Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Návrh a implementace aplikace typu CRM pro konkrétní firmu

Bc. Miloš Faltýnek

Diplomová práce

2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Miloš FALTÝNEK, DiS.**
Osobní číslo:	**I08355**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a implementace aplikace typu CRM pro konkrétní firmu**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Hlavním cílem diplomové práce je návrh a implementace aplikace pro řízení a kontrolu vybraných částí obchodních procesů ve firmě CSTechnologies. V úvodní teoretické části bude shrnut význam aplikace pro firemní marketing a vytvořen přehled základních požadavků na aplikaci. Na základě těchto požadavků bude definována základní charakteristika aplikace, její základní funkce a postup implementace. Součástí implementace bude analýza požadavků prostřednictvím UML diagramů pomocí softwarového nástroje Enterprise Architect, návrh datových tříd a jejich grafické vizualizace. Hlavním prvkem aplikace bude přehledný kalendář pro plánování a kontrolu událostí obchodního procesu. K tvorbě aplikace budou použity komponenty Devexpress. Nedílnou součástí bude i tvorba vlastních komponent a rozšíření. Aplikace bude naprogramovaná v jazyce C# (vývojové prostředí Microsoft Visual Studio) s podporou objektově-relačního mapovacího nástroje NHibernate pro .NET. Databázovou vrstvu zpřístupní systém řízení báze dat MS SQL Server. Při tvorbě aplikace je třeba důsledně dodržet koncepci CSTechnologies pro aplikaci klient-server. Rozhraní aplikace budou postavena na dosud používaných platformách firmy Centaur, Icarus a Zeus.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. GÁLA, Libor, POUR, Jan, TOMAN, Prokop. Podniková informatika. Mgr. Petr Mušálek. 1. vyd. Praha: Grada Publishing, 2006. 484 s. Management v informační společnosti. ISBN 80-247-1278-4. 2. ROBINSON, Simon. C# Programujeme profesionálně. 1. vydání. Brno: Computer Press, 2003. 1130 s. ISBN 80-251-0085-5. 3. CHLEBOVSKÝ, V. CRM - Řízení vztahů se zákazníky. 1. vydání. Praha: Computer Press, a. s., 2005. ISBN 80-251-0798-1.

Vedoucí diplomové práce: **Ing. Jana Holá, Ph.D.**
Katedra zdravotnické informatiky

Datum zadání diplomové práce: **30. října 2009**

Termín odevzdání diplomové práce: **21. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

doc. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 10. listopadu 2009

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Nesouhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 5. 2010

Bc. Miloš Faltýnek

ANOTACE

Diplomová práce popisuje implementaci systému CRM s názvem CSManager pro firmu CS Technologies. Pro databázové schéma byla použita relační databáze Microsoft SQL Server, nad kterou byl navržen a naprogramován systém v jazyce C#. Systém dále využívá technologie NHibernate a dodržuje standardy vývoje systémů ve firmě. Nedílnou součástí implementace byla i migrace dat.

KLÍČOVÁ SLOVA

crm, systém, implementace, databáze, c#, nhibernate

TITLE

Design and implementation of CRM application for a particular firm

ANNOTATION

Diploma paper describes implementing of CRM system with name CSManager for CS Technologies firm. For database schema was used Microsoft SQL Server database, on which was designed and programmed application in C# language. The system also uses technologies such as NHibernate and it adheres to standards of system development in the firm. An integral part of implementation was also migration of data.

KEYWORDS

crm, system, implementation, database, c#, nhibernate

Obsah

Seznam obrázků

Seznam zkratek a specifických pojmů

Termín	Význam
ADO.NET	Komponenty pro přístup k datům v prostředí .NET Framework.
API	Application Programming Interface – označuje rozhraní pro programování aplikací.
ASP.NET	Součást .NET Framework pro tvorbu webových aplikací a služeb.
CRM	Customer Relationship Management – řízení vztahů se zákazníky.
CSV	Comma Separated Values – je jednoduchý souborový formát určený pro výměnu tabulkových dat.
DLL	Dynamic Link Library – soubor knihovny sestavení programu.
DML	Data Manipulation Language – jazyk pro manipulaci s daty, určující jednotlivé operace s daty v databázi.
GUI	Graphical User Interface – grafické uživatelské rozhraní.
HQL	Hibernate Query Language – dotazovací jazyk technologie NHibernate používaný pro práci s daty.
HTML	Hyper Text Markup Language – jazyk pro vytvoření statických internetových stránek.
HTTP	Hyper Text Transfer Protocol – internetový protokol.
IDE	Integrated Development Environment – integrované vývojové prostředí.
IIS	Internet Information Server – webový server pro internetové informační služby.
LINQ	Language Integrated Query – jazyk integrovaný v .NET Framework pro dotazování nad jakýmikoliv daty.
.NET	Název pro soubor technologií v softwarových produktech od společnosti Microsoft, které tvoří celou platformu.
ODBC	Open Database Connectivity – standard definující rozhraní mezi databázovým serverem a klientem (aplikací).
RDBMS	Relational Database Management System – systém pro správu relační báze dat.
SQL	Structured Query Language – strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.
SOAP	Simple Object Access Protocol – protokol umožňující přenášet XML dokumenty.
TCP	Transmission Control Protocol – spojově orientovaný protokol pro přenos dat se spolehlivým doručováním.
TXT	Označení souboru uchovávající prostý text.
UML	Unified Model Language – standardní jazyk pro vytváření modelů obchodních a technických systémů.
UP	Unified Process – metodika vývoje software.
URL	Unique Resource Location – unikátní identifikátor označující umístění zdroje v rámci internetu.
WCF	Windows Communication Foundation – systém přenosu zpráv mezi programy pro podporu lokální i vzdálené spolupráce.
WPF	Windows Presentation Foundation – je grafický subsystém pro vykreslování uživatelského rozhraní ve Windows aplikacích.
WWW	World Wide Web – v překladu „celosvětová pavučina“, je označení pro aplikace internetového protokolu HTTP.
XLS	Přípona souborů vytvořených v aplikaci Microsoft Excel.
XML	eXtensible Markup Language – standardní formát určený pro výměnu dat mezi aplikacemi.

Úvod

Zadavatel CS Technologies s. r. o.

Zadavatel, společnost CS Technologies se zabývá tvorbou www stránek, e-shopů a web designem. Na správu webových prezentací nebo aplikací nabízí vlastní administrační systém. Jádro těchto vytvořených internetových aplikací je generováno v technologii ASP.NET. Veškerý software vyvíjený ve firmě používá zvláštní frameworks nazvané Centaur, Icarus a Zeus[1], poskytující specifické funkce pro dané rozhraní.

Cíl práce

Hlavním cílem diplomové práce je návrh a implementace aplikace typu CRM na základě stávajícího informačního systému a požadavků zadavatele. Úkolem je analyzovat a navrhnout koncepční informační systém pro podporu vybraných obchodních procesů souvisejících s marketingem firmy. Součástí práce je migrace stávajících dat do nového systému a jejich synchronizace. Předmětem této diplomové práce je programování v jazyce C#. Aplikace využívá nejmodernější technologie persistence dat NHibernate. Při vývoji je třeba důsledně dodržet standardy CS Technologies pro aplikace klient-server.

[1] Centaur, Icarus, Zeus frameworks vytvořené firmou CS Technologies programované v jazyce C# pro prostředí .NET Framework.

1 CRM systémy

Customer Relation Management neboli řízení vztahů se zákazníky je systém pro procesy evidence, správy a další využívání informací o zákazníkovi. Tyto informace se efektivně využívají podle požadavků ke spokojenosti zákazníka. Jde o součást marketingové strategie orientované na zákazníka, protože právě ten přináší peníze. Důležitým aspektem CRM systému jsou data o zákazníkovi dostupná v celé infrastruktuře podniku z dlouhodobého hlediska. Vzhledem k nasycenému trhu a silné konkurenci narůstá tlak na schopnosti firmy komunikovat se zákazníkem. Tato komunikace se uchovává v databázi pro další zpracování a analýzu. Tím je podpořena koordinace komunikace v podniku tak, aby nedocházelo k opakovaným dotazům na různé skutečnosti (1).

Základní charakteristiky hlavních součástí systému CRM:

Operační – část zahrnuje softwarové aplikace, které řeší operativní záležitosti a kontakty v kooperaci se zákazníkem. Tato oblast zahrnuje např. funkce podporující řízení obchodních kontaktů, řízení jednotlivých obchodních případů, vytváření marketingového plánu a kampaní, sledování konkurence, funkce pro specifikaci požadavků na zákaznický servis, jejich vyhodnocování apod.

Kooperační – část rozšiřuje základní kanály pro komunikaci se zákazníkem (pošta, fax, telefon, osobní schůzka) o internet, elektronickou poštu a zejména o interaktivní aplikace na webu. To vše je koordinováno a řízeno pomocí kontaktních center, což jsou aplikace a technologie v rámci CRM založené na centrálním přístupu zákazníka k firmě. V rámci kontaktních center se uchovávají a pravidelně aktualizují informace o jakémkoli kontaktu se zákazníky (vyřizování stížností, informace o předání a druhu nabídky, odeslání marketingových materiálů, informace o podpisu kontraktu). Kontaktní centra tedy obstarávají funkce, jako jsou podpora komunikace se zákazníkem, založená na integraci různých komunikačních kanálů, automatické interaktivní hlasové odpovědi, zpracování elektronické pošty, hlasová komunikace přes web, vedení marketingových kampaní a další.

Analytická – část již zahrnuje agregace a aplikace znalostí o zákazníkovi a zajišťuje funkce, např. segmentace zákazníků, analýzy marketingových kampaní, predikce chování zákazníků a další. Analytická část CRM se obvykle realizuje tak,

že využívá zákaznická data získaná z operačního a kooperačního CRM, případně z dalších aplikací. Pro zpracování analýz využívá technologií a aplikací BI (Business Intelligence). Taková kombinace CRM a BI se označuje jako CI (Customer Intelligence) a většinou se již chápe jako synonymum pro analytické CRM. CI v tomto směru představuje funkcionalitu zaměřenou na poznání zákazníka, jeho hodnoty, preferencí, rizikovosti nebo pravděpodobnosti odchodu ke konkurenci (2, s. 211).

1.1 Hlavní funkce CRM

Systém CRM by měl poskytovat funkce potřebné pro vytvoření a udržení dobrého přehledu o zákaznících, a to od prvního kontaktu přes prodej až po následnou péči o zákazníka, a také funkce orientované na správu marketingových kampaní, řízení obchodních aktivit a řešení servisních případů. Jednou z podstatných součástí systému CRM je podpora řízení a poskytování služeb zákazníkům, např. záručních i pozáručních služeb spojených s dodávanými produkty (2, s. 212).

CRM systém přináší firmám i jejich zákazníkům specifické možnosti:

- Obchodníci a další pracovníci firmy získávají a sdílejí detailní informace o zákaznících, o jejich požadavcích a potřebách, informace o obchodních příležitostech či o stavu a průběhu jednotlivých obchodních případů; na základě těchto informací jsou schopni uplatnit individuální přístup k jednotlivým zákazníkům.
- Díky správě obchodních příležitostí a jejich automatizovanému zpracování, řízení prodejních procesů a sledování konkurence se dosahuje zvýšení úspěšnosti prodeje.
- Komplexní informace o potenciálních i realizovaných obchodních aktivitách vytvářejí základ pro analýzy prodejní výkonnosti a tím poskytují nástroje pro její zvyšování. Umožňují vytvářet prognózy prodeje, sledovat obchodní aktivity a jejich efektivitu (2, s. 213).

2 Postup implementace systému CSManager

Tato kapitola popisuje postup a použité nástroje při tvorbě CRM systému pro firmu CS Technologies, který byl nazván CSManager.

Vývoj informačního systému probíhal v následujících krocích:

- Zjištění požadavků zadavatele a porovnání aktuálního stavu.
- Analýza požadavků.
- Obecná charakteristika funkcí a návrh řešení.
- Návrh databáze.
- Migrace dat.
- Implementace systému.
- Zaškolení uživatelů, testování systému a ostrý provoz.

2.1 Vývojové prostředky použité při tvorbě informačního systému

Technologie vývoje informačního systému byly předem stanoveny zadavatelem a stejně tak i vývojové prostředky. Vedoucí vývojového oddělení mě stručně seznámil s postupem vývoje, funkcemi a integrací software ve firmě CS Technologies.

Pro vývoj samotného systému CSManager byly využity technologie:

- Objektově orientovaný programovací jazyk C# určený k programování v prostředí .NET.
- NHibernate řešení objektově-relačního mapování pro .NET.
- Vizuální komponenty firmy DevExpress pro WinForms prostředí .NET.

Při vývoji systému CSManager byly konkrétně použity nástroje:

- SQL Manager 2008 for SQL Server (3.2.0.1): nástroj pro vývoj a správu Microsoft SQL Server databází.

- Microsoft Visual Studio 2008: vývojové prostředí pro psaní programů technologií .NET.
- Enterprise Architect verze 7.1: nástroj pro návrh a modelování systémů pomocí jazyka UML.
- TortoiseSVN verze 1.6.3: systém pro správu zdrojových kódů projektu.
- ToDoList 5.9.2: projektový organizátor úkolů.

3 Analýza projektu CSManager

Cílem projektu je vytvořit GUI[2] aplikaci klient-server, která má být dostupná nejen z vnitropodnikové sítě, ale i ze sítě internet. Firma CS Technologies pro své aplikace využívá serverových služeb u profesionálních poskytovatelů. V aplikaci založené na architektuře klient-server je výkonná část systému spolu s databází uložena na pronajatém serveru. Odlehčenou klientskou část aplikace má uživatel nainstalovanou na svém osobním počítači či notebooku. Klientská a serverová část spolu komunikují přes internetovou síť. Toto řešení umožňuje snadnější správu a sdílení informací více uživateli a přístup uživatele prakticky odkudkoliv.

CRM systém by měl spravovat odpovědnosti jednotlivých pracovníků za trvalou spokojenost zákazníka v dlouhodobém časovém horizontu. Zahájením marketingové činnosti pracovník obchodního oddělení vyhledává obchodní příležitosti pomocí dostupných informačních kanálů. Obchodní příležitost je prvním bodem spuštění obchodního procesu a od něj se vše odvíjí. Na úrovni získávání obchodních příležitostí by měla probíhat kontrola, zda obchodní příležitost potažmo zákazníka již nekontaktoval jiný obchodník. Pokud je obchodní příležitost volná, obchodník má za úkol oslovit kontakt obchodní příležitosti a nabídnout odpovídající produkt. Cílem obchodníka je domluvit si s potencionálním zákazníkem schůzku. Na schůzce obchodník přesvědčuje zákazníka o koupi produktu. Pokud je obchodník v nějaké fázi nabídky produktu neúspěšný, druhý obchodník může naopak uspět. Tím je na zákazníka vyvíjen určitý tlak a minimalizuje se možnost odmítnutí produktu. Celý proces komunikace je uložený v databázi a poskytuje tak přehled o interakci obchodníka s potencionálním zákazníkem. Firma pak neztratí přehled o stavu zakázky ani v případě, kdy je obchodník z nějakého důvodu v systému neaktivní. Určitým způsobem je také chráněno zcizení databáze kontaktů.

3.1 Stav při zahájení projektu

Každý obchodník spravuje svoji bázi kontaktů v tabulce MS Excel. Data jsou nekonsistentní a často chybná. Tento systém neumožňuje unikátní oslovení obchodní

[2]GUI (Graphical User Interface) – grafické uživatelské rozhraní.

příležitosti jedním zástupcem firmy. Zaměstnanci pro plánování událostí a úkolů používají odděleně aplikaci Alive Organizer. Což je výkonný systém, umožňující spravovat informace o zákazníkovi, plánovat úkoly a udržovat adresáře. Avšak tyto informace nijak nesdílí s ostatními uživateli pro kontrolu a plánování.

3.1.1 Nedostatky stávajícího řešení

Stávající kombinace řešení má tyto hlavní nedostatky, které byly podnětem pro provozovatele k zadání této práce.

- Informace nejsou přístupné přes internet. Chybí vzdálená správa a sdílení informací.
- Řešení neumožňuje sdílení vnitropodnikových informací mezi pracovníky. Všechny kontakty a informace o zákaznících jsou roztroušené. Je žádoucí, aby bylo možné analyzovat potřeby zákazníka a efektivně je uspokojit.
- Systém práce neposkytuje přehled o stavu a vedení jednotlivých obchodních příležitostí. Není možné kontrolovat a řídit jednotlivé aktivity obchodního procesu. Chybí sledování historie.
- Existuje vysoké riziko ztráty nebo odcizení kontaktů.
- Firma má vysoké náklady na údržbu decentralizované evidence obchodních příležitostí.
- Proces získávání a zpracování dat není uživatelsky přívětivý. Prostředky vykazují nízkou míru automatizace a chybí validace dat.
- Současný stav poskytuje nepřehledné výstupy, špatně čitelné informace, nejednotný formát dat a chybovost.

3.1.2 Výhody nasazení nového systému CRM

Nový systém by měl eliminovat všechny nedostatky stávajícího stavu a měl by zajistit informační toky zejména pro procesy:

- Správa a dostupnost kontaktů.
- Řízení a aktuální přehled vztahů se zákazníky.
- Integrita a sjednocení dat o zákaznících.

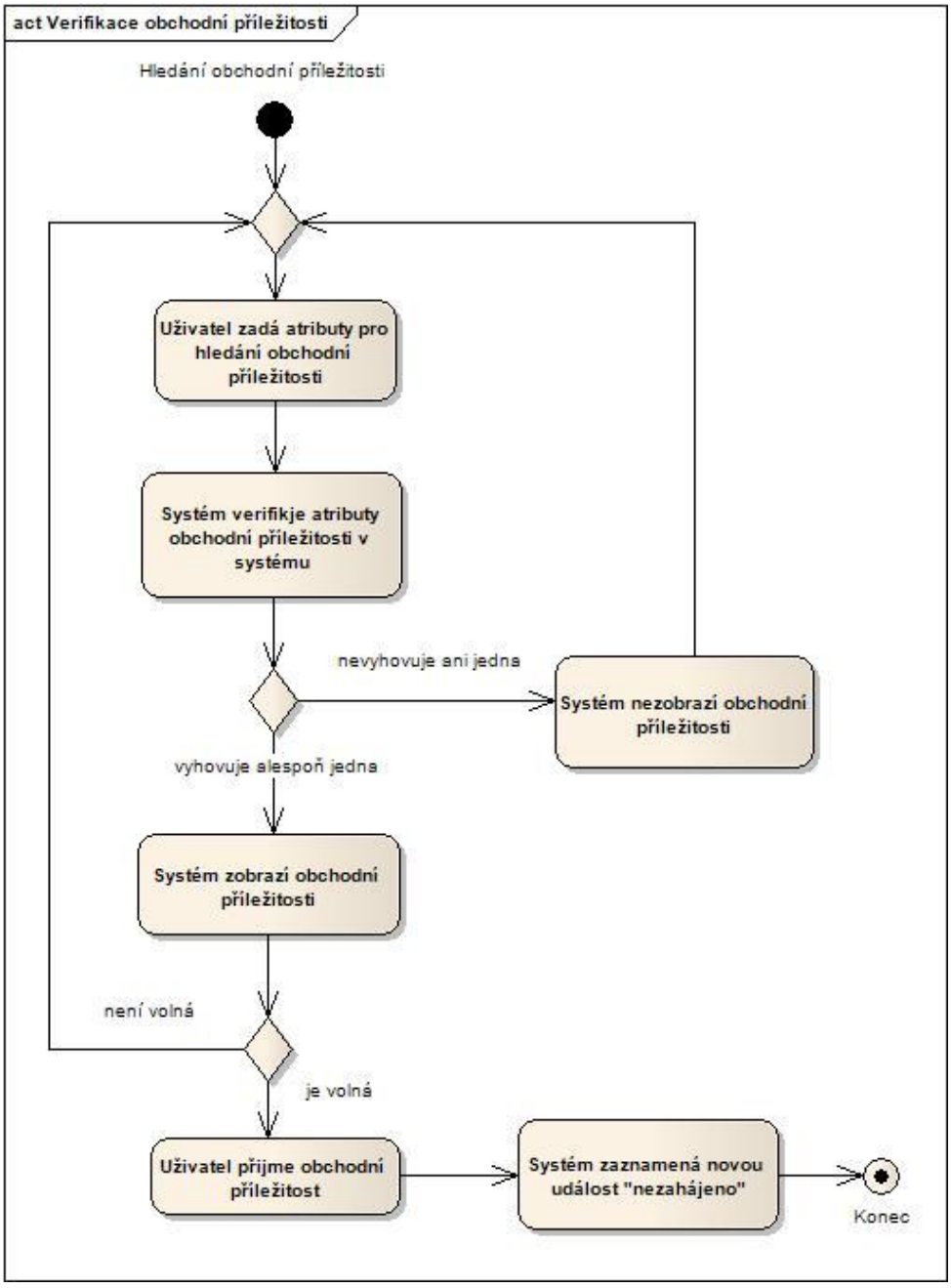
- Plánování a řízení obchodní činnosti podniku.
- Snadné hodnocení efektivity pracovníků.
- Marketing se zaměřením na potřeby zákazníka.
- Podpora zákazníka a zákaznických služeb.

3.2 Popis hlavních aktivit obchodního procesu pro nový systém CRM

Jak již bylo napsáno v úvodu této kapitoly, vyhledání a zaevidování obchodní příležitosti je hlavním impulsem spuštění obchodního procesu ve firmě CS Technologies. Pro snadné rozlišení, v jaké fázi obchodního procesu se obchodní příležitost nachází, je zaveden pojem stav obchodní příležitosti. Při vyplnění údajů a vytvoření obchodní příležitosti v systému má obchodní příležitost stav „nezahájeno“. Práci na obchodní příležitosti vyznačuje stav s názvem „pokračuje“. Stav „dokončeno“ vypovídá o úplném odmítnutí či nezájmu zákazníka o produkty firmy. Nejpodstatnější ukazatel stavu obchodní příležitosti je „objednávka“. Jednotlivé pracovní úkony vedoucí k objednávce se zaznamenávají pomocí události. Události jsou nejčastěji typu „e-mail“, „telefon“ nebo „schůzka“. Poslední událost označuje stav obchodní příležitosti.

3.2.1 Proces vyhledávání obchodních příležitostí

Proces spočívá v hledání kontaktů z dostupných informačních zdrojů (internet, zlaté stránky, obchodní rejstřík, živnostenský rejstřík). Zahrnuje zadávání kontaktů do systému a verifikaci, zda již nejsou vlastněny jiným obchodníkem. Při analýze nového informačního systému jsem pro snadné pochopení pomocí diagramu aktivit namodeloval některé procesy. Diagram aktivit na následujícím obrázku (Obr. 1) znázorňuje rozšířené aktivity, spojené s ověřením při zadávání nové příležitosti v novém projektovaném systému.

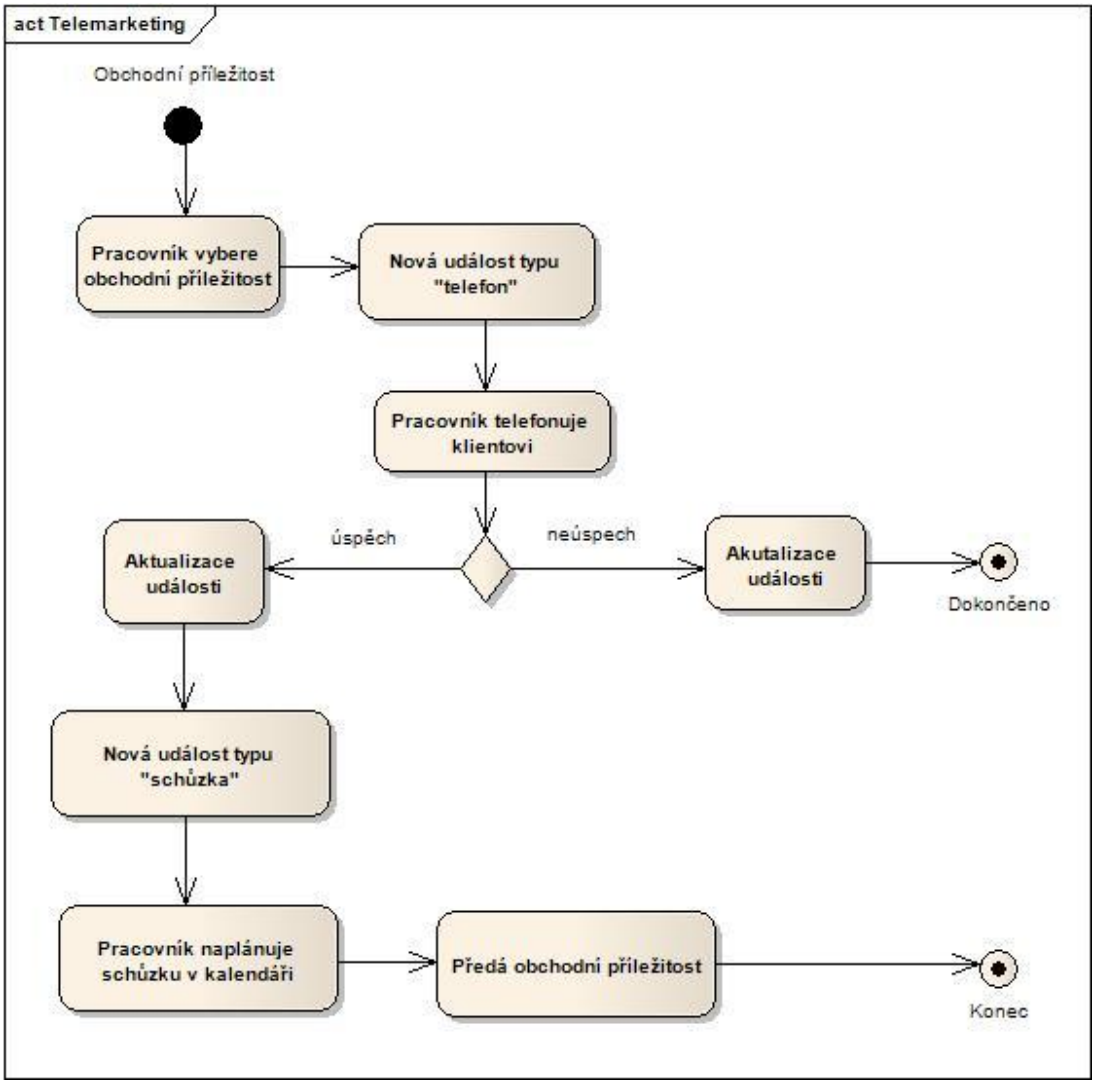


Obr. 1: Proces verifikace obchodních příležitostí v systému CRM

3.2.2 Proces telemarketingu obchodní příležitosti

Pracovník telemarketingu v CS Technologies spravuje zpravidla obchodní příležitosti ve stavu „nezahájeno“, které po telefonickém sjednání schůzky předá obchodním zástupcům. Rozdělování obchodních schůzek probíhá nejlépe podle místa konání schůzky tak, aby měl obchodník schůzky v jednom městě nebo jeho okolí. Obchodní zástupce může také předat své vybrané obchodní příležitosti pracovníkovi telemarketingu a využít tak jeho služby. Telemarketing šetří čas obchodním zástupcům, kteří nemusí neustále vyhledávat nové kontakty a domlouvat

setkání s potencionálním klientem. Místo toho se mohou více věnovat samotným schůzkám a nabízením produktu face-to-face. Následující diagram aktivit (Obr. 2), graficky popisuje činnosti pracovníka telemarketingu při sjednávání schůzky obchodnímu zástupci.



Obr. 2: Proces telemarketingu v systému CRM

3.2.3 Analýza obchodní příležitosti a návrh komplexního řešení

Pracovník obchodního oddělení analyzuje novou obchodní příležitost a předem navrhuje strategii a řešení pro potencionálního zákazníka. V případě, že obchodní příležitost nemá naplánovanou schůzku, učiní tak obchodní zástupce, který ji vlastní. S tímto procesem souvisí editace údajů o obchodní příležitosti a naplánovaných událostí. Jinou možností je předat příležitost pracovníkovi telemarketingu a požádat

o domluvení schůzky. Není však žádoucí, aby byl pracovník telemarketingu zahlcen velkým množstvím požadavků.

3.2.4 Ostatní aktivity obchodního procesu

- Plánování a realizace kontaktu s potencionálním zákazníkem. Proces zahrnuje především editaci události typu „schůzka“ a vyplnění výsledku schůzky. Pokud je schůzka úspěšná, je nutné změnit status schůzky na „objednávka“. Tím se změní také stav obchodní příležitosti vázané k události.
- Aktualizace informací o zákazníkovi. Tato činnost doplňuje správu obchodních příležitostí a měla by být provedena před sjednáním objednávky. Údaje o zákazníkovi je potřeba vyplnit pro účel pozdější fakturace (název firmy, adresu a IČO).
- Marketing – nabídka produktu pomocí dostupných komunikačních kanálů.
- Objednávka produktu. Při této aktivitě si zákazník vybere z nabídky produkt, o který má zájem. Obchodník sjedná objednávku se zákazníkem a zaeviduje ji do systému.

3.3 Specifikace požadavků na systém CSManager

Během fáze zahájení projektu jsem podrobil zkoumání stávající řešení, a v průběhu analýz jsem konzultoval a upravoval požadavky na nový systém se zadavatelem. Postupem vývoje systému vznikaly nové požadavky, které doplnily nebo modifikovaly určité funkce systému. Systémové požadavky jsou rozděleny podle metodiky UP[3], na funkční a nefunkční. Metodika UP zavádí formální přístup k požadavkům, z nějž vychází v modelování případů užití pomocí UML[4] (3, s. 79). Model požadavků je vytvořen textově s použitím jednoduchého formátu.

3.3.1 Funkční požadavky na systém

Následuje výčet funkcí systému, které by měl nabízet uživateli.

F01. Správa uživatelských účtů, skupiny uživatelů a jejich práv.

[3] UP (Unified Proces) je osnova pro proces vývoje softwaru
[4] UML (Unified Model Language) – jednotný vizuální modelovací jazyk pro tvorbu diagramů

F02. Zajištění chráněného přístupu k některým funkcím a modulům podle skupinových nebo uživatelských práv.

F03. Autorizace přihlášení k systému pomocí přihlašovacího jména a hesla.

F04. Zajištění integrity obchodních příležitostí pomocí verifikace obsazené obchodní příležitosti.

F05. Pro přehledy bude systém nabízet filtrování jednotlivých sloupců tabulky.

F06. Systém nabídne automatické vyhledávání v jednotlivých sloupcích tabulky přehledu.

F07. Uživatelské prostředí pro správu údajů o zákaznících.

F08. Správa kontaktních údajů zvlášť pro zákazníka a obchodní příležitosti.

F09. Uživatelské prostředí pro správu obchodních příležitostí.

F10. Systém nabídne filtrování obchodních příležitostí zvláště podle obchodních možností a odvětví.

F11. Evidence událostí obchodního procesu v přehledném kalendáři.

F12. Plánování událostí uživatelům bude zjednodušovat plánovací kalendář.

F13. Systém poskytne obchodnímu zástupci specifikovat typ a status události obchodního procesu.

F14. Tisk schůzek obchodního zástupce na den.

F15. Vytvoření nové události z přehledu obchodních příležitostí a přechod do kalendáře.

F16. Zobrazení pohledu na události v daném období podle obchodních příležitostí, zadavatele, vykonavatele, zákazníka.

F17. Upozornění na validitu zadaných informací k událostem a dodržování postupu vyplňování informací k obchodním příležitostem.

F18. Systém bude schopen vyčíslit report na měření efektivity obchodních zástupců.

3.3.2 Nefunkční požadavky na systém

Specifikace nefunkčních požadavků popisuje vlastnosti nebo omezující podmínky daného systému.

N01. Rozhraní aplikace budou postavena na dosud používaných platformách Centaur, Icarus a Zeus.

N02. Systém musí být jednoduše rozšiřitelný, tak aby v budoucnu bylo možné vyvíjet a přidávat nové moduly.

N03. Snadné přidávání nových číselníků (validační data) pomocí index tabulky.

N04. Volné rozšíření a napojení na jiné aplikace a báze dat.

N05. Aplikace je typu klient-sever a její správa je dostupná na internetové síti.

3.4 Model případů užití (Use Case)

Model případů užití grafickou formou zachycuje požadavky na systém a je hlavním zdrojem pro modelování objektů a tříd (3).

Model případů užití se skládá ze čtyř komponent:

- Hranice systému (system boundary): vymezená oblast působení subjektu.
- Aktéři (actors): osoby nebo předměty a jejich role používající systém.
- Případy užití (use cases): činnosti, které aktéři vykonávají v systému.
- Vztahy (relationships): vazby a vztahy mezi aktéry a případy užití.

Model případů užití, který jsem vytvořil při analýze systému CRM, obsahuje 10 balíčků modelů:

- Uživatelské role
- Kalendář
- Kontakty
- Obchodní příležitosti
- Osoby
- Pohled na události

- Report
- Uživatelé
- Zákazníci
- Číselníky

Následuje popis hlavních modelů případů užití pro systém CRM.

3.4.1 Modul Uživatelské role

Model (Obr. 3) zobrazuje hierarchii uživatelských rolí v systému. Na vrcholu hierarchie je obecný uživatel pro snadné znázornění odpovědností v ostatních modelech. Tato pokročilá metoda modelování se nazývá zobecnění aktéra. Obecný uživatel využívá základní funkce systému. Další uživatelé dědí od tohoto nadřazeného abstraktního předka všechny role a relace k případům užití. Kromě toho mají potomci na nižší úrovní přidané vazby na další případy užití (3, s. 116).

Administrátor

Uživatel v roli administrátor má plná a nejvyšší práva. Může provádět přidání a změny všech údajů. Systém dává administrátorovi k dispozici všechny moduly. Využívá všechny pohledy na data nad uživateli.

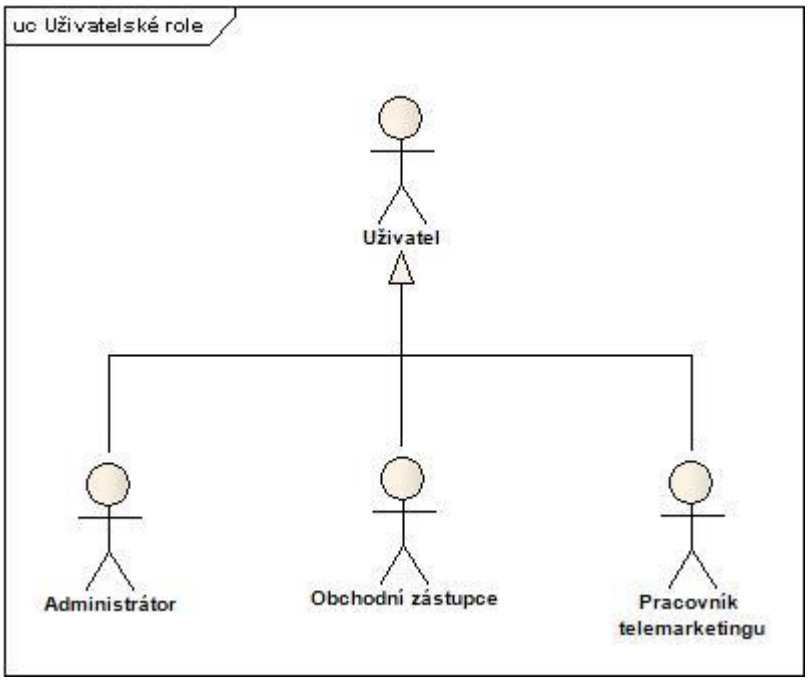
Obchodní zástupce

Je role pracovníka obchodního oddělení, který má na starosti celý obchodní proces. Má omezený přístup k modulům. Obchodní zástupce vkládá obchodní příležitosti a spravuje je. Může zobrazit jen své obchodní příležitosti, zákazníky a kontakty. Nemá možnost editovat a mazat některé údaje. Je zodpovědný za generování nových obchodních příležitostí. Řídí a plánuje telefonáty a schůzky. Uživatel v roli obchodního zástupce osobně jedná se zákazníkem a zaznamenává příslušné události související s jeho aktivitami do systému. Po úspěšné nabídce produktu a schválení balíčku produktu zákazníkem zaznamená objednávku v systému.

Pracovník telemarketingu

Jedná se o uživatele mající stejná práva jako obchodní zástupce jen s drobnými rozdíly. Může do systému zadávat nové obchodní příležitosti. Pracovník telemarketingu se ale na obchodním procesu podílí hlavně plánováním schůzek. Při jeho

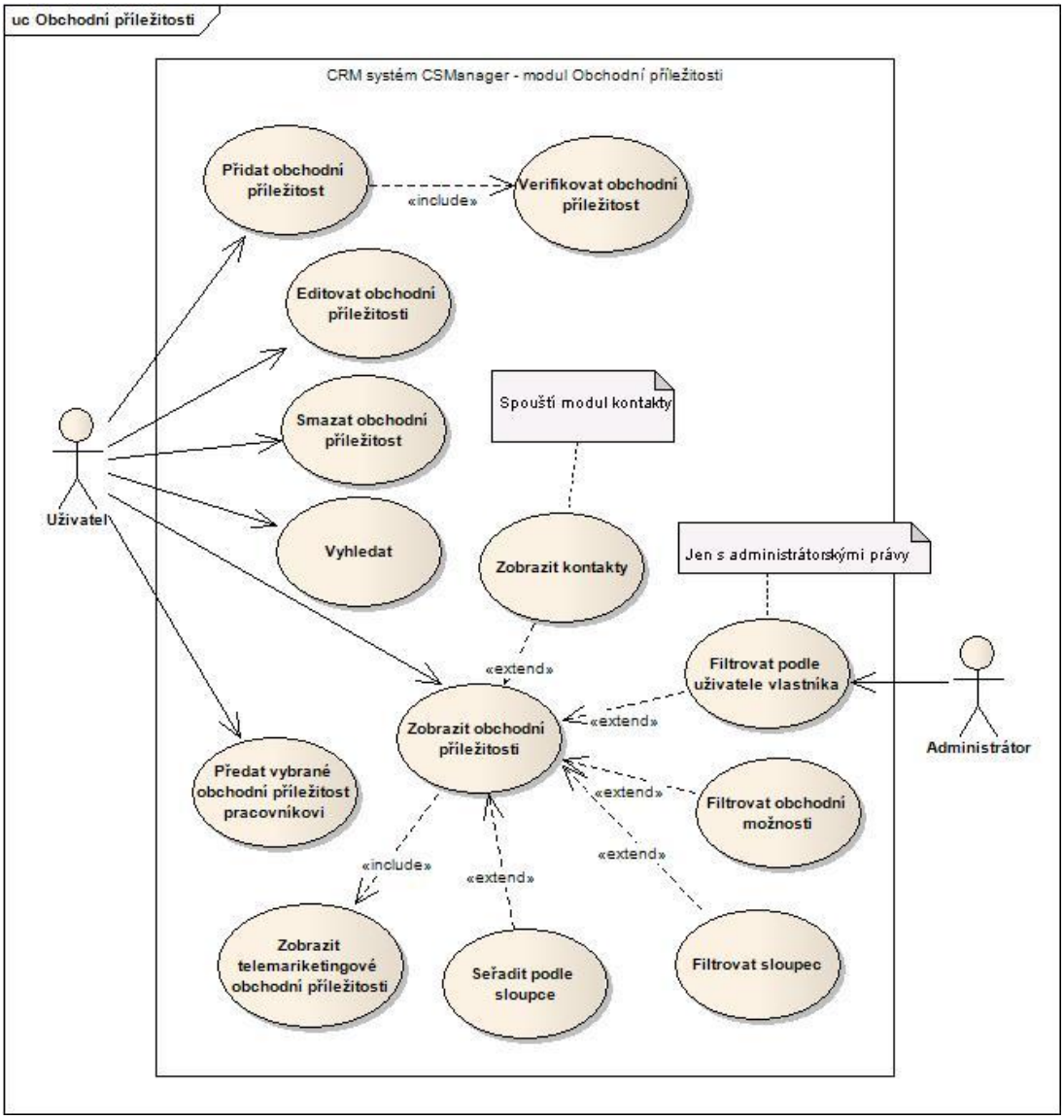
aktivitě vznikají události typu „telefon“ či „schůzka“. Při plánování schůzek má možnost vidět kalendáře ostatních uživatelů systému. V průběhu aktivit na obchodní příležitosti, většinou po domluvení schůzky, předá obchodní příležitost obchodnímu zástupci. Aby mohl být měřen pracovní výkon uživatele v roli telemarketingu, musí být všechny jeho zásahy správně zaznamenány. Efektivita práce je pak měřena podle úspěšnosti jím domluvených schůzek a tyto čísla jsou zobrazena v modulu Report.



Obr. 3: Hierarchický model Uživatelské role

3.4.2 Modul Obchodní příležitosti

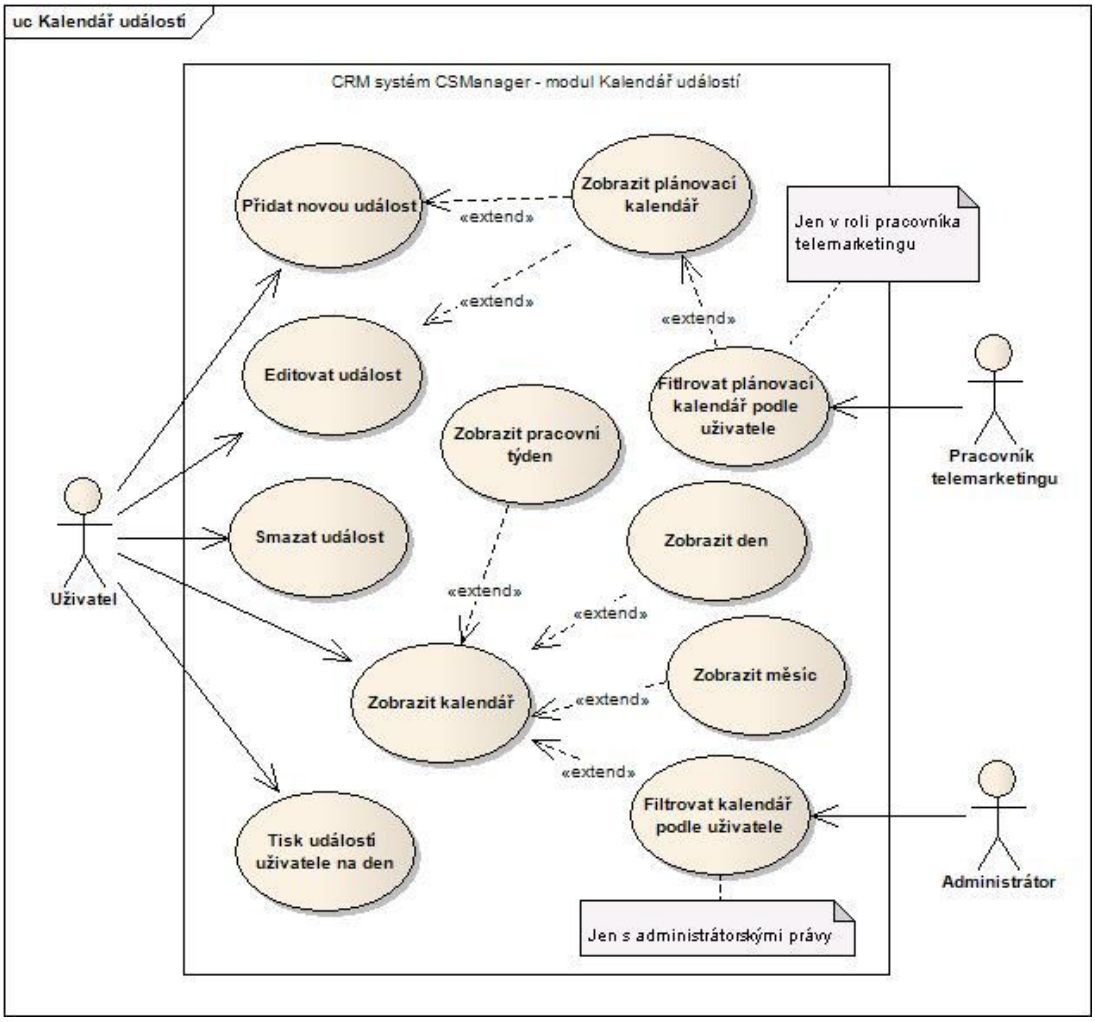
Model (Obr. 4) vyjadřuje případy užití vztahující se ke správě obchodních příležitostí. Aplikace umožní verifikaci obchodní příležitosti, zda je v databázi a nevlastní ji jiný obchodní zástupce. Pokud je obchodní příležitost volná, může ji pracovník zadat do systému s právem vlastníka a iniciovat nabídku produktu. Modul umožňuje zobrazit kontakty vztažené k dané obchodní příležitosti. Vztah typu *extend* rozšiřuje případ užití o nové chování. Uživatel v roli administrátora může procházet všechny obchodní příležitosti nebo je filtrovat podle vlastníka. Běžný uživatel může zobrazit jen své obchodní příležitosti, u kterých je vlastníkem. Užitečnou funkcí je předání vybraných obchodních příležitostí jinému pracovníkovi. Díky ní má vlastník obchodní příležitosti nebo administrátor možnost předat obchodní příležitost pracovníkovi telemarketingu s žádostí o domluvení schůzky. Modul také nabízí celou řadu funkcí pro zobrazení a filtrování seznamu obchodních příležitostí.



Obr. 4: Use Case – modul Obchodní příležitosti

3.4.3 Modul Kalendář událostí

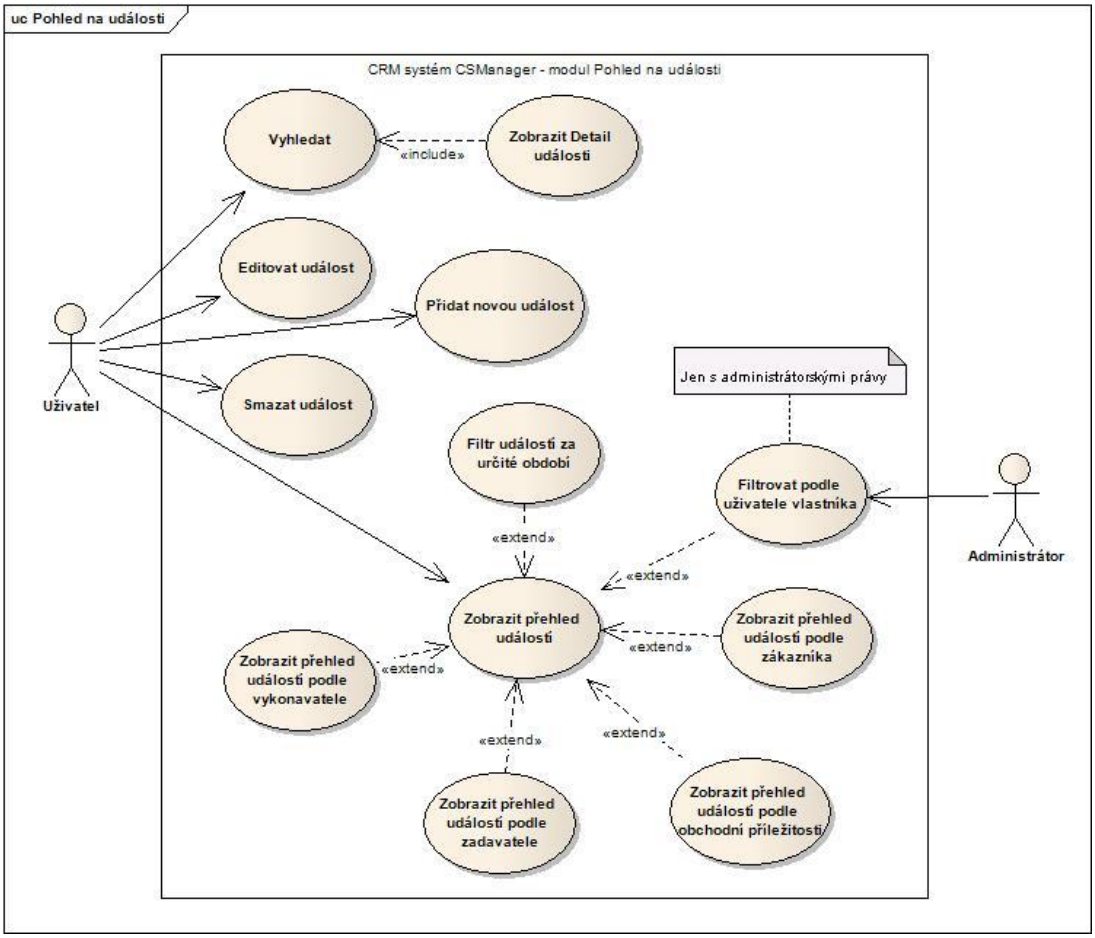
Modul Kalendář událostí (Obr. 5) by měl sloužit hlavně pro evidenci událostí obchodního procesu a v budoucnu také poskytne informace o jednotlivých fázích výrobního procesu. Kalendář poskytne tisk schůzek obchodního zástupce na den. Stěžejní funkcí kalendáře by mělo být plánování telefonů a schůzek pro obchodní příležitost. Pracovník telemarketingu v tomto modulu disponuje případem užití, ve kterém může pomocí plánovacího kalendáře přidávat schůzky ostatním uživatelům. Stejně tak uživatel v roli administrátora má práva na zobrazení kalendáře podle uživatele.



Obr. 5: Use Case – modul Kalendář událostí

3.4.4 Modul Pohled na události

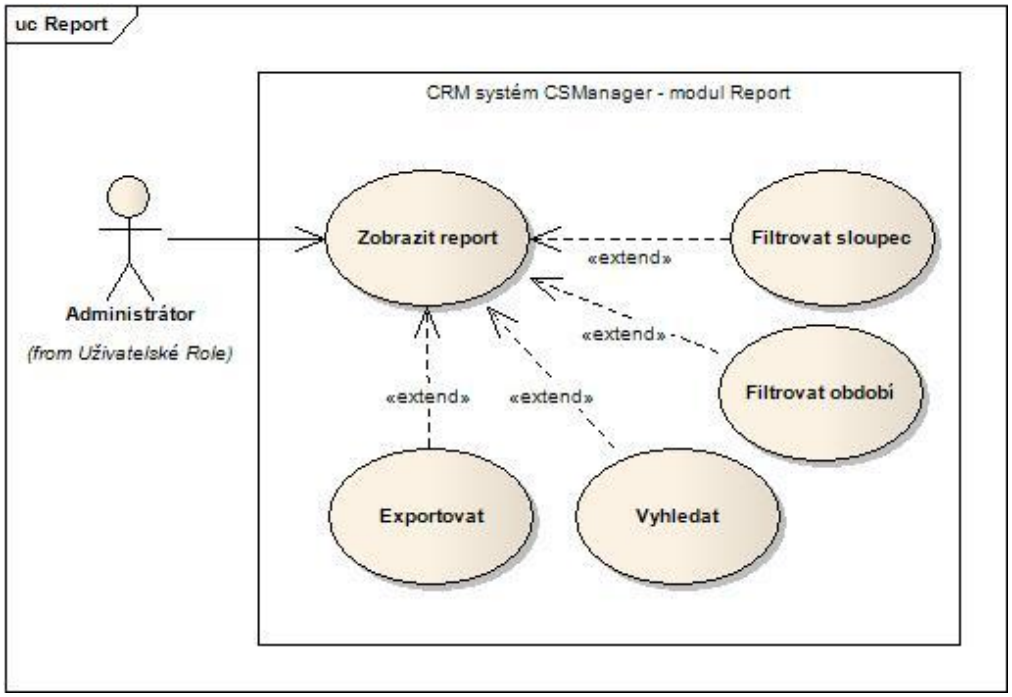
Tento modul (Obr. 6) nabízí komplexní zobrazení událostí podle zvolených parametrů. Základní funkcí pohledu je omezení prohlížených událostí jen z určitého období. Neméně důležité jsou také funkce přidání, editace a mazání události. Administrátor má právo zobrazovat události nejen podle běžných parametrů, ale může také použít filtr pro pohled nad uživateli vlastnícími obchodní příležitosti.



Obr. 6: Use Case – modul Pohled na události

3.4.5 Modul Report

Modul Report (Obr. 7) poskytuje informace o výkonech a efektivitě pracovníků. Přístup k tomuto modulu má pouze uživatel v roli administrátor. Měření se soustřeďuje na obchodní zástupce a na pracovníky telemarketingu. Mezi výpočty patří počet schůzek, telefonů a objednávek. Jednotlivé výkony pracovníka jsou měřeny v celkovém počtu nebo v procentech. Důležitou funkcí modulu je export dat do tabulkového procesoru MS Excel.



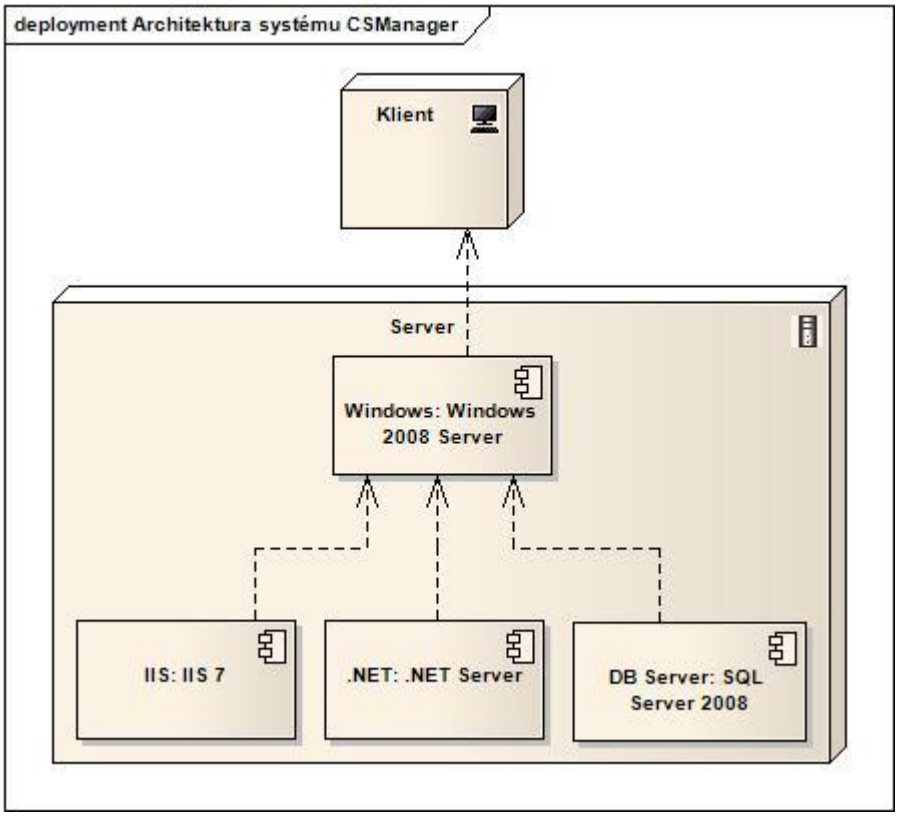
Obr. 7: Use Case – modul Report

3.5 Architektura systému CSManager

Architektura navrhovaného systému je dána prostředím vyvíjeným a používaným zadavatelem. Použitá technologie se odvíjí od základního požadavku zajistit přístup k systému pomocí klientské GUI aplikace. Výsledkem návrhu je tudíž třívrstvá centralizovaná architektura, která rozděluje zátěž systému mezi aplikační a databázový server (Obr. 8). Klientská aplikace může být díky technologii Mono provozována na počítačích s operačními systémy řady Microsoft Windows, ale i Linux, FreeBSD, UNIX, Mac OS X a Solaris (6).

Systém Windows Server 2008 představuje serverový operační systém navržený pro podporu nové generace sítí, aplikací a webových služeb. Integruje následující technologie:

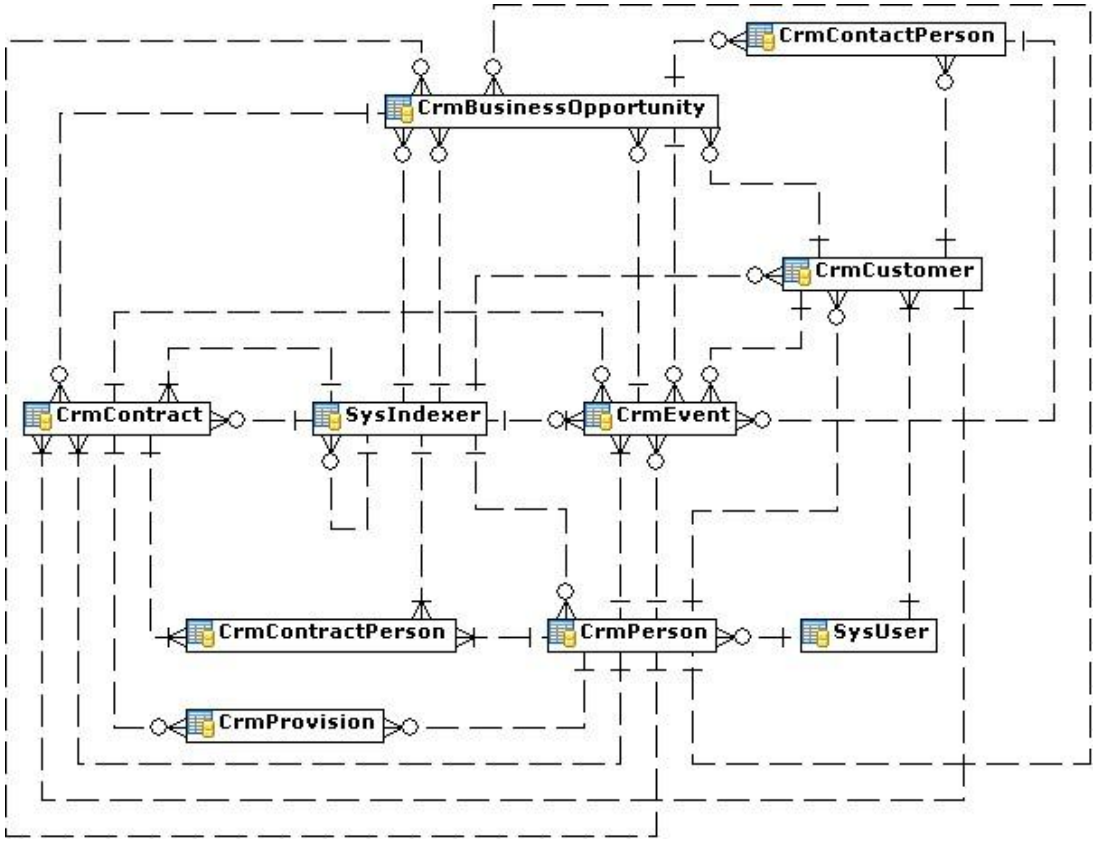
- IIS 7.0 (Internet Information Service) – webový server pro internetové informační služby. Balík programů umožňuje nasazovat a spravovat náročné internetové aplikace.
- Microsoft SQL Server 2008 – relační databázový systém od firmy Microsoft.
- .NET Server – hostování aplikací a podpora rozhraní Microsoft .NET Framework.



Obr. 8: Architektura navrhovaného systému CSManager

4 Návrh databáze

V této kapitole je popsán návrh databáze, datový model a jeho nejdůležitější tabulky. Nedílnou součástí každého systému je úložiště dat. Zadavatel používá pro správu databáze svých řešení relační databázový systém provozovaný na Microsoft SQL Server 2008. Tento produkt v sobě zahrnuje také manažer databáze, ve kterém jsem vytvořil datový model a vygeneroval SQL skript pro generování tabulek a jejich vazeb. MS databáze se dá lépe ovládat pomocí konkurenčního produktu SQL Manager 2008. Je to systém pro správu relační databáze (RDBMS), který se používá k vytvoření, udržování, modifikování a správě relační databáze (7).



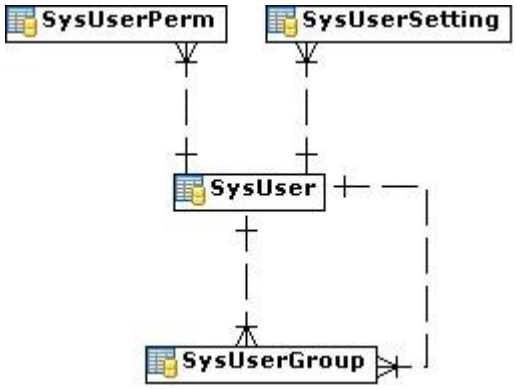
Obr. 9: Datový model systému CSManager

Ze specifikace požadavků na aplikaci jsem definoval potřebné entity datové struktury. Při návrhu jsem se snažil dodržet pravidla pro návrh relační databáze dle třetí normální formy. Avšak někdy jsem musel následovat doporučení vedoucího vývojového oddělení. Datový model (Obr. 9) byl vytvářen s přihlédnutím na fakt, že v aplikaci bude použita technologie NHibernate, o které pojednává samostatná kapitola 6.1 Technologie NHibernate. Protože framework Centaur je navržen jako

databázově nezávislý a některé databáze nepodporují technologii trigger[5], je business logika systému řešena programově v DAL (Data Access Layer). Na základě rozhovorů a konzultací se zadavatelem jsem k jednotlivým entitám doplnil atributy a jejich vazby, až vznikl datový model. V modelovacím nástroji SQL Manager 2008 jsem vytvořil datovou strukturu, která obsahuje deset vzájemně provázaných tabulek.

4.1 Datová tabulka Uživatel systému

Tabulka *SysUser* slouží ke správě uživatelských účtů systému. Rozšířenou datovou strukturu pro uživatelské účty jsem převzal z jiných softwarových řešení zadavatele a zachoval její koncepci. Tabulka *SysUser* je svázána s dalšími tabulkami, které řeší uživatelská práva, skupiny a nastavení. Struktura samostatného modelu je zřejmá z následujícího obrázku (Obr. 10).



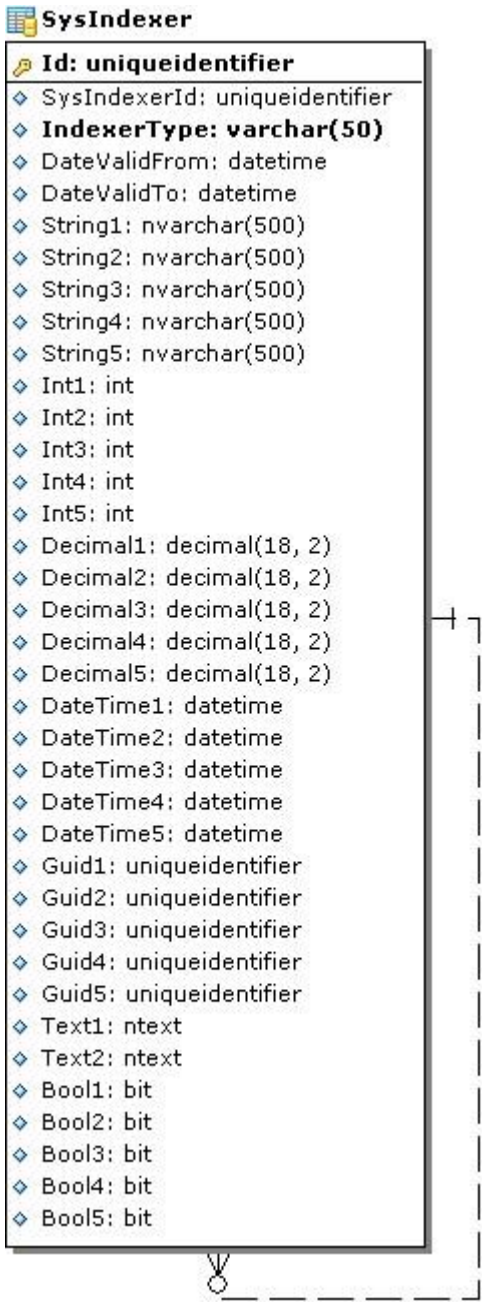
Obr. 10: Datový model uživatelské struktury

4.2 Datová tabulka Číselník

Tabulka *SysIndexer* (Obr. 11) byla navržena vývojovým týmem zadavatele a má plnit úlohu globálních číselníků nejen v rámci jedné aplikace. Jde vlastně o validační tabulku nebo také vyhledávací tabulku. Záznamy v této tabulce jsou trvalé a nemění se tak často. Koncept validační tabulky nabízí možnost definovat seznam hodnot různých typů, aniž by bylo předem zřejmé, které typy budou pro validační seznam potřeba. To zajišťuje modulárnost řešení jako takového. Kouzlo tohoto návrhu je ve správě těchto záznamů patřících do různých validačních

[5] Trigger (spoušť) v databázi definuje činnosti, které se mají provést v případě události nad databázovou tabulkou

seznamů. Jednoduše lze doplnit třeba informaci o tom, jaký záznam číselníku má ovlivnit chování aplikace pouhým nastavením sloupce *Bool1*. V praxi to znamená, že u každé položky validačního seznamu je možné definovat několik pravidel či omezení. Pravidla u položek mohou řešit uživatelská práva, ale také ovlivnit chování jiných komponent na formuláři. Například pokud se seznam použije v komponentě výběrového seznamu, který označuje typ události, je možné definovat ještě sloupec *DateTime1*. Existují tedy události typu „telefon“, „schůzka“ a „e-mail“. U každé události je zapotřebí nastavit předpokládaný čas trvání, který se při výběru typu události na formuláři automaticky přenese do pole s výběrem plánovaného času. Nespornou výhodou je také centrální správa takového číselníku a jeho využití v libovolném počtu jiných aplikačních systémů. Aplikace umožní specifikovat název použitého sloupce pomocí XML souboru a sloupec *IndexerType* říká, o jaký druh seznamu se jedná.

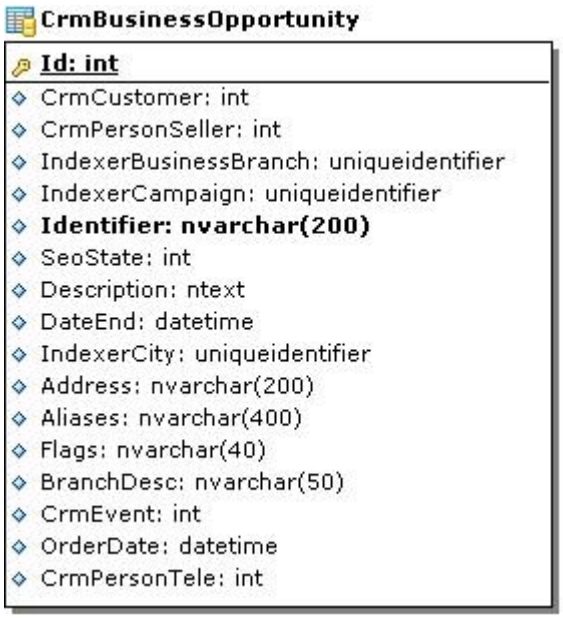


Obr. 11: Datová tabulka Číselník

Jednoznačný identifikátor *Id* je typu GUID (Globally Unique Identifier). Tento speciální typ identifikátoru poskytuje globálně unikátní identifikátor, který mírně snižuje výkon SQL dotazu. Záznam číselníku je identifikován vždy pomocí dvou klíčů *Id* a *IndexerType*, aby se zvýšila rychlost vyhodnocení dotazu. Navržený číselník má také vazbu sám na sebe a tím umožňuje větvení ve stromové struktuře. V systému CSManager je číselník používán pro uložení názvů měst a obcí nebo také již zmiňovaného typu události. Zvýrazněné položky v tabulce nemohou nabývat prázdných hodnot.

4.3 Datová tabulka Obchodní příležitost

Tabulka *CrmBusinessOpportunity* (Obr. 12) uchovává informace o všech obchodních příležitostech evidovaných v systému. U každého zákazníka může být několik obchodních příležitostí, a proto zde existuje relace 1:N. Atribut *CrmPersonSeller* je vazbou na tabulku *CrmPerson* a poskytuje tak údaje o obchodníkovi, který je vlastníkem. Sloupce tabulky s příponou *Indexer* obsahují specifické informace z globálního číselníku. Cizí klíč *IndexerBusinessBranch* určuje odvětví podnikání, do kterého patří obchodní příležitost. Položka *BranchDesc* slouží k upřesnění oblasti podnikání. Položka *Identifier* je doména nebo URL adresa internetových stránek obchodní příležitosti. Pole *SeoState* je číselný údaj, který udává procentuální výši optimalizace webových stránek pro vyhledávače. Do položky *Flags* se ukládají příznaky budoucích obchodních možností oddělené středníkem.



Obr. 12: Datová tabulka Obchodní příležitost

Seznam obchodních možností s jednotlivými příznaky je definován v globálním číselníku. Tyto informace o obchodní příležitosti je nutné svázat programově. Pro zjištění stavu obchodní příležitosti slouží reference na poslední událost cizího klíče *CrmEvent*. Z důvodů měření efektivity u dané obchodní příležitosti se do položky *OrderDate* zaznamenává datum objednávky v den, kdy obchodní zástupce

uzavře se zákazníkem objednávku a změní status události na „objednávka“. Další atributy informativního charakteru jsou zřejmé z jejich názvů.

4.4 Datová tabulka Zákazník

Každý řádek v tabulce *CrmCustomer* (Obr. 13) představuje informace o zákazníkovi. Cizí klíč *CrmPersonSeller* zde představuje obchodního zástupce, který je zodpovědný za zákazníka. Povinná položka *Name* uchovává název zákazníka neboli firmy. Jsou zde také nezbytné informace *CompanyId* (IČ) a *CompanyTaxId* (DIČ). Povinná položka *SysUserModified* uchovává referenci na uživatele, který naposledy upravoval údaje o zákazníkovi. Záznam o zákazníkovi vyžaduje přesnější údaje o sídle firmy. I v této tabulce informaci o městě poskytuje cizí klíč s vazbou na globální číselník měst *IndexerCity*. Tabulka ovšem nabízí ještě upřesnění městské části, kterou je možné zaznamenat do pole *City*. Důležité údaje sídla firmy jsou také telefonní spojení a e-mail. Položka *Recipient* by měla obsahovat jméno a příjmení jednatele firmy.

Obr. 13: Datová tabulka Zákazník

4.5 Datová tabulka Kontaktní osoba

Záznamy o kontaktech (Obr. 14) jsou jednou z nejdůležitějších tabulek CRM systému. Poskytuje informace o kontaktech, vztahujících se nejen k obchodní

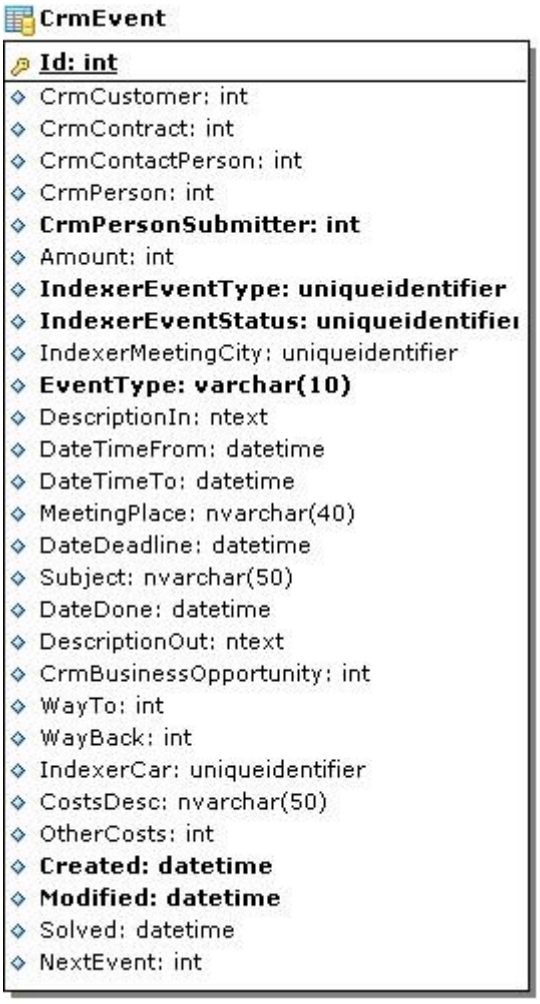
příležitosti, ale také k zákazníkovi. Na zákazníka nebo obchodní příležitost může být navázáno libovolné množství kontaktů. Položky *ActiveFrom* a *ActiveTo* typu datetime upřesňují období, ve kterém je kontakt aktivní. Položka *ActiveFrom* musí být vyplněna a je inicializována datem vytvoření kontaktu. Naopak sloupec *ActiveTo* nemusí obsahovat hodnotu a znamená to, že kontakt je v současnosti stále aktivní.

Obr. 14: Datová tabulka Kontaktní Osoba

4.6 Datová tabulka Událost

Nejpoužívanější tabulka (Obr. 15) v CRM systému. Každá obchodní nebo výrobní aktivita na obchodní příležitosti je do systému zanesena v podobě události. Do tabulky se zapisují všechny plánované události a v aplikaci slouží také jako kalendář úkolů. Tabulka je navržena, tak že při vzniku události se volí typ *EventType* na obchodní nebo výrobní. Každá událost má svého zadavatele uloženého v poli s názvem *CrmPersonSubmitter*. Zadavatel události (obvykle sám sobě) úkoluje nebo eviduje událost jako vykonanou. Informace o tom, kdo je vykonavatelem se zaznamenává v položce *CrmPerson*. Uživatel pak volí typ události *IndexerEventType* podle číselníku. Stav, v jakém se událost nachází, vyjadřuje číselník *IndexerEventStatus*. Při zahájení nebo plánování události je zapotřebí vést informace o vstupech a výstupech, k čemuž slouží textová pole *DescriptionIn* a *DescriptionOut*. Funkcionalita posloupnosti a řetězení jsou zajištěna relací události sama na sebe identifikátorem *NextEvent*. Časový rámec, do kterého událost spadá, se ukládá ve sloupcích *DateTimeFrom* a *DateTimeTo*. Atributy *Created* a *Modified* slouží k udržení data o vytvoření a změně události. U události, která již byla vyřešena, se nastaví datum u položky *Solved*. Při návrhu tabulky jsem zvažoval oddělení údajů o schůzce do jiné tabulky.

Po konzultaci se zadavatelem, jsem přihlédl k faktu, že vzhledem k častým operacím čtení a zápisu by technologie NHibernate (popisovaná v kapitole 6.1 Technologie NHibernate) v daném případě nemusela pracovat efektivně.



Obr. 15: Datová tabulka Událost

5 Migrace dat

V této kapitole popisuji způsob a provedení migrace stávajících dat do nové databáze. Databáze se musí naplnit daty z doposud vedené evidence dat v tabulkách MS Excel. Každý obchodník zaznamenává obchodní příležitosti ve své tabulce. Data jsou často chybná a nevalidní, a proto se musí naplnění dat provádět po částech. Při načítání hodnot z transformované tabulky musí probíhat kontrola validity záznamů, datového typu a dodržení datového formátu. Jelikož data v Excel tabulce jsou relačně neuspořádaná, je potřeba je správně rozdělit podle navržené datové struktury do několika tabulek. Rozdělení a následný převod se provádí mapováním sloupců. Po přečtení jednoho řádku a přiřazení všech položek se v rámci jedné transakce spustí vložení do databáze. Pokud se celá transakce z nějakého důvodu nepovede bezchybně, je zapotřebí ji odvolat. Při chybě se vyvolá výjimka a data musí být překontrolována. Výjimka zobrazí chybové hlášení o pádu transakce. Hlášení obsahuje informaci o kolizi, číslo řádku, kde kolize vznikla a její příčinu. Podle těchto indicií je snadné najít ve vzorku dat chybnou položku a nastavit správná pravidla pro transformaci.

5.1 Využití nástroje pro migraci dat

Pro migraci dat jsem využil volně dostupný program pro import dat z CSV souboru naprogramovaný v jazyce C# (5). Program obsahuje třídu a základní metody pro připojení CSV souboru pomocí ODBC a naplnění třídy DataSet. Rozšířil jsem ovládání programu o nezbytné funkce. Přidal jsem možnost čtení souborů typu XLS, jak je vidět v následující ukázce kódu (Obr. 16):

```
// import z XLS
DataSet ds = new DataSet();
OleDbConnection oleDbConn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;
                                                  DataSource=D:/Test/JakubMadr.xls;
                                                  Extended Properties=Excel 8.0");
oleDbConn.Open();
OleDbDataAdapter oleDA = new OleDbDataAdapter(" SELECT * FROM [List1$]", oleDbConn);
oleDA.Fill(ds, "Transfer");
oleDbConn.Close();
DataTable dataTable = ds.Tables["Transfer"];
```

Obr. 16: Ukázka kódu – načtení XLS souboru do DataSet

Nastavil jsem transformace a kontrolu validity jednotlivých sloupců. Přidal jsem také důležitý ovládací prvek, a to volbu uživatele vlastnícího obchodní příleži-

tosti tak, aby se při importu dat do tabulky *CrmBusinessOpportunity* správně nastavila reference na *CrmPersonSeller*. Součástí migrace bylo naplnění číselníku měst a porovnání validity přenášených dat do tabulky *CrmBusinessOpportunity* s vazbou na číselník *IndexerCity*. Rozdělil jsem data podle sloupců z tabulky Excel tak, aby jejich transport vytvořil záznamy a vzájemné vazby v tabulkách *CrmEvent*, *CrmCustomer*, *CrmPerson* a *CrmContactPerson*. Ošetřil jsem import pomocí transakcí na straně aplikace a přidal srozumitelné vypisování výjimek pro testování a odchytávání chyb (Obr. 17). Po odladění aplikace byla data úspěšně převedena.

```
// přípojení k databázi a zahájení transakce
SqlTransaction tn = sqlConn.BeginTransaction();
SqlCommand cmd = new SqlCommand();
cmd.Connection = sqlConn;
cmd.CommandType = CommandType.Text;
cmd.Transaction = tn;
try
{
   foreach (DataRow row in dataTable.Rows)
   {
      // transformační a validační příkazy
      // příkazy pro manipulaci s daty
      cmd.CommandText = @"SET DATEFORMAT DMY
            INSERT INTO CrmBusinessOpportunity
                        (CrmCustomer,CrmPersonSeller,IndexerBusinessBranch,Flags,
                        IndexerCity,Address,Identifier,Description)
            VALUES(@newCustomerId,@personSellerId,@idxBBranchGuid,@flags,
                     @idxCityGuid,@address,@Identifier,@Description)
            SELECT CAST(scope_identity() AS int)";
      int newBoId = (Int32)cmd.ExecuteScalar();
      // vrací nový identifikátor vloženého řádku
      // další příkazy
   }
   tn.Commit(); // potvrzení a ukončení transakce
   MessageBox.Show("Data úspěšně importována","Done",MessageBoxButtons.OK,
                     MessageBoxIcon.Information);
}
catch (Exception ex)
{
   MessageBox.Show("Chyba řádek " + countRow + ";" + ex, "Chyba",
                     MessageBoxButtons.OK, MessageBoxIcon.Error);
   tn.Rollback(); // vrácení transakce
}
finally
{
   sqlConn.Close();
}
```

Obr. 17: Ukázka kódu – import dat ze souboru XLS

6 Implementace systému CSManager

Při zahájení projektu jsem dostal k dispozici základní vývojovou platformu CS Technologies. Jedná se o nový projekt pro Visual Studio s integrovanými knihovnami pro vývoj aplikací v této firmě. Tento základ pro programování systému již obsahoval reference na součásti framework Centaur, Icarus a Zeus. Součástí také bylo přednastavené hlavní okno programu s integrovaným menu a pracovní plochou, kterou je možné spravovat pomocí záložek. Do programového menu byl již integrován modul pro správu uživatelů, který se používá ve všech programech vyvíjených ve firmě zadavatele. Systém CSManager je rozdělen na dvě části klient a server. Ve skutečnosti jsou to dva projekty s vlastními adresáři. V klientské části se buduje uživatelské prostředí, okna aplikace a formuláře. Klientské rozhraní používá framework Icarus. Framework Centaur se specializuje na komunikaci se serverem a všechny operace prováděné na serveru. Framework Zeus nabízí různé pomocné funkce například validaci a formátování dat. Na straně serveru je samostatná část s implementací nástroje NHibernate, který je podrobněji popsaný v následující kapitole 6.1 Technologie NHibernate. Z úvodního popisu v této kapitole je zřejmé, že architektura je rozdělená do několika vrstev, kde každá vrstva řeší příslušný problém. Tím je systém stabilnější a dá se lépe spravovat. Vrstvy mezi sebou komunikují a využívají navzájem svých služeb.

6.1 Technologie NHibernate

V dnešním světě informačních technologií je mnoho formátů pro ukládání dat a přibývají nové. Data se posílají přes síť do jiných aplikací nebo služeb, jako jsou relační databáze, adresářové servery nebo fronty zpráv. Vzniká potřeba ukládat data na několika různých místech nebo kombinovat všechny tyto možnosti v jedné aplikaci. Kritickou myšlenkou v této oblasti je správa persistentních dat nebo jednoduše řečeno jak data načítat, ukládat a chránit. Relační databáze jsou velmi osvědčeným nástrojem pro manipulaci s daty, ale je zde pořád mnoho možností a otázek, kterým se musí čelit při každodenní práci s nimi. Často je pokládána otázka, jestli používat uložené procedury. Mají se SQL dotazy psát vlastnoručně nebo je nechat automaticky generovat? Je užitečné vytvářet v kódu doménový model se třídami a spravovat načítání a ukládání dat z databáze? Odpovědí na tyto otázky je ORM

(Object Relational Mapping). Jedním ze sofistikovaných ORM nástrojů, který používá i firma CS Technologies ve svých aplikacích, je NHibernate. NHibernate je open source ORM nástroj pro .NET, port původního Hibernate napsaného v Javě. Nabízí kompletní řešení problému správy persistentních dat při práci s relační databází a doménovým modelem tříd (8, s. 4).

6.1.1 Hlavní charakteristiky technologie NHibernate

Vlastní objektově orientovaný model databázových relačních entit je definován v XML. Doménový model pomáhá výrazně zlepšovat znovu-použitelnost kódu a udržovatelnost. Není nutné vytvářet tento model ručně, stačí využít generátor pro vytvoření souborů mapování databázových tříd. Mapovací informace říká, jakým způsobem se mají entity načíst a uložit. ORM v podstatě funguje na transformaci dat z jedné reprezentace do druhé.

Jakmile má NHibernate informace o mapování entit, je možné vykonávat CRUD (Create, Read, Update, Delete) operace pouhým voláním příslušné metody. Automaticky se provádí komplexní složité DML (Data Manipulation Language) příkazy nad vzájemně propojenými entitami. Využívá se jednoho připojení a stav změn se udržuje do konce transakce.

NHibernate podporuje funkci zvanou lazy loading (líné načítání). Když se načítá objekt, je možné se rozhodnout, jestli se mají načíst také referenční objekty. Pokud se vybere funkce lazy loading, tyto objekty budou transparentně inicializovány, až když jsou potřeba.

Uložené procedury se neprovádí v databázi, ale v .NET. To umožňuje využívat sofistikované objektově orientované koncepty, jako je dědičnost a polymorfismus.

Dalším důležitým aspektem je nezávislost na databázovém systému. NHibernate odděluje aplikaci od různého SQL dialektu.

Dotazovací jazyk nad objekty se nazývá HQL (Hibernate Query Language). Dalším rozhraním pro dotazování nad daty je QBC (Query By Criteria). Ve vývoji je také NHibernate implementace jazyka LINQ pro dotazování nad různými typy datových zdrojů bez nutnosti použití dotazovacího řetězce.

NHibernate využívá cache paměť pro sdílení používaných dat mezi transakcemi a dokonce i aplikacemi. Data se při požadavku na zobrazení nemusejí pokaždé načítat. NHibernate spravuje mapu identit a efektivně pracuje s vyrovnávací pamětí, což vede ke zvýšení výkonu (8, s.15).

6.2 Využití NHibernate v systému CSManager

Datové schéma a datový model databáze nám poskytuje potřebné informace pro generování mapovacích dokumentů a kostru persistentních tříd. Existuje řada nástrojů pro generování kódu. Tyto nástroje umí generovat základní implementaci pro NHibernate. CS Technologies používá vlastní detailně rozpracované funkce NHibernate a tudíž potřebuje i vlastní generátor. Prvotním úkolem tedy před zahájením projektu bylo, naprogramovat generátor podle specifického vzorku souborů. Ve spolupráci s vývojovým týmem jsem vyvinul generátor všech potřebných souborů doménového modelu. Ke generování se používá předem definovaná šablona. Při změnách v datovém modelu a nutnosti znovu vygenerovat soubory, umožňuje nástroj zachovat ručně psané oblasti kódu.

6.2.1 Persistentní třídy

Aby aplikace mohla získávat data pro zobrazení a měnit je v databázi, potřebuje jednoduché persistentní třídy. NHibernate u třídy vyžaduje jeden neveřejný implicitní bezparametrický konstruktor. Třída nese název entity, kterou zastupuje a obsahuje všechny její položky jako vlastnosti. Vlastnosti mají nastavené metody *Get* a *Set*. Persistentní třída může také obsahovat business metody, které provádí vlastní zpracování dat.

6.2.2 Mapování tříd

Mapování třídy *CrmCustomer* je uloženo v souboru Objects.hbm.xml. Mapovací dokument (Obr. 18) ukazuje, že třída *CrmCustomer* se uchovává v tabulce se stejným názvem. U každé třídy je v hlavičce, pomocí atributu *table,* možnost nastavit, jakou tabulku bude třída mapovat. Atribut *lazy* nastavuje způsob načítání dat lazy loading, popsaný dříve v kapitole 6.1.1 Hlavní charakteristiky technologie NHibernate. Kromě toho lze také nastavit *dynamic-insert* nebo *dynamic-update,* které NHibernate udávají, zda zahrnout nezměněné hodnoty vlastností během SQL

insert nebo update. Vynechání sloupců při těchto operacích zlepšuje výkon obzvlášť u tabulek s mnoha sloupci. Vlastnosti třídy jsou definovány elementem *property* a reference na vazební třídy elementem *many-to-one*. NHibernate používá reflexi[6], aby zjistil typ vlastnosti a mohl tak odvodit mapování na sloupec v SQL. Díky tomuto mapování má NHibernate dostatek informací k vytváření instance *CrmCustomer* a generování kompletních CRUD databázových příkazů pro zajištění persistence.

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
      namespace="CSManager.Server.Crm.Model" assembly="CSManager.Server.Crm"
      default-access="nosetter.camelcase-underscore" >
   <class name="CSManager.Server.Crm.Model.CrmCustomer, CSManager.Server.Crm"
         table="CrmCustomer" lazy="true" dynamic-update="true" >
      <id name="Id" column="Id" >
         <generator class="native" >
            <param name="sequence">CrmCustomer</param>
         </generator>
      </id>
      <property name="SysUserModified" column="SysUserModified" />
      <property name="Name" column="Name" />
      <property name="CompanyId" column="CompanyId" />
      <property name="CompanyTaxId" column="CompanyTaxId" />
      <many-to-one name="CrmPersonSeller" column="CrmPersonSeller"
            class="CrmPerson" />
      <many-to-one name="IndexerCity" column="IndexerCity"
            class="Icarus.Server.Indexer.Model.SysIndexer,Icarus.Server.Indexer" />
   </class>
</hibernate-mapping>
```

Obr. 18: Definice mapování třídy CrmCustomer

6.2.3 Hlídání změn databázového schématu

Po nasazení aplikace je obtížné měnit schéma databáze, proto je v systému CSManager implementováno rozšíření hlídající vývoj schématu a jeho změny. Existuje soubor DatabaseSchema.xml, ve kterém jsou všechny změny označeny číslem nové verze. Vždy, když se aplikace spouští, dojde ke kontrole a provede se migrace ze staré verze do nové. Díky tomu, je databáze vždy aktuální bez nutnosti zásahu administrátora.

6.3 Použití framework Icarus a Centaur

V následujících odstavcích popisuji použití framework Icarus a Centaur, se kterými jsem pracoval při vývoji systému CSManager. Pro zobrazení dat v tabulce

[6] Reflexe je obecný termín zahrnující různé bázové třídy platformy .NET Framework, jejíž instance umožňují dynamicky vyhledávat informace o typech jak v programech , tak v sestaveních (4, s. 283).

XtraGrid je potřeba připojit datový zdroj a naplnit DataSet záznamy. V návrhovém režimu je také nutné do komponenty XtraGrid nadefinovat sloupce a jejich typy. Komponenty framework Centaur a Icarus nabízejí API (Application Programming Interface), které umožňují snadné získání a ovládání dat. Tyto komponenty mají mnoho užitečných funkcí dostupných také v režimu návrhu. Pro komunikaci klient-server se používá komponenta ModuleIndex, která v sobě nese nastavení připojení a informace o spojení. Komponentu pak dále využívá komponenta QxQuery. Ta obsahuje referenci QueryIdentifier na třídu s dotazem HQL uloženou v souboru na straně serveru. Tento soubor je využíván pro náročnější získání dat z databáze a podporuje speciální metody používané komponentou TableViewManager. QxQuery také spravuje informace o mapované tabulce a pomocí reflexe zjistí názvy vlastností a datový typ, potřebné právě v režimu návrhu. Získaná data převezme komponenta TableViewManager, spravující operace načítání, zobrazení, editování a ukládání dat. S touto komponentou je úzce spojen BarManager a zobrazovací komponenta XtraGrid popsaná v kapitole 6.4.2 Komponenta XtraGrid. Názvy sloupců tabulky a jejich typ jsou tak nastaveny automaticky na základě zvoleného názvu tabulky. ModuleIndex nabízí pro připojení se serverem čtyři druhy připojení (WCF, SOAP, TCP, HTTP). V systému CSManager jsem využil možnost připojení pomocí webové služby SOAP. Stačí zadat adresu zdroje služby, údaje pro autentizaci a potom stisknout tlačítko connect (připojit). Pokud se připojení povede na komponentě QxQuery, jsou k dispozici dvě tlačítka. První z nich „Fetch QueryDescription", zajišťuje přenesení popisných dat (názvy sloupců, datové typy) do všech svázaných komponent. Druhé tlačítko „Force table structure" slouží k naplnění tabulkových struktur popisnými daty. Tím je komponenta XtraGrid nastavena a připravena přijímat data. V módu návrhu u komponenty XtraGrid je možné vybrat zobrazované sloupce, které byly v předchozím kroku načteny. Jak již bylo řečeno QxQuery získává data z databáze. V následující ukázce (Obr. 19) je dotaz HQL na data uložený ve zvláštní třídě, která je potomkem třídy QxQueryHiber. Třída implementuje metodu BeforeSelect rodičovské třídy, ve které se po splnění podmínky přidá do dotazu další klauzule. Tento postup přidání omezení získání dat jsem při vývoji systému CSManager hojně využíval.

```
public class CrmBO_SelectViewContext : QxQueryHiber
{
      protected override void InitQuery()
      {
            HiberContextName = ModuleContext.HiberContextName;
            // dotaz na data pomocí HQL
            string sqlSelectBegin = @"select
                                          b.Id,
                                          b.Identifier,
                                          e.DateTimeTo as LastEventDate,
                                          es.String2 as StatusColor
                                       from CrmBusinessOpportunity as b
                                          left join b.CrmPersonSeller as s
                                          left join b.CrmCustomer as c
                                          left join b.CrmEvent as e
                                          left join e.IndexerEventStatus as es
                                       where 1=1";
            // klauzule pro obnovení řádku komponenty TableViewManager
            SqlRefreshRow = sqlSelectBegin + @" and (b.Id = :Id) ";
            FieldsListKey = new string[] { "Id" };
      }

      protected override void BeforeSelect()
      { // při splnění podmínky přidání klauzule do dotazu
         if (SqlParams.Contains("CrmPersonId"))
         {
            if (!String.IsNullOrEmpty(SqlParams["CrmPersonId"].ToString()))
            {
               Sql.Add(" and (b.CrmPersonSeller.Id = :CrmPersonId) ");
            }
         }
      }
}
```

Obr. 19: Ukázka kódu – dotaz pro QxQuery

6.3.1 Metody trigger framework Centaur

Framework Centaur má integrované programové triggers (spouštěče), které analogicky nahrazují všechny databázové operacemi insert, update a delete volané before a after činnosti. Centaur synchronizace spouštěčů využívá NHibernate objekty. Jedním z nich je session (sezení). Pojem session lze v tomto případě chápat jako něco mezi připojením a transakcí. Je to paměť obsahující načtené objekty, vztahující se k jedné transakci. NHibernate může detekovat změny objektů v paměti. Rozhraní ISession je také označováno jako manažer persistence, protože se stará o operace načítání a ukládání objektů. Stav persistentních instancí se synchronizuje s databází na konci transakce. Když se transakce ukončí, stav paměti se uloží do databáze provedením SQL insert, update nebo delete příkazů (8, s. 32). Například po vložení nové události do plánovacího kalendáře je zavolána metoda AfterInsert a její příkazy vykonají námi definované operace.

6.3.2 Metody ve třídě ModuleWorker

Serverová část systému CSManager zpřístupňuje některé metody pomocí webových služeb založených na SOAP. Třída ModuleWorker je umístěna na serveru

a slouží k definování těchto metod pro operace nad databází. Tyto metody mohou být využívány nejen klientskou aplikací CSManager, ale jakoukoliv jinou aplikací. Metoda je na serveru, vystavena jako služba. Klient pošle požadavek na server, ve kterém je název metody se vstupními parametry. Služba na serveru zpracuje požadavek a vrátí klientovi odpověď.

Popisovanou techniku jsem používal především pro získávání dat z databáze. Například pokud chci z formuláře pro editaci události zobrazit pohled na souhrn událostí vybrané obchodní příležitosti, potřebuji zjistit, kdo vlastní obchodní příležitost. Je několik možností, jak získat Id vlastníka obchodní příležitosti. Jednou z možností je použití DataContext nástroje NHibernate a poptat se na data pomocí manažeru dané třídy. NHibernate pokud má data v paměti, neváhá a vrátí požadovaná data, jinak provede potřebný dotaz pro získání dat z databáze. Druhou možností je využít session nástroje NHibernate a přímým dotazem potřebná data získat (Obr. 20). První způsob je rychlejší v případě, že jsou data již ve sdílené paměti NHibernate.

```
[CentaurServiceParams(Method = "get_seller_bo")]
public void get_seller_bo(DataRequestParams aRequest, DataResponseParams aResponse)
{ // vstupní parametry metody
   int? bOId = aRequest.Params.GetNullable<int>("BusinessOpportunityId");
   int? personSellerId = null;
   if (bOId != null)
   { // použití session
      using (IHiberSession session = ModuleContext.CreateHiberSession())
      {
         Params pars = new Params();
         pars.Add("BusinessOpportunityId", DataType.Int, bOId);
         string sqlCommand = @"SELECT o.CrmPersonSeller.Id
            from CrmBusinessOpportunity as o
            where o.Id =:BusinessOpportunityId";
         personSellerId =
         // vykonání dotazu
         DataUtils.GetNullable<int>(session.ExecuteScalar(sqlCommand, pars));
         // výstupní parametry metody
         aResponse.Params.Add("personSellerId", DataType.Int, personSellerId);
      }
   }
   else
   {
      aResponse.Params.Add("personSellerId", DataType.Int, personSellerId);
   }
}
```

Obr. 20: Ukázka kódu – dotazovací metoda třídy ModuleWorker

Ve třídě ModuleWorker je možné také definovat operace pro uložení nebo mazání dat. V metodě event_date_save (Obr. 21) je vidět použití DataContext nástroje NHibernate. Tento příklad demonstruje uložení změny časového rozsahu události, která vznikne při události AppointmentResized v plánovacím kalendáři událostí.

```
[CentaurServiceParams(Method = "event_date_save")]
public void event_date_save(DataRequestParams aRequest, DataResponseParams aResponse)
{ // vstupní parametry metody
   int? id = aRequest.Params.GetNullable<int>("Id");
   if (id != null)
   { // použití kontextu
      using (CrmContext dc = ModuleContext.CreateDataContext())
      { // získání záznamu události podle id pomocí manažeru
         CrmEvent recCrmEvent = dc.CrmEventManager.GetById((int)id);
         if (recCrmEvent != null)
         { // nastavení záznamů
            recCrmEvent.Modified = DateTime.Now;
            recCrmEvent.DateTimeFrom = (DateTime)aRequest.Params["DateTimeFrom"];
            recCrmEvent.DateTimeTo = (DateTime)aRequest.Params["DateTimeTo"];
            dc.CrmEventManager.Update(recCrmEvent); // update pomocí manažeru třídy
            aResponse.Params["Result"] = true; // výstupní parametry metody
         }
      }
   }
   else
   {
      aResponse.Params["Result"] = false;
   }
}
```

Obr. 21: Ukázka kódu – update metody event_date_save třídy ModuleWorker

Pro doplnění technik popsaných v této kapitole uvádím ještě ukázku (Obr. 22) volání metody event_date_save v klientské části systému CSManager.

```
private void schedulerControl_AppointmentResized(object sender,
   AppointmentResizeEventArgs e)
{ // nastavení vstupních parametrů metody
   Params pars = new Params();
   pars.Add("Id", (int)e.EditedAppointment.GetValue(schedulerStorage, "Id"));
   pars.Add("DateTimeFrom", e.EditedAppointment.Start);
   pars.Add("DateTimeTo", e.EditedAppointment.End);
   // vykonání metody na serveru s vrácením parametrů
   ParamsCallResponse res = moduleIndex.QxModule.ExecuteParams("event_date_save",
      pars);
}
```

Obr. 22: Ukázka kódu – reakce na událost změna času schůzky

6.4 Komponenty od firmy DevExpress

Vývojové oddělení CS Technologies pro své projekty používá komponenty od firmy DevExpress. Firma DevExpress vyvíjí vlastní vizuální komponenty zejména pro .NET, WPF, SilverLight nebo Delphi. Z jejich dílny také pochází nástroje pro zvýšení produktivity práce CodeRush a Refactor. Mým úkolem bylo využít vizuální komponenty pro uživatelské prostředí Windows Forms, obzvláště XtraScheduler, XtraGrid a XtraReports. Tyto komponenty mají speciální IDE plně integrované do Visual Studia. Toto prostředí velice ulehčí práci s takto sofistikovanými komponentami. Následuje stručný popis použitých vizuálních komponent.

6.4.1 Komponenta XtraScheduler

Komponenta XtraScheduler (plánovač) je klíčovou komponentou celého systému CSManager. XtraScheduler v sobě obsahuje kompletní plánovací kalendář, vzhledem podobný integrovanému v Microsoft Outlook. Podporuje různé typy zobrazení od měsíčního až po denní. Procházení kalendáře podle data usnadňuje DateNavigator. Díky přímé vazbě může svá data ukládat v databázi, XML nebo v objektu s IList rozhraním. Základním prvkem v kalendáři je Appointment (schůzka). Tento název je trochu zavádějící, ale může být chápán také jako úkol nebo událost, která se zobrazí v kalendáři. Dále budu používat termín událost, který tak zapadá do vytvářeného systému. Událost dovoluje přidat vlastní položky, přičemž výchozí data jsou používána k řízení události. Událost disponuje mnoha cennými vlastnostmi. Jednou z nich je například hledání volného místa v rozvrhu. Událost může být přiřazena k nějakému prostředku, a tím je zajištěno jeho rezervování. Pro ty zapomnětlivé lze nastavit jedno nebo více upozornění na plánovanou událost. Jako jiné .NET komponenty firmy DevExpress, XtraScheduler uživatelské rozhraní podporuje nastavení vzhledu a přizpůsobení stylů. Další výhodou je výměna dat s Microsoft Outlook kalendářem, která se dá provádět v obou směrech (9).

6.4.2 Komponenta XtraGrid

Ovládací prvek XtraGrid reprezentuje data ve formě tabulky. Použitím architektury pro přístup k datům a oddělením interních datových modulů od samotné prezentace dat, plně využívá výhody technologie ADO.NET. XtraGrid dovoluje vytvářet a presentovat master-detail informace ve více úrovních, tak že i ty nejsložitější sady vztahů se jednoduše ovládají. Veliký vliv na výkon XtraGrid tabulky má správa dat na straně serveru. Je to speciálně navržený režim pro podporu velkých datových množin (sestávající z více jak 50.000 záznamů). Načítá data po malých částech na vyžádání, provádí všechny dohlížecí operace na straně serveru. Tím zajistí rychlý přístup k datům, i když jsou setříděná, seskupená nebo filtrovaná. XtraGrid představuje architekturu založenou jak na klasickém pohledu na data s řádky a sloupci, tak i speciální kartový pohled nebo také detailní pohled v úrovních. Samozřejmostí je filtrování, přemisťování, třídění sloupců a v neposlední řadě také vyhledávání ve sloupci. Navíc se tyto operace nad pohledem ukládají do historie

a mohou být kdykoliv znovu použity. Předností komponenty je určitě i export do TXT, HTML, XML a MS Excel (9).

6.4.3 Komponenta XtraReport

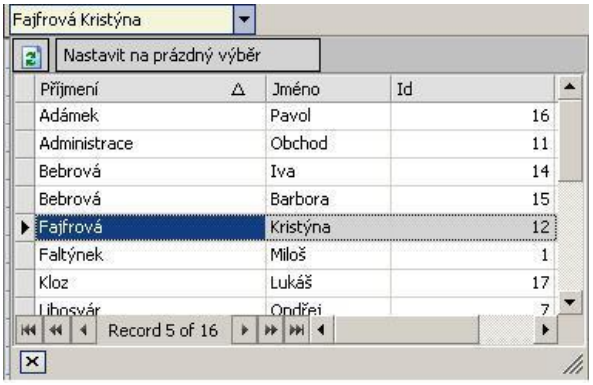
XtraReport je systém pro snadné vytváření sestav a jejich tisk. Pracuje se všemi objekty podporovanými Visual Studiem pro .NET. Je možné navázat report na XML data nebo na jakýkoliv datový objekt, implementující IList rozhraní. Osazení pole daty ve formuláři funguje tak, že stačí napsat do pole reportu název položky uzavřený do hranatých závorek. Vazba na data umožňuje naplnit report tak, že při vytvoření tiskové sestavy jsou pole osídlena položkami z datového zdroje za běhu. Kromě textových položek lze pouhým kliknutím vložit na vytvářenou sestavu různé další zobrazovací prvky. Sestavy je možné obohatit o grafy, které nabízí plně kompatibilní komponenta XtraCharts. Pokud je zvolen tisk sestavy, zobrazí se nejprve náhled, který lze pomocí událostí dynamicky měnit. Komponenta standardně nabízí export do mnoha textových nebo tabulkových formátů, ale také do spousty obrázkových formátů. Pro tisk reportu slouží knihovna XtraPrinting s řadou užitečných funkcí (11).

6.5 Realizace uživatelských ovládacích prvků systému CSManager

Pro aplikační formuláře v uživatelsky příjemném prostředí jsou důležité snadno ovladatelné prvky. Ovládací prvky na formuláři mezi sebou komunikují, poskytují a zpracovávají data. Při vývoji systému CSManager jsem vybudoval celou strukturu formulářů. Pro komponenty zobrazující data jsem v první řadě musel programově získat data. Po té jsem nastavil prostředky pro jejich chování a zobrazování. Interakci s uživatelem jsem zajistil vytvořením kontextových menu. Formuláře jsem opatřil metodami pro správu a manipulaci s daty. Obsluhu některých formulářů jsem obohatil o příslušná dialogová okna. Naprogramoval jsem a nastavil kontrolu uživatelských vstupů. K uživatelským akcím jsem přidal také chybová hlášení, která zachycují nežádoucí stavy.

V klientské části systému CSManager jsem potřeboval několik výběrových seznamů, které se plní daty z databáze. Visual Studio 2008 pro tvorbu vizuální

komponenty nabízí přednastavený projekt s názvem UserControl. Je to v podstatě základ pro vkládání ovládacích prvků. Výhody takto vytvořeného ovládacího prvku jsou jednotnost napříč aplikacemi, stejný vzhled a chování. Příkladem mnou vytvořeného ovládacího prvku je sel_SelectPerson výběrový rozbalovací seznam (Obr. 23), který zapouzdřuje skupinu ovládacích prvků. Po kliknutí na tento ovládací prvek se rozbalí seznam uživatelů, ze kterých je možné vybrat právě jednoho.



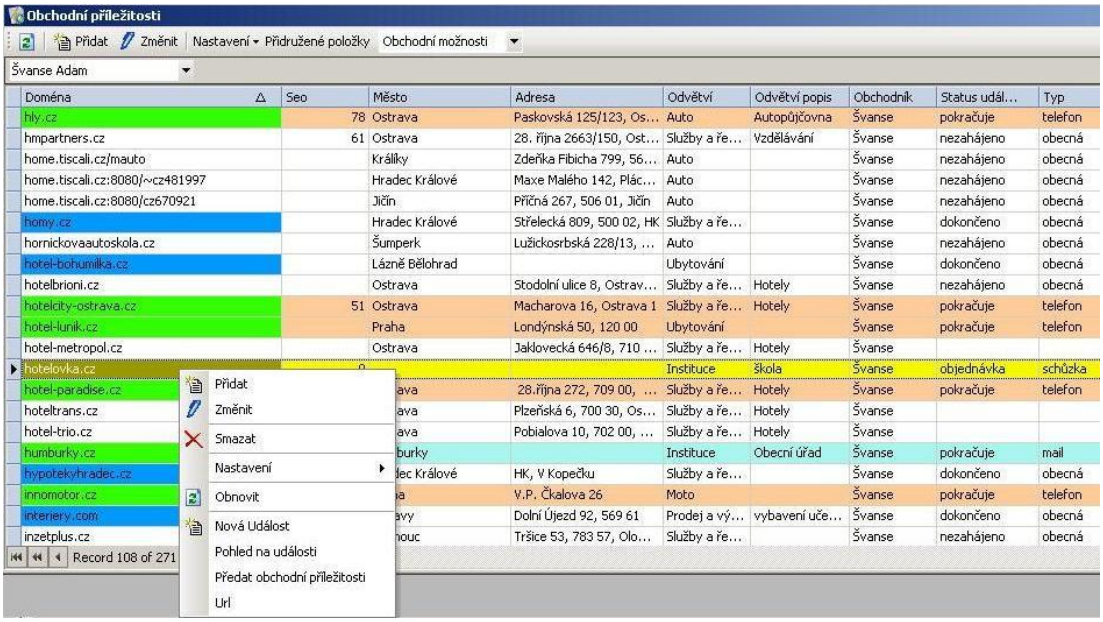
Obr. 23: Ukázka vlastního ovládacího prvku – rozbalovací seznam pro výběr osoby

Hlavní řídící složkou ovládacího prvku je tabulka XtraGrid, která je naplněna pomocí sdílené komponenty sd_SelectPerson, poskytující rozhraní pro práci s datovým zdrojem. K tomu, aby formulář zjistil, který záznam byl vybrán z rozbalovacího seznamu, slouží veřejná vlastnost SelectedId.

6.5.1 Ovládání formuláře Obchodní příležitosti

Formulář (Obr. 24) zobrazuje obchodní příležitosti v administrátorském režimu, kde je možnost vybrat obchodní příležitosti podle vlastníků. Obchodní zástupce zahajuje většinu svých aktivit v systému CSManager, právě na tomto formuláři. V případě, že chce uživatel přidat novou událost k obchodní příležitosti, stiskne tlačítko přidat v liště nebo v kontextovém menu formuláře. Poté se zobrazí modální okno s formulářem pro hledání obchodní příležitosti v systému. Formulář vyhledává obchodní příležitosti podle domény a obchodního identifikačního čísla. Obchodní zástupce si může zaregistrovat jen neobsazenou obchodní příležitost. Kontextové menu z tabulky obchodních příležitostí usnadňuje přístup ke speciálním funkcím systému. Následuje jejich popis:

- Předat obchodní příležitosti – splňuje požadavek na předání obchodní příležitosti pracovníkovi telemarketingu. Je možné zvolit a předat naráz více obchodních příležitostí.
- Pohled na události – otevře formulář s pohledem na události k vybrané obchodní příležitosti.
- URL – spustí výchozí prohlížeč internetových stránek operačního systému s URL vybrané obchodní příležitosti.



Obr. 24: Formulář Obchodní příležitosti

Obrázek (Obr. 24) demonstruje příklad řízení a správy obchodních příležitostí v prostředí CSManager.

6.5.2 Formulář Kalendář událostí

Kalendář událostí (Obr. 25) zobrazuje velký čitelný kalendář a umožňuje přidávat a odebírat události. Události mají nastavené vlastnosti pro snadné ovládání, jako jsou drag and drop (táhni a pusť) nebo resize (zvětšení). Uživatel tak pouhým tažením myší přesune nebo rozšíří událost. Přepínaní zobrazení kalendáře (den, týden, pracovní týden, měsíc, rozvrh) definuje časový interval zobrazení, elementy uživatelského rozhraní a chování ovládacích prvků specifických pro daný aktivní pohled. Každý pohled mění celkové zobrazení plánovacího kalendáře. V některých případech dochází ke zmenšení události a některé informace nejsou kompletně zobrazeny. Z tohoto důvodu jsem přidal metodu pro zobrazení detailní informace tzv. tooltip. Ke

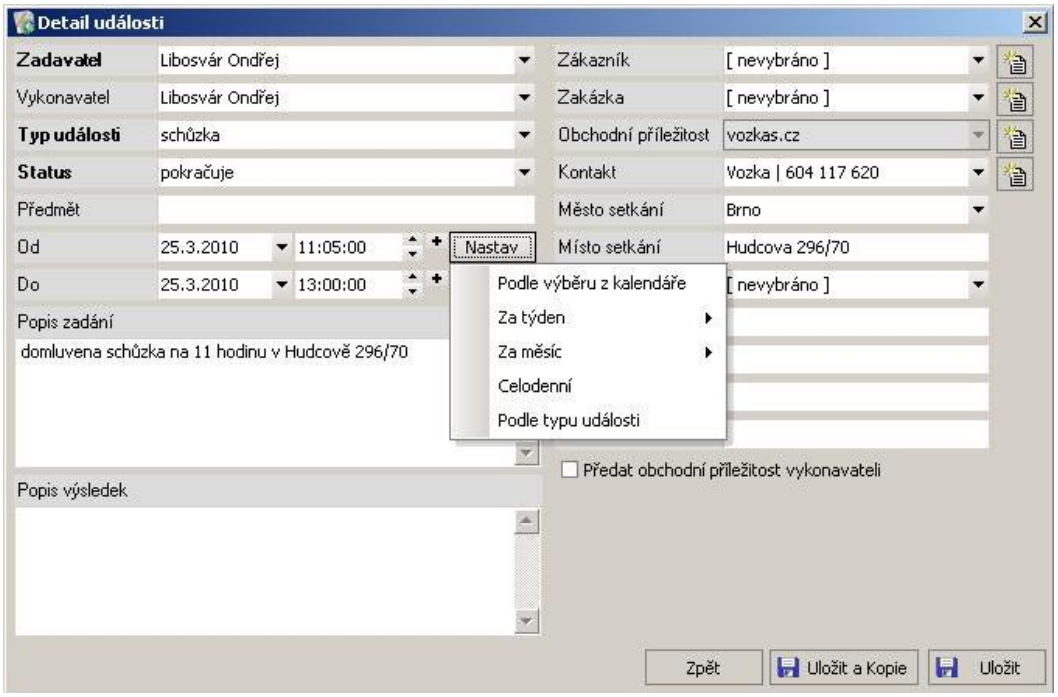
snadné navigaci v kalendáři slouží ovládací prvek DateNavigator umístěný v pravé části formuláře. Při implementaci jsou všechny ovládací prvky automaticky svázány s hlavním ovládacím prvkem kalendáře XtraScheduler, který se nachází na stejném formuláři.

Obr. 25: Formulář Kalendář událostí

6.5.3 Editační formulář události

Původně byly předem stanoveny požadavky, jaké funkce bude editační formulář události nabízet. Ale s používáním tohoto formuláře, vznikly nové nároky na funkcionalitu. Úkolem vyvíjeného dialogového okna bylo poskytnout nejen komplexní editaci události, ale také všech entit k ní vztažených. V současné době je z tohoto editačního formuláře (Obr. 26) možné přidávat a editovat entitu Zákazník, Zakázka, Obchodní příležitost a Kontakt. Plánování události zjednodušuje seznam funkcí s automatickým nastavením časového intervalu. Mezi tyto funkce patří obvyklé nastavení události za nějakou určitou dobu (např. za měsíc v pondělí). V případě označení časové události myší na ploše plánovacího kalendáře, je časový interval události inicializován ze zvoleného rozsahu. Aby bylo plánování události a v podstatě tento editační formulář použitelný z různých modulů systému CSManager, musel jsem přidat volbu nastavení trvání události podle výběru z kalendáře. Volba otevře zjednodušenou formu plánovacího kalendáře v dialogovém okně. Uživatel v tomto okně nemůže nijak měnit události, má právo jen vybrat časový

interval a potvrzením se vybraný rozsah přenese na formulář „Detail události“ do časových editačních polí „Od“ a „Do“. Zjednodušení přináší i funkce nastavení trvání události podle typu události. Typ události má v číselníku definován název typu, barvu zobrazení události a omezení z hlediska uživatelských práv. Pro funkci nastavení časového rozsahu bylo zapotřebí přidat do číselníku k typu události další položku typu DateTime. Když uživatel zvolí typ události a v menu je vybráno „Nastav – Podle typu události“, nastaví se do časového trvání události předem definované hodnoty z číselníku. Pro uživatele s příslušným oprávněním editační formulář disponuje zaškrtávacím ovládacím prvkem „Předat obchodní příležitost vykonavateli“. Rychlé vytváření kopie události šetří uživateli čas automatickým vyplněním položek události.

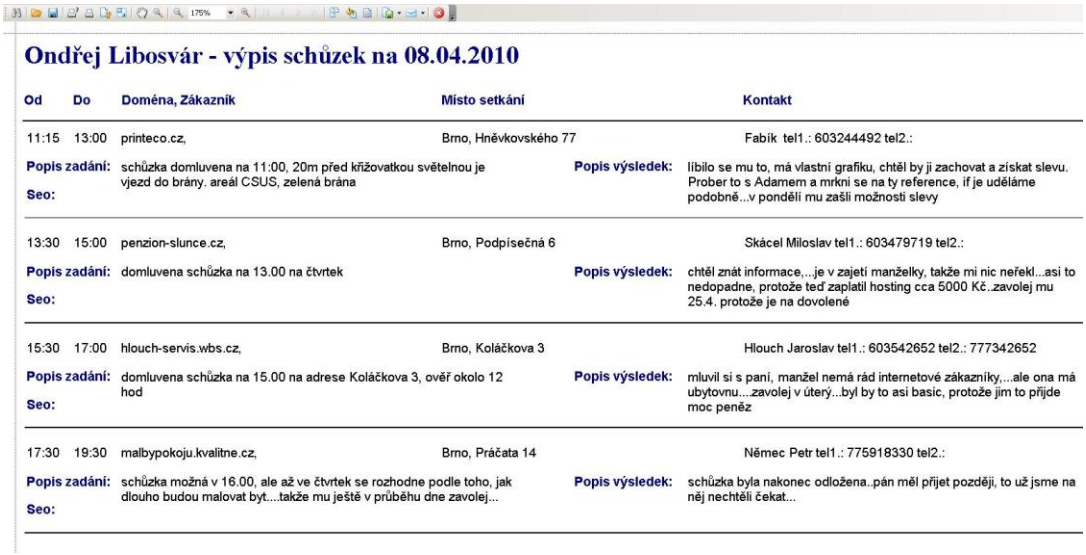


Obr. 26: Formulář editace události

6.5.4 Tisk schůzek na den

Jedním z požadavků na aplikaci CSManager bylo také tisknutí schůzek obchodních zástupců na den. Funkce tisku schůzek slouží nejen obchodním zástupcům, aby měli seznam naplánovaných schůzek na den, ale i nadřízeným pracovníkům k souhrnné kontrole výsledků schůzek. Pro zobrazení informací schůzek a tisk jsem využil komponentu XtraReport popsanou v kapitole 6.4.3 Komponenta XtraReport. Ve formuláři Kalendář Událostí uživatel označí den se schůzkami

a stiskne ikonu s obrázkem tiskárny. Z formuláře Kalendář událostí jsou komponentě XtraGrid předány parametry potřebné k získání dat. Potom komponenta generuje sestavu položek, které osidluje daty z datového zdroje. Následně je zavolána metoda pro zobrazení náhledu (Obr. 27). Náhled umožňuje především úpravu vlastností dokumentu před samotným tiskem, ale také export sestavy do různých textových formátů.



Ondřej Libosvár - výpis schůzek na 08.04.2010

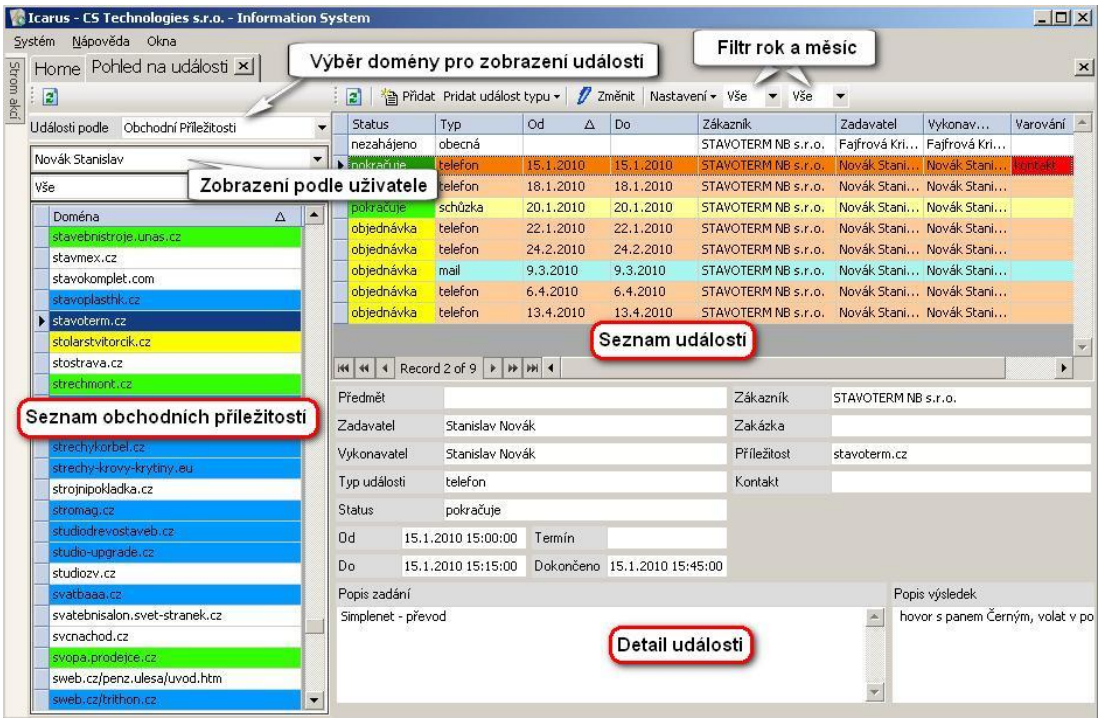
Od	Do	Doména, Zákazník	Místo setkání	Kontakt
11:15	13:00	printeco.cz,	Brno, Hněvkovského 77	Fabík tel1.: 603244492 tel2.:
Popis zadání:		schůzka domluvena na 11:00, 20m před křižovatkou světelnou je vjezd do brány. areál CSUS, zelená brána	**Popis výsledek:**	líbilo se mu to, má vlastní grafiku, chtěl by ji zachovat a získat slevu. Prober to s Adamem a mrkni se na ty reference, if je uděláme podobně...v pondělí mu zašli možnosti slevy
Seo:				
13:30	15:00	penzion-slunce.cz,	Brno, Podpísečná 6	Skácel Miloslav tel1.: 603479719 tel2.:
Popis zadání:		domluvena schůzka na 13.00 na čtvrtek	**Popis výsledek:**	chtěl znát informace,...je v zajetí manželky, takže mi nic neřekl...asi to nedopadne, protože teď zaplatil hosting cca 5000 Kč..zavolej mu 25.4. protože je na dovolené
Seo:				
15:30	17:00	hlouch-servis.wbs.cz,	Brno, Koláčkova 3	Hlouch Jaroslav tel1.: 603542652 tel2.: 777342652
Popis zadání:		domluvena schůzka na 15.00 na adrese Koláčkova 3, ověř okolo 12 hod	**Popis výsledek:**	mluvil si s paní, manžel nemá rád internetové zákazníky,...ale ona má ubytovnu...zavolej v úterý...byl by to asi basic, protože jim to přijde moc peněz
Seo:				
17:30	19:30	malbypokoju.kvalitne.cz,	Brno, Práčata 14	Němec Petr tel1.: 775918330 tel2.:
Popis zadání:		schůzka možná v 16.00, ale až ve čtvrtek se rozhodne podle toho, jak dlouho budou malovat byt....takže mu ještě v průběhu dne zavolej...	**Popis výsledek:**	schůzka byla nakonec odložena..pán měl přijet později, to už jsme na něj nechtěli čekat...
Seo:				

Obr. 27: Náhled výpisu schůzek před tiskem

6.5.5 Zpracování formuláře Pohled na události

Z funkčních požadavků na CSManager vyplývá, že uživatelé systému potřebují mít přehled nad událostmi. Události je potřeba seskupovat zejména podle obchodní příležitosti, zadavatele a vykonavatele. Tento modul si vyžadoval koncepční přístup vytváření UserControl. Ovládací prvky formuláře (Obr. 28) jsem rozdělil do několika závislých částí. Složky formuláře spolu komunikují pomocí událostí, sdílených metod nebo vlastností. V levé části formuláře jsem vytvořil panel s ovládacím prvkem pro výběr události podle referenční domény. V tomto popisu se omezuji pouze na výběr události podle obchodní příležitosti. Při výběru se inicializuje ovládací prvek se seznamem obchodních příležitostí a je nastavena obsluha události změny výběru obchodní příležitosti ze seznamu. Každý uživatel vidí v seznamu jen ty obchodní příležitosti, u nichž je vlastníkem. Procházet seznamy obchodních příležitostí vlastníků, může jen uživatel se speciálním oprávněním. Pravá část formuláře slouží pro zobrazení seznamu událostí s filtrem období, ve kterém nastaly. Zobrazované období se určuje rokem nebo měsícem. Formulář reaguje na výběr události ze seznamu tím,

že ve spodní části zobrazí její detail. Zobrazení poskytuje master-detail pohled podle zvolené entity. Uživatel tak má chronologický přehled událostí podle entit, ke kterým se událost váže. Lišta tlačítek formuláře nabízí také vytvoření nové události nebo editaci.



Obr. 28: Formulář Pohled na události

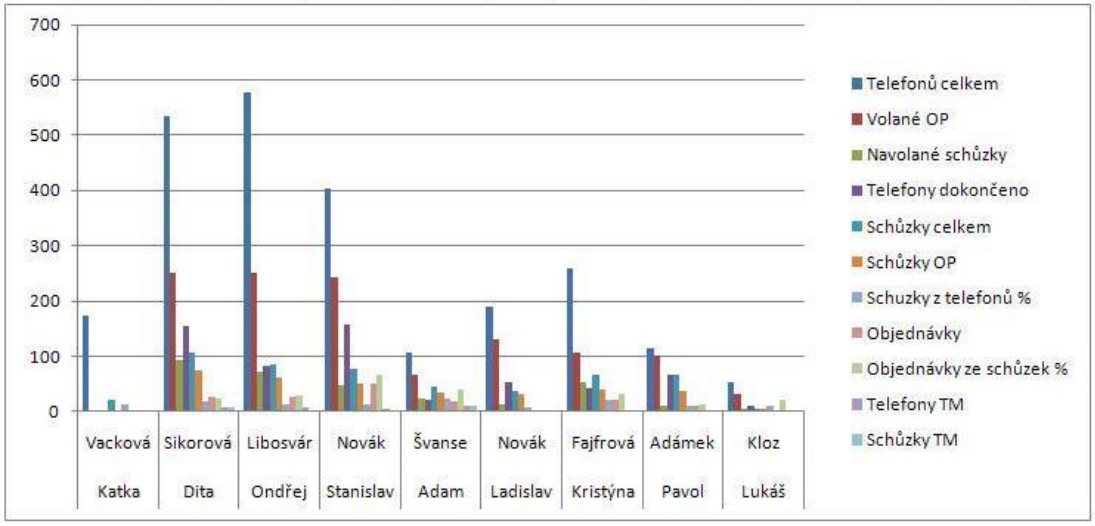
Barva obchodní příležitosti označuje status poslední události. V seznamu událostí levá část zabarvení odráží status a druhá typ události. Barvy statusů a typů událostí jsou nastavitelné s příslušnými právy v modulu „Správa číselníků“. Poslední sloupec v tabulce událostí s názvem „Varování“ vyjadřuje splnění jednotlivých podmínek na data v systému. Jednou z podmínek pro událost typu telefon je vyplněný kontakt. Uživatelé mají za úkol varování sledovat a chyby eliminovat.

6.5.6 Report efektivity pracovníků

Modul slouží k měření efektivity pracovníků. Modul obsahuje tabulku zobrazující různé ukazatele produktivity práce pro obchodní oddělení. Výsledky se pomocí filtru dají zúžit na určité časové období s přesností na dny. V současné době měření podléhají pracovníci telemarketingu a obchodní zástupci. Obě skupiny jsou měřeny odlišným způsobem. Zatím co u obchodních zástupců se efektivita zjišťuje zejména počtem objednávek, úspěch pracovníka telemarketingu spočívá ve sjednávání schů-

zek. Na následujícím obrázku (Obr. 29) je vidět vyexportovaná tabulka v programu MS Excel. Graf není součástí exportu a vytvořil jsem jej ručně pro názornost efektivity jednotlivých pracovníků.

Jméno	Příjmení	Telefonů celkem	Volané OP	Navolané schůzky	Telefony dokončeno	Schůzky celkem	Schůzky OP	Schuzky z telefonů %	Objednávky	Objednávky ze schůzek	Telefony TM	Schůzky TM
Katka	Vacková	172	0	0	0	22	0	12,79	0	0	0	0
Dita	Sikorová	535	251	92	154	106	75	17,2	26	24,53	8	7
Ondřej	Libosvár	576	250	72	83	86	62	12,5	25	29,07	7	2
Stanislav	Novák	404	243	48	156	77	49	11,88	51	66,23	6	2
Adam	Švanse	107	67	24	21	45	33	22,43	18	40	11	9
Ladislav	Novák	190	131	13	53	38	31	6,84	1	2,63	0	0
Kristýna	Fajfrová	259	107	54	41	65	40	20,85	21	32,31	3	0
Pavol	Adámek	115	101	11	65	67	37	9,57	9	13,43	0	2
Lukáš	Kloz	54	31	5	11	5	5	9,26	1	20	0	0



Obr. 29: Ukázka exportu dat z modulu Report do MS Excel

7 Spuštění a testování systému CSManager

Systém CSManager byl instalován a spuštěn po splnění všech požadavků zadavatele a dokončení vývoje všech důležitých součástí. Instalace zahrnovala nejprve serverovou část, její součástí bylo nahrát soubory sestavení na server a nastavit všechny potřebné parametry serveru a IIS. V systému bylo nastaveno spojení na databázi s aktuálními daty. Pro klientskou část systému jsem vytvořil adresář s potřebnými soubory aplikace a ten zkopíroval všem uživatelům. Systém byl opatřen administrátorským účtem, který sloužil k zavedení ostatních uživatelských údajů do systému a nastavení jejich práv k jednotlivým modulům a funkcím. Před spuštěním systému do ostrého provozu, proběhlo zaškolení uživatelů, kteří budou systém používat. Během několika dnů se CSManager začal plně používat. Některé funkce systému vyžadují testování opravdovým uživatelem a při testování vznikají další požadavky na vlastnosti systému. Po čase ostrého provozu vzniklo několik návrhů na vylepšení, které jsem znovu analyzoval a konzultoval se zadavatelem. Jedním z nich byl i požadavek na otevírání prohlížeče s URL adresou obchodní příležitosti. Pro spouštění a testování systému CSManager jsem používal nástroj pro správu verzí TortoiseSVN. Ten umožňuje sledování a vracení změn ve zdrojovém kódu. Framework Icarus a Centaur také implementuje nástroj pro správu změn sestavení a knihoven. Tento nástroj je velice užitečný při vývoji a vylepšování aplikace. Update aplikace se totiž provede při inicializaci spuštění na každém klientském počítači, kde je nainstalována. Vždy, když jsem vytvořil novou verzi systému s nějakými funkcemi navíc, zkopíroval jsem nové sestavení knihoven s příponou dll na server. Při spuštění klienta systému CSManager na uživatelském počítači se vždy provede kontrola změn v sestavení a provede se update systému na novější verzi.

Závěr

V rámci diplomové práce jsem vyvinul modulární informační systém typu CRM pro správu a řízení obchodních procesů ve firmě CS Technologies. Informační systém s pracovním názvem CSManager ve firmě zvýšil efektivitu obchodního procesu konkrétními funkcemi a nahradil předešlé nevyhovující řešení ve firmě. Mým vlastním přínosem v rámci vytvořeného software bylo rychlé pochopení architektury a efektivní naprogramování systému CSManager postaveném na frameworks Centaur, Icarus a Zeus firmy CS Technologies. Pronikl jsem do pro mne dosud neznámé technologie NHibernate a využil její nástroje pro správu persistentních dat. Z uživatelského pohledu jsem pomocí komponent DevExpress dodal aplikaci profesionální funkce a vzhled.

Motivací bylo naplnit požadavky vycházející z potřeb zadavatele. Při vývoji systému byl kladen důraz na jeho budoucí rozšiřitelnost, prostřednictvím možnosti přidávat další moduly pro další požadované funkce. Projekt vedl k naplnění všech požadavků zadavatele. Nový systém zcela pokryl nedostatky původního systému a vyhovuje všem potřebám zadavatele. Dodržení standardů pro tvorbu systémů firmy CS Technologies usnadní v budoucnu jejímu vývojovému oddělení doplnění systému o další funkce nebo rozšiřující moduly. V současnosti systém obsahuje moduly pro správu uživatelů, obchodních příležitostí, událostí, zákazníků a zakázek. Zaměstnanci firmy jsou schopni systém plně spravovat a ovládat.

Systém by mohl mít vylepšení plánování schůzek pomocí kalendáře nebo tabulky nabízející den, ve kterém jsou schůzky ve shodném městě. Vyplňování údajů o zákazníkovi by mohlo být prováděno automaticky z dostupných informačních kanálů. Plánovaná je také integrace s dalšími informačními systémy v podniku. Systém bude v budoucnu opatřen i uživatelským návodem k obsluze.

Tato diplomová práce a celý projekt mi přinesl mnoho cenných zkušeností. Součástí práce nebylo jen použití mých dosavadních znalostí, ale také studium, zkoumání a implementace nových technologií.

Seznam použité literatury

1. Customer relationship management In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1. 9. 2004, 14. 1. 2010 [cit. 2010-03-27]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Customer_relationship_management>.
2. GÁLA, Libor; POUR, Jan; ŠEDIVÁ, Zuzana. *Podniková informatika : 2., přepracované a aktualizované vydání*. Druhé vydání. Praha : Grada Publishing, 2009. Řízení vztahů k zákazníkům (CRM), s. 496. ISBN 978-80-247-2615-1.
3. ARLOW, Jim; NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací : Objektově orientovaná analýza a návrh prakticky.* Vydání první. Brno : Computer Press, 2007. 568 s. ISBN 978-80-251-1503-9.
4. ROBINSON, Simon. *C# Programujeme profesionálně*. 1. vydání. Brno : Computer Press, 2003. 1130 s. ISBN 80-251-0085-5.
5. *CodeProject* [online]. 25.8.2005 [cit. 2010-04-08]. Importing CSV Data and saving it in database. Dostupné z WWW: <http://www.codeproject.com/KB/database/FinalCSVReader.aspx>.
6. Mono (platforma) In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8. 4. 2006, 9. 2. 2010 [cit. 2010-04-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Mono_(platforma)>.
7. HERNANDEZ, M. J. *Návrh databází*. Praha : Grada Publishing, 2006. 408 s. ISBN 80-247-0900-7.
8. KUATÉ, Pierre Henri, et al. *NHibernate in Action*. Greenwich (Connecticut) : Manning Publications, 2009. 400 s. ISBN 978-1-932394-92-4.
9. *Online Documentation - Developer Express Inc.* [online].DevExpress, 2010 [cit. 2010-03-30]. XtraScheduler Overview. Dostupné z WWW: <http://www.devexpress.com/Help/?document=XtraScheduler/CustomDocument1715.htm>.
10. *Online Documentation - Developer Express Inc.* [online].DevExpress, 2010 [cit. 2010-03-30]. XtraGrid Overview. Dostupné z WWW: <http://www.devexpress.com/Help/?document=XtraGrid/CustomDocument1236.htm>.
11. *Online Documentation - Developer Express Inc.* [online].DevExpress, 2010 [cit. 2010-03-30]. XtraReports Overview. Dostupné z WWW: <http://www.devexpress.com/Help/?document=XtraReports/CustomDocument2161.htm>.

Příloha A – Obsah přiloženého CD

Adresáře:

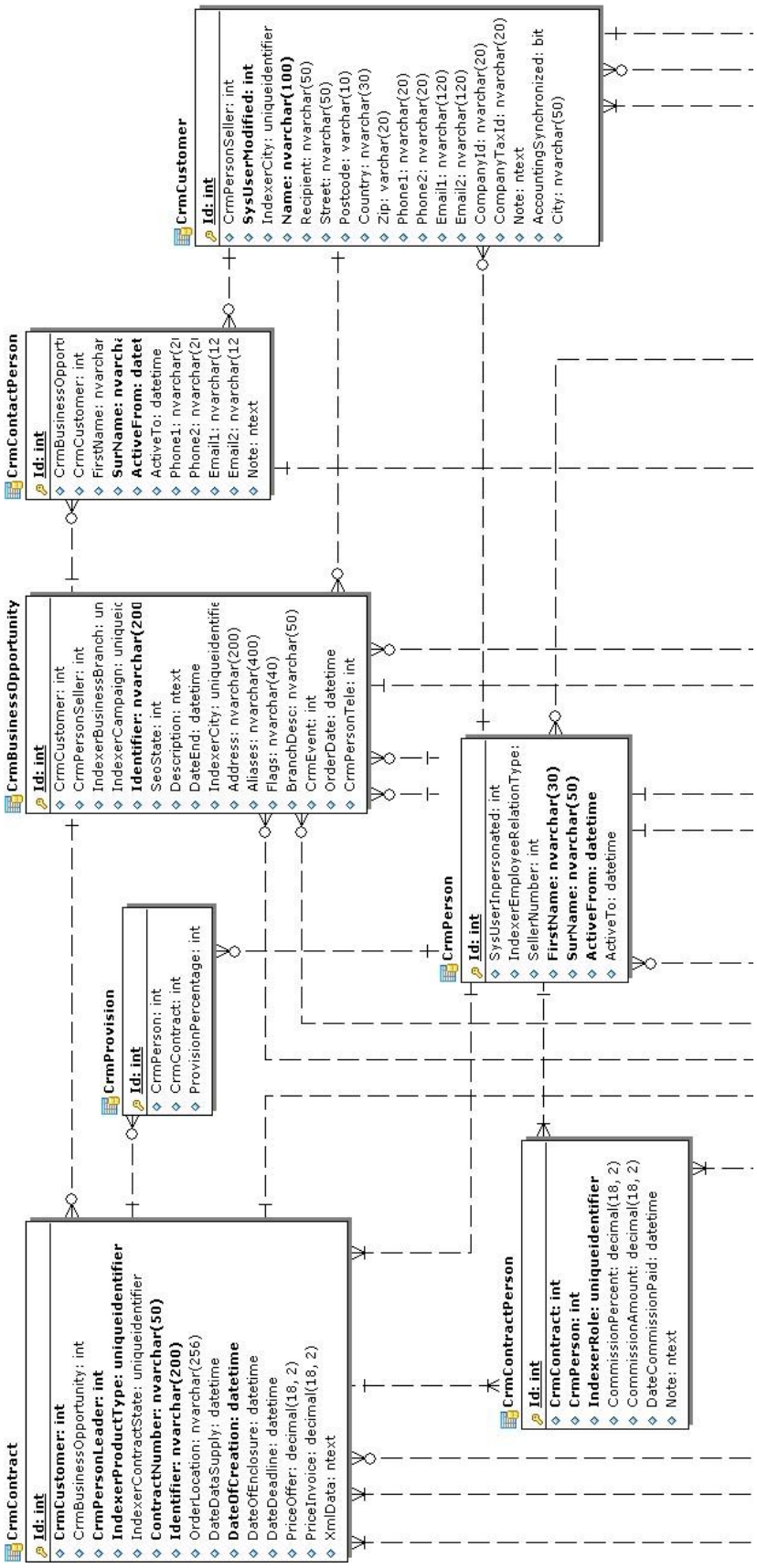
CSManager – obsahuje zip soubor se všemi adresáři a soubory se zdrojovým kódem CRM systému CSManager.

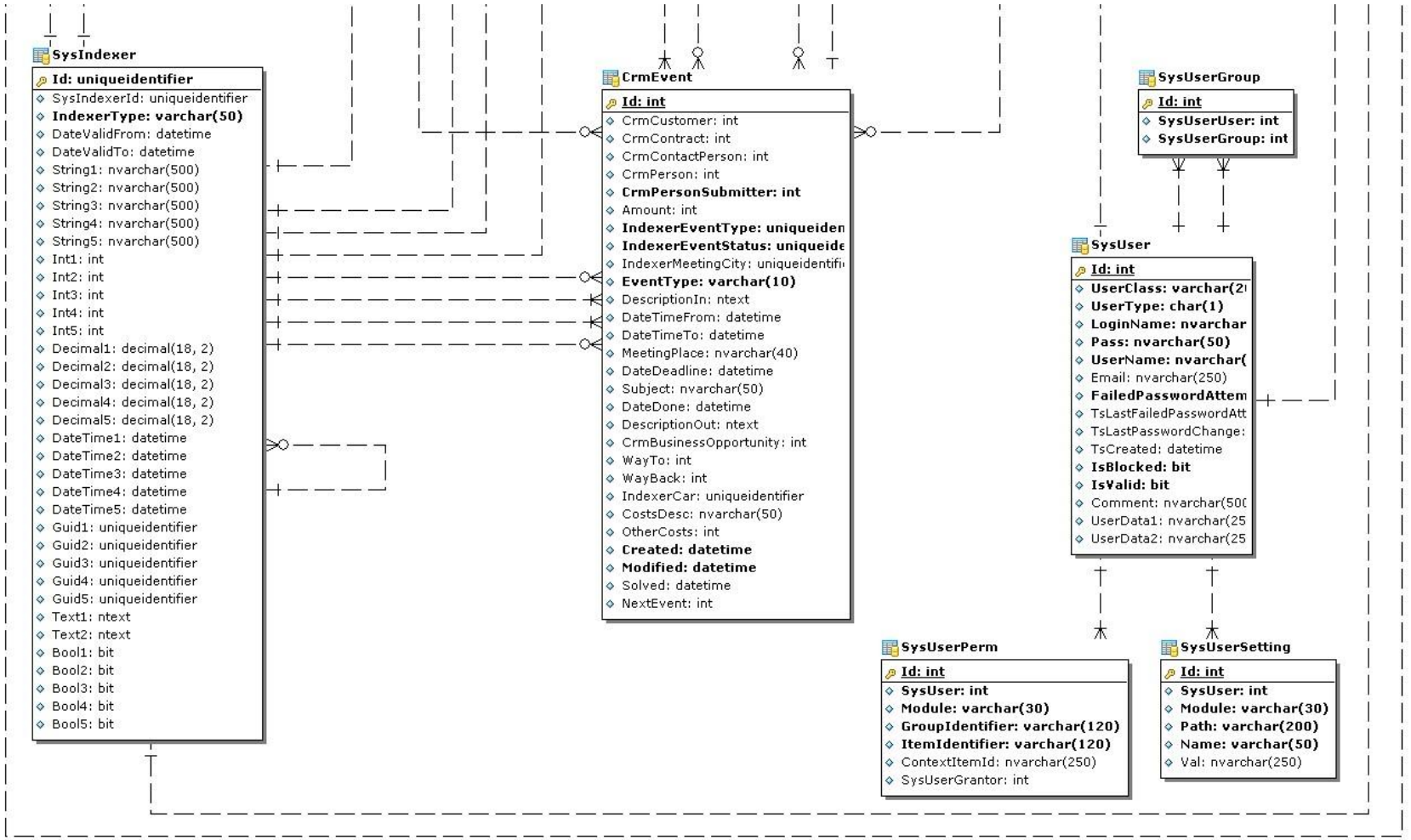
Doc – diplomová práce ve formátu PDF a soubor s UML diagramy pro Enterprise Architect 7.1.

Database – obsahuje souboru databáze, soubor datového modelu pro SQL Manager 2008 a soubor s procedurami použitými při vývoji.

Pictures – obrázky klientské aplikace CSManager.

Příloha B – Fyzický datový model




SysIndexer
Id: uniqueidentifier
SysIndexerId: uniqueidentifier
IndexerType: varchar(50)
DateValidFrom: datetime
DateValidTo: datetime
String1: nvarchar(500)
String2: nvarchar(500)
String3: nvarchar(500)
String4: nvarchar(500)
String5: nvarchar(500)
Int1: int
Int2: int
Int3: int
Int4: int
Int5: int
Decimal1: decimal(18, 2)
Decimal2: decimal(18, 2)
Decimal3: decimal(18, 2)
Decimal4: decimal(18, 2)
Decimal5: decimal(18, 2)
DateTime1: datetime
DateTime2: datetime
DateTime3: datetime
DateTime4: datetime
DateTime5: datetime
Guid1: uniqueidentifier
Guid2: uniqueidentifier
Guid3: uniqueidentifier
Guid4: uniqueidentifier
Guid5: uniqueidentifier
Text1: ntext
Text2: ntext
Bool1: bit
Bool2: bit
Bool3: bit
Bool4: bit
Bool5: bit
CrmEvent
Id: int
CrmCustomer: int
CrmContract: int
CrmContactPerson: int
CrmPerson: int
CrmPersonSubmitter: int
Amount: int
IndexerEventType: uniqueiden
IndexerEventStatus: uniqueide
IndexerMeetingCity: uniqueidentifi
EventType: varchar(10)
DescriptionIn: ntext
DateTimeFrom: datetime
DateTimeTo: datetime
MeetingPlace: nvarchar(40)
DateDeadline: datetime
Subject: nvarchar(50)
DateDone: datetime
DescriptionOut: ntext
CrmBusinessOpportunity: int
WayTo: int
WayBack: int
IndexerCar: uniqueidentifier
CostsDesc: nvarchar(50)
OtherCosts: int
Created: datetime
Modified: datetime
Solved: datetime
NextEvent: int
SysUserGroup
Id: int
SysUserUser: int
SysUserGroup: int
SysUser
Id: int
UserClass: varchar(2
UserType: char(1)
LoginName: nvarchar
Pass: nvarchar(50)
UserName: nvarchar(
Email: nvarchar(250)
FailedPasswordAttem
TsLastFailedPasswordAtt
TsLastPasswordChange:
TsCreated: datetime
IsBlocked: bit
IsValid: bit
Comment: nvarchar(50(
UserData1: nvarchar(25
UserData2: nvarchar(25
SysUserPerm
Id: int
SysUser: int
Module: varchar(30)
GroupIdentifier: varchar(120)
ItemIdentifier: varchar(120)
ContextItemId: nvarchar(250)
SysUserGrantor: int
SysUserSetting
Id: int
SysUser: int
Module: varchar(30)
Path: varchar(200)
Name: varchar(50)
Val: nvarchar(250)