UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Informační systém pro správu zakázek

Lukáš Salfický

Bakalářská práce
2013

Prohlášení autora

Prohlašuji, že jsem tuto práci 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.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 06. 05. 2014 Lukáš Salfický

Poděkování

Chtěl bych poděkovat panu Ing. Josefu Brožkovi za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

Anotace

Obsahem této práce je vytvoření vnitropodnikového webového informačního systému pro správu a vedení zakázek. Systém dále zaměstnancům poskytuje informace o tom, na jakých úkolech mají v daný čas pracovat. V teoretické části této práce jsou popsány základní pojmy a popis ostatních podobných aplikací, které jsou poté porovnány. V praktické části se poté nachází informace a popis zhotovení aplikace samotné.

Klíčová slova

Informační systém, správa zakázek, C#, MVC, MS SQL.

Title

Information management system for procurement

Annotation

The content of this thesis is to develop in-house web based information system for the administration and management of contracts. The system also provides information to employees about what the tasks are not working. In the theoretical part of this thesis describes the basic concepts and description of other similar applications, which are then compared. The practical part is then located and description of the fabrication of the application itself.

Keywords

Information system, order management, C #, MVC, MS SQL.

Obsah

Seznam zkratek

APS	Advanced planning systém
CLS	Common Language Specification
CRM	Customer relationship management
CSS	Cascading style sheets
ERP	Enterprise resource planning
HRM	Human resource management
HTML	HyperText markup language
IS	Information systém
LINQ	Language integrated query
MIS	Management information systém
MSIL	Microsoft intermediate language
MVC	Model-View-Controller
ORM	Object relational mapping
SAAS	Software as a service
SASS	Syntactically awesome style sheets
SQL	Structured query language
VS	Visual Studio

Seznam obrázků

Seznam tabulek

Úvod

Cílem této práce je vytvoření informačního systému pro vedení zakázek, který by firmám a podnikům usnadnil vedení záznamů o zákaznících, právě vedených zakázkách a souvisejících informacích. Zároveň by zavedl přehledný seznam zaměstnanců, kteří na těchto projektech pracují, umožnil zavést systém jednoznačných odpovědností a rychlého úkolování.

Aplikace by měla umožnit výše postaveným zaměstnancům vedení záznamů o projektech a vytváření úkolů nezbytných pro dokončení projektu. Tyto úkoly poté mohou přidělit zaměstnancům, kteří si mohou v systému prohlédnout, jaké úkoly jim nadřízený přidělil a na čem mají právě pracovat.

Jedním z cílů práce je informovat čtenáře o tom co je informační systém a vysvětlit jeho význam pro společnosti, podniky a organizace. Druhá část bude popis samotného infomačního systému a jeho funkcionalit. Práce rovněž ozřejmí, za pomocí jakých nástrojů bude tento systém vybudován.

Velmi důležitým krokem bude správný návrh databáze, se kterou bude aplikace pracovat a využívat ji k ukládání dat. Pokud totiž při tvorbě aplikace dojde k zjištění, že návrh databáze nebyl zcela korektní a tudíž obsahuje nedostatky, bude se muset databáze pracně upravovat.

Uživateli bude vytvořeno jednoduché uživatelské rozhraní, které bude intuitivní a snadné pro použití. Pomocí uvedeného rozhraní bude uživatel přistupovat k datové části aplikace.

1 Teoretická část

1.1 Vytyčení pojmů

V následující kapitole bylo čerpáno z [1], [6], [7], [8], [11], [12], [13], [21] a [22].

1.1.1 Systém

Pod pojmem systém si můžeme představit soubor prvků, které se vzájemně ovlivňují a ovlivňují tak chování systému samotného. Každá část sytému je závislá na alespoň na jednom z dalších prvků systému. Tyto jednotlivé prvky mohou být sloučeny do tzv. podsystémů. V případě, že dojde k rozdělení systému na jednotlivé prvky, ztrácí tak své vlastnosti. Příkladem takovéhoto podsystému může být počítač. Počítač se skládá z jednotlivých částí, které se sjednocují do podsystému. Všechny tyto části jsou závislé na dalších částech počítače. Ovšem pokud počítač rozdělíme na jednotlivé části, ztratí svůj smysl a přestane fungovat.

1.1.2 Informační systém

Pod pojmem informační systém si nepochybně každý člověk představí například jakousi obsáhlou aplikaci, kupříkladu nějakou aplikaci pro řízení ekonomiky podniku. Tato představa však není dostačující. Teď už tedy k tomu, co si pod pojmem informační systém představit. Pojmem informační systém se nejčastěji rozumí nějaká soustava zdrojů a procesů, které s těmito zdroji pracují. „Asi nejvýstižnější definicí je ta, která pod informačním systémem rozumí široký komplex lidí, informací, vlastního systému řízení (tedy programového vybavení), technické prostředky (převážně pak hardwarové pozadí) a systém organizace práce uživatele v příslušné oblasti. Účelem celého komplexu je pak známá šestice - sběr, přenos, aktualizace, uchování a další zpracování dat za účelem tvorby a prezentace informací, které by měly zlepšit výkonnost uživatelů.“

1.1.3 Programování

Programováním rozumíme proces, při kterém se snažíme navrhnout efektivní algoritmus, který popisuje obecné řešení dané. Zároveň jím rozumíme proces, který se snaží převést tento algoritmus do příkazů programovacího jazyka. S tím také souvisí důkladné otestování programu a jeho vyladění. Cílem programování je vytvořit program, který bude vykazovat chování, které od něj očekáváme.

1.1.4 Programovací jazyk

Programovací jazyk je jazyk sloužící k zápisu algoritmů tvořících program. Zároveň je to nástroj, který slouží ke komunikaci mezi počítačem a programátorem.

1.1.5 Relační databázový systém

Databáze je místo, kam si mohou být uloženy nějaké informace, přičemž databázový systém je aplikace umožňující snadné ukládání, úpravu a výběr dat z databáze (relační databáze se tak skládá ze samotných dat a aplikace, která slouží k jejich přístupu – označována jako SŘDB nebo DBMS). Relační databáze patří k nejrozšířenějšímu druhu databáze, i když v dnešní době se také začínají prosazovat objektové databáze nebo

objektově-relační. Tyto typy databází se snaží převzít chování z objektového programování.

Základ relačním databázím položil E. F.Codd již v roce 1970. Základem relačních databází je tabulka, která obsahuje sloupce a řádky. Jednotlivé sloupce představují atributy a jednotlivé řádky tabulky poté můžeme chápat jako záznamy uložené do databáze. Tabulky jsou mezi sebou propojeny tzv. relacemi, které nám umožňují vzájemně propojit související data.

Zástupci relačních databází jsou například produkty Oracle, MySql, MS SQL a další.

1.1.6 Architektonický vzor MVC

Tento architektonický vzor je v dnešní době velmi často používaný a to zejména na poli webových aplikací. Dokonce je součástí mnoha velmi populárních frameworků pro tvorbu webových aplikací jako je například Nette, Zend nebo ASP. NET MVC. Hlavní myšlenkou toho vzoru je oddělení datového modelu, logiky pro řízení dat a uživatelského rozhraní. Aplikace je tedy rozdělena na tři základní komponenty a to Model, Controller a View.

1.1.6.1 Komponenta Model

Tato komponenta udržuje informace o modelech a jejich vlastnostech a vůbec neví o nějakém výstupu pro uživatele. V modelech by se také měly nalézat metody pro práci s atributy modelu.

1.1.6.2 Komponenta View

Komponenta View se stará o zobrazení stránky uživateli a měla by obsahovat co nejméně logiky. Samotná komponenta neví, odkud jí přišla data, jejím hlavním úkolem je, aby je zobrazila uživateli na obrazovce. Tato komponenta se tedy stará o zobrazení dat, které se nacházejí v komponentě Model.

1.1.6.3 Komponenta Controller

Controller je takovým spojovacím prvkem mezi komponentou Model a View. Tato komponenta se tedy vlastně stará o samotný běh aplikace. Controller zajišťuje získání potřebných dat z Modelu, popřípadě je nějak upraví a tyto data předá komponentě View, která je poté reprezentuje uživateli.

1.1.6.4 Princip funkce

Princip tohoto vzoru si popíšeme na příkladu. Řekněme, že uživatel zadá do prohlížeče nějakou URL adresu. Podle zadaných parametrů aplikace pozná, který *controller* má zavolat, a jsou mu předány další zadané parametry. Poté dojde k zavolání *modelu*, který nám poskytne žádaná data. Tyto informace si *controller* uloží a vyrenderuje *view*, kterému jsou předány informace, aby je mohl vložit do připravené šablony.

1.1.7 ORM

Objektově relační mapování si můžeme představit jako techniku, která nám při objektově orientované programování usnadňuje práci s databází. V relační databázi jsou vlastně data uložena ve formě řádků v tabulkách, ale v objektovém programování se na data díváme jako na objekty, které můžou být v databázi uloženy ve vícero tabulkách. ORM nám usnadní konverzi právě mezi daty uloženými v databázi a objekty v objektovém programování.

1.1.8 Entity framework

Entity framework je sada technologií pro vývoj na platformě ADO. NET a jedná se o jeden z ORM frameworků. První verze tohoto frameworku se nacházela již v. NET verze 3.5 a postupem času se velmi vyvíjí. Nyní si můžeme stáhnout již šestou verzi tohoto frameworku. V této době je Entity framework vyvíjen přímo společností Microsoft a je napsán v jazyce C#. V roce 2013 se společnost Microsoft rozhodla vydat Entity Framework jako open source projekt. Tím je zajištěna širší podpora a různé vylepšení ze strany samotných uživatelů. Nyní všechny z nejpoužívanějších databází (MS SQL, Oracle i MySQL) podporují vyžití Entity frameworku, nicméně Entity framework byl primárně vyvíjen pro databázi MS SQL, proto se této databázi dostává největší podpory. Pokud chcete využít Entity framework například s databází MySQL je potřeba stahovat další dodatečné knihovny pro podporu funkčnosti.

1.1.9 CSS

CSS je jazyk, který programátorovi umožňuje zápis kódu, který bude popisovat způsob, jak budou napsané webové stránky vypadat. Velkou výhodou je, že programátorovi umožňuje oddělit obsah stránky od toho, jak vlastě bude design stránky vypadat.

1.1.10 CSS preprocesory

CSS preprocesory nám z našeho zapsaného kódu v určité syntaxi vygeneruje kaskádové styly, které poté použijeme pro prohlížeč. V současné době jsou nejrozšířenějšími preprocesory hlavně Less a Sass. A co vlastně takové preprocesory umí? Mají hned několik výhod oproti klasickým kaskádovým stylům, dovolí nám například deklarovat proměnné. To je velmi užitečná vlastnost. Umožní nám si deklarovat nějakou proměnnou pro barvu a poté ji používáme, kde potřebujeme a pokud se nám poté barva přestane líbit, tak nemusíme procházet celé CSS, ale úpravu barvy provedeme jen na jednom místě. Pro tyto preprocesory není problém ani definice uživatelských funkcí, které můžete volat na různých místech kde je to právě zapotřebí. Zvládají i různé matematické operace. Pro Sass dokonce není problém ani použití podmínek nebo cyklů.

1.1.11 Compass

Compass je Sass (problematika Saas byla vysvětlena v kapitole 1.1.10) framework, který obsahuje různé nástroje pro zjednodušení a zefektivnění práce při vytváření kaskádových stylů za pomoci Sass.

1.1.12 Framework

Pod pojmem Framework by se dala představit sada nástrojů, knihoven a konvencí, které by měli programátorovi usnadnit vývoj aplikace. Poskytuje programátorovi základní kostru programu. Využití najde hlavně u rozsáhlých projektů, kde dokáže ušetřit velmi mnoho času.

1.2 Důležitost informačního systému pro firmy

V následující kapitole je čerpáno z [10].

Informační systémy jsou v dnešní době nedílnou součástí běhu mnoha různých firem a společností. Dříve byly informační systémy výsadou hlavně velkých firem, ale postupem času se začaly informační systémy zavádět i v menších firmách. A z jakého důvodu jsou vlastně informační systémy tak hojně využívány?

Hlavní funkcí informačního systému a také přínosem, by mělo být poskytnutí informace o stavu podniku. Většina firem od informačního systému také očekává, že budou komplexní. Další výhodou informačních systémů je evidence dat a jejich pozdější vyhodnocení. To také například umožňuje vytvořit si obrázek o aktuálním běhu společnosti a usnadňuje budoucí strategické rozhodnutí kam firmu směřovat nebo výrobě jakého produktu se v daný čas věnovat. Můžeme si také udělat přibližný odhad o tom, kam se firma bude v budoucnu ubírat, popřípadě zjistit co je třeba změnit, aby na tom firma měla vyšší finanční výkonnost.

Další výhodou je například automatizované zpracování údajů dodaných do informačního systému. Informační systém pak automaticky může provádět různé výpočetní operace nad těmito daty a po výpočtu vytvořit grafický výstup výsledku.

Informační systém také poskytuje informace o zaměstnancích a jejich práci. Každý zaměstnanec je odpovědný za nějakou část práce. Za pomoci informačního systému můžeme zjistit, jak si stojí v porovnání s jinými zaměstnanci ve firmě na stejné pozici. Dokonce nám může být poskytnuta informace o tom, v kolik hodin daný zaměstnanec přišel do práce a v kolik hodin z ní odcházel.

V dnešní době se stalo celkem běžné provozování informačního systému formou outsourcingu (využíváním modelu SaaS). To znamená, že si daný informační systém vlastně jen pronajmeme. Hlavní výhody této formy může nalézt v personální oblasti. Odpadá tím potřeba zaměstnávat vlastní tým odborníků, protože o informační systém se starají zaměstnanci firmy, která pronájem poskytuje.

1.2.1 Nevýhody využití informačních systémů

Mezi hlavní nevýhody patří, že v informačních systémech se nachází mnoho velmi citlivých informací a jejich zneužití by mohlo mít pro firmu fatální následky. Tyto data musí být dostatečně ochráněna a to jak před zneužití uživatelem, ale také například proti tomu, kdyby došlo k selhání techniky. A právě proto firmy využívají různé ochranné prvky, aby zamezili ztrátě nebo zneužití dat souvisejících s firmou. Důležité je to, aby si

firmy využívající informační systémy uvědomily, že jim nehrozí útok nejen zvenčí, ale velmi často dochází k útoku a zneužití dat vlastními zaměstnanci.

1.2.2 Bezpečnostní opatření

Informační systém by neměl do aplikace pustit uživatele, který nebyl přihlášen pomocí přihlašovacího jména a hesla. Každý uživatel by měl mít jasně přidělené práva za pomoc přidělení role tomuto uživateli. Přičemž každá role by měla mít práva jen na činnosti, které jsou nezbytně nutná k vykonávání jeho pracovní činnosti.

1.2.3 Způsoby ochrany dat

Mezi hlavní způsoby ochrany patří:

- zálohování

- správné přidělení rolí

- nutnost přihlášení uživatele

- ukládání informací o činnosti uživatele v systému

1.2.4 Základní druhy informačních systémů

Informační systémy se dělí do různých skupin a to především podle toho, k jakému hlavnímu záměru byly navrženy. Níže budou popsány základní druhy informačních systému.

V této kapitolo bylo čerpáno z [5], [23] a [24].

1.2.5 ERP

Enterprize resources planing, neboli systém na správu zdrojů firmy je velmi obecně zaměřený a tím pádem také komplexní informační systém integrující různé činnosti probíhající ve společnosti. Zaměřuje se například na logistiku, správu skladu, fakturaci nebo účetnictví. Tyto systému mohou také velmi pomoci při optimalizaci procesů nebo jejich automatizaci. Tento typ informačního systému také velmi dobře poslouží při kontrole toho, jestli projekt nebo nějaká činnost probíhá tak jak má a nevznikl jakýkoliv problém. Výhodou takového systému je, že poskytuje aktuální pohled na výkonnost společnosti. ERP systémů je velké množství a liší se hlavně svojí komplexností, což je zapříčiněno hlavně tím, že systém pro velké společnosti bude muset obsahovat více funkcí z více oblastí než systém, který bude využívat jeden živnostník nebo malá firma.

Příklad ERP systémů:

- ABRA G4
- K2 SOFTWARE
- Money S4 a S5
- Pohoda

1.2.6 MIS

Management information system, neboli manažerský informační slouží především pro zkvalitnění organizace a běhu firmy. Hlavním účelem těchto systému je

shromažďování informací v podobě dat například v databázi. Tato data jsou poté použita, pro zkvalitnění běhu společnosti. Pomáhají při plánování a předpovědi situací, což zvyšuje schopnost společnosti reagovat na různé situace, které mohou nastat popř., které by mohly v budoucnosti nastat. Velmi se uplatní také při určení správné marketingové taktiky. Stejně jako v případě ERP systémů, můžeme na trhu najít mnoho rozmanitých druhů těchto systému, které mají pomoci ve vedení společnosti.

Příklad MIS systémů:

- Corporate Financial Management (CFM)
- HELIOS
- KYBOS
- ARBES BI

1.2.7 APS

Advanced planning system, neboli systém pokročilé plánování slouží především pro zkvalitnění fáze plánování. Jejich účelem je zkvalitnit a zrychlit proces plánování. Pomáhají také odhadnout, kdy přibližně dojde k dokončení daného procesu nebo projektu. Tento informační systém bývá velmi často spojován se systémem ERP. Při správné implementaci společně dokáží ještě více zdokonalit běh společnosti a zlepšit kontrolu nad probíhajícími procesy.

Příklad APS systémů:

- Součástí ERP systému Infor SyteLine
- Součástí ERP systému Infor ERP
- APS KARAT
- APS One

1.2.8 CRM

Customer relationship management, je systém zaměřený na management vztahu se zákazníkem. Jeho hlavním účelem je uchování dat o zákaznících a jejich zpracování a utřídění. Tyto data se potom velmi často statisticky zpracovávají. Za pomoci těchto dat se dají jednoduše sledovat veškeré obchodní aktivity společnosti.

"Cílem CRM je především zlepšit cílení služeb, lépe porozumět zákazníkům a identifikovat jejich konkrétní potřeby. To umožňuje budovat dlouhodobě prospěšné vztahy se zákazníky a tím vytěžit z jednoho zákazníka větší zisk. Protože stávající zákazníci jsou pro firmu nejhodnotnější, vyplatí se pomocí CRM systémů zajistit si jejich věrnost a důkladně o ně pečovat."[1]

Příklad CRM systémů:

- BLUEJET CRM
- RAYNET CRM
- Součást ERP systému K2
- InTouch CRM

1.2.9 Time and attendance system

Tyto docházkové systémy se velmi často používají ve spojení s například čipovou kartou nebo jiným zařízením. Hlavním účelem je monitorování toho, v jaký čas

zaměstnanci přišli do práce a kdy z ní také odešli. Tento systém se také velmi často využívá k monitorování toho, kdo vstoupil do jaké části podniku.

Příklad docházkových systémů:

- Alveno
- ETEND
- systém od společnosti CoNet
- OKbase

1.2.10 HRM

Human resource management jsou systémy, které pomáhájí řídit lidské zdroje ve společnosti. Zaměřuje se hlavně na zlepšení vnitrofiremní komunikace a správnou a efektivní dělbu práce a tím se snaží rychleji dosáhnout požadovaných strategických cílů.

Příklad systémů pro HRM:

- HR software
- BambooHR

1.3 Podobné informační systémy a jejich porovnání

1.3.1 Výběr IS pro správy zakázek

Na internetu bylo k nalezení okolo 10 nejrůznějších aplikací pro správu zakázek nebo projektů. Mnoho z nich, ale nebyly aplikace určené pouze pro tento účel, ale byly součástí nějaké komplexnější aplikace. Z těchto aplikací byly vybrány čtyři nejzajímavější z autorova pohledu a také těch, u kterých byla možnost si je bezplatně vyzkoušet nebo byla aplikace pečlivě popsaná a ukázaná na webové stránce. Následně bylo provedeno srovnání těchto aplikací.

1.3.2 Porovnání informačních systémů

1.3.2.1 EasyProject

První testovaná aplikace byla aplikace od s názvem EasyProject. Tato aplikace je vyvíjená českou společností Easy Software s.r.o. Společnost je dodavatelem webových a e-commerce aplikací. Webové stránky této společnosti jsou přehledné a uživatelsky přívětivé.

Společnost dodává software pro řízení projektů ve čtyřech licencích Mini, Standard, Business a Individual. První tři mají pevně danou kostru, ale poslední jmenovaná verze je společností upravena do podoby, jakou si uživatel sám určí.

Cena prvních tří jmenovaných softwarů je 3000,- Kč a jednotlivé verze se liší měsíčním poplatkem za údržbu a provoz softwaru. Poslední verze nemá pevně stanovenou cenu, ta se odvíjí až dle náročnosti zpracování resp. specifikace uživatele.

Společnost Easy Software s.r.o. jako jediná z testovaných, poskytuje i zkušební verzi tohoto systému online. Samotný informační systém má velmi pěkné uživatelské rozhraní, které je však o něco složitější na zorientování, kde se jaká funkce nachází. Aplikace poskytuje širokou škálu funkcí, které uživateli poskytují různé funkce v oblasti plánování projektů, přiřazování úkolů zaměstnancům nebo plánování různých pracovních schůzek. Kromě těchto výše uvedených funkcí poskytuje i velké množství dalších a systém obsahuje i pěkné administrační rozhraní a nalezneme zde i docházkový systém.

1.3.2.2 Firon

Další aplikaci, kterou jsem otestoval, byla aplikace od společnosti UVM intractive s.r.o. Tato společnost nabízí software nazvaný Firon. Společnost se zabývá hlavně vývojem internetových prezentací, využívajících redakční systémy. Mimo jiné také vyvíjí e-shopy a věnuje si i tvorbě grafiky pro klienty a množství dalších činností. Společnost má pěkné a přehledné webové stránky. Podle tvůrce aplikace je tento nástroj určen pro malé a střední společnosti.

Ceny licence informačního systému Firon začínají na částce 19 900,- Kč bez DPH. Informační systém poskytuje množství modulů pro řízení projektů, správu úkolů v týmu, vedení zákaznické evidence, záznam kontaktů s dodavateli a mnoho dalšího.

Informační systém této společnosti nabídne uživateli také vcelku širokou škálu funkcí, ale oproti ostatním testovaným systémům jich uživateli nenabídne tolik. Také uživatelské rozhraní tohoto systému nijak nenadchne, ale také neurazí.

1.3.2.3 Twis

Dalším testovaným informačním systémem je systém Twis od české společnosti TWIS systém s.r.o. Společnost se zabývá hlavně řešením systému pro správu projektů a vnitro-firemní komunikaci. Webové stránky této společnosti mají velmi pěkně vypadající design, jsou přehledné, ale přeci jenom mají jednu vadu a tou je nevhodně zvolené písmo, které se jeví jako kostrbaté a některá písmenka pak vypadají jako by byla zvýrazněna tučně.

Společnost nabízí systém za měsíční poplatek odvozený podle počtu licencí, které byly zakoupeny a podle dodatečných modulů, které si zákazník vybere. Tyto moduly jsou však v měsíčním poplatku zahrnuty jen jednou za celý systém nezávisle na počtu licencí. Z toho vyplývá, že cena funkce na jednu licenci klesá s počtem zakoupených licencí. Může být zakoupen libovolný počet licencí s tím, že společnost poskytuje prvních pět zdarma. Cena jedné licence bez přídavných funkcí vyjde na 195,- Kč bez DPH. Celková cena za všechny přídavné moduly vyjde na 2470,- Kč měsíčně bez DPH.

Samotný informační systém této společnosti se skládá z různých modulů. Informační systém je velmi přehledný, a tudíž intuitivní na použití. Systém poskytuje širokou škálu funkcí, které může uživateli poskytnout. Avšak oproti informačnímu systému EasyProject neobsahuje tolik rozmanitých funkcí.

1.3.2.4 Evidio

Posledním testovaným informačním systémem je EvidioApp.cz od společnosti evidio s.r.o. Společnost evidio s.r.o. se specializuje především na vývoj informačních systémů a webových portálů. Její webové stránky byly ohodnoceny jako nejhorší, jejich design se jeví jakoby bez nápadu a v podstatě se dá říci, že uživatele moc nezaujmou. V této oblasti je tato společnost o krok pozadu oproti webovým stránkám společností, jejichž informační systém byl hodnocen.

Informační systém společnosti je dostupný ve čtyřech odlišných tarifech, odlišných hlavně tím, kolik uživatelů může informační systém používat a velikostí poskytnutého diskového prostoru. Prvním je tarif Solo a jak už název napovídá je jen pro jednoho uživatele. Tento tarif je dostupný za cenu 1250,- Kč měsíčně. Dalším tarifem je Basic, který je určen až pro pět uživatelů a je dostupný za 1500,- Kč měsíčně. Tarif Plus je až pro patnáct uživatelů a společnost ho nabízí za cenu 2500,- Kč měsíčně. Posledním tarifem je Premium, který je až pro čtyřicet uživatelů a je k dispozici za částku 6000,- Kč měsíčně. Všechny uvedené ceny jsou bez DPH.

Informační systém této společnosti nabídne uživateli dostatek rozmanitých funkcí pro řízení projektů a úkolů. Stejně jako informační systém od společnosti UVM intractive s.r.o. je design tohoto systému trochu pozadu oproti zbylým dvěma, které byly testovány.

1.3.3 Výsledek porovnání

Hodnocení jednotlivých aplikací bylo provedeno pomocí přiřazování bodů, přičemž nejmenší možný počet bodů, které mohla aplikace při hodnocení obdržet, bylo 1 a maximální možná známka byla 10. Hodnoceny byly jednak webové stránky výrobce a pak samozřejmě informační systém samotný.

Název aplikace	Easy Project	Twis	Evidio	Firon
Vzhled webu	9	7	5	8
Přehlednost aplikace	5	6	6	7
Množství funkcí	8	6	6	5
Vzhled aplikace	8	7	5	6
Cena licence	7	8	9	3
Celkové hodnocení	7,4	6,8	6,2	5,8

Tabulka 1 - Hodnocení jednotlivých aplikací

V testu informačních systémů obstály všechny čtyři testované aplikace. Všechny obsahují dostatečné množství funkcí pro správu zakázek, projektů a úkolů, které uživateli nabídnou. Jako vítěz z porovnaných informačních systému byl zvolen informační systém společnosti Easy Software s.r.o. Systém této společnosti nabídne uživateli početnou škálu možností, jak spravovat běh společnosti a to vše za rozumnou cenu. Některé funkce a

jejich množství by mohli být pro uživatele trochu matoucí, ale společnost ke všem licencím nabízí v rámci ceny licence i možnost proškolení uživatele jak systém používat. Jako druhý se umístil systém Twis, který uživateli sice nenabídne tolik funkcí k využití, ale celá aplikace má vzhledově velmi příjemné prostředí a je přehledná. Také cena licence jedné je velmi zajímavá. Jako třetí se umístila aplikace Evidio, která uživateli nabídne dostatečné množství funkcí, aby byl schopen dobře vést zakázky a projekty. Tato aplikace má také nejnižší cenu za licenci, ze všech testovaných aplikací. V čem tato aplikace ovšem zaostává, je hlavně vzhled. Jako poslední se umístila aplikace Firon, která sice nabídne uživateli dostatek funkcí, ale design aplikace není moc hezký a hlavně cena jedné licence je oproti ostatním testovaným aplikacím mnohem vyšší.

2 Základní použité prostředky

V této kapitole bylo čerpáno z [1], [4], [17], [18], [19] a [20].

2.1 Visual Studio

Jde o vývojové prostředí od společnosti Microsoft a jde asi o nejlepší sadu nástrojů pro vytváření aplikací pro platformu Windows. Jak již bylo řečeno, slouží hlavně k vývoji různých aplikací jako například desktopových aplikací, webových aplikací, aplikací pro mobilní telefony nebo konzolových aplikací. Mimo to ale poskytuje možnost tvorby mnoha dalších aplikací.

VS obsahuje editor kódu, který funguje podobně jako jiné editory jiných nástrojů. Pomáhá programátorovi v mnohých směrech a to hlavně tím, že kontroluje syntaxi napsaného kódu a podtrhává špatné konstrukce nebo automaticky doplňuje různé konstrukce. V podstatě stačí, znát jen příslušnou zkratku, kterou programátor napíše a VS vygeneruje konstrukci za něj. Velmi užitečné je i automatické napovídání a doplňování kódu ze seznamu, kterým VS napovídá.

VS také obsahuje velmi užitečný nástroj a to debugger, který využije asi každý programátor. Umožňuje nastavení breakpointu nebo watche a velmi tím usnadňuje práci programátorovi, který někde udělal chybu a snaží se ji najít a opravit.

Ve VS najdeme také designer, který slouží hlavně pro desktopové aplikace, ale najde uplatnění i u jiných aplikací. Designer nám umožní snadné rozmístění vybraných komponent na formuláři a tím ušetří programátorovi hodně času.

Prostředí VS je možné také hodně měnit a přizpůsobit tomu, jak to bude programátorovi nejvíce vyhovovat. Jednoduše lze změnit barvu celého rozhraní ze světlé na tmavé, které hodně programátorů preferuje, jelikož hodně šetří unavené oči. Lze však měnit i menší detaily, jako například velikost písma, barvu písma nebo to jakou barvou se budou podtrhávat syntakticky špatné konstrukce.

VS v dnešní době nabízí mnoho produktů jako například Microsoft Visual C++, Microsoft Visual C# nebo Microsoft Visual Basic .NET. První verze VS byla vydána v roce 1997 a jmenovala se VS 97. Následovaly další verze tohoto úspěšného produktu a poslední verze byla pojmenována VS 2013 a jak již název napovídá, byla vydána na konci roku 2013. Každá z verzí přinesla řadu užitečných změn a rozšíření. Ruku v ruce s vývojem VS šel i vývoj .NET frameworku. VS najdeme i v různých verzích, které se liší tím, jak mnoho nástrojů programátorovi poskytnou. Jednotlivé verze jsou Express, Professional, Premium, Ultimate a Test Professional. Verze Express je k dostání zdarma, dokonce ke komerčnímu použití, ale samozřejmě nemá plnou podporu jako ostatní produkty. Nejvyšší podporu a počet rozšíření má verze VS Ultimate.

2.2 C#

C# je objektově orientovaný programovací jazyk od společnosti Microsoft. První verzi tohoto jazyka vydala společnost Microsoft v roce 2002 a byla nazvána C# 1.0. Tento programovací jazyk se vyvíjí společně s .NET Frameworkem. Tento programovací jazyk vznikl hlavně na základě jazyků Java a C++ a z obou se snaží brát to nejlepší. Prakticky ihned po vydání si získal mnoho obdivovatelů a v dnešní době je to jeden z nejpoužívanějších programovacích jazyků. Využívá se k programování nejrůznějších aplikací pro počítače nebo mobilní zařízení. Nynější verze jazyka je verze C# 5.0 a chystá se verze C# 6.0, která přinese další množství rozšíření jazyka, stejně jako předešlé verze.

2.2.1 Verze jazyka

C# 1.0

První verze jazyka obsahovala základní konstrukce potřebné pro objektové programování, a jak již bylo zmíněno, vychází z programovacího jazyka C++ a Java.

C# 1.2

První vylepšení tohoto jazyka, které bohužel nepřineslo žádné významné novinky, jež by programátorovi ulehčilo práci.

C# 2.0

Největší novinkou této vydané verze je generika, která odstraňuje většinu přetypování a zlepšuje tím potenciální rozbor kódu. Další novinkou je například přidání nullable typů nebo anonymních metod.

C# 3.0

Tato verze přinesla hlavně vylepšení týkající se LINQ jako například inicializátory objektů a kolekcí, anonymní typy nebo lambda výrazy.

C# 4.0

Největší novinkou této verze je dynamické typování. Tyto typy hlavně usnadňují práci s objekty z prostření Python, Ruby nebo Javascript.

C# 5.0

V této verzi je největší novinkou způsob volání a používání asynchronních metod.

2.3 .NET

Je to softwarová platforma, která umožňuje vývoj mnoha různých druhů aplikací. Často je používána k vývoji klasických, webových nebo mobilních aplikací. Součástí je také velká sada knihoven, které obsahují širokou škálu tříd a funkcí, které se snaží usnadnit práci a čas programátorovi.

První verze tohoto frameworku se objevila v roce 2002. Postupem času byly vydávány další verze, které již byly součástí systému Windows a rozšiřovaly množství funkcí, které framework obsahuje. Poslední vydanou verzí je .NET 4.5.1, který byl vydán koncem roku 2013 a přinesl další řadu změn a vylepšení jako předcházející verze frameworku.

Při psaní programů pro .NET lze využít různých programovacích jazyků. Podmínkou je to, že tyto programovací jazyky musí splňovat určité specifikace, které se nazývají Commnon Language Specification (CLS). Tyto specifikace splňují například jazyk Visual Basic, C++, C#, Jscript nebo J#.

Kompilace programů v .NET frameworku probíhá trochu odlišně například od kompilace programu napsaného v C++. Výsledkem kompilace v C++ je strojový kód, ale v .NET frameworku tomu tak není. Kompilace zde probíhá velmi podobně jako u jazyka Java. Programy nejsou kompilovány přímo do strojového kódu, ale do kódu jemu podobného a to MSIL, který je nezávislý na zvolené platformě. Tento kód je po překladu zabalen s dalšími soubory do takzvané assembly. A teprve při spuštění se provádí překlad do strojového kódu, přičemž se překládá jen ta část kódu, která je zrovna potřeba.

2.4 HTML

V překladu je HTML značkovací jazyk pro hypertext. Jde o jazyk učený pro vytváření webových stránek. Jeho hlavní účel je popsání jednotlivých částí dokumentu značkami specifickými pro danou verzi, definující strukturu a části dokumentu popisující webovou stránku.

Struktura HTML dokumentu:

- Doctype (označuje použitou verzi HTML a jak má dokument zpracovat)
- Kořenový element
- Hlavička
- Tělo dokumentu

2.4.1 Verze HTML

HTML 1.0

Tato verze byla vydána mezi rokem 1989 a rokem 1994. Tato verze byla velmi omezená v oblasti vytváření stylu webové stránky, například nešlo vůbec používat tabulky, měnit pozadí stránky nebo měnit použitý font.

HTML 2.0

HTML 2.0 byla vydána v roce 1995 a vylepšil přechozí verzi o mnoho užitečných prvků.

HTML 3.2

Tato verze byla vydána v roce 1997 a opět vylepšila přechozí verzi HTML a konečně přinesla podporu používání tabulek, obrázků a formulářů. V této verzi byla také již možnost změnit pozadí stránky, rovněž se také objevila podpora využívání CSS.

HTML 4.01

Tato verze byla vydána roku 1999 a vylepšila podporu využívání stylů. Snaží se oddělit obsah stránek a to jak bude stránka vypadat. Také přibila podpora skriptování.

HTML 5.0

Tato verze byla vydána v roce 2012 a je stále ve vývoji, doporučení k jejímu používání se předpokládá na konec roku 2014. Přináší mnoho změn, novinek a vylepšení. Tato verze se zaměřila na zlepšení přehlednosti kódu a přináší mnoho nových prvků jako je například

- <aside>
- <section>
- <header>
- <footer>
- <article>
- <nav>

2.5 LINQ

LINQ

LINQ je skupina nástrojů určená pro dotazování nad daty, která také velmi usnadňuje práci s nimi. LINQ je od verze C# 3.0 a .NET 3.5 integrován přímo v jazyce C#, což přináší tu výhodu, že syntaktická kontrola může probíhat při překladu.

Existují tři různé implementace toho jazyka a to:

- LINQ to Object
- LINQ to XML
- LINQ to ADO.NET

LINQ to Object

Umožňuje vytvářet dotazy kolekcemi v paměti a dalšími třídami, které implementují rozhraní IEnumerable.

LINQ to XML

Pracuje s daty uloženými jako XML a umožňuje tento dokument přeměnit na kolekci objektů typu XElement a nad nimi provádět dotazy.

LINQ to ADO.NET

Tato implementace se rozděluje na další tři a to LINQ to SQL, LINQ to Entities a LINQ to DataSet.

LINQ to SQL

Stará se o mapování mezi vlastní typy v rozhraní. NET Framework a tabulkami na SQL Serveru.

LINQ to Entities

Jedná se o objektově relační mapování, které namísto použití fyzické databáze používá konceptuální Entity Data Model, který popisuje strukturu dat.

LINQ to DataSet

Implementuje použití LINQ pro ADO .NET datasety, což jsou vlastně sady dat.

LINQ dotaz poté může mí dvě různé podoby. První podoba bude vypadat asi následovně:

```
var dotaz = from j in jmena where(j.Length > 8) select j;
```

nebo za použití lambda výrazů a dotaz poté bude vypadat takto:

```
var dotaz = jmena.Where(j => j.Length > 5).Select(j => j);
```

Přičemž z prvního dotazu C# vygeneruje kód, který souhlasí s druhou podobou LINQ dotazu.

3 Praktická část

3.1 Představení aplikace

Informační systém, který byl vytvářen v rámci této práce, je webovou aplikací. Požití této formy má oproti desktopové aplikaci hlavní výhodu v tom, že se aplikace nemusí na všechny počítače instalovat, ale stačí mít webový prohlížeč a přístup k internetu.

Aplikace by měla sloužit jako informační systém určený hlavně k správě zakázek, projektů a věcí, které s tím úzce souvisí.

3.2 Využití

Informační systém by měl najít využití hlavně u malých a středních firem, které by chtěly lépe a rychleji organizovat, své fungování.

Aplikace má dvě hlavní využití, které se liší tím, s jakou rolí se uživatel do systému přihlásí. Pokud se přihlásí uživatel s právy na přístup do administrační části, tak bude moci upravovat různé data, které se týkají vedení projektů a zakázek. Může do systému přidávat nové projekty nebo je upravovat, může upravovat finance resp. náklady a příjmy týkající se daného projektu. Také může přidat nového zákazníka do databáze nebo data sjednaných schůzek. Tento uživatel má také právo k projektům přidávat konkrétní úkoly, které jsou nezbytné udělat pro dokončení projektu a tyto úkoly poté přiřadí zodpovědným uživatelům. Mimo tyto funkce může také posílat zprávy ostatním uživatelům a tímto způsobem s nimi komunikovat. Druhý pohled uvidí uživatel, který je běžným zaměstnancem. Tomuto uživateli je zakázán přístup do administrační části aplikace. Uživatele s těmito právy by měla zajímat hlavně ta část aplikace, kde mu jsou zobrazeny úkoly které, mu byly nadřízeným přiřazeny. U těchto úkolů si uživatel může prohlédnout detail tohoto zadání a poté co úkol dokončí, může ho označit jako dokončený. Uživatel v této roli má také možnost posílat zprávy ostatním uživatelům. Může se tak například zeptat nadřízeného na nějaký úkol, pokud mu nebude jasné co má v rámci úkolu dělat.

3.3 Teoretické zabezpečení

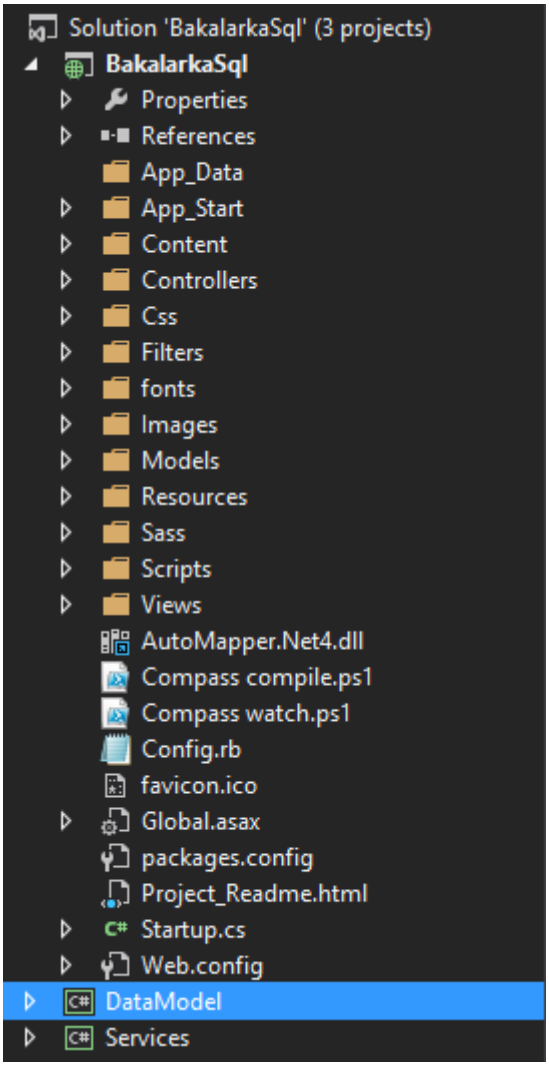
V aplikaci je potřeba zajistit zabezpečení v podobě nutnosti přihlášení do systému za pomoci přihlašovacího jména resp. emailu a hesla. Zároveň je třeba rozdělit práva uživatelů na různé činnosti podle toho, jakou roli bude uživatel po přihlášení zastupovat.

3.4 Realizace aplikace

Při tvorbě aplikace byl využit software od společnosti Microsoft a to konkrétně Visual Studio 2013. Aplikace byla napsána za pomoci programovacího jazyka C# a frameworku ASP. NET MVC. Tato volba byla provedena hlavně z důvodu, že autor měl zkušenost s tímto frameworkem z práce a z důvodu, že autor chtěl v rámci této práce dál rozvíjet svoje schopnosti a dovednosti v programování webových aplikací za pomoci tohoto frameworku.

3.5 Adresářová struktura aplikace

Ihned po založení projektu Visual Studio vygeneruje poměrně velké množství složek a souborů, které nám usnadní orientaci v projektu. Můžeme zde například nalézt základní složky Controllers, Models a Views, kam se vytváří jednotlivé komponenty architektonického vzoru MVC. Mimo tyto složky zde najdeme i velmi důležitý soubor Web.config, kde například nalezneme konfiguraci spojení s databází nebo mnohé další konfigurace.



Obrázek 1- Adresářová struktura

3.6 Nasazení webu

Před nahráním aplikace na hosting, je nutné zjistit, jestli tento hosting podporuje technologie, které jsou potřebné pro nasazení. Je potřeba najít hosting, který podporuje technologie ASP. NET specifické verze, ASP. NET MVC 5 a také jestli podporuje

databázi, kterou používáme, v našem případě MS SQL 2012. Pro nahrání aplikace lze zvolit dva hlavní různé postupy, první je že se soubory nahrají na server ručně. Druhou a vhodnější cestou je, že se aplikace na server nahraje pomocí funkce, kterou přímo obsahuje Visual Studio a to za pomoci funkce Publish. Při použití této funkce se nastaví základní potřebné informace, hlavně tedy URL serveru kam se má aplikace nahrát a na jaké doméně bude dostupná. Mimo to se také nastavuje přístup k databázi. Díky této funkci nám Visual Studio ušetří práci s nahráváním databáze, protože ji samo nahraje spolu s aplikací.

3.7 Uživatelské role

Administrátor (Admin)

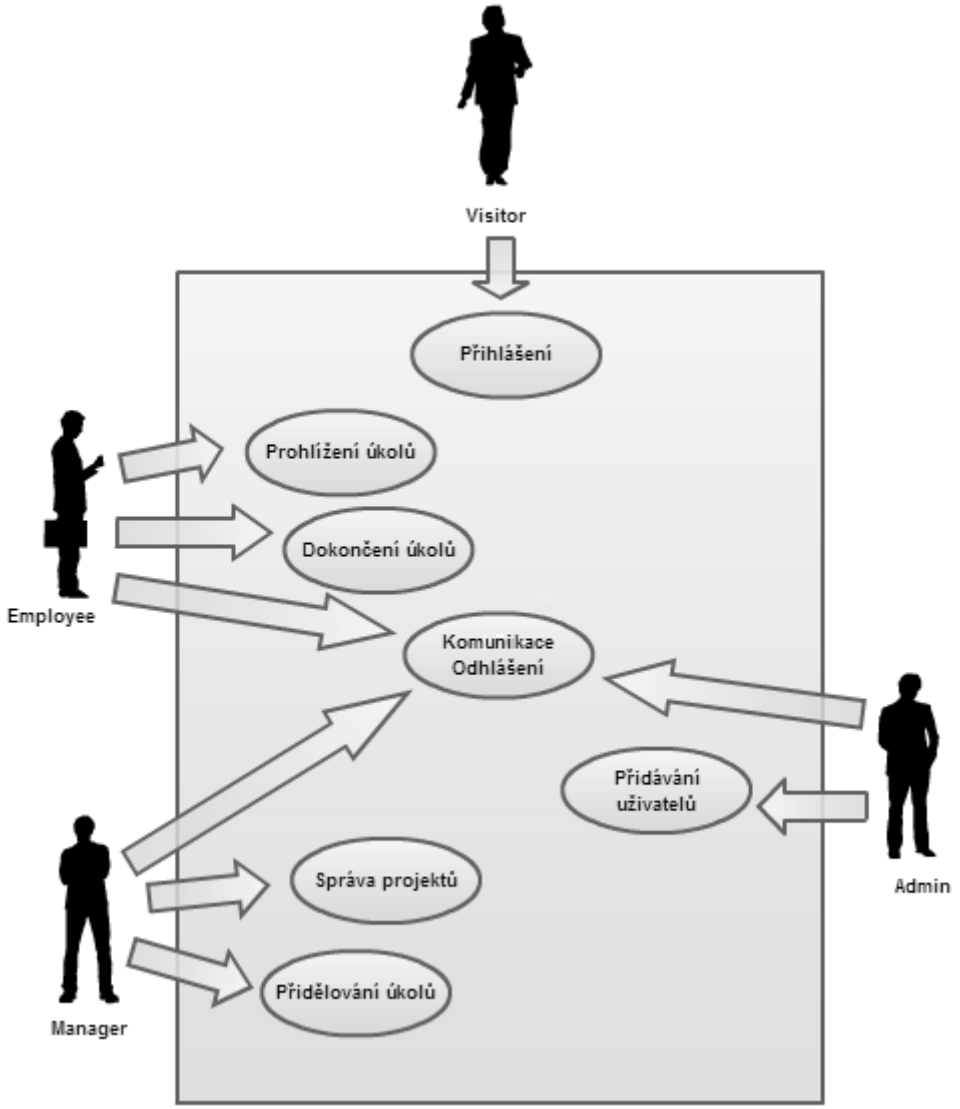
Administrátorovi byla udělena práva na přidávání osob do databáze a jejich úpravu.

Manažer (Manager)

Manažerovi byla udělena práva na přidávání, úpravu a mazání různých údajů, týkajících se zakázek. Zároveň je oprávněn přidělovat úkoly zaměstnancům. Rovněž tak může posílat zprávy zaměstnancům nebo jiným manažerům.

Zaměstnanec (Employee)

Zaměstnanec se po přihlášení do systému může podívat na jakých projektech či úkolech má zrovna pracovat. Pokud nějaký úkol dokončí, může ho označit jako dokončený a objeví se mu v tabulce dokončených úkolů. Také může posílat zprávy ostatním zaměstnancům nebo manažerům.



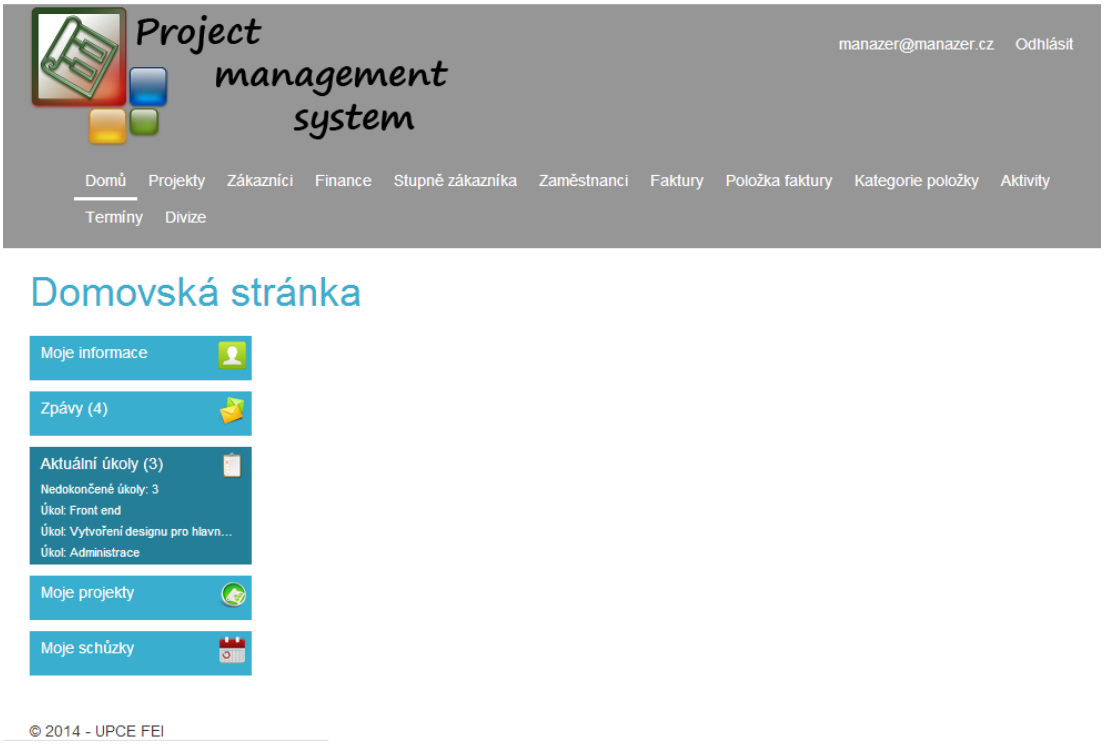
Obrázek 2 - UseCase diagram

3.8 Frontend

Při tvorbě frontendu této webové aplikace byl od začátku hlavním záměrem vytvořit uživatelsky přívětivou aplikace, která bude snadná na ovládání. Vzhled aplikace se liší podle toho, jaká role je přihlášenému uživateli přidělena.

Každému uživateli se po přihlášení zobrazí dvě základní menu, jedno je ve formě klasického navigačního menu v hlavičce stránky a druhé je přímo v těle stránky. Navigační menu se liší pro jednotlivé role. Toto menu je určené hlavně pro roli manažera, který tu najde záložky, potřebné pro vytvoření nových projektů, úkolu a mnoho dalšího. Druhé menu je určené hlavně pro klasického zaměstnance. V tomto menu jsou umístěny záložky, pomocí kterých se zaměstnanec dostane na stránku, kde budou vypsány úkoly, které musí zaměstnanec dokončit. Mimo jiné se odtud dá dostat na stránku se zprávami, které usnadňují vnitrofiremní komunikaci.

Uživatelské rozhraní bylo nastylováno za pomoci CSS respektive za pomoci CSS preprocesoru Sass.



Obrázek 3 - Frontend

3.9 Databáze

3.9.1 Výběr databáze

Při výběru databáze bylo vybíráno mezi databází MySQL a databází MS SQL. Oba dva typy jsou volně ke stažení i pro komerční použití. Rozhodování to nebylo vůbec jednoduché, hlavní výhodou databáze MS SQL byla autorova zkušenost s touto databází a to, že databáze MS SQL má v MVC velmi kvalitní podporu. I přesto první pokus o zprovoznění aplikace proběhl společně s databází MySQL. Avšak protože byl použit Entity Framework 6, který vyšel jen krátkou chvíli před počátkem vývoje aplikace, tak musely být použity zatím jen testovací knihovny zajišťující podporu databáze MySQL. Po chvíli vývoje aplikace se ovšem vyskytly první problémy s touto knihovnou, která měla problémy již s generováním skriptů pro tvorbu tabulek. To bylo nakonec vyřešeno napsáním vlastního generátoru skriptů, ale poté se objevily další problémy. Tento fakt nakonec rozhodl, že databáze použitá k tomuto projektu je databáze od firmy Microsoft a to MS SQL.

3.9.2 Návrh databáze

Na návrh databáze (viz. Příloha F) byl udělán v softwaru MySQL Workbench. Ten byl použit hlavně pro to, aby si byl vytvořen smysluplný návrh databáze a bylo dobře známo, jak bude vypadat konečné schéma.

Na nic jiného tento software prakticky nebyl použit, protože v práci byl využit Entity framework, konkrétně jeho Code First přístup, který umožňuje prezentovat tabulky databáze jako třídy, podle kterých Entity framework následně vygeneruje do databáze příslušné tabulky. Šel by využít i obrácený přístup tzv. Database First, kdy Entity framework vygeneruje z existující databáze třídy do projektu, ale kvůli autorovým zkušenostem s Code First přístupem k databázi, byl využit právě tento způsob.

Všechny tabulky byly navrhnuty tak, aby splňovaly druhou normální formu, a naprostá většina z nich splňuje i třetí normální formu. Jediná tabulka, která nesplňuje třetí normální formu je tabulka „Address", ve které by se dala najít závislost mezi sloupci „City" a „Zip". Měla by tak správně vzniknout nová tabulka, ale na druhou stranu by se tím snížila výkonnost databáze resp. dotazů a proto byly tyto sloupce ponechány v tabulce „Address".

3.9.3 Tabulky

V této části práce budou popsány všechny tabulky, které byly v práci navrhnuty a použity.

Legenda
Primání klíč (PK)
Cizí klíč (FK)
PK a FK zároveň

Tabulka 2 - Legenda

Project

Tabulka „Project" je jednou z hlavních tabulek celé databáze a tomu odpovídá i její rozsáhlost. Uchovávají se v ní mnoho informací popisující daný projekt. Najdeme v ní například jméno projektu, jeho popis, co je cílem, kdy byl začat, jestli byl ukončen a pokud ano tak kdy a další informace. Tabulka také nese informaci o tom, který zákazník si projekt vyžádal. Nalezneme zde také cizí klíče, které nám umožní dohledat záznamy o financích daného projektu nebo o divizi pro kterou je projekt určen.

Project	
ProjetId	INT
Name	VARCHAR
ProjectNumber	VARCHAR
Description	VARCHAR
Task	VARCHAR
DealerNote	VARCHAR
BusinessNote	VARCHAR
RatingPersonResponsible	VARCHAR
RatingPersonSupervising	VARCHAR
CustomerFeedBack	VARCHAR
CreationDate	DATETIME
EndDate	DATETIME
DateOfAddressing	DATETIME
DateOfSubmission	DATETIME
DateOfCustomerReplies	DATETIME
Division_DivisionId	INT
Finance_FinanceId	INT
Customer_CustomerId	INT

Tabulka 3 - Project

Customer

Tato tabulka je další z hlavních tabulek celé databáze. Jsou v ní uloženy informace o zákaznících, kteří chtějí vypracovat nějaký projekt. V tabulce najdeme záznamy jako název zákazníka potažmo společnosti, IČO nebo DIČ. V tabulce najdeme dva cizí klíče, kde první nám umožní najít adresu, kde zákazník sídlí a druhý určí jaký je stupeň zákazníka.

Customer	
CustomerId	INT
Name	VARCHAR
FormalName	VARCHAR
Type	VARCHAR
PhoneNumber	INT
IC	INT
DIC	VARCHAR
VAT	BOOL
CustomerDegree	VARCHAR
Address_AddressId	INT
DegreeOfCustomer_DegreeOfCustomerId	INT

Tabulka 4 - Customer

DegreeOfCustomer

Jde o velmi jednoduchou tabulku, která se skládá jen z primárního klíče, názvu a popisu. Umožňuje definovat různé stupně zákazníka.

DegreeOfCustomer	
DegreeOfCustomerId	INT
Name	VARCHAR
Description	VARCHAR

Tabulka 5 - DegreeOfCustomer

Address

V této tabulce se nachází záznamy, které určují adresu zákazníka nebo adresu jednoho ze zaměstnanců, který pracuje u zákazníka. V tabulce můžeme nalézt název státu, města, ulice a směrovací číslo.

Address	
AddressId	INT
City	VARCHAR
Country	VARCHAR
Zip	INT
Street	VARCHAR
NumberOfDescriptive	INT

Tabulka 6 - Address

Employee

Tabulka „Employee“ obsahuje informace o lidech, kteří pracují pro zákazníky. V tabulce se nacházejí základní informace jako jméno, příjmení, email nebo na jaké pozici pracují a jaké jsou jejich funkce.

Employee	
EmployeeId	INT
FirstName	VARCHAR
LastName	VARCHAR
Email	VARCHAR
Phone	INT
Function	VARCHAR
Rights	VARCHAR
Customer_CustomerId	INT
Address_AddressId	INT

Tabulka 7 - Emloyee

Finance

Jde o tabulku, kde jsou shromažďovány informace týkající se finanční stránky projektu. Jsou tu k nalezení informace typu, jako jestli jde o příchozí platbu, kdy daná platba proběhla nebo o jakou sumu se jednalo, případně jestli bylo placeno i DPH.

Finance	
FinanceId	INT
Name	VARCHAR
Income	BOOL
Date	DATETIME
DocumentNumber	VARCHAR
Amount	DECIMAL
PayVat	BOOL
Vat	DOUBLE
AmountSum	DECIMAL

Tabulka 8 - Finance

Invoice

Tabulka „Invoice“ je tabulkou, kde najdeme informace ohledně faktury, týkající se daného projektu. V tabulce nalezneme informace o tom, kdy byla faktura vystavena a kolik bylo zaplaceno.

Invoice	
InvoiceId	INT
Name	VARCHAR
DateOfIssue	DATETIME
DueDate	DATETIME
Paid	INT

Tabulka 9 - Invoice

InvoiceItem

V tabulce „InvoiceItem“ nalezneme informace o konkrétní položce, která se nachází na faktuře. O položce se v ní evidují data jako název, popis, cena nebo počet kusů.

InvoicedItem	
InvoicedItemId	INT
Name	VARCHAR
Description	VARCHAR
Count	INT
Prize	DECIMAL
CompanyId	VARCHAR
Invoice_InvoiceId	INT

Tabulka 10 - InvoiceItem

OtherActivities

Tabulka „OtherActivities“ slouží pro ukládání dat o úkolech, které je na projektu potřeba udělat. Tyto úkoly mohou být poté přiděleny nějakému zaměstnanci. Ukládají se informace o popisu úkolu, o tom jestli byl úkol dokončen nebo o odhadu doby který je na úkol potřebná.

OtherActivities	
OtherActivitiesId	INT
Name	VARCHAR
Description	VARCHAR
Done	BOOL
Estimate	INT
User_UserId	INT
Project_ProjectId	INT

Tabulka 11 - OtherActivities

OtherTerms

V této tabulce jsou uložena data, týkající se dalších důležitých termínu souvisejících s projektem, jako například domluvené schůzky. O termínech se eviduje hlavně datum a nějaký stručný popis, popř. kdo je za schůzku zodpovědný.

OtherTherms	
OtherThermsId	INT
Name	VARCHAR
Description	VARCHAR
Date	DATETIME
User_UserId	INT
Project_ProjectId	INT

Tabulka 12 - OtherTerms

Message

Tabulka „Message“ je velmi důležitou tabulkou, která eviduje zprávy sloužící k vnitrofiremní komunikaci mezi zaměstnanci. U zprávy se zaznamenává hlavně předmět a text zprávy, ale můžeme v ní nalézt i příznak jestli byla zpráva přečtena. Tabulka obsahuje cizí klíče, které nám umožňují dohledat adresáta zprávy a také jeho autora.

Message	
MessageId	INT
Subject	VARCHAR
Text	VARCHAR
Readed	BOOL
Date	DATETIME
UserId	INT
AuthorId	INT

Tabulka 13 - Message

Division

Poslední tabulkou je tabulka „Division“, ve které jsou pouze uloženy názvy a popisy divizí, na které je společnost rozdělena.

Division	
DivisionId	INT
Name	VARCHAR
Description	VARCHAR

Tabulka 14 - Division

3.9.4 Speciální tabulky

__MigrationHistory

Tato tabulka vzniká automaticky, při povolení migrací v projektu. Ukládají se do ní informace o proběhlých migracích neboli změnách ve struktuře databáze. Tabulka obsahuje záznamy MigrationId, ContextKey, Model a ProductVersion.

__MigrationHistory	
MigrationId	VARCHAR
ContextKey	VARCHAR
Model	VARCHAR
ProductVersion	VARCHAR

Tabulka 15 - MigrationHistory

Následujících pět tabulek bude vytvořeno automaticky po vytvoření prvního uživatele. Těchto pět tabulek si vytvoří ASP.NET Identity, který v těchto tabulkách uchovává potřebná data o uživatelích, jako přihlašovací jména, role nebo hesla. Tabulka AspNetUsers je poté spojena s tabulkou UserData za pomoci cizího klíče.

AspNetRoles

Tato tabulka je velmi jednoduchá a udržuje informaci o existujících rolích. Obsahuje pouze primární klíč a název role.

AspNetRoles	
RoleId	VARCHAR
Name	VARCHAR

Tabulka 16 - AspNetRoles

AspNetUserClaims

Do této tabulky se dají ukládat podrobnější informace o uživateli, které se později mohou využít. ASP.NET dokonce povoluje autentizaci za pomoci skupiny záznamů v této tabulce, které náleží uživateli.

AspNetUserClaims	
Id	INT
ClaimType	VARCHAR
ClaimValue	VARCHAR
User_Id	VARCHAR

Tabulka 17 - AspNetUserClaims

AspNetUserLogins

Tabulka „AspNetUserLogins" je vytvořena zejména pro účely externí autentifikace, jako například přihlašování za pomoci účtu na Facebooku nebo Googlu.

AspNetUserLogins	
UserId	VARCHAR
LoginProvider	VARCHAR
ProviderKey	VARCHAR

Tabulka 18 - AspNetUserLogins

AspNetUserRoles

Vznik této tabulky je zapříčiněn typem vazby mezi tabulkou „AspNetUsers" a tabulkou „AspNetRoles", kde jednomu uživateli může být přiřazeno více rolí a jedna role pravděpodobně bude přiřazena u uživatelů.

AspNetUserRoles	
UserId	VARCHAR
RoleId	VARCHAR

Tabulka 19 - AspNetUserRoles

AspNetUsers

Tabulka „AspNetUsers" ukládá základní údaje o uživateli, které jsou potřebné k jeho přihlášení do systému. Kromě primárního klíče obsahuje hlavně záznam UserName, který představuje přihlašovací jméno uživatele a PasswordHash, kde je uloženo uživatelovo heslo.

AspNetUsers	
Id	VARCHAR
UserName	VARCHAR
PasswordHash	VARCHAR
SecurityStamp	VARCHAR

Tabulka 20 - AspNerUsers

3.9.5 Vazby

Customer_Project

Mezi těmito tabulkami je binární relace typu jedna k více, přičemž v této vazbě je povinná účast rodiče, ale potomka nikoli.

Customer_Emloyee

Mezi tabulkami byla opět vytvořena binární relace typu jedna k více a účast rodiče je povinná, ale potomka už ne.

Address_Customer

Tyto tabulky mají mezi sebou binární relaci typu jedna k jedné a je opět povinná účast rodiče, ale potomka ne.

Address_Emloyee

Tabulky „Address" a „Emloyee" mají mezi sebou binární relaci typu jedna k jedné a je v ní povinná účast pouze u rodiče, ale potomka ne.

DegreeOfCustomer_Customer

Tyto tabulky mezi sebou mají binární relaci typu jedna k více, kde je povinná účast rodiče a potomek již povinnou účast nemá.

Project_Finance

Tyto tabulky „Project" a „Finance" mají mezi sebou binární relaci typu jedna k více. Ve vazbě je povinná účast rodiče. Potomkova účast povinná není.

Invoice_Project

Tyto dvě tabulky spojuje binární relace typu jedna k jedné, kde je povinná účast rodiče i potomka.

Invoice_InvoicedItem

Tabulka „InvoicedItem“ a „Invoice“ jsou spojeny za pomoci binární relace typu. Ve vazbě je povinná účast rodiče a potomek povinnou účast nemá.

ItemCategory_InvoicedItem

Tyto tabulky spojuje binární relace typu jedna k více, kde je povinná účast rodiče, ale potomka nikoliv.

Division_Project

Tabulky „Division“ a „Project“ jsou spojeny binární relací typu jedna k více, kde je povinná účast rodiče, ale potomka ne.

Division_UserData

Tyto tabulky jsou spojeny binární relací typu jedna k více. Rodič má ve vazbě účast povinnou, ale potomek ne.

Project_OtherActivity

Tyto tabulky jsou spojeny za pomoci binární relace typu jedna k více, přičemž účast rodiče je ve vazbě povinná a potomka ne.

Project_OtherTerms

Tabulky „Project“ a „OtherTerms“ jsou spojeny za pomoci binární relace typu jedna k více, kde je účast rodiče povinná a potomka nikoli.

UserData_OtherActivity

Tabulky jsou spojeny pomocí binární relace typu jedna k více. Účast rodiče je ve vazbě povinná a potomka ne.

UserData _OtherTerms

Tyto tabulky spojuje binární relace typu jedna k více, kde je účast rodiče povinná a potomka nikoli.

UserData_Message

Tato vazba se v databázi vyskytuje hned dvakrát a to z důvodu, že uživatel má k dispozici dvě kolekce zpráv. Jedna je pro odeslané zprávy a druhá pro přijaté. Tabulky jsou tedy spojeny binární relací typu jedna k více, kde účast rodiče je povinná, ale potomka ne.

AspNetUsers_UserData

Tyto tabulky jsou spojeny binární relací typu jedna k jedné, kde je účast jak rodiče, tak potomka povinná.

AspNetUsers_ AspNetUserClaims

Mezi těmito tabulkami se nachází binární relace typu jedna k více. Účast rodiče je ve vazbě povinná, ale potomka nikoli.

AspNetUsers_ AspNetUserLogins

Tabulka „AspNetUsers“ a tabulka „AspNetUserLogins“ jsou spojeny binární relací typu jedna k více, kde je účast rodiče povinná, ale potomka ne.

AspNetUsers _AspNetUserRoles

Tyto tabulky jsou spojeny binární relací typu jedna k více, kde je povinná účast rodiče, ale potomek povinnou účast nemá.

AspNetRoles _AspNetUserRoles

Mezi tabulkami se nachází binární relace typu jedna k více. Ve vazbě je povinná účast rodiče, ale potomka ne.

3.10 Aplikované zásady zabezpečení

3.10.1 Přístup do aplikace

Zabezpečení aplikace funguje na principu, že do systému se přihlásí jen uživatel, kterému byl vytvořen profil administrátorem a zároveň mu byla přidělena určitá práva specifikující úkony, které budou uživateli v rámci aplikace poskytnuty. Poté se již může uživatel přihlásit do aplikace za pomoci vytvořeného účtu a hesla. Každá role má dovolena přístup do různých sekcí aplikace. Například základnímu zaměstnanci se vůbec nezobrazí v menu položky, které by mu umožnily se dostat například do sekce vytváření projektu. I kdyby se zaměstnanec pokusil dostat do sekce, kam nemá přístup pomocí URL adresy, tak mu bude zamítnut přístup, respektive dostane možnost se znovu přihlásit s jiným účtem, který by měl roli umožňující přístup do této sekce.

3.10.2 SQL injection

Této technice při, které dochází databázové vrstvy za pomoci vsunutého SQL dotazu, je v práci zabráněno za pomoci výše zmíněného Entity Frameworku. Ten poskytuje tíženou obranu proti napadení a získaní tajných dat, které by uživatel mohl zneužít.

4 Závěr

V teoretické části této práce byla popsána důležitost použití informačních systémů společnostmi v dnešní době, ale zároveň také jaká plynou rizika jejich použití. Taktéž byly blíže popsány základní druhy informačních systémů včetně toho, k čemu především slouží.

V další části byly vybrány různé informační systémy sloužící podobnému účelu, jako informační systém vytvářený v této práci. Tyto informační systémy byly porovnány a na základě výsledků porovnání jednotlivých aplikací byl vybrán nejlepší produkt.

V praktické části práce je poté popsán samotný informační systém a jeho základní funkce a využití. V této části byla detailně popsána zejména také databáze, která je nedílnou součástí a stavebním kamenem informačního systému.

Vytvořený systém je funkční a připravený na použití v praxi. Potenciálem budoucího rozvoje je implementace dalších funkcionalit, které by rozšířily komplexnost systému. V současné době se však o něčem takovém neuvažuje, neboť v současné podobě poskytuje systém jednoduché a intuitivní rozhraní se všemi základními funkcionalitami – o tyto benefity by při rozšiřování celé řešení mohlo přijít.

5 Soupis použité literatury

[1] PIALORSI, Paolo a Marco RUSSO. *Programming Microsoft LINQ in Microsoft .NET Framework 4*. Sebastopol, Calif.: O'Reilly, 2010, xxiv, 675 p. ISBN 07-356-4057-2. Dostupné z: http://it-ebooks.info/book/1436/

[2] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. *ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně*. Vyd. 1. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 880 s. Encyklopedie Zoner Press. ISBN 978-80-7413-131-8.

[3] FREEMAN, Adam. *Pro ASP.Net MVC 5*. New York: Apress, 2013. ISBN 978-1-4302-6529-0.

[4] LUBBERS, Peter, Brian ALBERS a Frank SALIM. *HTML5: programujeme moderní webové aplikace*. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.

[5] CRM. *Adaptic* [online]. [2005] [cit. 2014-04-18]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/crm/

[6] Co je systém?. *Vítejte na zemi* [online]. 2008 [cit. 2014-04-18]. Dostupné z: http://www.vitejtenazemi.cz/krajina/index.php?article=56

[7] Pojem "Informační systém". *Tovarna.cz* [online]. [2005] [cit. 2014-04-18]. Dostupné z: http://www.tovarna.cz/cz/slovnik-pojmu/17-informacni-system/

[8] Co vlastně je informační systém a jak souvisí s řízením?. *Živě.cz* [online]. 1998 [cit. 2014-04-18]. Dostupné z: http://www.zive.cz/clanky/co-vlastne-je-informacni-system-a-jak-souvisi-s-rizenim/sc-3-a-4436/default.aspx

[9] K čemu jsou podnikové informační systémy. *BusinessVize* [online]. 2010 [cit. 2014-04-18]. Dostupné z:http://www.businessvize.cz/informacni-systemy/k-cemu-jsou-podnikove-informacni-systemy

[10] *Přínosy informačních systémů pro řízení podniku* [online]. 2005 [cit. 2014-04-18]. ISSN 1210-0889. Dostupné z:http://www.odbornecasopisy.cz/index.php?id_document=26295

[11] VACEK, Václav. Učebnice programování Atmel s jádrem 8051. 1. vyd. Praha: BEN - technická literatura, 2001, 141 s. ISBN 80-730-0043-1.

[12] MVC architektura. *Devbook.cz* [online]. [2013] [cit. 2014-04-18]. Dostupné z: http://www.devbook.cz/mvc-architektura-navrhovy-vzor

[13] SASS, LESS, STYLUS NEBO ČISTÉ CSS?. *PhpFasion* [online]. 2012 [cit. 2014-04-18]. Dostupné z:http://phpfashion.com/sass-less-stylus-nebo-ciste-css-2

[14] SASS – snadnější psaní CSS. *3sixtyBLOG* [online]. 2012 [cit. 2014-04-18]. Dostupné z: http://www.3sixty.eu/blog/sass-snadnejsi-psani-css

[15] Deset důvodů proč používat CSS generátory – Less a Sass. *Spectrum* [online]. 2011 [cit. 2014-04-18]. Dostupné z:http://www.jakubskvara.cz/deset-duvodu-proc-pouzivat-css-generatory-less-a-sass/

[16] *Podpora podnikových procesů v ERP aneb Nezbytnost jménem workflow* [online]. 2012 [cit. 2014-04-18]. ISSN 1802-615X. Dostupné z: http://www.systemonline.cz/erp/podpora-podnikovych-procesu-v-erp.htm

[17] Verze jazyka c#. *DOTNETPORTAL.CZ* [online]. 2013 [cit. 2014-04-18]. Dostupné z:http://www.dotnetportal.cz/blogy/12/Ondrej-Janacek/3967/Verze-jazyka-C-

[18] C Sharp (programming language). *Wikipedia* [online]. 2014 [cit. 2014-04-18]. Dostupné z:http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[19] Versions of HTML. *ScriptingMaster* [online]. ©2014 [cit. 2014-04-18]. Dostupné z:http://www.scriptingmaster.com/html/versions-of-HTML.asp

[20] Úvod do .net frameworku. *DOTNETPORTAL.CZ* [online]. 2009 [cit. 2014-04-27]. Dostupné z:http://www.dotnetportal.cz/clanek/125/Uvod-do-NET-Frameworku

[21] Databáze a jazyk SQL. *Interval.cz* [online]. 2000 [cit. 2014-05-03]. Dostupné z: http://interval.cz/clanky/databaze-a-jazyk-sql/

[22] ORM - základní informace. *Jan* [online]. 2009 [cit. 2014-05-03]. Dostupné z: http://jan.baresovi.cz/dr/cs/ORM-zakladni-informace

[23] Seznam zkratek ve světě informačních technologií. *Bpsoft* [online]. [2011] [cit. 2014-05-03]. Dostupné z:http://www.bpsoft.cz/cz/seznam-zkratek-ve-svete-informacnich-technologii/

[24] Co nabízí komplexní manažerský informační systém (MIS). *Sefima* [online]. 2011 [cit. 2014-05-03]. Dostupné z:http://www.sefima.cz/co-nabizi-komplexni-manazersky-informacni-system-mis-cln6.php

Příloha A – Obsah controlleru Project

Tato příloha obsahuje část kódu, který je obsažen v controlleru Project. Obsahuje metodu pro zobrazení formulářové stránky a metodu pro zpracování dat, které byly odeslány z formuláře na server.

```
[HttpGet]
public ActionResult CreateProject()
{
	var project = new ProjectModel();

	project.CustomerList = this.CustomerService.GetQueryable().ToList().Select(d =>
	new SelectListItem
	{
		Text = d.Name,
		Value = d.CustomerId.ToString(CultureInfo.InvariantCulture)
	});

	DateTime date;
	DateTime.TryParse("01.01.1900", out date);
	project.CreationDate = date;
	project.EndDate = date;
	project.DateOfAddressing = date;
	project.DateOfCustomerReplies = date;
	project.DateOfSubmission = date;

	return this.View(project);
}

[HttpPost]
public ActionResult CreateProject(ProjectModel model)
{
	Contract.Requires(model != null);

	if (model.CustomerId == 0)
	{
		this.ModelState.AddModelError("CustomerId", "Zadejte zákazníka");
	}

	if (ModelState.IsValid)
	{
		var projectNew = this.ProjectService.Create();
		AutoMapper.Mapper.DynamicMap(model, projectNew);
		this.ProjectService.Insert(projectNew);

		this.UnityOfWork.Save();
		return this.RedirectToAction("Index");
	}
```

```
        model.CustomerList = this.CustomerService.GetQueryable().ToList().Select(d =>
        new SelectListItem
        {
                Text = d.Name,
                Value = d.CustomerId.ToString(CultureInfo.InvariantCulture)
        });

        return this.View(model);
}
```

Příloha B – Zdrojový kód modelu Message

```
public class MessageModel
{
        [ScaffoldColumn(false)]
        public int MessageId { get; set; }

        [Display(ResourceType = typeof(BakalarkaResource), Name = "MessageSubject")]
        [Required(ErrorMessageResourceType = typeof(BakalarkaResource),
        ErrorMessageResourceName = "MessageSubjectRequiredErrorMessage")]
        public string Subject { get; set; }

        [DataType(DataType.Date), DisplayFormat(DataFormatString = "{0:yyyy-MM-
        dd}", ApplyFormatInEditMode = true)]
        [Display(ResourceType = typeof(BakalarkaResource), Name = "MessageDate")]
        [Required(ErrorMessageResourceType = typeof(BakalarkaResource),
        ErrorMessageResourceName = "MessageDateRequiredErrorMessage")]
        public DateTime Date { get; set; }

        [Display(ResourceType = typeof(BakalarkaResource), Name = "MessageText")]
        [Required(ErrorMessageResourceType = typeof(BakalarkaResource),
        ErrorMessageResourceName = "MessageTextRequiredErrorMessage")]
        public string Text { get; set; }

        public bool Readed { get; set; }

        public int UserDataId { get; set; }

        public UserRegisterModel UserData { get; set; }

        public string AuthorName { get; set; }

        public int AuthorId { get; set; }

        public UserRegisterModel Author { get; set; }

        public string AddressEmail { get; set; }
}
```

Příloha C – Zdrojový kód Sass

```
.myBtn{
	display: inline-block;
	margin-bottom:10px;
	$btn-color: rgb(150,150,150);
	background: $btn-color;
	padding: 5px 10px;
	@include transition();
	text-decoration: none;
	color: white;
	@include border-radius(5px);
	border: none;

	&:hover{
		background-color: darken($btn-color, 15%);
		text-decoration: none;
		color: white;
	}
}
```

Příloha D – Ukázka z komponenty view

```
@model IEnumerable<BakalarkaSql.Models.DivisionModel>

@{
   ViewBag.Title = "Divize";
   var sortModel = ViewData["sort"] as string;
   var filterName = ViewData["filterName"] as string;
}

<div id="division">

   <h2>Divize</h2>

   <p>
         @Html.ActionLink("Vytvořit divizi", "CreateDivision", null, new { @class =
         "myBtn" })
   </p>

   @using (Html.BeginForm("Index", "Division", FormMethod.Get))
   {
      @Html.Hidden("sort", sortModel)
      <table class="searchTable">
         <thead></thead>
         <tbody>
            <tr>
               <td>
                  Vyhledat podle názvu
               </td>
               <td>
                  @Html.TextBox("filterName", filterName)
               </td>
            </tr>
         </tbody>
      </table>

      <input type="submit" class="myBtn" value="Vyhledat" />
   }

   <table class="table">
      <tr>
         <th>
            <span class="theadSpan">@Html.DisplayNameFor(model =>
model.Name)</span>
            <div class="sortDiv">
               <a href="@Url.Action("Index", new { sort = "name", filterName = filterName
})" id="up"></a>
               <a href="@Url.Action("Index", new { sort = "nameDesc", filterName =
filterName })" id="down"></a>
```

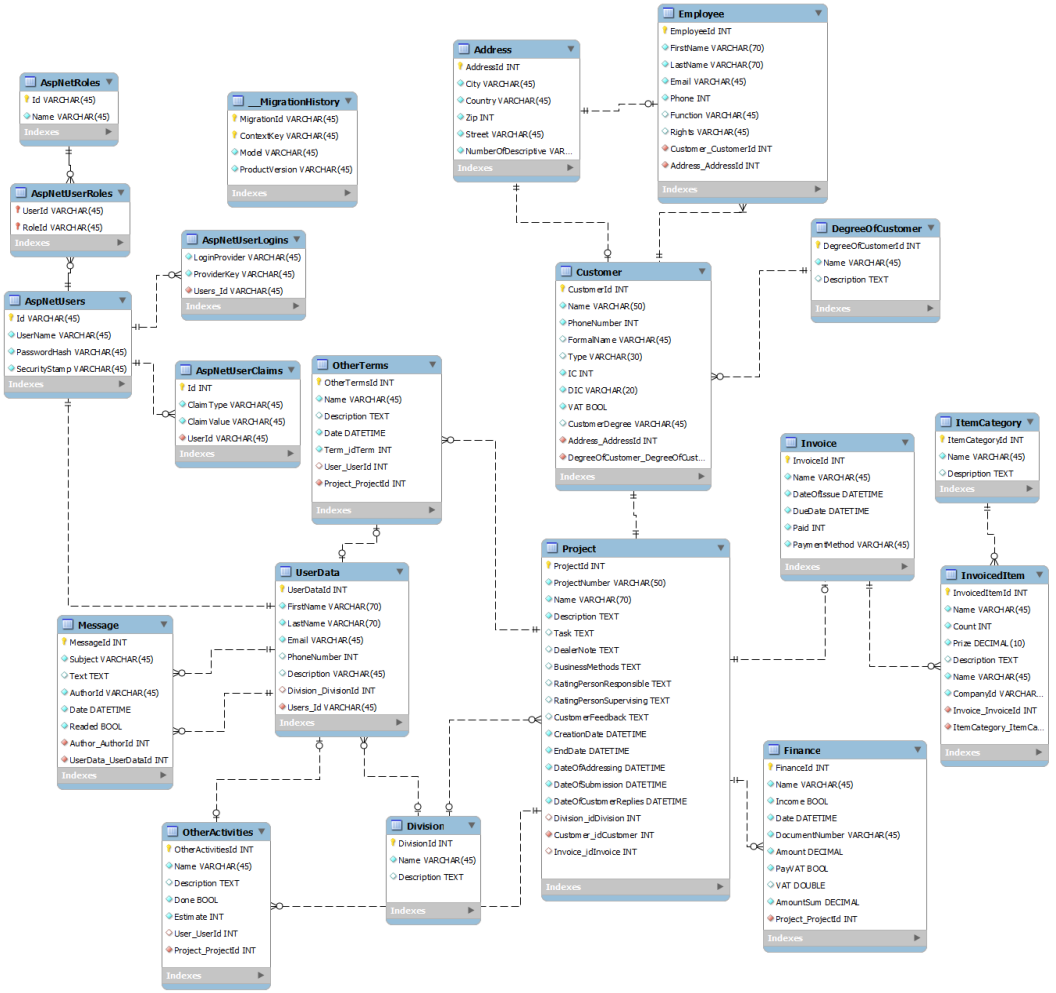
```
            </div>
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Description)
        </th>
        <th></th>
    </tr>

    @foreach (var division in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => division.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => division.Description)
            </td>
            <td>
                @Html.ActionLink("Upravit", "Edit", new { id = division.DivisionId }, new {
@class = "myBtn" })

                <input type="submit" data-id="@division.DivisionId" class="myBtn
removeBtn" value="Odstranit" />
            </td>
        </tr>
    }

</table>

</div>

@section scripts{
    <script type="text/javascript">
        $(function () {
            $(".removeBtn").on("click", function (e) {
                e.preventDefault();
                if (confirm("Opravdu chcete smazat tuto divizi ?")) {
                    var ri = $(this);
                    var Id = ri.data('id');
                    var url = "@Url.Action("Delete","Division")";
                    $.ajax({
                        type: "POST",
                        datatype: 'json',
                        data: JSON.stringify({ id: Id }),
                        url: url,
                        success: function(e) {
                            if (e.success) {
                                var url = '@Url.Action("Index", "Division")';
                                window.location.href = url;
```

```
                    }
                },
                contentType: 'application/json; charset=utf-8'
            });
        }
    });
});
</script>
}
```

Příloha E – Ukázka třídy, ze které je vygenerována tabulka

```
public class Invoice
{
	public virtual int InvoiceId { get; set; }

	[Required]
	public virtual string Name { get; set; }

	[Required]
	public virtual DateTime DateOfIssue { get; set; }

	[Required]
	public virtual DateTime DueDate { get; set; }

	[Required]
	public virtual decimal Paid { get; set; }

	[Required]
	public virtual string PaymentMethod { get; set; }

	public virtual ICollection<Project> Projets { get; set; }

	public virtual ICollection<InvoicedItem> InvoicedItem { get; set; }
}
```

Příloha F – Návrh databáze



Obrázek 4 - Návrh databáze