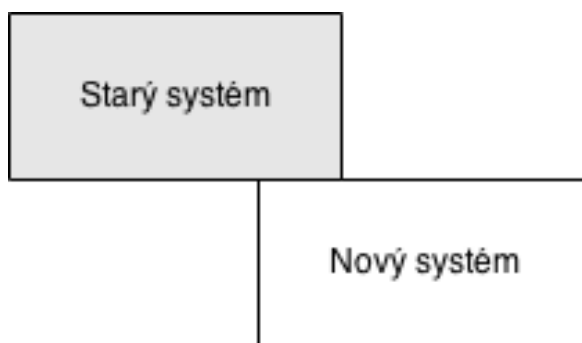
- *validaci*, že produkt odpovídá představám uživatele ve všech možných případech,
- *verifikaci*, že produkt odpovídá specifikaci ve všech možných případech,
- *testování* pomocí konečné sady příkladů, díky níž se snažíme dokázat správnost programu.

1.3.5 Předání do provozu, provoz a údržba

Předposlední fází je zavedení systému do provozu. Aby však mohl systém korektně fungovat, nestačí jej jen nainstalovat a spustit. Je třeba jej napojit na okolí systému a také připravit jeho uživatele na nový způsob práce. Příprava uživatelů je většinou provedena formou školení.[1]

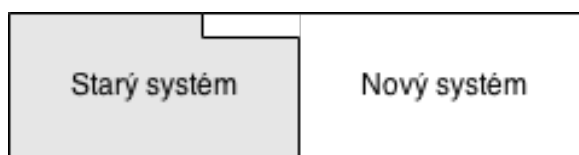
Jestliže existuje nějaký dosavadní IS, je třeba zvolit strategii zavedení nového. Každá strategie má své výhody i nevýhody. Mezi ty nejběžněji používané patří *souběžné*, *pilotní*, *postupné* a *nárazové zavádění*. V praxi je mnohdy nutno kombinovat jednotlivé postupy.

Při *souběžném zavádění* je IS (obr. 1.8) zaveden najednou na všech pracovištích. Je vhodné jej použít u jednodušších IS, které nevyžadují složitá školení a konverzi dat z předchozích IS.



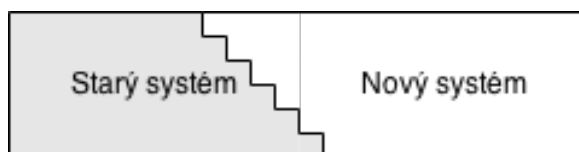
Obr. 1.8 Souběžné zavedení IS

Při *pilotním zavádění* (obr. 1.9) je IS nejprve zaveden na jednom, již připraveném pracovišti. Následuje ověřovací provoz a posléze zde probíhá zacvičování pracovníků ostatních pracovišť. Často používáno při zavádění kvalitativně odlišných IS, které vyžadují rozsáhlé testování nového IS v provozních podmínkách. Tato strategie vyžaduje postupnou transformaci z předchozích IS. V závěru pilotní fáze dochází k zavádění IS na ostatní pracoviště.

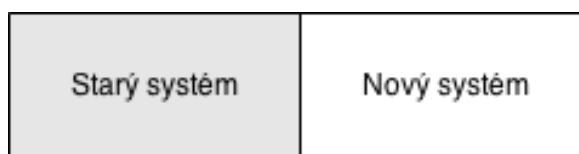


Obr. 1.9 Pilotní zavádění IS

Postupné zavádění (obr. 1.10) probíhá bez pilotní fáze. Rychlost zavádění je závislá na připravenosti jednotlivých pracovišť a složitosti IS. Vhodné u systémů, které nepotřebují provozní ověřování.



Obr. 1.10 Postupné zavádění IS



Obr. 1.11 Nárazové zavádění IS

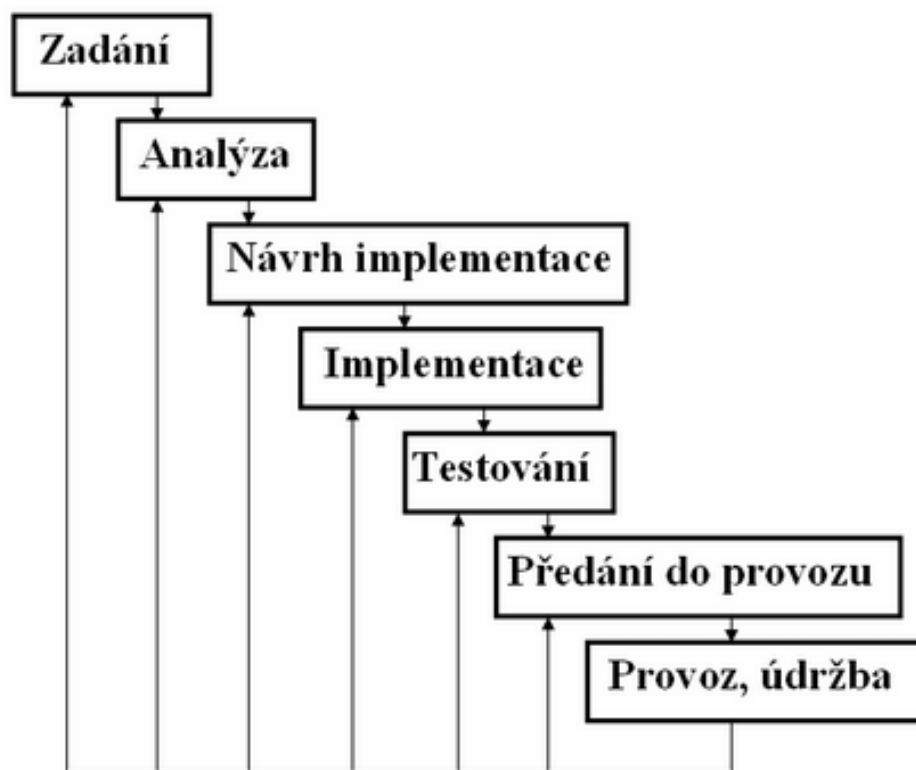
Nárazová strategie (obr. 1.11) se vyznačuje ukončením činnosti starého IS a zavedení nového. Tento postup je riskantní a používá se tam, kde souběh IS není možný.[13]

Poslední fázi životního cyklu vývoje IS je provoz, údržba a rozvoj. V tomto kroku se sleduje provoz a provádí se opravy chyb, které se nepodařilo odhalit při testování. Také se registrují připomínky, návrhy na zlepšení a doplnění systému.[1]

1.4 Modely životního cyklu

V praxi nejsou jednotlivé fáze životního cyklu řešeny v lineárním pořadí, tak jak jsou uvedeny v předchozí kapitole. Obvykle dochází k překrývání či opakování fází. V takovém případě hovoříme o modelech životního cyklu. Těchto modelů existuje několik, za všechny uveďme například *model vodopád*, *iterativní model* či *model výzkumník*. Při reálné práci dochází mnohdy ke kombinaci těchto modelů.

Model vodopád, jehož diagram vývoje je na obr. 1.12, se vyznačuje striktním oddělením jednotlivých fází. Postup do další fáze řešení nastává až po úspěšném vyřešení předchozí fáze. Realizace je možná pouze pokud zadavatel zná přesně svoje potřeby a ty se v čase nemění.



Obr. 1.12 Vodopádový model životního cyklu vývoje IS (zdroj: [1])

Iterativní model vyžaduje aktivní účast zadavatele projektu. Po provedení analýzy, návrhu a implementace jsou shromážděny požadavky uživatelů a je provedena nová iterace cyklu analýza, návrh, implementace. Po zapracování připomínek zadavatele je vytvořena finální verze. Výhodou tohoto modelu je spoluodpovědnost zadavatele za rozsah řešených funkcí. Tento model také poskytuje zadavateli lepší přehled o postupu prací na projektu.

U *modelu výzkumník* je možné zasahovat do již dříve provedených etap a v nich provádět modifikace. Na rozdíl od iterativního modelu nejsou prováděny znovu všechny fáze analýzy, návrhu a implementace. [14]

2 PŘEHLED WEBOVÝCH TECHNOLOGIÍ

Dnešní webové aplikace, potažmo IS, již zdaleka nejsou tvořeny statickými stránkami a ke svému chodu používají nejrůznější technologie a nástroje. Tato kapitola nemá za účel představit všechny dostupné technologie, ale slouží k základnímu představení těch, kterých se tato práce dotýká.

ASP.NET (Active Server Pages) je prostředí pro tvorbu dynamických webových aplikací od firmy Microsoft. Při tvorbě umožňuje využívat programovací jazyky kompatibilní s .NET Frameworkem (C#, Visual Basic .NET). Aplikace založené na této technologii jsou předkompilovány, což přináší výhody v podobě rychlosti a odhalení více chyb již při vývoji.

Pro tvorbu statických webových stránek je obvykle používán *HyperText Markup Language* (HTML). HTML je značkovací jazyk umožňující propojení stránek hypertextovými odkazy. Seznam značek (tzv. *tagů*) je pevně dán specifikací verze HTML a nelze vytvářet vlastní. V roce 2015 je aktuální verze HTML 5.0.

JavaScript je skriptovací jazyk zcela nezávislý na operačním systému. Je používán na straně klienta, nikoliv serveru jako např. PHP. Syntakticky vychází z jazyka C.

JavaServer Pages (*JSP*) je technologie vyvinutá firmou Sun Microsystems určená k tvorbě dynamických stránek. Programovacím jazykem je zde *Java*, na kterém je i založena. Výhodou *JSP* je nezávislost na platformě serveru i klienta. [18]

PHP, neboli *Hypertext preprocessor*, je skriptovací programovací jazyk určený především pro tvorbu dynamických stránek. Je založen na vkládání skriptového kódu do *HTML* stránky. Jde o interpretovaný jazyk, nikoliv kompilovaný. Syntaxe jazyka je založena na kombinaci rysů C, Javy a Perlu. Umožňuje psát objektově orientovaný kód.

Simple Network Management Protocol, zkráceně *SNMP* je protokol sloužící pro správu zařízení přes síť. Je používán pro sběr informací ze zařízení a jejich nastavení. Zařízeními mohou být například servery, tiskárny, huby, switche nebo routery. [20]

XML (eXtensible Markup Language) je značkovací jazyk, podobně jako *HTML*, ale na rozdíl od něj nejsou značky pevně dány. Význam těchto značek určuje autor. Určen pro výměnu a sdílení informací, pro dokumenty obsahující strukturované nebo semistrukturované informace.

XPath (XML Path Language) je dotazovací jazyk umožňující adresovat jednotlivé části *xml* dokumentu.

Zdroj kromě *jsp* a *SNMP*: [19]

3 SWOT ANALÝZA

Vzhledem k faktu, že pro základ budoucího IS je vybíráno z již existujících řešení, může nastat situace, kdy si je více systémů rovno z hlediska námi kladených požadavků. V takovém případě jsou systémy podrobeny dodatečné analýze. Porovnávají se *silné stránky* (strengths), *slabé stránky* (weaknesses), *příležitosti* (opportunities) a *hrozby* (threats). Dle počátečních písmen anglických slov je tato analýza označována jako *SWOT analýza*.

Základ metody spočívá v klasifikaci a ohodnocení jednotlivých faktorů, které jsou rozděleny do čtyřech výše uvedených skupin. Vzájemnou interakcí faktorů silných a slabých stránek na jedné straně vůči příležitostem a nebezpečím na straně druhé lze získat nové kvalitativní informace, které charakterizují a hodnotí úroveň jejich vzájemného střetu. Členění SWOT analýzy nalezneme v tab. 3.1.

Tab. 3.1 SWOT analýza

		Interní analýza	
		S: Silné stránky	W: Slabé stránky
Externí analýza	O: Příležitosti	Strategie SO Vývoj nových metod, které jsou vhodné pro rozvoj silných stránek společnosti (projektu).	Strategie WO Odstranění slabín pro vznik nových příležitostí.
	T: Hrozby	Strategie ST Použití silných stránek pro zamezení hrozeb.	Strategie WT Vývoj strategií, díky nimž je možné omezit hrozby, ohrožující naše slabé stránky.

V rámci SWOT analýzy je vhodné hledat vzájemné vazby mezi silnými a slabými stránkami, příležitostmi a silnými stránkami apod. Tyto vazby pak vzápětí mohou být použity pro stanovení strategie a rozvoje firmy. Volbou vhodných parametrů lze použít analýzu pro naše potřeby, výběr vhodného IS.[15]

4 ZAŘÍZENÍ TOSHIBA

Firma, pro kterou je systém vyvíjen, je dlouholetým autorizovaným prodejcem a servisním střediskem kopírovacích a faxových přístrojů *TOSHIBA*. Na komunikaci se zařízeními této značky je systém zaměřen. Komunikace je prováděna za účelem získání informací o stavu zařízení. Komunikaci dělíme na aktivní a pasivní. Aktivní komunikaci inicializuje obecně jakákoliv osoba, pasivní inicializuje samo zařízení. Komunikace se tedy dělí na:

- pasivní, zasláním e-mailové zprávy,
- aktivní, přes protokol Simple Network Management Protocol (SNMP),
- aktivní, přes webové rozhraní *TopAccess*.

E-mailová zpráva obsahuje v příloze soubor typu *text/xml* s informacemi o stavu zařízení. Tento soubor lze jednoduše procházet za pomoci *XPath* (viz kapitola 2). *SNMP* je standardizovaný protokol a proto komunikaci tímto způsobem není třeba popisovat.

The screenshot displays the Toshiba TopAccess web interface. At the top, there is a navigation bar with tabs for 'Device', 'Job Status', 'Logs', 'Registration', and 'Counter'. The 'Device' tab is selected. Below the navigation bar, there is a 'Device' section with a 'REFRESH' button and a small image of a Toshiba copier. To the right of the image is a 'Device Information' table. Below the image are two smaller tables: 'Options' and 'Toner'. To the right of the 'Options' and 'Toner' tables is a 'Paper' table.

Device Information	
Status	Ready
Name	MFC708510
Location	
Copier Model	TOSHIBA e-STUDIO3000C
Serial Number	CLF00000
MAC Address	00:00:01:6C:20:7E
Main Memory Size	2048 MB
Page Memory Size	1024 MB
Save as File & e-File Space Available	26201 MB
Fax Space Available	978 MB
Contact Information	
Phone Number	0
Message	
Alerts	•

Options	
Finisher	None
Hole Punch Unit	None
Fax	Installed

Toner	
Yellow(Y)	100%
Magenta(M)	100%
Cyan(C)	100%
Black(K)	100%

Paper				
Drawer	Size	Type	Capacity	Status
Drawer 1	A4	Plain	540	Paper Available
Drawer 2	A3	Plain	540	Near Empty
Drawer 3	A4R	Plain	540	Paper Available
Drawer 4	A4R	Plain	540	Near Empty

Obr. 4.1 Rozhraní TOSHIBA TopAccess (zdroj: [21])

TopAccess je nástroj dovolující získat informace o zařízení a tiskových úlohách přes webový prohlížeč. Obsahuje dva režimy přístupu podle druhu uživatele:

- koncový uživatel (běžný uživatel),
- správce.

Koncovému uživateli je umožněno:

- zobrazení obecných informací o zařízení, včetně jeho stavu, konfigurace příslušenství a stavu papíru,
- zobrazit a spravovat své tiskové, faxové a skenovací úlohy,
- registrovat a upravovat šablony,
- přidávat a upravovat kontakty a skupiny kontaktů v adresáři,
- zobrazit stavy počítačů (počty provedených kopií, skenování).

Správci je po přihlášení dovoleno provádět pokročilou konfiguraci zařízení (konfiguraci sítě, konfigurace e-mailových účtů).

Příklad rozhraní *TopAccess* je na obrázku 4.1. Jde o *HTML* stránku, je proto možné ji parsovat a získat z ní potřebné informace. [21]

5 SUITECRM

SuiteCRM je vývojová větev známějšího SugarCRM vyvíjená společností SalesAgility sídlící ve Velké Británii. SalesAgility se již před vznikem SuiteCRM podílela na vývoji modulů pro SugarCRM. První verzi SuiteCRM, pod označením 7.0, vydala v říjnu 2013. Verze byla založena na SugarCRM Community Edition obohacená o vlastní moduly jako byly například: produkty, pracovní postupy, smlouvy, pdf šablony, reporty, události a pokročilé vyhledávání. SalesAgility sama staví SuiteCRM proti SalesForce Professional, Microsoft Dynamics a SugarCRM Professional Edition.

Tab. 5.1 Vlastnosti SuiteCRM

Licence	GNU/AGPL3
Technologie	PHP 5.3
Databáze	MySQL 5.0x, 5.1
	MSSQL 2005, 2008
Poslední stabilní verze	11.3.2015

5.1 MVC

Již při popisu systému bylo zmíněno, že staví na produktu *SugarCRM*, a proto zachovává i jeho jmenné prostory a názvosloví, z čehož mj. vyplývá jméno použitého frameworku - Sugar framework. Jednou z vlastností tohoto frameworku je podpora *návrhových vzorů*, konkrétně vzoru známého jako *MVC* (viz obr. 5.1).

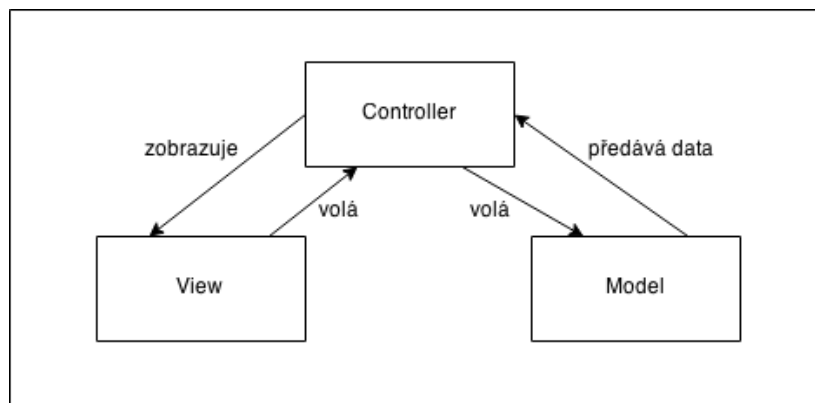
MVC, z anglického Model-View-Controller, je velmi oblíbený architektonický vzor, který se uchytil zejména při vývoji webových aplikací. Jeho základní myšlenkou je oddělení logiky od výstupu. Jak je již z názvu patrné, skládá se ze tří komponent, které jsou vysvětleny dále.

Model obsahuje logiku a vše co do ní spadá - výpočty, databázové dotazy, validace a podobně. Naopak neví vůbec nic o výstupu (o tom jak data budou vypsána a zpracována) ani odkud data v parametrech přišla. V SuiteCRM je pak tento model zastoupen třídou *SugarBean*, který koresponduje přímo s databázovými tabulkami (objektově-relační mapování).

View má na starosti zobrazení výstupu uživateli. Nejčastěji se jedná o šablonu obsahující HTML stránku s tagy některého značkovacího jazyka a zobrazovač výstupu. Zobrazovač obsahuje jen minimální množství logiky, která je pro výpis nutná. V našem IS je pro šablonování použit *Smarty Template Engine* a třída *SugarView* pro zobrazení.

Poslední komponentou je *Controller*, na něhož zbyla role prostředníka, se kterým komunikuje uživatel, model i view. Drží tedy celý systém pohromadě a komponenty

propojuje.[16]



Obr. 5.1 Diagram MVC

5.2 Ukázkový modul

Před samotným návrhem architektury (především modelů) je vhodné se blíže seznámit s frameworkem, který SuiteCRM používá a návrh případně přizpůsobit jeho specifikům či případným omezením. K tomu slouží vytvoření ukázkového modulu. V předchozí kapitole jsou popsány teoretické základy systému a jeho návrhového vzoru, které jsou zde převedeny do praktického použití. Pro správnou funkčnost je třeba ve většině případů dodržovat jak předepsanou adresářovou strukturu, tak i konkrétní názvy souborů a tříd.

5.2.1 Model

Nejprve stanovíme název modulu a objektu (modelu) v něm obsaženého. Doporučeno je použít množné číslo pro název modulu (*Soucastky*) a jednotné pro konkrétní objekt (*Soucastka*). Nyní vytvoříme složku s názvem modulu ve složce modules, tedy:

```
modules/Soucastky
```

a v ní vytvoříme soubor *Soucastka.php* s následujícím obsahem:

```
<?php
class Soucastka extends Basic
{
```

Zde popisujeme samotný model - *Soucastka*, který dědí třídu *Basic* a ta dále dědí *SugarBean*, tedy předka všech modelů. V praxi to pak znamená, že třída *Basic* již obsahuje některé předdefinované vlastnosti, jež jsou vhodné použít pro každý objekt - např. datum vzniku, autora či zda byla tato položka smazána (více o mazání v kapitole 5.2.3).

Dále jsou popsány námi definované názvy, navíc s názvem tabulky v databázi. Vlastnost *new_schema* pak značí použití nového formátu dat.

```
var $new_schema = true;
var $module_dir = 'Soucastky';
var $object_name = 'Soucastka';
var $table_name = 'soucastky';
```

Pak již následuje pouze definice vlastností charakteristických pro náš model společně s konstruktorem (z hlediska php jde o proměnné, avšak pro naše účely je nazveme vlastnostmi objektu, resp. vlastnosti - lépe vyhovují zamýšlenému objektovému chování):

```
var $soucastka_ref_cislo;
var $soucastka_lokace;
public function __construct()
{
    parent::Basic();
}
```

přičemž vlastnosti zde mají spíše kosmetický význam - usnadňují následnou práci logiky. Jejich definování je provedeno později. Nyní následuje příklad logiky, jež se stará o uložení dat - klíčem je přepsání funkce *save*:

```
public function save($check_notify = false)
{
    if (empty($this->soucastka_lokace) ) {
        if (strpos($this->soucastka_ref_cislo,'7-') !== FALSE )
            $this->soucastka_lokace = 'Sklad A';
        elseif (strpos($this->soucastka_ref_cislo,'8-') !== FALSE )
            $this->soucastka_lokace = 'Sklad B';
    }
    return parent::save($check_notify);
}
```

Obsah této funkce není třeba nikterak zvlášť popisovat - zajímavý je až poslední řádek, tedy *parent::save(\$check_notify)*, který volá tuto metodu v předku - *Basic*, poražmo *SugarBean*, jež provede samotné uložení do databáze.

SuiteCRM staví samotné databázové úložiště do pozadí a vývojáře tak ve většině případech vůbec nemusí zajímat jaká databáze je použita, díky čemuž se pro základní případy obejde bez jediného řádku sql kódu. Definice vlastností objektů se provádí v souboru *vardefs.php* (*modules/Soucastky/vardefs.php*), jež obsahuje kolekci *dictionary*.

```
$dictionary['Soucastka'] = array(
    'table' => 'soucastky',
    'audited' => true,
```

Klíč kolekce je název objektu, následuje nastavení jména databázové tabulky a zda má být objekt auditován. V praxi to pak znamená, že uživatel má k dispozici seznam změn společně s jejich autory (vše jedním řádkem kódu, o zbytek se postará framework).

```
'fields' => array(
    'soucastka_ref_cislo'=> array (
        'name' => 'soucastka_ref_cislo',
        'vname' => 'LBL_SOUCASTKA_REF_CISLO',
        'type' => 'varchar',
        'len' => '255',
    ),
    'soucastka_lokace'=> array (
        'name' => 'soucastka_lokace',
        'vname' => 'LBL_SOUCASTKA_LOKACE',
        'type' => 'enum',
        'len' => '50',
        'options' => 'soucastky_soucastka_lokace_dom',
    ),
),
```

Klíč *fields* obsahuje pole (polí) s definicemi vlastností. Klíč *name* značí jméno popisované vlastnosti, *vname* je *id v jazykovém balíčku* s textem zobrazeným uživateli, *type* je typ proměnné (obvykle datový, ale může jít i o funkci) a konečně *len* jeho délka. U typu *enum* ještě následuje *options*, jež se znovu odkazuje na jazykový balíček, v tomto případě na asociované pole, kde klíče představují hodnotu, pod kterou je veden v systému (uložen do databáze) a musí být pro všechny jazyky stejné. Jejich hodnoty jsou texty, které se zobrazí uživateli.

Nejsou to jediné klíče se kterými se lze ve *fields* setkat, ale jejich kompletní popis se všemi možnostmi by vydal na samostatnou knihu, a to není účelem této práce. V případě zájmu doporučuji knihu *The Definitive Guide to SugarCRM* od *J. Mertic* [17], z níž je čerpána inspirace i pro tento příklad.

```
'indices' => array(
    0 => array(
        'name' => 'soucastky_pk',
        'type' => 'primary',
        'fields'=> array('id')
    ),
),
'relationships' => array(
)
);
```

Klíč *indices* obsahuje databázové klíče (primary, unique a index), v našem případě primární klíč nad *id*. Při vynechání, je automaticky vytvořen primární klíč stejný jako v ukázce. Druhý klíč, *relationships*, definující vztahy mezi objekty, jež je další položkou v *dictionary*, zde přeskočíme. Pro náš jednoduchý případ není třeba a na *relationships* se podíváme v samostané kapitole 5.2.2.

Závěrem souboru *vardefs.php* předáme *správci definic* informace o našem objektu, který je tímto kompletní:

```
require_once('include/SugarObjects/VardefManager.php');
VardefManager::createVardef('Soucastky', 'Soucastka',
    array('basic', 'assignable'));
```

Pro doplnění - správci říkáme, že jsme definovali objekt *Soucastka* modulu *Soucastky*, který vychází z *basic* (více o *basic* v kapitole 8.2) a je mu možno přiřazovat uživatele (*assignable*). Příkladem použití přiřazování může být situace, kdy vedoucí pracovník ráno rozděljuje úkoly podřízeným.

Pro fungování modulu po stránce modelu chybí již jen poslední tři kroky. Prvním z nich je registrace modulu do systému (říkáme *tady máš modul, použij jej*). Pokud se podíváme na obsah souboru

```
custom/application/Ext/Include/modules.ext.php
```

zjistíme, že zde je seznam všech registrovaných modulů. Tento soubor však může být přepsán (důvod je vysvětlen v posledním kroku) a použijeme raději

```
custom/Extension/application/Ext/Include/rozsireni_soucastky.php.
```

Zde je jedno z mála míst, kde si můžeme zvolit vlastní pojmenování souboru (systém prochází celou složku), důležitý je jeho obsah:

```
$beanList['Soucastky'] = 'Soucastka';
$beanFiles['Soucastka'] = 'modules/Soucastky/Soucastka.php';
$moduleList[] = 'Soucastky';
```

V druhém kroku jsou vytvořeny informace pro jazykový balíček. Použijeme soubor, v jehož názvu je uveden prefix pro identifikace jazykového balíčku *cs_cz* (oddělený tečkou), příponou *php* a zbytek je libovolný, tedy např. včetně cesty

```
custom/Extension/application/Ext/Language/cs_cz.soucastky.php
```

s obsahem

```
$app_list_strings['moduleList']['Soucastky'] = 'Součástky';
$app_list_strings['moduleListSingular']['Soucastky'] = 'Součástka';
```

Pro popisy zobrazených polí se poté využívá souborů uložených v

```
modules/Soucastky/language/cs_cz.lang.php
```

resp.

```
custom/Extension/modules/Soucastky/Ext/Language/cs_cz.soucastky.php
```

ve kterých se definuje kolekce *mod_strings* ve tvaru:

```
$mod_strings['LBL_SOUCASTKA_REF_CISLO'] = 'Referenční číslo'.
```

Třetí, poslední krok, se provádí přes uživatelské rozhraní - v administrátorském menu je třeba zvolit *Oprava* a provést *Rychlou opravu a rekonstrukci* (v závislosti na použitém jazykovém balíčku), jenž mj. zajistí promítnutí změn do systému a provede vytvoření tabulek (či jejich aktualizaci) v databázi. Jedním z efektů je přepsání právě souboru *modules.ext.php* a tím pádem ztráty naší informace o registraci modulu. Tuto akci je třeba provést po většině zásahů do systému.

5.2.2 Relationship

Vztahy mezi objekty je možné definovat vyplněním klíče *relationship* v samotném *vardefs.php*, případně pro rozsáhlejší definice je vhodnější vytvořit soubor v *custom/metadata*. Pro náš úvodní příklad si definujeme vztah mezi modulem *Soucastky* a *Vyrobci*. Jedna součástka zjevně může mít jen jednoho výrobce, ale výrobce může vyrábět součástek několik - jde tedy o vztah 1:N (v SuiteCRM se tento vztah označuje jako *one-to-many*). Definice takového vztahu je následující:

```
'relationships' => array(
    'soucastky_related_vyrobci' => array(
        'lhs_module' => 'Soucastky',
        'lhs_table' => 'soucastky',
        'lhs_key' => 'id',
        'rhs_module' => 'Vyrobci',
        'rhs_table' => 'vyrobci',
        'rhs_key' => 'related_vyrobce_id',
        'relationship_type' => 'one-to-many',
    )),
```

Klíče s prefixem *lhs* popisují levou stranu vztahu, zatímco *rhs* pravou. Názvy *module* i *table* jsou dostatečně samopopisné a *key* nastavuje jméno primárního sloupce na příslušné straně vztahu.

S pouhým definováním vztahu se však pravděpodobně nespokojíme - chceme dát uživateli možnost jej využít, tedy na některém z *view* zobrazit či vybrat daného výrobce. To je možno provést přidáním dalších položek do *fields*:

```
'related_vyrobce_id' => array(
    'name' => 'related_vyrobce_id',
    'rname' => 'name',
    'id_name' => 'related_vyrobce_id',
    'vname' => 'LBL_VYROBCE_ID',
    'type' => 'relate',
    'table' => 'vyrobci',
    'module' => 'Vyrobci',
    'isnull' => 'true',
    'dbType' => 'id',
```

```

),
'related_vyrobce_name' => array(
  'name' => 'related_vyrobce_name',
  'required' => true,
  'link' => 'related_vyrobce_link',
  'vname' => 'LBL_VYROBCE',
  'rname' => 'name',
  'type' => 'relate',
  'source' => 'non-db',
  'table' => 'vyrobci',
  'id_name' => 'related_vyrobce_id',
  'module' => 'Vyrobci',
),
'related_vyrobce_link' => array(
  'name' => 'related_vyrobce_link',
  'type' => 'link',
  'relationship' => 'soucastky_related_vyrobci' ,
  'vname' => 'LBL_VYROBCE',
  'link_type' => 'one',
  'module' => 'Vyrobci',
  'bean_name' => 'Vyrobce',
  'source' => 'non-db',
  'rname' => 'name',
  'id_name' => 'related_vyrobce_id',
  'table' => 'vyrobci',
),
),

```

Účelem prvního, *related_vyrobce_id*, je vytvoření vlastnosti do které je ukládáno *id* výrobce a jako jediná je opravdu uložená v databázi (analogií v sql je cizí klíč) - zbylé obsahují klíč *'source' => 'non-db'*. Druhý odkazuje na jméno výrobce (*Vyrobce.name*), případně jiný identifikátor, který je použit pro zobrazení na konkrétním view. Poslední je potom vazba na vytvořený vztah (*relationship*).

5.2.3 Mazání cenných dat

Existence vlastnosti *deleted* - tedy zda je záznam smazán může znít zvláště, ale je nutno si uvědomit, že data jsou pro tyto systémy to nejcennější a proto nejsou ve většině případů úplně mazána, ale pouze skryta. Pokud je systém dobře navržen, není se třeba obávat ani rychlého úbytku místa na datovém úložišti.

Uvažujme např. hypotetickou situaci, kdy se rozhodujeme od kterého ze dvou výrobců odebírat konkrétní (stejnou) součástku pro náš výrobek. V případě poruchy výrobku zákazník pošle e-mail na reklamační oddělení, které jej posoudí, označí štítkem (např. „reklamace výrobce b“) a předá dál. Po dobu celého reklamačního řízení je jistě důležitý i text odeslaného emailu včetně připojených příloh. Z dlouhodobého hlediska má však pouze význam onen štítek a třeba fakt, že byla přijata e-mailem (po roce nikoho nezajímá rozhořčený e-mail zákazníka). V tomto příkladě bychom tedy mohli vytvořit dva modely - *Reklamace* a *ReklamaceDetail*. První obsahující od koho a jak byla reklamace přijata, důvod závady (naš štítek) a zda je smazána. Druhý pak text e-mailu se všemi přílohami. Po uzavření reklamačního řízení či uplynutí určitého

časového období *Reklamaci* označíme jako smazanou - nezobrazuje se tedy v seznamu reklamací a smažeme (tentokrát opravdu) odpovídající detail. Důležitá data zůstala zachována a my stále můžeme zjistit, jakého výrobce si vybrat a přesto nemusíme udržovat všechna data, což prospěje úložišti i rychlosti aplikace samotné.

5.2.4 Views

V předchozí kapitole jsme definovali model. Víme jaká data model obsahuje a nyní je třeba definovat kdy a jak chceme zobrazit konkrétní data. *SuiteCRM*, ve snaze ušetřit čas při vývoji, nabízí tři základní views:

- *ListView* pro zobrazení seznamu objektů (zobrazen jako defaultní),
- *DetailView* pro zobrazení detailu konkrétního objektu,
- *EditView* pro editaci konkrétního objektu.

Všechny popisné soubory se umísťují do složky *metadata* daného modulu, pro definici *ListView* tedy použijeme soubor:

```
modules/Soucastky/metadata/listviewdefs.php.
```

Stejně jako u modelu je použita kolekce, v tomto případě *listViewDefs*. Obsah souboru pokud chceme zobrazit *name* (jméno součástky), *soucastka_ref_cislo* a *soucastka_lokace*:

```
$listViewDefs['Parts'] = array(  
    'NAME' => array(  
        'width' => '32',  
        'label' => 'LBL_NAME',  
        'link' => true  
    ),  
    'SOUCASTKA_REF_CISLO' => array(  
        'width' => '30',  
        'label' => 'LBL_SOUCASTKA_REF_CISLO',  
    ),  
    'SOUCASTKA_LOKACE' => array(  
        'width' => '30',  
        'label' => 'SOUCASTKA_LOKACE',  
    );  
);
```

Za povšimnutí stojí použití velkých písmen v názvu vlastností. Pomocí *width* je definována šířka (defaultně v procentech). *Label* určuje *id* v jazykovém balíčku. Klíč *link* značí, že v tomto sloupci je umístěn odkaz a je možno otevřít (defaultně) *DetailView* daného objektu. Ten lze definovat v

```
modules/Soucastky/metadata/detailviewdefs.php
```

s následující strukturou:

```
$viewdefs[Soucastky]['DetailView'] = array(
    'templateMeta' => array(
        'form' => array(
            'buttons' => array(
                'EDIT',
                'DUPLICATE',
                'DELETE',
            ),
        ),
        'maxColumns' => '2',
        'widths' => array(
            array('label' => '10', 'field' => '30'),
            array('label' => '10', 'field' => '30')
        ),
    ),
    'panels' => array(
        array(
            'name',
            'soucastka_ref_cislo',
        ),
        array(
            'soucastka_lokace',
            'related_vyrobce_name'
        ),
    ),
);
```

V první části (*templateMeta*) je obsažen popis formu a rozložení, přesněji:

- *buttons* jsou popisy tlačítek (v tomto případě jde o předdefinované akce),
- *maxColumns* určuje do kolika sloupců maximálně výpis rozdělit,
- *widths* nastavuje rozdělení velikosti sloupců.

V druhé části (*panels*) najdeme vlastnosti objektu k zobrazení. Struktura definičního souboru *editviewdefs.php* pro *EditView* je shodná s *DetailView*, a proto není třeba ji popisovat.

5.2.5 Controller

Třetí částí návrhového vzoru MVC je *Controller* - řadič. V našem krátkém a jednoduchém případě není třeba žádný definovat - systém použije předdefinované chování.

5.3 Upgrade-safe customizace

Při tvorbě vlastního modulu upravujeme až na výjimky (registrace modulu) jen soubory, které jsme si vytvořili a je tak malá pravděpodobnost, že by je někdo jiný, např. při aktualizaci systému, přepsal. Jiná situace ovšem nastane, pokud chceme upravit

již existující modul. Pro takové případy je zde složka *custom/modules*, obsahující podobnou strukturu, jako původní složka *modules*. Rozhodneme-li se např. nadále nezobrazovat v některém z pohledů libovolnou vlastnost či přeorganizovat rozložení, stačí upravit konkrétní definiční soubor pohledu a uložit jej do příslušné složky v *custom*. Systém v takovém případě dá přednost tomuto upravenému.

Nejde o jediný způsob použití této složky. Může nastat případ, kdy budeme požadovat uložení dat v jiném formátu (či před uložení chceme provést ještě jinou akci) - pro tyto scénáře slouží *hooks*. Umožní definovat akce ve stylu *před uložení proved' tuto operaci* apod.

Naším cílem je využít co možná největší část již existujícího řešení/kódu a zasahovat (v ne-upgrade-safe smyslu) do něj v co nejmenší možné míře (pokud to je možné), aby kvůli aktualizacím, které si klient může sám provést nepřišel o námi provedené změny. Tato cesta je sice mnohdy složitější, ale jediná správná.

II. PRAKTICKÁ ČÁST

6 ANALÝZA POŽADAVKŮ

6.1 Stávající řešení

Systém je, alespoň co se týče digitální formy pro uchovávání záznamů, stavěn od základů. Nyní jsou využívány záznamy pouze v papírové formě, jejich příklad je na obr. 6.1. Hlavička obsahuje základní informace o klientovi jako je název firmy, adresa a odpovědná (kontaktní) osoba společně s telefonním číslem. Dále také informace o spravovaném zařízení, tedy v tomto případě o jaký jde model (typ), sériové číslo a datum instalace, případně instalované příslušenství. Jednotlivý záznam poté obsahuje datum jeho pořízení, stav celkového počítadla kopií, jaký servisní zásah byl proveden a použitý materiál. V optimálním případě by tento záznam měl být prostudován servisním technikem (dále jen „technik“) před každou cestou ke klientovi a měl by se na ni náležitě vybavit - ať již spotřebním materiálem či v rámci plánování času, pokud se blíží nutnost pravidelné údržby. Takové plánování přináší klientovi značné finanční i časové úspory, minimálně v tom, že nejsou nutné dvě cesty technika.

Firma:			
Adresa			
Telefon			
Odpovědná osoba:			
Typ stroje:	TOSHIBA c-studio 165		Příslušenství:
Výrobní číslo:	CXL 63 38 22		Datum instalace:
			31/5 2007
Datum	Počet kopií	Provedený zásah	Použitý materiál
24/9 07	51 349	PROFYLAKTICKÁ	OD 1600
1/10 07	61 764	PROFYLAKTICKÁ, VÝMĚNA	OD 1600, BL 2320, B 2320
2/9 07	88 827	PROFYLAKTICKÁ	
7/7 07	115 733	PROFYLAKTICKÁ, VÝMĚNA	OD 1600, BL 2320, B 2320
27/10 07	146 627	PROFYLAKTICKÁ	
15/11 07	166 616	VYČISTĚNÍ KOPÍ PEDAGOGICKE I OPTICKY	
19/2 07	187 032	PROFYLAKTICKÁ	
12/6 07	211 288	-	OD 1600, BL 2320
10/11 07	247 457	PROFYLAKTICKÁ	OD 1600, BL 2320, B 2320
10/2 10	270 918	VYČISTĚNÍ, REGISTR, PODÁV. ÚPRAVA KAPETI	

Obr. 6.1 Příklad se záznamy o konkrétním zařízení

Některá zařízení, která mají povolen přístup na Internet, umožňují zasílat různá hlášení (reporty) na zadané e-mailové adresy. Tato hlášení mohou být pravidelná (např. každý měsíc) obsahující souhrnné informace či zasláná na základě vzniklé události. Takovou událostí může být chyba (např. zaseknutý papír) či docházející spotřební materiál (nejčastěji toner).

Tyto zprávy pak některý z techniků přebere a posoudí, zda je nutno věnovat jim pozornost (pravidelné hlášení to obvykle nevyžaduje). Při větším počtu zpráv je ovšem tato činnost časově náročná a je snadné přehlédnout některé důležitější zprávy. Dalším potenciaálním problémem může být přístup více techniků najednou - e-mailová schránka k tomu není uzpůsobena.

6.2 Analýza požadavků

Před samotným porovnáním již existujících systémů, je vhodné nejprve stanovit požadavky kladené na informační systém. Umožní nám to lépe se zaměřit na konkrétní aspekty porovnávaných systémů a ulehčí jejich hodnocení. Požadavky lze rozdělit na dvě základní kategorie - *funkční* a *nefunkční*, přičemž je možno je dále členit do podkategorií. Míra splnění těchto požadavků pak slouží jako jedno z hodnotících kritérií pro výběr vhodného řešení.

6.2.1 Funkční požadavky

Uživatelé

- vedení seznamu uživatelů
- přiřazování uživatelských oprávnění (minimálně na úrovni přístupu k modulu)

Nastavení systému

- vypínání/zapínání modulů
- nastavení kontrolované e-mailové schránky
- nastavení dalších konfigurovatelných parametrů (defaultní lokalizace, vzhled systému, ...)

Systémový plánovač

- vedení seznamu plánovaných úloh a jejich spouštění v nastavených časech či v zadaných intervalech (např. výběr e-mailové schránky)

E-mailová schránka (poštovní klient)

- stahování zpráv z poštovního serveru včetně možnosti jejich následného zpracování

Klienti

- vedení seznamu klientů s možností jejich vyhledání
- nastavení kontaktních údajů
- vedení seznamu asociovaných zařízení

Konvertory reportů

- definice rozhraní pro převod zasílaných e-mailových zpráv do čitelné podoby

Šablony zařízení

- správa parametrů společných pro dané zařízení (výrobce, modelová řada)
- výběr asociovaného konvertoru
- vedení seznamu příloh - souborů s manuály a návody

Servisní záznamy

- vedení seznamu servisních záznamů

Správce zařízení

- správa aktuálního klienta
- výběr šablony zařízení
- vedení seznamu provedených servisních úkonů (servisní záznamy)
- nastavení parametrů pro *pasivní* komunikaci se zařízením - e-mailová adresa, ze které chodí zasílané hlášení
- nastavení parametrů pro *aktivní* komunikaci se zařízením - pro přímé připojení k zařízení

Jelikož by systém měl dle dohody se zadávající společností v prvotní fázi nahrazovat systém stávající, je tedy kladen větší důraz na *pasivní* zpracování komunikace.

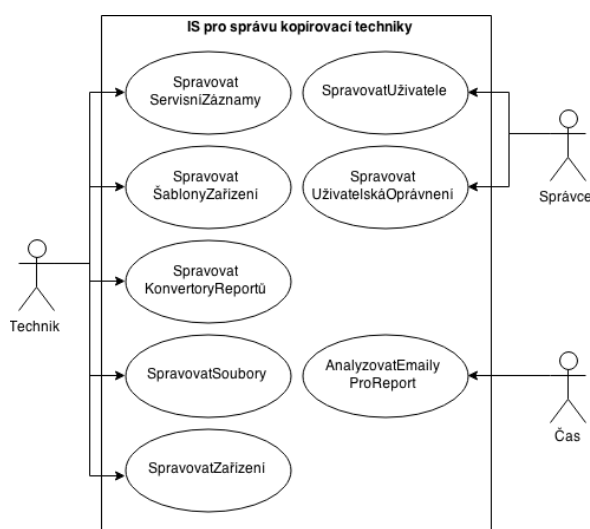
6.2.2 Nefunkční požadavky

- typ aplikace - webová aplikace
- přístup z mobilních zařízení (responzivní design webové aplikace či samostatný klient pro platformu Android)
- česká lokalizace
- jednoduchost ovládání
- otevřená licence

6.2.3 Případy užití (UC diagram)

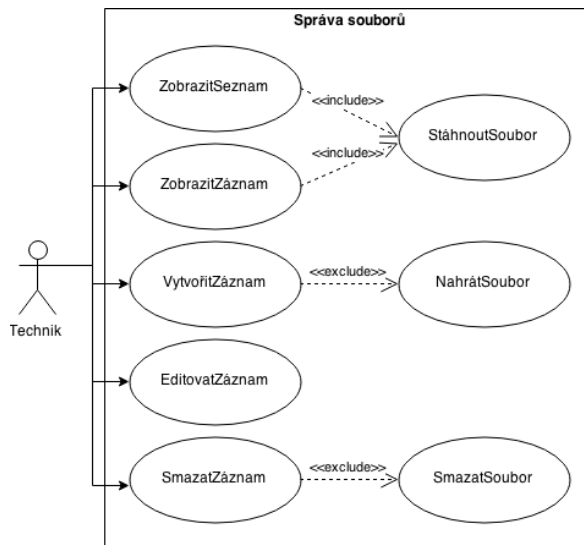
V rámci funkční analýzy si nyní zpracujeme možné *případy užití*. Začneme s hledáním skupin uživatelů - aktérů, kteří mají v systému figurovat. První skupinou jsou *administrátoři* - správci, kteří mají na starosti obecné nastavení systému a udílení oprávnění ostatním uživatelům. Druhou skupinu tvoří běžní uživatelé - *technici*, jenž mají na starosti správu zařízení, záznamů apod. Poslední, třetí skupina, spouští úlohy systémového plánovače. Nejde v tomto případě o uživatele, ale reprezentuje akci spuštěnou v daném čase. V UC diagramech se tyto aktéři obvykle označují jako *čas*.

Na obr. 6.2 je vyobrazen letmý pohled na celý, námi konstruovaný, systém. Jednotlivé případy užití jsou zde uvedeny na nejvyšší úrovni abstrakce, pod pojmem *Spravovat* si tedy v nejjednodušším případě lze představit alespoň základní *CRUD* (create, read, update, delete) akce. *Správu uživatelů a udělování oprávnění* bude systém již pravděpodobně obsahovat, proto není třeba tento případ užití uvádět podrobněji.

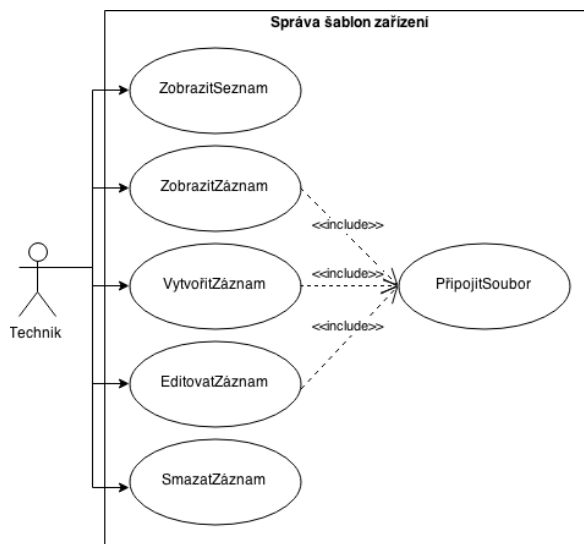


Obr. 6.2 UC celého systému

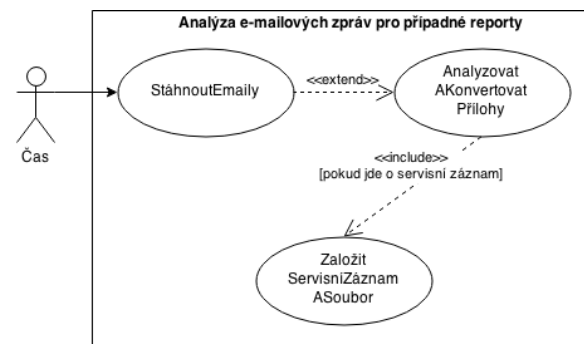
Na obr. 6.3 je diagram pro detailnější správu souborů se všemi akcemi, které lze provádět. Z diagramu je mj. patrné, že soubor je možno stáhnout jak ze seznamu, tak v jeho detailu. Obr. 6.4 vyobrazuje použití při správě šablon, které je až na drobné odlišnosti analogické u všech ostatních případů použití prováděných technikem. Odlišná situace je až při analýze emailů, která je na obr. 6.5. Stažení nových e-mailů vyvolá spuštění jejich analýzy. Pokud se podaří ve zprávě nalézt servisní záznam, dojde k jeho konvertování a následnému uložení.



Obr. 6.3 UC správy souborů



Obr. 6.4 UC správy šablon



Obr. 6.5 UC analýzy e-mailových zpráv

7 ANALÝZA VYBRANÝCH INFORMAČNÍCH SYSTÉMŮ

Z požadavků vyplývá, že nejsme omezeni použitou technologií (či programovacím jazykem), ale systém by měl obsahovat některé základní stavební prvky používané v CRM systémech, proto je výběr cílen tímto směrem.

7.1 Funkční část

Níže je uveden seznam systémů v abecedním řazení společně s klíčovými vlastnostmi důležitými pro splnění požadavků, případně krátký komentář proč je dané řešení vhodné, nevhodné či některé odlišnosti, jimiž vyniká oproti konkurenci. U názvu je vždy uveden slogan, kterým se daný produkt charakterizuje - mělo by tak být možné vytvořit si alespoň základní představu o jeho zařazení. Téměř každý systém obsahuje adresář kontaktů, umožňuje pracovat s více uživateli s možností alespoň základní správy oprávnění a má implementovanou podporu pro více jazyků (jazykové balíčky), většinou bez českého překladu. Tyto vlastnosti jsou proto zmiňovány pouze pokud je systém nějakým způsobem výrazněji rozšiřuje či naopak postrádá.

Navzdory krátkému popisu, daným počtem účastníků - systémů, pevně věřím, že by tento seznam mohl sloužit alespoň jako odrazový můstek se základním přehledem a mohl tak někomu usnadnit práci s hledáním „svého ideálního“ informačního systému.

Přehledná verze tohoto seznamu je uvedena v tabulce, která je součástí přílohy (viz přílohy 1.1 a 2.1).

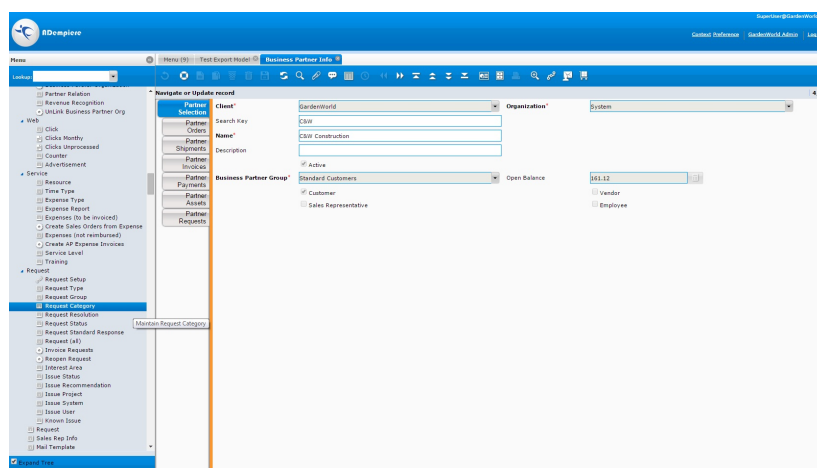
Adempiere („Midgets sitted over a giant shouldrs can see farest than the giant.“)

- webová aplikace má oddělené rozhraní pro mobilní zařízení
- správa uživatelů včetně správy rolí
- správa poštovní schránky

Výhodou je implementace plánování. Obsahuje inteligentní vyhledávač s našeptávačem. Umožňuje otevření více panelů v jednom okně. Uživatelské menu je prezentováno ve stromové struktuře, což vzhledem k počtu položek není vhodná volba (viz obr. 7.1).

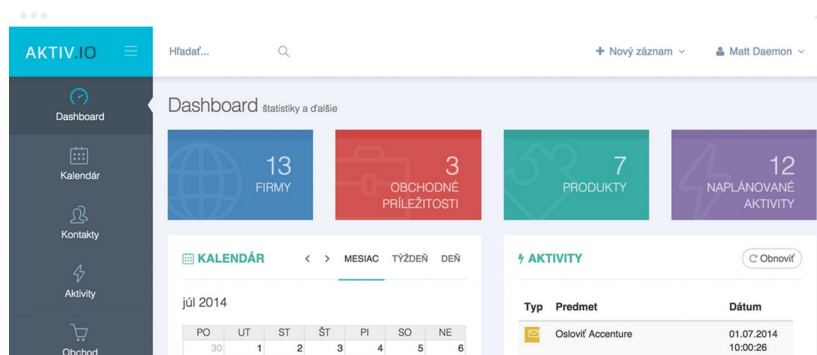
Aktiv.io („Pochopte svojich zákazníkovi, rozumejte im už dnes.“)

Tento slovenský produkt bohužel není šířen pod otevřenou licenci, ale jen ve formě *Software as a service* (SaaS). Uživatel si takový software pronajímá jen jako službu a proto jej není možné použít. V přehledu je zařazen pouze pro srovnání, především



Obr. 7.1 Uživatelské rozhraní Adempiere (zdroj: demo.testadempiere.com)

díky velmi příjemnému a modernímu uživatelskému rozhraní (viz obr. 7.2). To je optimalizováno pro použití na mobilních zařízeních. Čistě ze zajímavosti je tedy možno uvést funkcionalitu: firmy & kontakty, úlohy & sledování událostí, synchronizace kalendářů, mailová asistentka (kontrola a přiřazování emailů jednotlivým zákazníkům), analytické sestavy.



Obr. 7.2 Uživatelské rozhraní Aktiv.io (zdroj: aktiv.io)

CiviCRM („Growing and Sustaining Relationships“)

- webová aplikace není optimalizována pro mobilní zařízení (testováno na verzi drupal), existence API
- správa uživatelů s nastavováním přístupu (zobrazení/editace)
- správa poštovní schránky
- z větší části přeloženo do češtiny

Funguje v propojení s redakčními systémy Drupal, Joomla, WordPress. Obsahuje správu příspěvků, událostí, korespondence, členství, kampaní, případů a reportů.

Coevery („Coevery - free ASP.NET CRM“)

- ačkoliv je avizováno, v demo instalaci není responzivní design
- správce uživatelů s přiřazováním rolí

Jeden z mála systémů, který umožňuje vytvářet jednoduché moduly přímo v uživatelském rozhraní (definování objektů, vzájemných vztahů a následných pohledů). Systém je již téměř rok (od ledna 2014) ve fázi alpha verze bez viditelné aktivity v repozitáři.

Epesi („EPESI - web CRM/ERP & PHP/AJAX framework“)

- kompletně oddělené rozhraní mobilní verze
- uživatelé s pokročilým správcem oprávnění
- nejbližší požadavku „zařízení“: modul „Inventory“, obsahující historii transakcí a prodejů, naskladněné zboží a jeho lokace - pouze jako placený modul (600 USD)
- podpora pro CRON (kontrola schránek), emailový klient (využívající Roundcube.net)
- částečný překlad do češtiny

Obsahuje integraci s Google Docs, dashboard, sdílený kalendář s upozorněními, sdílený úkolovník, záznam telefonních hovorů, poznámky a soubory - správa dokumentů, snadné vyplňování formulářů, sledování změn záznamů, shoubox, přizpůsobení klávesových zkratk.

ERPNext („Open Source ERP built for the Web. Uses Python, MariaDB.“)

- webová aplikace je optimalizována i pro mobilní zařízení - existuje responzivní design i pro mobilní telefony a tablety
- uživatelé s pokročilým správcem oprávnění
- nejbližší požadavku „zařízení“: modul „Stock“, obsahující historii transakcí a prodejů v kombinaci s evidencí plánovaných údržeb

ERPNext umožňuje sledování plateb, faktur a stavu skladů, komunikace se zákazníkem - řešení problémů, úkolovník - přiřazování úkolů a sledování stavu splnění, příprava nabídek, správa příspěvků na blogu, inteligentní vyhledávání s návazností na akce (např. vytvoření úkolu).

EspoCRM („Increase profitability through customer loyalty“)

- webová aplikace je optimalizována i pro mobilní zařízení - existuje responzivní design i pro mobilní telefony a tablety
- správa uživatelů s přiřazováním rolí (rozlišení: přístup k modulu, čtení, zápis, smazání)
- nejbliže požadavku „zařízení“: modul „Opportunities“
- správa poštovní schránky

Svížný a moderně vypadající systém s responzivním designem.

Fat Free CRM („You earned your clients with diligence; keep them with Fat Free CRM.“)

- webová aplikace není optimalizována pro mobilní zařízení, k dispozici je však mobilní aplikace pro Android včetně zdrojových kódů
- správce uživatelů s přiřazováním rolí
- nejbliže modulu „zařízení“: „Příležitosti“
- správa poštovní schránky
- dostupný český překlad

Neumožňuje sice tvorbu jednoduchých modulů jako např. Coevery, umí ale ke stávajícím objektům (modulům) přidávat nové vlastnosti.

Fedena („The all-in-one software to manage schools and colleges.“)

- mobilní verze pouze za poplatek
- uživatelé s možností nastavení typu (admin/učitel/student) s předdefinovanými oprávněními

IS je určen speciálně pro školy, a proto je uzpůsoben jejich potřebám - rozvrhy, zkoušky, propojení na moodle.

FreenetIS („Informační systém pro neziskové sítě.“)

- webová aplikace je částečně responzivní (při vyšším rozlišení je i na mobilním telefonu zobrazena desktopová verze)
- správa uživatelů s přiřazováním oprávnění dle skupin uživatele
- z velké části implementuje požadavek „Zařízení“
- správa poštovní schránky
- jedná se o český produkt, čeština je tedy dostupná

Systém určený především pro správu síťových zařízení, umožňuje s nimi také komunikaci (zjišťování stavu, nastavení, . . .). Z hlediska nutnosti doimplementace funkcí potřebných pro splnění požadavků patří k těm lepším.

FrontAccounting („Accounting on the Web has never been easier. . .“)

- mobilní verze není dostupná
- nejbližší požadavku „zařízení“: inventář a zásoby, obsahující historii transakcí a prodejů v kombinaci s evidencí plánovaných údržeb

Další vlastnosti: prodej a pohledávky, nákup a závazky, správa výroby, správa bankovních účtů, generování reportů (tisk, pdf, email).

Group-Office („Your online office“)

- webová aplikace není optimalizována pro mobilní zařízení, ale je z větší části použitelná; je dostupné API
- správa uživatelů, oprávnění udržováno pomocí rolí uživatelů
- správa poštovní schránky
- přeloženo do češtiny

Poměrně přehledný systém integrující podporu e-mailu. Sdílení souborů, funkce kalendáře. Možnost otevření více panelů, při jejich přepínání zůstává zachován stav (simuluje multitasking). Vyhledávání není „inteligentní“. Součástí systému je správce modulů, který umožňuje instalovat jak oficiální (většinou je třeba koupě licence), tak komunitní moduly.

HeliumV („the open source ERP system for small and medium-sized enterprises is freely available under the AGPL licence“)

- mobilní verze není dostupná, ale existuje REST API
- nejbližší požadavku „zařízení“: modul „Device“ umožňující vést evidenci údržby

HeliumV obsahuje správu dokumentů (export do různých formátů), prodeje a distribuce (nabídky, objednávky, . . .), nákupů, veřejných zakázek (dle německých předpisů). Dále nabízí plánování zdrojů, produkce, času a správu stížností zákazníků, údržbu a analýzu nákladů. Uživatelská dokumentace je dostupná pouze v německém jazyce.

iDempiere („iDempiere Business Suite ERP/CRM/SCM done the community way.“)

- mobilní verze není dostupná, ale existuje definice služeb pro komunikaci přes SOAP a nativní aplikace pro Android
- správa poštovní schránky

Obsáhlý ERP systém spojující finanční účetnictví (faktury, platby, objednávky, aktiva, . . .), CRM (vyřizování požadavků zákazníků) a Supply Chain Management (SCM - díly, testování kvality, výroba, . . .).

JFire („JFire - the all-rounder“)

- mobilní verze není dostupná
- správa uživatelů - možnost tvorby uživatelských skupin

Další vlastnosti: vývoj byl ukončen v roce 2009.

LedgerSMB („The foundation for your business“)

- mobilní verze není dostupná
- uživatelé s pokročilým správcem oprávnění (umožňuje nastavit oprávnění pro každou akci)
- částečný překlad do češtiny

LedgerSMB spravuje účetní operace (pohledávky, závazky), vede evidence zboží a služeb a umožňuje generování nabídek. Velmi jednoduchý a zastaralý vzhled. Aktualizace probíhají spíše nárazově (od roku 2012 byly vydány celkem tři verze).

Odoo/OpenERP („One need, one app. Integration has never been so smooth.“)

- mobilní verze webového rozhraní není dostupná, je však možno použít oficiální nativní aplikaci pro platformu Android
- uživatelé s pokročilým správcem oprávnění, možnost nastavení různých úrovní přístupu k jednotlivým modulům, případně jej zamezit úplně
- nejbližší požadavku „zařízení“: modul „Fleet“ pro správu vozového parku, obsahující záznamy o stavu tachometru, nákupech benzínu a servisních záznamech
- správa poštovní schránky
- částečný překlad do češtiny

Projekt patří k těm rozsáhlejším - pokrývá oblasti více oblastí ERP jako je CRM, správa obsahu (Content Management System, CMS) a účetnictví. Jde rozdělit do šesti základních skupin, kterými jsou: front-end aplikace (web, blog), řízení prodeje, obchodní operace, aplikace pro marketing, správa lidských zdrojů a aplikace podporující produktivitu (zasílání instantních zpráv, poznámky, . . .). V současné době existuje 260 základních (core) modulů a asi 4000 modulů vytvořených komunitou. Systém samotný je aktivně vyvíjen.

Opentaps („Most advanced Open Source ERP+CRM solution“)

- webová aplikace není optimalizována pro mobilní zařízení, existence oficiální mobilní aplikace pro Android a iOS
- správce uživatelů, správa oprávnění dle rolí
- nejbližší požadavku „zařízení“: modul „Kampaně“
- částečně přeloženo i do českého jazyka

Projekt je aspoň v komunitní verzi aktualizován velmi sporadicky (souzeno dle historie změn v repozitáři).

Phreedom („Small Business Solutions“)

- mobilní verze není dostupná
- uživatelé s pokročilým správcem oprávnění, různé druhy přístupů pro dané moduly - plný, editace, přidávání, pouze pro čtení, bez přístupu

- nejbliže požadavku „zařízení“: modul „Inventory“, obsahující historii transakcí a prodejů, naskladněné zboží v jednotlivých skladech, statistiky prodejů

Obsahuje bankovní funkce, správu lidských zdrojů, podporu více společností a sledování zásilek. Bylo by nutno implementovat poměrně hodně funkcí, layout - rozvržení stránky je dosti nešťastné a není optimalizované pro mobilní zařízení ani neposkytuje API.

SimplERP („Fast deploying ERP solution“)

Tento systém byl z porovnání vyřazen kvůli nedostupným zdrojovým kódům a místy i nekompletní lokalizaci alespoň v anglickém jazyce.

SplendidCRM („SplendidCRM is the clear and obvious choice for companies running Windows.“)

- existuje rozhraní optimalizované pro mobilní zařízení, ovšem s omezenou funkcionalitou (je dostupná i oficiální mobilní aplikace pro Android, ale bez zdrojových kódů), možnost využít REST/SOAP API
- správce uživatelů s velmi podrobným nastavením oprávnění
- nejbliže požadavku „zařízení“: vzdáleně podobné „Příležitosti“ či „Kampaně“
- k dispozici je i čeština (pravděpodobně se ale jedná o strojový překlad); lokalizace je uložena v databázi, takže správce má možnost upravit překlad přímo z administrace systému

Umožňuje nastavit oprávnění pro přístup, čtení, editaci, smazání, import, export. Chybí možnost hromadného nastavení (viz obr. 7.3). Aplikace šířena pod GNU Affero General Public Licence, ale pro stažení aktuální verze je třeba se registrovat. Neexistuje repozitář, což značně znesnadňuje vývoj a následnou aktualizaci.

SQL-Ledger® / SQL-Ledger Network („Double entry accounting and ERP system“)

SQL-Ledger Network vychází z původního SQL-Ledger, který se nyní ubírá více komerčním směrem.

- mobilní verze není dostupná
- základní správa uživatelů - oprávnění zde nejsou v pravém smyslu slova (zakázané moduly jsou danému uživateli pouze skryty)

	Access	View	List	Edit	Delete	Import	Export
Accounts	Enabled	All	All	All	All	All	All
Bug Tracker	Enabled	All	All	All	All	All	All
Call Marketing	Enabled	All	All	All	All	All	All
Calls	Enabled	All	All	All	All	All	All
Campaign Log	Enabled	All	All	All	All	All	All
Campaigns	Enabled	All	All	All	All	All	All
Campaign Trackers	Enabled	All	All	All	All	All	All
Cases	Enabled	All	All	All	All	All	All
Charts	Enabled	All	All	All	All	All	All
Chat Channels	Enabled	All	All	All	All	All	All
Chat Dashboard	Enabled	All	All	All	All	All	All
Chat Messages	Enabled	All	All	All	All	All	All
Contacts	Enabled	All	All	All	All	All	All
Contracts	Enabled	All	All	All	All	All	All
Credit Cards	Enabled	All	All	All	All	All	All
Dashboard	Enabled	All	All	All	All	All	All
Documents	Enabled	All	All	All	All	All	All
Email Client	Enabled	All	All	All	All	All	All
Email Marketing	Enabled	All	All	All	All	All	All
Emails	Enabled	All	All	All	All	All	All
Email Templates	Enabled	All	All	All	All	All	All
Employees	Enabled	All	All	All	All	All	All
Forums	Enabled	All	All	All	All	All	All
moduleListImages	Enabled	All	All	All	All	All	All
Invoices	Enabled	All	All	All	All	All	All
Invoices Line Items	Enabled	All	All	All	All	All	All
Leads	Enabled	All	All	All	All	All	All
Meetings	Enabled	All	All	All	All	All	All
Notes	Enabled	All	All	All	All	All	All
Opportunities	Enabled	All	All	All	All	All	All
Orders	Enabled	All	All	All	All	All	All
Orders Line Items	Enabled	All	All	All	All	All	All

Obr. 7.3 Nastavení oprávnění v SplendidCRM (zdroj: demo.splendidcrm.com)

- nejbližší požadavku „zařízení“: modul „Goods & Services“ (zboží a služby)
- čeština je dostupná pro starší verzi

Prostředí aplikace, především její menu, se snaží působit „čistým“ dojmem, avšak toto menu obsahuje 25 položek, které se dále ve stromové struktuře větví, což může následně stěžovat orientaci.

SugarCRM - Community Edition („CRM software trusted by millions worldwide.“)

- webová aplikace není přímo optimalizovaná pro mobilní zařízení, ale obsahuje oddělené rozhraní ve formě doplňku, který obsahuje alespoň částečnou funkcionalitu a neoficiální aplikace dostupná pro Android (QuickCRM for SugarCRM); dále je možno využít SOAP a REST API
- správa uživatelů, oprávnění dle rolí
- správa poštovní schránky, včetně nastavení plánovače (CRON) pro pravidelné kontroly přímo z uživatelského rozhraní
- čeština dostupná pro starší verzi (komunitní doplněk)

CRM s rozsáhlou komunitou a spoustou doplňků (sugarforge.com). Instalace bezproblémová a rychlá. Z hlediska nutnosti doimplementace funkcí patří k průměru, existence API je výhodou. Pro placené verze existuje oficiální mobilní aplikace (Android

i iOS), pro Community Edition (které jediná je open-source) nutno hledat v doplňcích. Umožňuje tvorbu jednoduchých modulů z uživatelského rozhraní.

SuiteCRM („The open source alternative to Salesforce, Microsoft Dynamics and SugarCRM Professional“)

Jedná se o rozšířenou verzi SugarCRM - nejlépe tomu asi odpovídá popis samotných tvůrců „SuiteCRM is SugarCRM Supercharged“. Obsahuje veškerou funkcionalitu jako systém ze kterého vychází a se kterým je i zpětně kompatibilní. Jako jeden z mála prošel za dobu analýzy výraznějšími změnami. Tou nejvýraznější změnou je především responzivní design.

Tryton („Is a three-tier high-level general purpose application platform under the license GPL-3 written in Python and using PostgreSQL as database engine. It is the core base of a complete business solution providing modularity, scalability and security.“)

- desktopová aplikace dostupná pro Windows, Linux, MacOS X, BSD, existuje i neoficiální klient pro Android (bez dostupnosti zdrojových kódů), možnost použití RPC
- uživatelé s pokročilým správcem oprávnění, umožňuje udělovat oprávnění pro jednotlivé akce modulů
- nejbliže požadavku „zařízení“: modul „Inventory & stock“
- podpora pro CRON (kontrolu schránek), emailový klient (využívající Roundcube.net)
- čeština je dostupná

Tryton obsahuje moduly pro účetnictví (a následnou analýzu), fakturace, řízení nákupu, prodeje, zásob, výroby, projektů a příležitostí.

Vtiger („Win More Deals“)

- webová aplikace není optimalizována pro mobilní zařízení, možnost použít REST API, k dispozici jsou oficiální aplikace pro Android, iPhone a iPad, ovšem bez zdrojových kódů
- správa uživatelů s pokročilým správcem oprávnění - na základě rolí nastavuje atributy pouze čtení, či čtení a zápis
- nejbliže požadavku „zařízení“: např. modul „Produkty“ a „Servis“, ten umožňuje vést historii oprav a přikládat i dokumenty

- operace s poštovní schránkou
- přeloženo do slovenštiny

Celkově přehledný systém s možností správy poštovní schránky. Výhodou je i možnost vytvořit jednoduché „moduly“ přímo v uživatelském rozhraní (definice objektů a vztahů mezi nimi).

WebERP („webERP is a mature open-source ERP system providing best practise, multi-user business administration and accounting tools over the web.“)

- webová aplikace není optimalizována pro mobilní zařízení
- správa uživatelů se základním nastavením oprávnění (povolení/zakázání přístupu k modulu)
- nejbližší požadavku „zařízení“: modul „items“, evidující množství na skladě, pohyby zboží (prodeje), z čeho je položka složena (z jakých dílů)
- správa poštovní schránky
- minimální překlad do češtiny

xTuple PostBooks® („World’s #1 Open Source ERP“)

- aplikace obsahuje jak desktopového klienta, tak webové rozhraní přizpůsobené i pro mobilní telefony
- uživatelé s pokročilým správcem oprávnění, umožňuje udělovat oprávnění pro jednotlivé akce modulů
- nejbližší požadavku „zařízení“: modul „Inventory“ (zásoby), obsahující historii transakcí/prodejů

Obsahuje moduly účetnictví, prodej, CRM, výrobu, nákupy, zásoby a produkty. Modul CRM obsahuje mj. úkolovník, kalendář, správce událostí, správu předprodejních akcí jako jsou nabídky, příležitosti („opportunities“), ale také správu událostí/incidentů a správu projektů.

Zurmo („The New Open Source CRM: Gamified, Mobile, Social.“)

- webová aplikace je optimalizována i pro mobilní telefony - responzivní design s možností ručního přepnutí

- správce uživatelů s přiřazováním rolí
- nejbližší požadavku „zařízení“: modul „Kampaně“
- správa poštovní schránky
- částečně přeloženo i do češtiny

Poměrně přehledný, jednoduše ovladatelný systém s velkými možnostmi přizpůsobení. Zajímavým prvkem je zde „*gamifikace*“, která má zábavným způsobem přimět uživatele k používání systému - sbírání odznaků, odměny za plnění úkolů. Z hlediska nutnosti doimplementování se řadí k „lepšímu“.

O něco komplikovanější je ovšem instalace (navíc ne všechny chybové hlášení jsou zrovna dobře napsány, např.: „Database default collation is: utf8_general_ci . Database default collation should not be in: utf8_general_ci“) a bez memcache je systém obtížně použitelný kvůli pomalým odezvám. To, společně s využíváním sql procedur, může být i v případě placeného hostingu umezujícím prvkem.

7.2 Nefunkční část, zhodnocení

Předchozí část se věnovala především funkční části srovnání a některým prvním postřehům při používání. Při výběru vhodného systému je třeba přihlédnout i k jiným kritériím. Řadíme zde např. snadnost obsluhy, atraktivita vzhledu uživatelského rozhraní, odezvy aplikace (rychlost) či použitou technologii. Jelikož se v našem případě jedná o open-source produkty, je pro nás - jako vývojáře, také velmi důležitá komunita či kvalita dokumentace. Většina repozitářů také umožňuje zobrazit historii změn a je možno si tak udělat obrázek o četnosti aktualizací - jak aktivně je daný systém vyvíjen či jak rychle je reagováno na nahlášenou chybu.

Souhrn všech těchto hodnocení je možné nalézt v tabulkách přílohy (viz 1.1 a 2.1). Po sečtení všech ohodnocení se jako nejpříjemnější jeví použití některého ze systémů *CiviCRM*, *FreenetIS*, *SugarCRM* a z něj vycházejícího *itSuiteCRM*. Tuto čtveřici je podrobena SWOT analýze. Na výsledek se můžeme podívat v tabulkách 7.1, 7.2, 7.3 a 7.4.

Po zhodnocení všech dostupných údajů se jako nejlepší jeví *SuiteCRM*, který vyhrál jak po stránce tabulkového hodnocení (tedy jak po stránce funkční, tak nefunkční), tak má mírně navrch i při SWOT analýze. Kromě silné vývojářské komunity je zde plusem také existující OnDemand řešení (tedy poskytování hostingu aplikace), což by mělo dokázat zajistit financování a následný vývoj. Následkem toho je pak minimalizace hrozby - závislosti na *SugarCRM*, u kterého práce na vývoji nových verzí stagnuje (neplatí pro ostatní, placené edice). Slabá stránka, tedy neoptimalizování pro PHP 5.5+, se vyskytovala u většiny testovaných systémů - nejedná se tak o fatální nedostatek.

Tab. 7.1 SWOT analýza pro *CiviCRM*

CiviCRM	
Silné stránky	Slabé stránky
Zaměření na komunitní organizace	Neoptimalizováno pro mobilní zařízení
Odezva na požadavky komunity	Žádné trvalé finanční příjmy
Příležitosti	Hrozby
Vytvoření alespoň SaaS varianty	Roztříštěnost (Drupal/Joomla/WordPress)

Tab. 7.2 SWOT analýza pro *FreeNetIS*

FreeNetIS	
Silné stránky	Slabé stránky
Vyvíjeno na akademické půdě	Spíše lokálního charakteru
Podobné IS většinou komerčně zaměřeny	Neexistence rozšíření třetích stran
Příležitosti	Hrozby
Dotazení vzhledu pro mobilní zařízení	Malý vývojový tým
	Ve hře přechod na jinou architekturu

Tab. 7.3 SWOT analýza pro *SugarCRM*

SugarCRM	
Silné stránky	Slabé stránky
Silná vývojářská komunita	Odladěno jen na PHP 5.3
Existence spousty rozšíření	
Rozšířenost	
Příležitosti	Hrozby
Placené verze - financování rozvoje	Jeden z mnoha produktů na trhu

Tab. 7.4 SWOT analýza pro *SuiteCRM*

SuiteCRM	
Silné stránky	Slabé stránky
Silná vývojářská komunita	Odladěno jen na PHP 5.3
Existence spousty rozšíření	
Rozšířenost	
Připraveno na mobilní zařízení	
Příležitosti	Hrozby
Existence OnDemand - financování rozvoje	Jeden z mnoha produktů na trhu
	Postaveno nad produktem jiné firmy

8 IMPLEMENTACE

Návrh architektury řešení je z větší části dán výběrem daného systému. K dispozici je *Sugar framework* a návrhový vzor *MVC*.

8.1 Stávající moduly

Nejprve je třeba vycházet z toho, co je již implementováno a zjistit do jaké míry je to použitelné pro naše účely. Projdeme tedy podrobněji již existující moduly a vybereme ty, jež budou vhodné pro naše zamýšlené využití. Následuje jejich seznam s krátkým komentářem.

8.1.1 Uživatelé (Users)

Tvoří základní kámen systému a dostatečně pokrývá všechny naše požadavky jak na správu uživatelů, tak na správu oprávnění. Není tedy třeba provádět žádné úpravy a je možno jej použít v současném stavu.

8.1.2 Poznámky (Notes)

Umožňují tvorbu poznámek společně s příkládáním souborů. Zároveň slouží jako odkladiště souborů pro ostatní moduly (Chyby, Emaily - ukládání příloh, Hovory atd.). Pro zamýšlené použití (příkládání příloh, ukládání hlášení) by mohlo být pro uživatele zbytečně složité a neumožňuje např. zobrazit náhled souboru.

8.1.3 Dokumenty (Documents)

Poskytují uživateli možnost verzovat nahrané dokumenty, třídit je do kategorií, nastavovat spřízněné dokumenty a další. Stejně jako u předchozího je i tento modul zbytečně složitý a taktéž nepodporuje náhledy souborů.

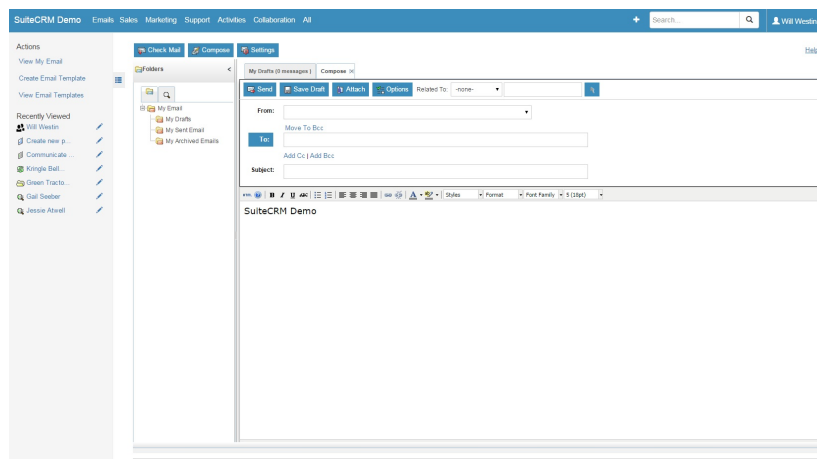
8.1.4 Plánovač (Scheduler)

Rozhraní díky němuž je možno v uživatelském prostředí spravovat úkoly spouštěné přes systémový plánovač. Kromě tvorby úkolu či úpravu stávajícího není třeba zasahovat a je možno použít v současném stavu pro plánování úloh (analýza a konverze e-mailových zpráv pro případné reporty).

8.1.5 Pošta (Emails)

Slouží jako uživatelské rozhraní pro správu poštovních schránek. Umožňuje zobrazit zprávy v ní obsažené (doručené, odeslané, koš, ...), aktualizovat obsah a psát zprávy.

Spravuje jak účty osobní, tak účty doručené pošty (viz dále). Vzhledově sice působí mírně zastarale¹⁾, nicméně po funkční stránce je použitelná bez větších změn. Jak vypadá uživatelské rozhraní tohoto modulu je zobrazeno na obr. 8.1.



Obr. 8.1 Uživatelské rozhraní modulu Pošta

8.1.6 Doručená pošta (Inbound Mail)

Použito k nastavení společných emailových účtů, které mohou být dále vybírány za pomoci *plánovače*. Účty je možno řadit do několika kategorií - ty ovšem nelze definovat. Až na poslední zmíněné je po drobných úpravách použitelné v kombinaci s Plánovačem pro stažení a následné předání ke zpracování zasílaných hlášení.

8.1.7 Klienti (Accounts)

Správa klientů (firem) evidovaných v systému. Po vytvoření vazby slouží jako vlastník daného zařízení.

8.1.8 Kontakty (Contacts)

Podobně jako u *Klientů*, jen s tím rozdílem, že slouží k evidenci osob. Ačkoliv tento požadavek nebyl vznesen, je možno po vytvoření vazby využít jako kontaktní osobu pro dané zařízení.

8.2 Implementované moduly

Pro splnění všech funkčních požadavků je třeba implementovat *Konvertory reportů*, *Šablony zařízení*, *Servisní záznamy* a *Správce zařízení*. Názvy požadavků v tomto případě

¹⁾Vývojáři o tomto nedostatku vědí a zařadili jej na seznam věcí (<https://suitecrm.com/community/roadmap>), které jsou v plánu; i když zatím bez plánovaného data splnění.

odpovídají pojmenování nově vytvořených modulů. Dále vytvoříme modul *Správce souborů*, jelikož žádný existující nevyhovuje našim požadavkům.

Při tvorbě je využito možnosti vytvářet modely na základě rozšíření šablony modelu a doplnění námi požadovaných vlastností, příp. předefinování stávajících. Použita je šablona *basic* s následujícími dispozicemi (viz tab. 8.1).

Tab. 8.1 Šablona objektů *basic*

Název vlastnosti	Typ	Typ v databázi	Komentář
id	id	guid	Jednoznačný identifikátor objektu (guid).
name	name	varchar	Jméno objektu.
date_entered	datetime	datetime	Datum a čas vložení do systému.
date_modified	datetime	datetime	Datum a čas poslední úpravy záznamu.
modified_user_id	assigned_user_name	id	Vazba na uživatele.
modified_by_name	relate	-	Pomocné pro vazbu na uživatele, neukládá se do databáze. Jméno uživatele.
created_by	assigned_user_name	id	Vazba na uživatele.
created_by_name	relate	-	
date_modified	datetime	datetime	Pomocné pro vazbu na uživatele, neukládá se do databáze. Jméno uživatele.
description	text	varchar	Poznámka.
deleted	bool	tinyint	Je záznam smazán?

Pro vytvoření skutečné vazby jsou dále vytvořeny vlastnosti *created_by_link* a *modified_user_link*, jež definují skutečný vztah mezi modulem a modulem *Uživatelé*. Automaticky je pak vytvořen i primární klíč nad sloupcem *id*.

Šablonu *basic* je dále možno doplnit za pomoci rozhraní *assigned*, jež ji dále rozšiřuje o vlastnosti uvedené v tabulce 8.2 a vlastnost *assigned_user_link*, opět pro skutečnou definici vztahu.

Všechny nové moduly používají šablonu *basic* a u kterých je to vhodné taktéž roz-

Tab. 8.2 Šablona rozšíření *assignable*

Název vlastnosti	Typ	Typ v databázi	Komentář
assigned_user_id	relate	guid	Vazba na uživatele.
assigned_user_name	relate	-	Pomocné pro vazbu na uživatele, neukládá se do databáze. Jméno uživatele.

hraní *assigned*. Protože nemá smysl pro každý modul znovu opisovat definici této šablony, jsou dále při popisu modelu uvedeny pouze něčím význačně vlastnosti pro ten který modul. Taktéž jsou vynechány definice vztahů, těm patří vlastní kapitola (8.3).

8.2.1 Správce souborů

Cílem je umožnit uživateli (technik) nahrávat soubory, pojmenovat je vlastním jménem, případně opatřit poznámkou. Nahrané soubory poté prohlížet a podle mime typu pak u vybraných zobrazit jejich náhled (typicky image/jpeg, text/xml) či stahovat. Soubory je možno vytvářet samostatně či je přikládat k *Servisním záznamům* a *Šablonám zařízení*. Datový model tohoto modulu nalezneme v tabulce 8.3.

Tab. 8.3 Model *SpravceSouboru*, šablona *basic*

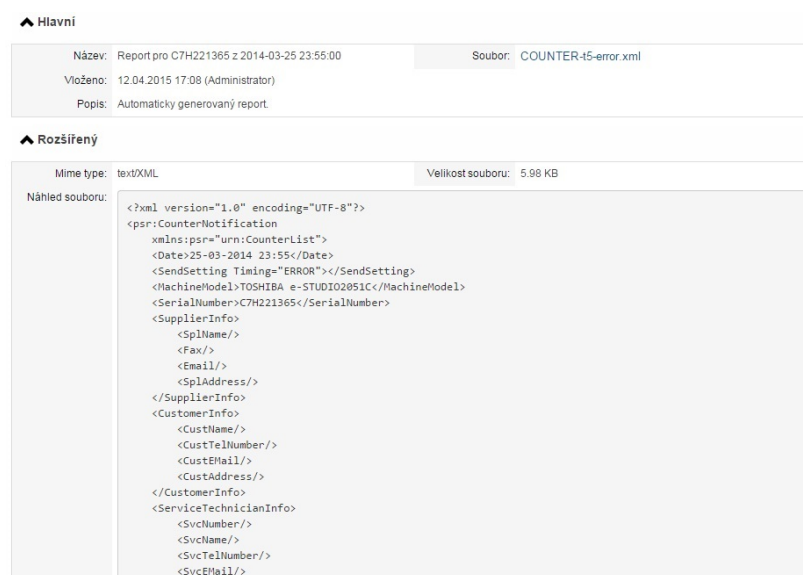
Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	varchar	Uživatelské pojmenování souboru.
description	text	varchar	Komentář k souboru.
filename	file	varchar	V modelu reprezentuje field pro vložení souboru. V databázi je originální název souboru.
file_mime_type	varchar	varchar	Mime type.
file_url	function	-	Funkce vracející odkaz na soubor ve tvaru <i>upload://id</i>

Při načtení *DetailView* je připojen javascriptový soubor *SpravceSouboruDetailView.js*, který obsahuje volání

```
$$SUGAR.util.doWhen("typeof $ != \'undefined\'", function () {..});
```

jež po splnění podmínky (první parametr) vykoná funkci (druhý parametr). Zde tedy po načtení *jQuery*²⁾ dojde k zobrazení náhledu souboru (změnou html elementů). Načítání je možné provést i na straně php, ovšem takto je načítání náhledů přesunuto až po zpracování stránky.

Zasílaná hlášení ve formátu xml často postrádají uživatelsky přívětivé formátování, proto je použit javascriptový doplněk *vkbeautify*³⁾, jež přidává chybějící konce řádků a odsazení. Orientace v takto upraveném souboru je mnohem snazší a rychlejší (příklad formátovaného výstupu viz obr. 8.2).



Obr. 8.2 Ukázka fomátovaného xml

8.2.2 Konvertory reportů

Slouží ke konverzi nahraných souborů (složka *upload*, ukládají se zde přílohy e-mailových zpráv po jejich stažení) na *ServisniZaznam* (v případě, že je soubor ve známém formátu). Konverzi je možno provádět za pomoci vytvoření třídy implementující dané rozhraní nebo zadáním *XPath*⁴⁾ cest pro xml soubory s hlášeními. Datový model je v tabulce 8.4.

Soubor *ObecnyKonvertor.php* obsahuje rozhraní, které musí implementovat jednotlivé konkrétní konvertory pro dané zařízení (pro php převod). Obsahuje funkci *konvertuj*, která obstarává samotný převod a funkce *vrat_XY*, které vrací *XY hodnotu* nalezenou v souboru (např. *vrat_seriove_cislo* vrací *seriové číslo*). Konkrétní konvertory jsou umístěny ve složce

²⁾<http://jquery.com>

³⁾<https://github.com/vkiriyukhin/vkBeautify>

⁴⁾XPath (XML Path Language) je počítačový jazyk, pomocí kterého lze adresovat části XML dokumentu.

Tab. 8.4 Model *KonvertoryReportu*, šablona *basic*

Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	varchar	Uživatelské pojmenování konvertoru.
konvertor_type	enum	varchar	Typ konvertoru - php soubor/xpath.
konvertor_name	enum	varchar	Enum se všemi konvertory (konvertor_type=php).
xpath_to_*	varchar	varchar	Jednotlivé xpath cesty v xml souboru.

modules/KonvertoryReportu/konvertory/

a jejich název (bez přípony) musí odpovídat třídě v nich obsažené.

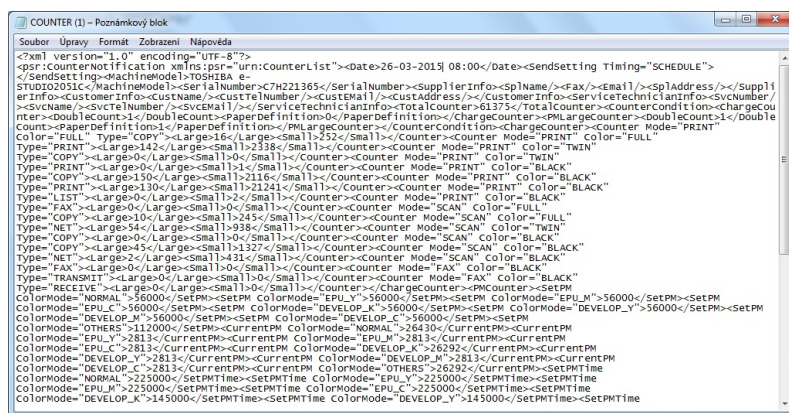
Vzhled *EditView* se musí přizpůsobit vybranému typu konvertoru. Pro tyto účely existuje soubor *KonvertoryReportu.js*, který tuto změnu obstarává. Jeho úkolem je jak změnit viditelné pole, tak je přidat do validace.

Konverzi nespouští uživatel samotný, ale jejím účelem je kontrolovat příchozí e-mailové zprávy a v jejich přílohách se pokoušet nalézt zaslané hlášení. Jelikož již existuje v plánovači úloha, která kontroluje poštovní schránky a stahuje nové zprávy, je jednou z možností upravit tento úkol a např. po stažení danou zprávu nejen uložit, ale i zkontrolovat. To by však vyžadovalo úpravu, která by s největší pravděpodobností nebyla upgrade-safe. Druhou možností je vytvoření vlastního úkolu - zde je však nutnost sladit tyto úkoly tak, aby šly po sobě (uživatel může změnit) a také evidence již zkontrolovaných zpráv. Pro implementování je zvolena třetí možnost, jež spočívá ve vytvoření *hook*-u po uložení zprávy (*after_save*). Hook je zaregistrován v systému následující úpravou souboru *custom/modules/Emails/logic_hooks.php*:

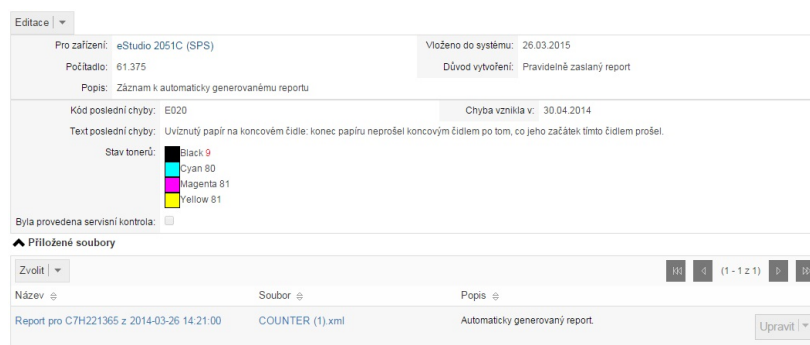
```
$hook_array['after_save'][] = Array(
    //Poradí ve kterém ma být zpracovan.
    20,
    //Textovy popis pro identifikaci.
    'after_save_email_convert',
    //Cesta k souboru ve kterém je implementovan.
    'custom/modules/Emails/logic_hooks_convert.php',
    //Trida.
    'ZpracovaniKonverze',
    //Volana metoda.
    'zpracuj_konverzi'
);
```

Podívejme se nyní jak činnost tohoto modulu vypadá v praxi. Konvertor dostane na vstupu soubor, jehož obsah je na obr. 8.3. V tomto případě se jedná o xml soubor, ale

obecně se může jednat o jakýkoliv jiný typ souboru. Konvertor se tento soubor pokusí převést a v případě úspěchu vytvoří příslušný servisní záznam, který je na obr. 8.4.



Obr. 8.3 Ukázkový soubor hlášení



Obr. 8.4 Převedené hlášení na servisní záznam

8.2.3 Šablony zařízení

Účelem je sjednotit prvky společné pro dané zařízení - výrobce, série, manuály, návody, konvertor a další. Manuály, návody a jim podobné mohou být libovolné soubory v libovolném počtu. Příslušný datový model je v tabulce 8.5.

8.2.4 Servisní záznamy

Uchovává servisní záznamy pro jednotlivá zařízení, ať již vytvořená ručně technikem (v takovém případě je požadováno vyplnění stavu celkového počítadla a popisu) či vzniklá konverzí (rozsah odpovídá rozhraní implementovanému v *ObecnýKonvertor*).

Stejně jako u ostatních, i zde je možno přikládat libovolný počet souborů ke každému záznamu (např. fotografie pořízené technikem přímo na místě - dokumentace stavu před servisním zásahem).

Tab. 8.5 Model *SablonyZarizeni*, šablona *basic*

Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	varchar	Uživatelské pojmenování šablony.
manufacturer	varchar	varchar	Název výrobce.
series	varchar	varchar	Název série.
service_after	int	int	Počet kopií po kterých je doporučováno provést servisní údržbu.
description	text	varchar	Poznámka k šabloně zařízení.

V tomto případě je model rozdělen na dvě části. V tabulce 8.6 vidíme položky, které jsou vyplňovány vždy (technik i hlášení). V druhé tabulce (viz 8.7) jsou ty, které jsou vyplňovány v případě, že tento záznam vznikl z hlášení.

Tab. 8.6 Model *ServisniZaznamy*, šablona *basic*

Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	-	Jednotlivé záznamy není třeba pojmenovávat.
reason	enum	varchar	Důvod vytvoření záznamu - technikem, pravidelný report, chyba či docházející spotřební materiál.
description	text	varchar	Popis chyby či servisního zásahu.
total_counter	int	int	Stav celkového počítadla v době pořízení záznamu.
service_check	bool	tinyint	Byla provedena servisní kontrola?

Dashlets

Modul *ServisniZaznamy* obsahuje také *Dashlety*, které je možno přirovnat k widgetům známým např. z mobilní platformy Android. Zjednodušeně se dají chápat jako přehled posledních či důležitých událostí svázaných s daným modulem, který se zobrazuje na domovské stránce systému.

V našem případě jde o dashlet zobrazující nevyřešená zasláná hlášení, které vznikla

Tab. 8.7 Model *ServisniZaznamy*, rozšíření pro případ zaslání hlášení

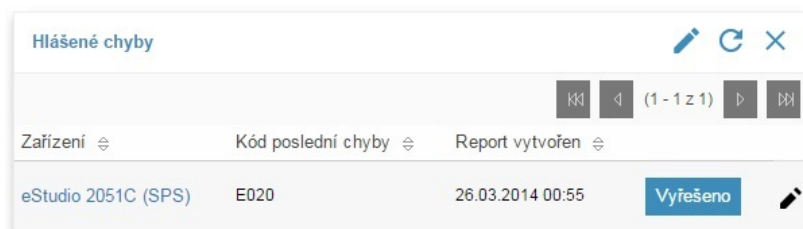
Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	-	Jednotlivé záznamy není třeba pojmenovávat.
date_report_created *	datetime	datetime	Datum a čas vytvoření hlášení (odlišný od data vytvoření záznamu v systému).
last_error_code*	varchar	varchar	Kód poslední chyby v hlášení (např. E220).
date_last_error_code*	datetime	datetime	Datum a čas vzniku poslední chyby v hlášení.
last_error_text*	varchar	-	Odpovídající text ke kódu chyby.
remaining_levels*	varchar	varchar	Zbývající stav tonerů či jiného spotřebního materiálu (př.: "Black": 9, "Cyan": 80, "Magenta": 81, "Yellow": 81).
solved*	bool	tinyint	Je chyba vyřešena? Jestliže jde o hlášení, které bylo inicializováno chybou, technik označí v případě vyřešení.

na základě chyby (*ZaslaneChybyDashlet*), poslední zaslání hlášení (*NoveZaznamyDashlet*) a přehled zařízení u kterých se blíží pravidelný servis (*BrzkyServisDashlet*).

Všechny jsou umístěny ve složce *modules/ServisniZaznamy/Dashlets/NazevDashletu* a jejich základ tvoří vždy tři soubory:

- *NazevDashlet.data.php*,
Definice sloupců, které má dashlet zobrazovat.
- *NazevDashlet.meta.php*,
Meta-data o dashletu - jak je identifikován v systému (název, ikona, ...)
- *NazevDashlet.php*.
Samotná logika dashletu.

Ačkoliv je jejich kód poměrně jednoduchý, lze na nich předvést variabilitu celého systému. *NoveZaznamyDashlet* využívá přepsání funkce *buildWhere*, jež umožňuje vytvořit vlastní část *where* řetězce, který je proveden v databázi. Zde je však nutno brát v podtaz možnost, že systém odstiňuje veškerou práci s databází a náš dotaz pak nemusí být kompatibilní se všemi.



Obr. 8.5 Ukázka dashletu *ZaslaneChybyDashlet*

ZaslaneChybyDashlet je v tomto směru konzervativnější - k zúžení množiny zobrazených dat využívá funkci *process*, ve které předává definici filtrů. Příkladem takové definice filtrů může být např.

```
$this->filters['solved'] = 'false';
```

Jak tento dashlet vypadá v praxi se můžeme podívat na obr. 8.5. Za zmínku zde stojí využití asynchronního volání pro označení hlášení jako vyřešeného za pomoci javascriptu (tlačítko „Vyřešeno“). Skript je přiložen nastavením *\$this->hasScript = true;* v třídě *ZaslaneChybyDashlet*, jež následně volá funkci *displayScript*. Obsahem je funkce, jež ve svém těle provádí samotné volání pomocí

```
YAHOO.util.Connect.asyncRequest('POST', 'index.php',
    {success: function() { /*zkraceno */ },
      failure: function() { /* zkraceno */ } },
    postData);
```

s následujícími předanými daty

```
postData = 'module=Home&action=CallMethodDashlet&method=markAsSolved'
          . '&id=' + id + '&id_solved=' + id_solved;
```

Action *CallMethodDashlet* je akce definovaná v systému, která zavolá dashlet dle *id*, *id_solved* je *id* konkrétního záznamu a *method* je název volané funkce.

Využití *YAHOO.util.Connect.asyncRequest* (součástí *YUI Library*⁵⁾) není nutné, stejně tak je možné využít pravděpodobně známější *jQuery* a jde v tomto případě spíše o preferenci pisatele kódu.

⁵⁾<http://yuilibary.com/>

Poslední, *BrzkyServisDashlet*, pak demonstruje možnost využít také tvorby vlastních sql dotazů, jež je provedeno v *process()*. Pro zběhlé v databázích to nesporně přináší větší možnosti a rychlost zpracování, avšak je o něco komplikovanější daná data následně zobrazit.

8.2.5 Správce zařízení

Modul obsahující informace o konkrétním zařízení (např. sériové číslo, vlastník,...) a spojující hlášení o něm. V budoucnu by měl také umět s těmito zařízeními komunikovat a vytvářet hlášení na základě dat z této komunikace. Model tohoto modulu je uveden v tabulce 8.8.

Tab. 8.8 Model *SprávceZařízení*, šablona *basic* společně s *assignable*

Název vlastnosti	Typ	Typ v databázi	Komentář
name	name	varchar	Uživatelský název zařízení.
serial_number	varchar	varchar	Sériové číslo.
own_address	bool	tinyint	Má zařízení vlastní adresu - je jiná než adresa klienta?
address_street, address_city, address_zip, address_country	varchar	varchar	Adresa - lokace zařízení, pokud má vlastní adresu.
reporting_email	varchar	varchar	E-mail ze kterého klientovo zařízení zasílá hlášení.
ip_address	varchar	varchar	IP adresa zařízení.
port	varchar	varchar	Port zařízení.
username	varchar	varchar	Uživatelské jméno pro přístup k zařízení.
password	varchar	varchar	Přístupové heslo.
total_counter	int	-	Nejvyšší hodnota počítadla v servisních záznamech (pro <i>List-View</i>).

Modul není ještě kompletní, proto je podpora pro komunikaci se zařízeními v experimentální fázi - byla přidána podpora pro *snmp*. Za tímto účelem je vytvořen pohled *view.snmp.php*. Tento view vyžaduje podporu v controlleru, konkrétně přidání funkce *action_snmp*, jež obstarává zobrazení view a funkce *action_snmpwalk*, která vrací výpis z *snmpwalkoid* formátovaný v tabulce pro následný výpis. Tato akce je vyvolána

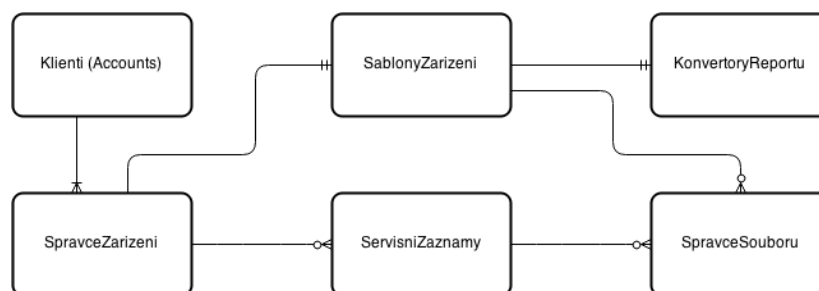
podobně jako u *ZaslaneChybyDashlet*.

Budoucí rozšíření

Po dokončení by modul měl umět analyzovat odpověď zařízení z *snmp* (v součinnosti s *KonvertoryReportu*) a umožnit volajícímu vytvořit *ServisniZaznam* bez nutnosti data ručně opisovat. Dále velká část zařízení poskytuje mnoho informací i přes webové rozhraní, které je možné využít stejně jako *snmp*. Tyto styly komunikace - tedy aktivně komunikaci inicializovat - měly poměrně malou prioritu řešení vzhledem k počtu klientů umožňující tento vzdálený přístup.

8.3 Vztahy mezi moduly (ER diagram)

Aby mohly moduly vzájemně koexistovat, musí o sobě nějakým způsobem vědět - mít definovaný vztah. Pro tento popis použijeme ER diagram. Jak vypadá jejich propojení se můžeme podívat na obr. 8.6, k značení je použita Martinova notace. Do diagramu je přidán již existující modul *Klienti* (Accounts), kterému jsou vytvořená zařízení přiřazována (zjevně jeden klient může mít libovolný počet zařízení). Kód samotný pro jejich vytvoření je analogický jako v 5.2.2 a proto jej zde není uveden.



Obr. 8.6 ER diagram systému

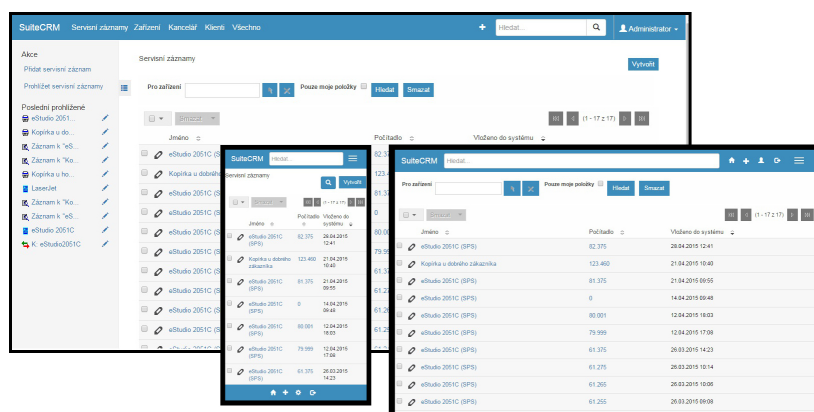
8.4 Mobilní přístup

Jedním z požadavků je také umožnit technikům přístup z mobilních zařízení, ten je však již splněn výběrem vhodného systému, jehož design je responzivní. Jak vypadá vzhled na různých typech zařízení je vyobrazeno na obr. 8.7.

V případě, že nechceme použít webovou aplikaci, je k dispozici *API* webových služeb integrovaných v systému postavených nad *REST* či *SOAP*. Následuje jednoduchá ukázka za pomoci *REST* a využití knihovny *cURL*, nejprve definice funkce *call*, která slouží k vyslání požadavku a přijetí odpovědi.

```

function call($method, $parameters, $url) {
    ob_start();
    $curl_request = curl_init();
  
```



Obr. 8.7 Vzhled na různých zařízeních

```

curl_setopt($curl_request, CURLOPT_URL, $url);
curl_setopt($curl_request, CURLOPT_POST, 1);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
            CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, 1);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
$jsonEncodedData = json_encode($parameters);
$post = array(
    "method" => $method,
    "input_type" => "JSON",
    "response_type" => "JSON",
    "rest_data" => $jsonEncodedData
);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
$result = curl_exec($curl_request);
curl_close($curl_request);
$result = explode("\r\n\r\n", $result, 2);
$response = json_decode($result[1]);
ob_end_flush();
return $response;
}

```

Prvním parametrem je název metody, kterou chceme volat, druhý obsahuje předané parametry a třetí je adresa služby, tedy např. *example.com/service/v4_1/rest.php* (*v4_1* je verze rozhraní). Příklad předaných parametrů a zavolání *call* pro přihlášení vypadá takto:

```

$login_parameters = array(
    "user_auth" => array(
        "user_name" => $username,
        "password" => md5($password),
        "version" => "1"
    ),
    "application_name" => "RestTest",
    "name_value_list" => array(),
);
$login_result = call("login", $login_parameters, $url);

```

Výsledkem takového volání je

```
stdClass Object
(
    [id] => dih3kl936rue17lkruc6fc01e0
    [module_name] => Users
    [name_value_list] => stdClass Object
    ( /* zkraceno */ )
),
```

přičemž *id* je identifikátorem session, které se použije při dalších voláních, např. při získání všech dostupných modulů - předané parametry tedy budou vypadat takto:

```
$get_available_modules_parameters = array(
    //Session id
    "session" => $login_result->id,
    //Module filter. Possible values are 'default', 'mobile', 'all'.
    "filter" => 'all');
```

Vzhledem k popularitě systému (resp. systému nad kterým je postaven) je možno využít i již existujících klientů v nejrůznějších prostředí, např. Java Client API for SugarCRM, Microsoft .NET Wrapper for SugarCRM SOAP API či SugarCRM REST API Client in node.js.

8.5 Současnost a možnosti budoucího rozšíření

Během následujících měsíců dojde k testovacímu nasazení tohoto produktu u zákazníka a připomínkovému období, během kterého budu sbírat vzniklé podměty, s nimiž bude následně vypořádáno. Vzhledem k plánovanému skokovému přechodu na nový systém a neexistenci digitalizovaného předchůdce odpadla nutnost řešit import starších dat.

Kromě samotné podrobnější analýzy sbíraných dat (kde je nejprve nutno zjistit jaká data za to stojí analyzovat) se díky možnostem systému samotného otvírají další způsoby jeho budoucího rozšíření, jež některé z nich budou jako příklady uvedeny v následujících odstavcích.

Novější zařízení umí kromě zasílaných reportů také dle nastavení vytvořit e-mailovou zprávu, která má sloužit jako objednávka docházejícího spotřebního materiálu. Poštovní schránka by na tyto zprávy také mohla být kontrolována, po jejich přijetí by mohl být zákazníkovi odeslán e-mail o přijaté objednávce a ve vhodné formě by byla předána informace obchodnímu oddělení. U zařízení, která tuto funkcionalitu nepodporují by mohlo dojít k odhadu, kdy daný materiál dojde a vytvořit záznam - např. v již existujícím modulu *Příležitosti*.

Jelikož je i v tomto případě uživatel systému pouze přeprodejce spotřebního materiálu, je i v jeho zájmu nabídnout zákazníkovi nejnižší cenu. Za tímto účelem by

mohl vzniknout modul, který eviduje tento spotřební materiál (i vzhledem ke konkrétnímu zařízení) a před jeho nákupem kontrolovat jeho cenu u jednotlivých dodavatelů (v případě, že dodavatel poskytuje vhodné rozhraní k svému systému).

Jednotlivým klientům také přijít vhod přehled o svém zařízení - nejen nákup materiálu, kdy může zjistit nevhodné využívání zařízení, ale také např. počtem neplánovaných servisních zásahů, jež také stojí nemalé částky. Těm sice nejde zcela předcházet, ale při pravidelných kontrolách bude toto číslo jistě menší. Tento přístup by mohl být realizován zadáním kódu, jež by byl klientovi sdělen při instalaci zařízení.

Dalším vhodným krokem je oddělení takto vzniklých modulů a vytvoření samostatného instalačního balíčku, který by bylo možno distribuovat zvlášť a instalovat za pomoci správce modulů obsaženého v SuiteCRM.

ZÁVĚR

Hlavním úkolem této bakalářské práce bylo nalézt vhodný informační systém, který by se dal přizpůsobit pro firmu zabývající se prodejem a servisem kancelářské techniky. Cílem tohoto systému je zpřehlednění a ucelení evidence servisních zásahů, tedy spojení papírové evidence s příchozími hlášenými, která zasílají sama zařízení e-mailem. Samotnému hledání systému musela předcházet analýza požadavků a vytvoření hodnotících kritérií. Při sestavování seznamu systémů k testování bylo na výběr ze dvou možností - vybrat pár nejznámějších adeptů a ty podstoupit podrobnému zkoumání nebo se pokusit obsáhnout co největší skupinu a zaměřit se jen na nejdůležitější části významné pro naše použití. Jak je již patrné z obsahu práce, byla zvolena druhá možnost. U této možnosti existují dvě rizika - při takovém počtu je pravděpodobnější přehlédnutí některé funkce systému (není možné se mu věnovat tak dlouhou dobu) a tvorby „nekonečného“ seznamu, jež spočívá v nacházení stále nových kandidátů při prověřování těch starých. Na počátku byla snaha tyto systémy také zařadit, s blížícím se koncem vyhrazeného času pro tento bod zadání byl však seznam uzavřen úplně. Volba „více systémů“ se na konec ukázala jako správná, protože vítězný systém SuiteCRM byl přibrán až mezi posledními.

Implementační část úkolu potvrdila často vznášený argument proti používání frameworků, tedy že čas potřebný k nastudování je u malých projektů vyšší než tvorba projektu bez něj. První modul tedy vznikl nemalou dobu, avšak při pochopení techniky bylo vytvoření základních koster dalších modulů jen otázkou minut.

Vzhledem k licenci, pod kterou je SuiteCRM distribuován, jsou všechny přidané zdrojové kódy šířeny také pod GNU/AGPLv3 a je možné je získat na <https://bitbucket.org/pavelbalcarek/suitecrmbeta>.

Protože šlo pouze o doplnění již existujícího systému, nebylo cílem vytvořit ani správcovskou (ta je součástí již existujícího), ani uživatelskou příručku (tu je možno v případě potřeby vytvořit dodatečně pro nové moduly). Dále také nebylo řešeno hledisko bezpečnosti, kdy veškerá komunikace probíhá mezi klientem (prohlížečem) a serverem v nešifrované formě. Proto bylo doporučeno ke zvážení využití protokolu *https*.

SEZNAM POUŽITÉ LITERATURY

- [1] ŠARMANOVÁ, Jana. *Databázové a informační systémy*. Ostrava: Vysoká škola báňská - Technická univerzita, 2007, 1 CD-R. ISBN 978-80-248-1499-5.
- [2] STEINER, František. *Komunikace v průmyslové organizaci - přednášky* [online]. [cit. 2015-05-09]. Dostupné z: http://home.zcu.cz/~steiner/KOPO/Prednasky/2_Prednaska%202.pdf
- [3] SODOMKA, Petr. *Informační systémy v podnikové praxi*. Vyd. 1. Brno: Computer Press, 2006, 351 s. ISBN 80-251-1200-4.
- [4] SODOMKA, Petr a Hana KLČOVÁ. Personální informační systém budoucnosti. [online]. [cit. 2015-05-11]. Dostupné z: <http://www.systemonline.cz/hrm-personalistika/personalni-informacni-system-budoucnosti.htm>
- [5] MAZLOVÁ, Tamara. PLM systémy pro řízení životního cyklu výrobku. [online]. [cit. 2015-05-11]. Dostupné z: <http://www.systemonline.cz/rizeni-vyroby/plm-systemy-pro-rizeni-zivotniho-cyklu-vyrobku.htm>
- [6] GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009, 496 s. Expert (Grada). ISBN 978-80-247-2615-1.
- [7] NÁPLAVA, Petr. *Tvorba informačních systémů - přednášky* [online]. [cit. 2015-05-09]. Dostupné z: <http://czm.fel.cvut.cz/vyuka/BI-TIS/Download/Prednaska06.pdf>
- [8] VASILEV, Julian. The change from ERP II to ERP III systems. 2013. DOI: 10.13140/2.1.5109.7609. Dostupné z: http://www.researchgate.net/publication/267448145_The_change_from_ERP_II_to_ERP_III_systems
- [9] AUTOR NEZNÁMÝ. The Best fit Customize ERP / MRP II Alternative in Thailand [online]. [cit. 15.5.2015]. Dostupné z: <http://www.vision4.co.th/>
- [10] ŠEDIVÁ, Zuzana. *Aplikace podnikové informatiky - studijní texty* [online]. [cit. 2015-05-11]. Dostupné z: http://www.vsem.cz/data/data/sis-texty/studijni-texty-bc/st_pis_api_sediva.pdf
- [11] DOHNAL, Jan. *Řízení vztahů se zákazníky: procesy, pracovníci, technologie*. 1. vyd. Praha: Grada, 2002, 161 s. ISBN 80-247-0401-3.

- [12] KLIMEŠ, Cyril. *Informační systémy 1: texty pro distanční studium* [online]. Ostrava: Ostravská univerzita v Ostravě, Přírodovědecká fakulta [cit. 2015-05-15]. Dostupné z: <http://www1.osu.cz/~zacek/infos1/skripta-old.pdf>
- [13] MOLNÁR, Zdeněk. *Moderní metody řízení informačních systémů*. 1. vyd. Praha: Grada, 1992, 347 s. ISBN 80-856-2307-2.
- [14] RYBIČKA, Jiří. *Informační systémy. Materiály k výuce*. [online]. [cit. 2015-05-13]. Dostupné z: <https://akela.mendelu.cz/rybicka/prez/infosyst.pdf>
- [15] DĚDKOVÁ, Jaroslava a Iveta HONZÁKOVÁ. *Základy marketingu*. Vyd. 1. Liberec: Technická univerzita, Hospodářská fakulta, 2001, c2000, 176 s. ISBN 80-708-3433-1.
- [16] ČÁPKA, David. *MVC Architektura*. [online]. [cit. 2015-05-15]. Dostupné z: <http://www.itnetwork.cz/mvc-architektura-navrhovy-vzor>
- [17] MERTIC, John. *The definitive guide to SugarCRM better business applications*. Berkeley, CA: Apress, 2009. ISBN 978-143-0224-402.
- [18] *JavaServer Pages Technology* [online]. [cit. 2015-05-13]. Dostupné z: <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>
- [19] HRONEK, Jiří. *Informační systémy - skripta*. [online]. Olomouc, 2007 [cit. 2015-05-13]. Dostupné z: <http://phoenix.inf.upol.cz/esf/ucebni/infoSys.pdf>
- [20] What Is SNMP?. *TechNet* [online]. [cit. 2015-05-14]. Dostupné z: <https://technet.microsoft.com/en-us/library/cc776379%28v=ws.10%29.aspx>
- [21] TOSHIBA. *TopAccess Guide: Multifunctional Digital Systems*. 2011.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AGPL	Affero General Public License
API	Application Programming Interface
APS	Advanced Planning and Scheduling
ASP	Active Server Pages
CPL	Common Public License
CRM	Customer Relationship Management
CRON	Systémový plánovač úloh
CRUD	Create, Read, Update, Delete
DFD	Date Flow Diagram
ER	Entity-Relationship
ERD	Entity-Relationship Diagram
ERP	Enterprise Resource Planning
GNU	GNU's Not Unix!
GPL	General Public License
GUI	Graphical User Interface
HRM	Human Resource Management
HTML	HyperText Markup Language
HW	Hardware
IS	Informační systém
LGPL	Lesser General Public License
MIT	Massachusetts Institute of Technology (License)
MPL	Mozilla Public License
MRP	Material Requirements Planning
MRPII	Manufacturing Resource Planning
MVC	Model-View-Controller
PHP	Hypertext Preprocessor, skriptovací programovací jazyk
PLM	Product Lifecycle Management
REST	Representational State Transfer
RPC	Remote Procedure Call
SaaS	Software as a Service
SCM	Supply Chain Management
SOAP	Simple Object Access Protocol
SRM	Supplier Relationship Management
STD	State Transition Diagram
SW	Software
UC	Use Case
XML	Extensible Markup Language, rozšiřitelný značkovací jazyk

SEZNAM OBRÁZKŮ

Obr. 1.1	Hierarchie IS ve firmě (zdroj: [6])	12
Obr. 1.2	HRM: Pokryté oblasti (zdroj: [4])	15
Obr. 1.3	Některé typické části ERP systémů (zdroj: [9])	16
Obr. 1.4	Příklad ER diagramu při tvorbě IS pro knihovnu (zdroj: [1])	17
Obr. 1.5	Diagram datových toků DFD (zdroj: [1])	18
Obr. 1.6	Příklad UC modelu pro jednání knihovny (zdroj: [12])	18
Obr. 1.7	Příklad STD	19
Obr. 1.8	Souběžné zavedení IS	20
Obr. 1.9	Pilotní zavádění IS	21
Obr. 1.10	Postupné zavádění IS	21
Obr. 1.11	Nárazové zavádění IS	21
Obr. 1.12	Vodopádový model životního cyklu vývoje IS (zdroj: [1])	22
Obr. 4.1	Rozhraní TOSHIBA TopAccess (zdroj: [21])	25
Obr. 5.1	Diagram MVC	28
Obr. 6.1	Příklad se záznamy o konkrétním zařízení	38
Obr. 6.2	UC celého systému	41
Obr. 6.3	UC správy souborů	42
Obr. 6.4	UC správy šablon	42
Obr. 6.5	UC analýzy e-mailových zpráv	42
Obr. 7.1	Uživatelské rozhraní Adempiere (zdroj: demo.testadempiere.com)	44
Obr. 7.2	Uživatelské rozhraní Aktiv.io (zdroj: aktiv.io)	44
Obr. 7.3	Nastavení oprávnění v SplendidCRM (zdroj: demo.splendidcrm.com)	51
Obr. 8.1	Uživatelské rozhraní modulu Pošta	57
Obr. 8.2	Ukázka fomátovaného xml	60
Obr. 8.3	Ukázkový soubor hlášení	62
Obr. 8.4	Převedené hlášení na servisní záznam	62
Obr. 8.5	Ukázka dashletu <i>ZaslaneChybyDashlet</i>	65
Obr. 8.6	ER diagram systému	67
Obr. 8.7	Vzhled na různých zařízeních	68

SEZNAM TABULEK

Tab. 1.1	Pojetí přístupů k zákazníkům (zdroj: [11])	14
Tab. 3.1	SWOT analýza	24
Tab. 5.1	Vlastnosti SuiteCRM	27
Tab. 7.1	SWOT analýza pro <i>CiviCRM</i>	55
Tab. 7.2	SWOT analýza pro <i>FreeNetIS</i>	55
Tab. 7.3	SWOT analýza pro <i>SugarCRM</i>	55
Tab. 7.4	SWOT analýza pro <i>SuiteCRM</i>	55
Tab. 8.1	Šablona objektů <i>basic</i>	58
Tab. 8.2	Šablona rozšíření <i>assignable</i>	59
Tab. 8.3	Model <i>SpravceSouboru</i> , šablona <i>basic</i>	59
Tab. 8.4	Model <i>KonvertoryReportu</i> , šablona <i>basic</i>	61
Tab. 8.5	Model <i>SablonyZarizeni</i> , šablona <i>basic</i>	63
Tab. 8.6	Model <i>ServisniZaznamy</i> , šablona <i>basic</i>	63
Tab. 8.7	Model <i>ServisniZaznamy</i> , rozšíření pro případ zaslaného hlášení	64
Tab. 8.8	Model <i>SprávceZařízení</i> , šablona <i>basic</i> společně s <i>assignable</i>	66
Tab. 1.1	Hodnocení systémů, část první	78
Tab. 2.1	Hodnocení systémů, část druhá	79

SEZNAM PŘÍLOH

- P I. Tabulkový přehled hodnocených systémů (část první)
- P II. Tabulkový přehled hodnocených systémů (část druhá)

PŘÍLOHA P I. TABULKOVÝ PŘEHLED HODNOCENÝCH SYSTÉMŮ (ČÁST PRVNÍ)

Tab. 1.1 Hodnocení systémů, část první

	Adempiere	CiviCRM	Coevery	Epesi	EspoCRM	FatFreeCRM	FreeNetIS	FrontAccounting	Group-Office	iDempiere	JFire
Obecné požadavky											
Licence	GPL	AGPL	CPL	MIT	GPL	MIT	GPL	GPL	AGPL	GPL	LGPL
Platforma	PHP	PHP	ASP	PHP	PHP	Ruby	PHP	PHP	PHP	Java	Java
Aktualizace	8.12.2014	4.2.2015	27.1.2014	7.1.2015	29.1.2015	22.1.2015	29.1.2015	29.9.2014	12.11.2014	31.10.2014	25.9.2009
Programátorské hledisko											
Podrobná vývoj. dok.	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✗
Komunita	✗	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗
Repozitář	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Pokrytí/vhodnost funkčních požadavků											
Uživatelé a oprávnění	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Správa klientů	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Info o klientovi	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Práce s mapy	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
Možnost lokalizace	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Správa zařízení	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Komunikace se zařízením	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Plánování úloh (CRON,...)	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗
Modulární	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Email	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗
Pokrytí/vhodnost nefunkčních požadavků											
Resp. design	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗
API	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗
Snadnost použití	3	2	2	3	1	3	2	3	1	2	3
Změny šablon v gui	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
Oficiální market	✓	✓	✗	✓	✗	✓	✗	✓	✓	✓	✗
Neoficiální rozšíření	✓	✓	✗	✓	✗	✓	✗	✓	✓	✓	✗

PŘÍLOHA P II. TABULKOVÝ PŘEHLED HODNOCENÝCH SYSTÉMŮ (ČÁST DRUHÁ)

Tab. 2.1 Hodnocení systémů, část druhá

	Odoo	Opentaps	Phreedom	SplendidCRM	SQL-Ledger N.	SugarCRM	SuiteCRM	Vtiger	WebERP	Zurmo
Obecné požadavky										
Licence	AGPL	AGPL	GPL	AGPL	GPL	AGPL	AGPL	MPL	GPL	AGPL
Platforma	Python	Java	PHP	ASP	Perl	PHP	PHP	PHP	PHP	PHP
Aktualizace	17.1.2015	9.4.2014	13.11.2014	-	17.10.2014	15.12.2014	19.1.2015	28.1.2015	27.1.2015	24.12.2014
Programátorské hledisko										
Podrobná vývoj. dok.	✓	✓	✗	✓	✗	✓	✓	✓	✗	✓
Komunita	✓	✗	✗	✓	✗	✓	✓	✓	✓	✓
Repozitář	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓
Pokrytí/vhodnost funkčních požadavků										
Uživatelé a oprávnění	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Správa klientů	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Info o klientovi	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Práce s mapy	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
Možnost lokalizace	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Správa zařízení	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Komunikace se zařízením	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Plánování úloh (CRON,...)	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗
Modulární	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Email	✓	✗	✗	✓	✗	✓	✓	✓	✓	✓
Pokrytí/vhodnost nefunkčních požadavků										
Resp. design	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓
API	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓
Snadnost použití	1	2	4	3	3	3	2	1	2	1
Změny šablon v gui	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Oficiální market	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
Neoficiální rozšíření	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓