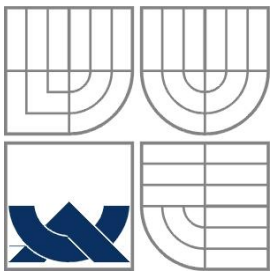
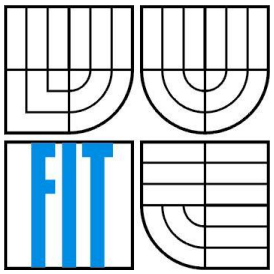


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO MANAGEMENT PROJEKTŮ

PROJECT MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PETR ŠPAČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LADISLAV RUTTKAY

BRNO 2009

Zadání bakalářské práce/8857/2008/xspace02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Špaček Petr**
Obor: Informační technologie
Téma: **Systém pro management projektů**
Kategorie: Web

Pokyny:

1. Prostudujte problematiku správy projektů a efektivního vyhodnocování informací.
2. Prostudujte problematiku webových služeb, MS SQL 2008.
3. Vytvořte návrh a design webové aplikace pro management projektů.
4. Implementujte systém, který bude sestávat z lokální a serverové časti. Lokální část bude zobrazovat informace o průběhu projektu a umožní jeho změny. Po připojení na internet budou informace synchronizované se serverem. Server bude poskytovat webové rozhraní pro správu.

Literatura:

- Kačmář, D.: Programujeme .NET aplikace. Computer Press, Praha, 2001
- Meilir,P. J.: Základy objektově orientovaného návrhu v UML. Grada Publishing, Praha, 2001
- Prosise, J.: Programování v Microsoft .NET - Webové aplikace v C#, ASP.NET a .NET Framework. Computer Press, Praha, 2003
- Schmuller, J.: Myslíme v jazyce UML. Grada Publishing, Praha, 2001.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Ruttkay Ladislav, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

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

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá problémem řízení projektů. Navržený software slouží k usnadnění řízení a plánování projektů ve firmě jednoduchým grafickým uživatelským rozhraním. K implementaci jsou zejména použity technologie Microsoft .NET, Microsoft SQL Server 2008.

Klíčová slova

Řízení projektu, projekt, webové služby, .NET, MSSQL 2008.

Abstract

The work deals with project management. The design software helps to manage and plan projects by using a simple graphic user interface. For implementation there are used technologies like Microsoft .NET, Microsoft SQL Server 2008.

Keywords

Project management, project, web services, .NET, MSSQL 2008.

Citace

ŠPAČEK, P. *Systém pro management projektů.* Vysoké učení technické v Brně, Fakulta informačních technologií, 2009.

Systém pro management projektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ladislava Ruttkaye.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Špaček
18. 5. 2009

Poděkování

Děkuji Ing. Ladislavu Ruttkayovi za vedení mé bakalářské práce a za jeho spolupráci.

Obsah

1 Úvod

Důležitou součástí firmy, která potřebuje mít přehled o dění uvnitř, je bezesporu informační systém. Ten je určen pro sběr, uchovávání, zpracování, poskytování a vyhodnocování dat. Data jsou nepostradatelným podkladem pro získání informací v rozhodovacích a řídících procesech firmy.

Aby firma byla schopna efektivně spravovat své projekty, potřebuje nástroj, který bude schopen plánovat, koordinovat a řídit úlohy a zdroje pro dosažení definovaného cíle obvykle v daném čase, s definovanými zdroji a s vymezenými finančními prostředky. Právě k tomuto účelu bude sloužit tento Systém pro management projektů.

Cílem bylo vytvořit software, který bude sloužit, jak k ulehčení práce vedoucím projektů, tak i k ulehčení práce pro běžného pracovníka, který se systémem bude pracovat. Vedoucí bude moci jednoduchým způsobem vytvářet strukturu projektu. Poté kontrolovat a řídit chod takového projektu. Zaměstnanci zde naleznou základní údaje o svých spolupracovnících, o zákaznících. Budou vědět, na koho se obrátit v případě problému. Naleznou zde potřebné pracovní dokumenty. Software jim umožní poslat zprávu kolegovi.

V bakalářské práci jsem se zejména zaměřil na jednoduchost a intuitivnost ovládání. Tato kritéria jsou bezesporu jedny z primárních požadavků na jakýkoli moderní systém, se kterým bude uživatel pracovat. Proto celou strukturu projektů lze tvořit a editovat přes jednoduchá grafická rozhraní.

Zajímavou možností je také použití softwaru v režimu offline, kdy uživatel nemusí být připojen k internetu a přesto může provádět změny v projektu. Až se uživatel připojí k internetu, aplikace mu nabídne možnost synchronizace, kdy se lokálně upravená data přenesou do databáze na serveru.

V první kapitole se nachází úvod, který popisuje stručné uvedení do problematiky. Jsou zde popsány jednotlivé kapitoly.

Druhá kapitola se zaměřuje na prostředky, které jsou používány pro návrh systémů. Je zde stručně popsán modelovací jazyk UML, který nám pomůže formálně definovat požadavky na software, vztahy mezi entitami, mezi objekty a mnoho dalších. Dále jsou zde popsány principy objektově orientovaného návrhu.

Třetí kapitola pojednává o prostředcích, které jsou využity při implementaci bakalářské práce. Jedná se o použité technologie, na kterých bude software vyvíjen. Značná část pojednává o technologii Microsoft .NET, dále pak o webových službách, o Microsoft SQL serveru a prostředku k návrhu databáze.

Ve čtvrté kapitole jsou uvedena pravidla řízení projektů, které nám pomohou k dosažení zdárného konce díla. Na řízení softwarových projektů se podíváme z různých pohledů. V druhé části

kapitoly jsou uvedeny techniky k modelování a plánování projektů. Jsou zde popsány diagramy, které právě slouží k takovému plánování. Jedná se o diagram PERT a známý Ganttův diagram.

V páté kapitole se nachází analýza systému, kde jsou popsány požadavky na systém a jejich následné zpracování do formální podoby pomoci diagramu případu užití. Pojednává o způsobech získávání informací od potencionálních budoucích uživatelů, kteří jsou zdrojem cenných informací.

Návrh systému je popsán v šesté kapitole. Jedná se o další krok ve vývoji softwaru. Je zde zobrazena struktura celého systému, její rozdělení na fyzicky oddělitelné části jako je serverová, lokální a webová část. Pomocí E-R diagramu je zde modelována struktura databáze, ve které budou uložena potřebná data.

Sedmá část popisuje realizaci systému, ve které jsou stručně popsány rozdíly mezi návrhem a realizací.

V kapitole osmé jsou shrnuty odezvy uživatelů na téměř hotovou verzi systému, které jsou podkladem pro finální dokončení a doladění.

A v závěrečné deváté části jsou zhodnoceny dosažené výsledky. Jsou zde nastíněny další možnosti a směr vývoje vytvořeného produktu. Je zde zhodnocen přínos pro můj vzdělávací životní cyklus.

2 Prostředky pro návrh

Každý program, ať už je to informační systém nebo například obyčejná kalkulačka, si zaslouží návrh, podle kterého se budeme při vytváření aplikace řídit. Je potřeba navrhnout databázi, rozvrhnout vztahy mezi objekty, navrhnout diagramy užití. Pro tento účel nám velmi dobře poslouží technologie UML, objektový návrh a posléze objektové programování. V této kapitole je popsána teorie těchto návrhů.

2.1 Jazyk UML

UML (Unified Modeling Language) je grafický jazyk, který vytváří pohledy nad systémem. Pohledy na systém jsou modelovány pomocí vhodných diagramů definovaných v UML. Slouží jak k modelování softwaru, tak i k návrhu hardwaru, procesů i komunikace. UML poskytuje objektový přístup k návrhu, analýze a ke struktuře vytvářeného systému. Jeho spektrum využití je velmi rozsáhlé a v praxi používané a vyžadované.

Standard UML 2.0 popisuje několik diagramů. Diagramy tvoří nejznámější část toho jazyka, které lze rozčlenit do skupin.

- ***Strukturální diagram*** popisuje vrstvu mezi objekty a třídami.
    - *Diagram tříd.*

- *Diagram komponent.*
- *Diagram nasazení.*
- *Diagram balíčků.*

- ***Diagram chování*** určuje akce, které systém provádí na základě vnějších podnětů.
  - *Objektový diagram.*
  - *Diagram případů užití.*
  - *Stavový diagram.*
  - *Diagram aktivit.*
- ***Diagram interakce*** popisuje, jak systémové objekty interagují mezi sebou.
  - *Sekvenční diagram.*
  - *Diagram komunikace.*
  - *Diagram spolupráce.*
  - *Diagram časování.*

2.2 Objektová orientace

V této kapitole jsou popsány základní informace o objektovém přístupu, zvláště pak o objektovém návrhu a o objektovém programování. Objektová orientace je přístup založený na několika základních principech.

2.2.1 Základní princip a pojmy objektové orientace

Objektový přístup zahrnuje následující pojmy, které je nezbytné pochopit.

- **Třída** je šablonou, podle které jsme schopni sestrojit objekt. Popisuje jeho strukturu, jeho chování a jakým způsobem lze s vytvořeným objektem komunikovat.
- **Objekt** je instancí třídy. Spojuje vlastnosti objektu (data) s jeho chováním (metody). Jak v reálném světě nemusíme znát detaily konstrukce zařízení, tak i na objekty v objektovém programování lze pohlížet jako na černou skříňku (black box).
- **Abstrakce.** Cílem abstrakce je zjednodušit pohled na daný problém a určit ty vlastnosti a kritéria, která jsou pro řešení problému klíčová a nezbytná.
- **Zapouzdření.** Pomocí zapouzdření jsme schopní implementační detaily schovat. K samotnému objektu se přistupuje pomocí zpráv (metod).
- **Polymorfismus** neboli mnohotvárnost. Umožňuje odlišným objektům reagovat různě na stejnou zprávu. To znamená, že můžeme různým objektům zasílat konkrétní zprávu, aniž bychom znali jejich třídu (typ). Zpráva bude zpracována způsobem definovaným třídou.

- **Dědičnost** umožňuje vytvářet a definovat třídy na základě již existujících. Tím usnadňuje polymorfismus a zapouzdření. Používáním dědičnosti vytváříme hierarchii vztahů mezi třídami.

2.2.2 Objektově orientované modelování a návrh

Objektová orientace nepopisuje jen způsob tvorby a zápisu programu, ale i postup, jakým způsobem návrhář o problematice objektového přístupu přemýšlí. Přemýšlí nad reálným světem, který chce modelovat a následně implementovat do vytvářeného systému. Přebírá ze vzorového světa jeho vlastnosti a chování.

Nejčastěji pro jeho návrh používáme jazyk UML, který byl popsán výše.

2.2.3 Objektově orientované programování

Skupina objektově orientovaných jazyků rozšiřuje koncepci modulárních jazyků o možnost spojit data s operacemi, které s daty pracují, do jednoho celku. Do této skupiny lze zařadit mnoho programovacích jazyků. Z těch nejznámějších jsou to například C++, Java, Phyton, Smalltalk, PHP, případně C#. Právě v tomto jazyku bude tato lokální aplikace napsána.

3 Použité prostředky pro implementaci

V této kapitole jsou popsány využité technologie, pomocí kterých bude nebo již je software vyvíjen. Především se jedná o technologie, prostřednictvím kterých lze navrhnout databázi, diagramy, psát samotné programy a vytvořit grafický návrh aplikace.

3.1 Microsoft .NET

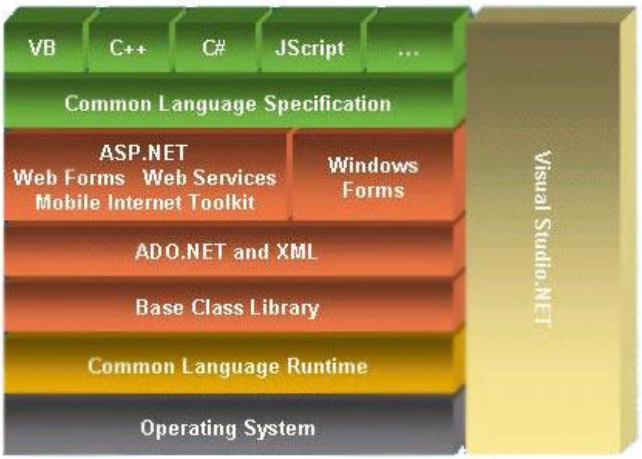
Platforma .NET byla oficiálně představena firmou Microsoft v roce 2000 jako klíčový produkt, jehož rozvoj a propagace je součástí dlouhodobé strategie firmy. Microsoft .NET znamená novou generaci systému vývoje aplikací pro operační systémy Windows, Windows mobile, který je založen na řízeném běhovém prostředí, obohaceném o sadu základních tříd nesoucím jméno *.NET Framework* pro operační systémy Windows a *.NET Compact Framework* pro Windows Mobile.

Princip řízených běhových prostředí, použitý právě u platformy .NET, ale také u platformy Java firmy Sun Microsystems, přidává k převodu zdrojového kódu do kódu strojového ještě jednu vrstvu. Touto vrstvou je mezikód, do kterého jsou zdrojové kódy zkompilovány, a tento mezikód je běhovým prostředím na cílové platformě (Windows, Linux) převeden do strojového kódu.

Tento převod je na cílové platformě realizován vždy při spouštění konkrétní aplikace. Mínusem tohoto překladu je vyšší náročnost na výkon uživatelského počítače, a z tohoto důvodu se tento způsob nepoužívá pro vývoj výpočetně náročných aplikací (například počítačové hry). S jeho častým použitím se naopak můžeme setkat spíše u obchodních aplikací, které nejsou tak náročné na výpočetní výkon a rychlost běhu daných aplikací je naprosto vyhovující.[1]

3.1.1 .NET Framework

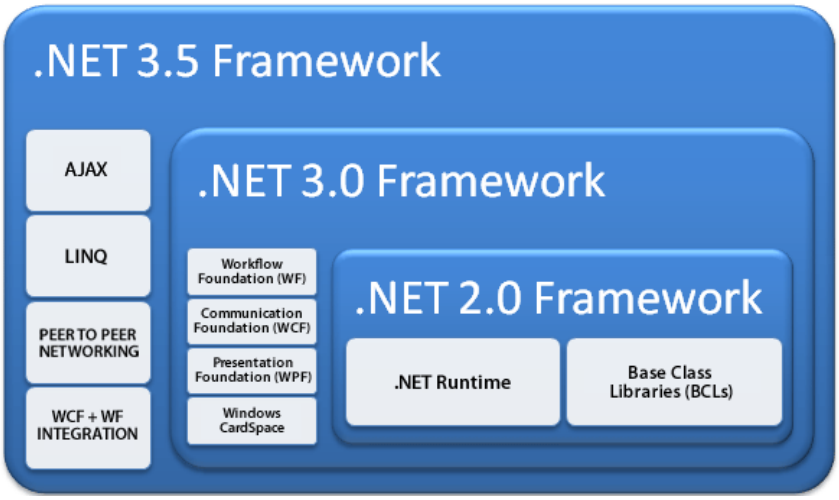
.NET Framework je prostředí, které umožňuje vyvíjení aplikací na poměrně vysoké abstraktní úrovni. Poskytuje rozhraní ke spouštění aplikací a také potřebnou sadu knihoven. Je integrován do operačních systémů Windows.



Obrázek 3.1: Názorná struktura .NET Frameworku. [2]

Pro použití v systémech Unixového typu je k dispozici platforma Mono, která je produktem nezávislé open source iniciativy, která pro zmiňované systémy implementuje .NET runtime. [3]

Aktuální verze této platformy je 3.5. Ve vývoji je verze 4.0.



Obrázek 3.2: Vývoj verzí .NET Framework. [4]

3.1.1.1 Visual Studio .NET

Microsoft Visual Studio je softwarový balík produktů, nástrojů a technologií pro rychlou a efektivní tvorbu aplikací. [5]

3.1.1.2 Programovací jazyk pro .NET

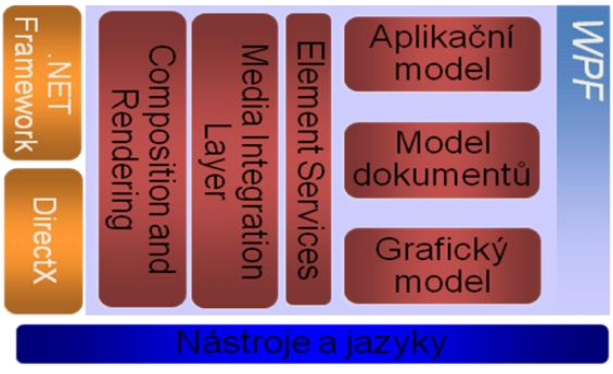
.NET nepředepisuje jeden konkrétní programovací jazyk. Umožňuje aplikace psát v několika programovacích jazycích vyšší úrovně. Nejčastější z nich jsou například:

- C#, jazyk vzniklý zároveň s platformou .NET.
- Visual Basic .NET, nová generace jazyku Visual Basic.
- F#, funkcionální programovací jazyk.
- J#, varianta k Javě, umožňuje používat její knihovny.
- Managed C++, upravené C++ pro použití v prostředí .NET.
- Delphi.

Možnost psát programy pomocí různých programovacích jazyků nám umožňuje vrstva Common Language Specification. (Společná jazyková specifikace). Jejím výsledkem je rovnocennost zmiňovaných jazyků.

3.1.1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) je grafický subsystém pro psaní programů. WPF se považuje za primární rozhraní API (application programming interface) systému Microsoft Windows Vista. Aplikace však lze spouštět i na systémech, které mají nainstalovány .NET Framework 3.0 a vyšší. WPF v praxi obsahuje dvě souběžná programovací rozhraní. Programy postavené na WPF lze psát pomocí jazyků podporovaných v CLS (Common Language Specification) nebo pomocí nového značkovacího jazyka, založeného na XML, který se nazývá XAML (Extensible Markup Language). Ten umožňuje oddělit logiku aplikace od samotného vzhledu. [6]



Obrázek 3.3: Struktura WPF. [7]

Za předchůdce Windows Presentation Foundation můžeme považovat WinForms. Prozatím je ale nenahrazuje. Slouží zatím spíše jako dokonalejší alternativa k tvorbě uživatelského rozhraní, ve

kterém lze pohodlně používat multimediální data, vektorovou grafiku, efekty k vytváření 2D nebo 3D aplikace.

3.2 Webové služby

Webová služba je sbírka protokolů a standardů, které slouží pro spolupráci počítačů v síti. Jsou nezávislé na platformě. Z toho důvodu lze uskutečnit komunikaci mezi různými operačními systémy.

Jedná se o komunikaci serveru s klientem, kde server představuje poskytovatele dat a klient uživatele, který má o data zájem. Komunikace probíhá v otevřeném formátu standardu XML.

Využití webových služeb je celá řada a v dnešní době se již stávají standardem. Velmi efektivně lze přes ně řešit například stahování aktualizací, výměnu a přístup k dokumentům, čtení aktuálních novinek pomocí kanálu RSS, přístup k databázovému systému a mnoho dalších.

Pro přenos těchto služeb se zpravidla používají standardní komunikační protokoly (například: HTTP, FTP, SMTP a další).

Mezi komunikační protokoly, které používají pro dorozumívání serveru s klientem XML zprávy, se řadí XML-RPC a SOAP. SOAP je nástupcem XML-RPC a řeší jeho některé nedostatky. Je standardizován konsorciem W3C. Pracuje na principu vzdáleného volání procedur (RPC). Poskytovatel klientům nabízí množinu funkcí, ke kterým je možné přistupovat vzdáleně ze sítě.

Pro popis služeb slouží WSDL (Web Services Description Language), který využívá SOAP pro komunikaci. WSDL slouží k popisu komunikace, formátu zprávy, názvu procedur, parametrů a dalším, které webová služba nabízí.

Webové služby je možno vyhledávat pomocí registrů služeb, do kterých poskytovatelé služeb vkládají jejich adresy a popis. Jako příklad poslouží otevřený obchodní registr UDDI (Universal Description, Discovery and Integration).

Firma Microsoft tuto technologii nabízí v .NET platformě již v základu. Je jisté, že potenciál webových služeb v budoucnu bude růst a nabídne tak mnoho možností pro nové vyvíjené systémy.

3.3 Microsoft SQL Server

Microsoft nabízí jako svůj produkt databázi MSSQL. Ta je založena na relačním modelu. Umožňuje ukládat jak relační data tak i nerelační (například XML, klasické soubory). Dovede pracovat s geografickými daty. S databází komunikuje pomocí programu v jazyce T-SQL (Transact-SQL), který rozšiřuje standardní SQL. Lze využívat serverových databázových procedur, které se ukládají přímo na server.

K datům v technologii .NET je možné přistupovat pomocí souboru tříd, který se nazývá ADO.NET. Je nástupcem svého předchůdce ADO (ActiveX Data Object).

3.4 CASE Studio

je program, který slouží k vizuálnímu navrhování databázových struktur na základě E-R diagramu. Umožňuje pracovat s velkou škálou databázových systémů. Jeho výhodou je generovaní SQL kódu z E-R diagramu. A také je schopen vytvořit model struktury z již existující databáze. [8]

4 Řízení projektů

V této kapitole, je uvedena teorie řízení projektů z pohledu různých aktérů firmy. Jsou zde nastíněna pravidla a rady, kterými by se měli tvůrci softwaru řídit. V další části kapitoly jsou popsány způsoby plánování projektu pomocí různých diagramů.

4.1 Teorie řízení projektů

Tato podkapitole je zpracována ze studijní opory Bohuslava Křeny a Radka Kočího pro předmět Úvod do softwarového inženýrství [9].

Nejprve je nutné upřesnit, co to projekt vůbec je. „*Projekt je časově ohraničené úsilí, které se vyvíjí s cílem vytvořit jedinečný výsledek (např. výrobek nebo službu).*“[10] Projekt je tedy ohraničený časem. Lze u něj určit začátek a konec. Při splnění všech cílů nebo naopak když se ukáže, že cíle jsou nesplnitelné, lze projekt prohlásit za ukončený. Důležité je, uvědomit si, že projekt je unikátní a s jistotou nikdy nemůžeme říct, zda bude probíhat přesně podle plánu.

4.1.1 Řízení softwarových projektů

Řízení softwarových projektů můžeme rozdělit na tři části.

- Vývoj projektu.
- Management projektu.
- Řízení kvality projektu.

4.1.1.1 Vývoj projektu

Na vývoj softwarového projektu dohlíží softwarový inženýr (vedoucí vývojového týmu). Ten by měl mít značné technické znalosti v oboru. Musí odhadnout schopnosti lidí, se kterými spolupracuje. Musí disponovat komunikačními dovednostmi.

Musí být schopný vytvořit vhodný poměr mezi níže uvedenými veličinami, které se vyskytují v každém softwarovém projektu.

- Cena,
- čas,
- rozsah,
- kvalita.

Tyto parametry jsou mezi sebou provázány. Při úpravě některé z veličin jsou ovlivněny i ostatní. Snížíme-li náklady, poklesne kvalita a zmenší se rozsah projektu. Zkrátíme-li čas dodáni projektu, povede to ke snížení kvality, rozsahu a pravděpodobně vzroste i cena projektu. Mohlo by se zdát, že při navýšení nákladů, se kvalita zlepší a čas zkrátí, ale z praxe vyplývá, že tomu tak není.

Nízká kvalita může znamenat nižší počáteční náklady, ale v pozdějších fázích vývoje a údržby se nevyplácí.

U menších projektů nejsou výše určená pravidla tak vypovídající. Lze totiž vyvinout menší projekt rychleji, za nízké náklady a s kvalitou, která je vyšší než u projektů rozsáhlých.

Je tedy nutné tyto parametry nastavit tak, aby odpovídaly konkrétnímu projektu a nenastala situace, kdy by některá z vlastností projektu byla v rozporu. Je běžné, že proměnné určují různí účastníci obchodu. Zákazník například definuje rozsah a čas a softwarový inženýr určí cenu a kvalitu.

Sestavení vývojového týmu

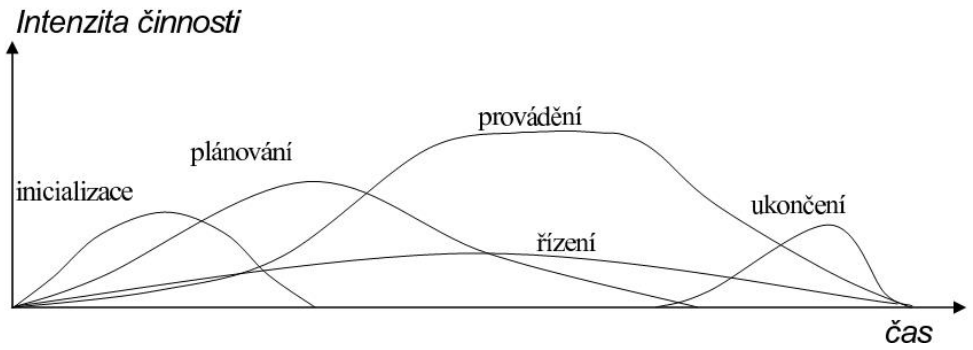
Nesmíme zapomenout, že správně sestavený tým je základem funkčního projektu. Týmová spolupráce je důležitější než práce nespolupracujících individualistů. Do týmu musíme obsazovat takové členy, u nichž se dá předpokládat nejen dostatečná odbornost a schopnost spolupráce, ale i členy, kteří jsou přesvědčeni o potřebnosti projektu.

Schopnost vnitřní komunikace a spolupráce je ovlivněna velikostí týmu. Malý tým je schopný pracovat daleko efektivněji než tým velký. Je proto levnější. Naopak velký tým je schopen práci odvést rychleji, ale s většími finančními náklady.

4.1.1.2 Management projektu

„*Management je proces koordinace činnosti skupiny lidí, který realizuje jednotlivec nebo skupina lidí za účelem dosažení stanovených cílů.* “[11] Na rozdíl od softwarového inženýra nejsou pro manažera důležité hluboké technické znalosti, ale především umění jednat s lidmi.

Etapy projektu z pohledu managementu



Obrázek 4.1: Intenzita činnosti v průběhu projektu. [12]

- **Inicializace** je prvotní fáze projektu, kdy na základě sesbíraných informací rozhodneme, zdali projekt můžeme spustit.

- **Plánování** slouží k lepšímu pochopení cílů projektu. V této etapě je nejdůležitějším úkolem stanovit si realistická očekávání. Nerealistická očekávání, která jsou založena na nepřesných odhadech, jsou největší příčinou selhání projektu. Definují se zde požadavky na zdroje, práci, určují se kvalitativní, kvantitativní a ekonomické cíle. Plánování představuje obvykle menší a časově kratší úsek projektu.
- **Řízení** je průběžná kontrola prováděného projektu. Jeho účelem je sběr dat, pro další vývoj a kontrolu následujících kroků. Na základě porovnání aktuálního stavu a vytvořeného plánu je lepší provádět průběžná opatření, než řešit případné problémy. Z obrázku je patrné, že řízení začíná v čas zahájení projektu a končí při jeho ukončení.
- **Prováděním** se rozumí realizace plánu projektu. Je časově i finančně nejnáročnější. Manažer projektu koordinuje činnosti projektu, přiděluje úkoly, stanovuje priority, sleduje postup prací a jeho snahou je řídit tento proces podle plánu tak, aby jeho výsledkem bylo ukončení díla (výrobku nebo služby).
- **Ukončením** končí smlouva mezi dodavatelem a zákazníkem. Spolupráce zpravidla nekončí, neboť zákazník je v kontaktu s dodavatelem i nadále a mohou spolu řešit následné dílčí poznatky, které jsou užitečné pro budoucí projekty.

4.1.1.3 Kvalita softwarových projektů

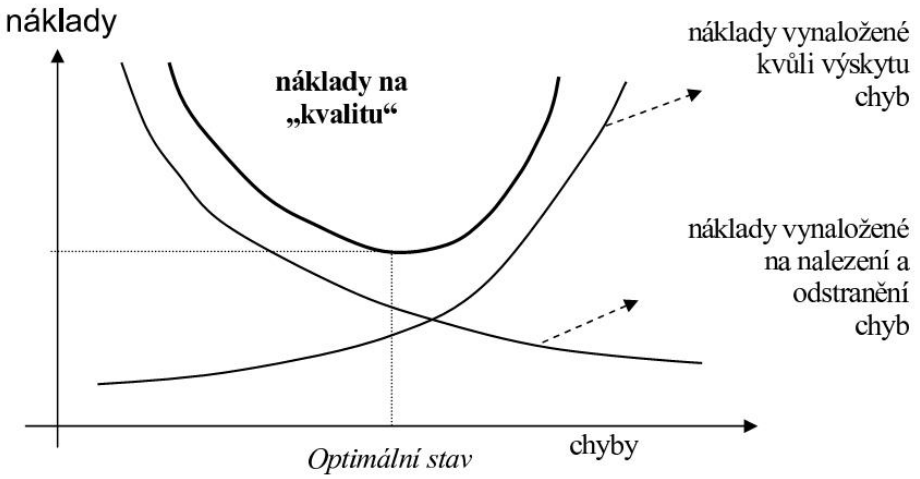
„*Obvyklý postup při tvorbě softwaru spočívá v co nejrychlejší implementaci programu neznámé kvality, v rychlém testování s cílem najít a odstranit chyby a nedostatky. V žádném jiném technickém oboru se nevytvářejí výrobky nekontrolovatelné kvality, která se zpravidla odhaluje až při testování.*“[13]

Uživatel kvalitu může vnímat s pohledu jednoduchosti obsluhy, přehlednosti systému nebo jeho spolehlivosti. Uživatel bude ve výsledku software využívat, proto by se na tyto kritéria mělo důsledně myslet.

S pohledu administrátora a tvůrce systému je kvalitou také myšlena přehlednost a čitelnost modifikovatelného zdrojového kódu. Důraz je také kladen na kvalitní dokumentaci.

Manažer vnímá kvalitu z pohledu spíše ekonomického. Důležitý je termín dodání s dodrženým rozpočtem a splněnou funkčností.

Množství chyb v softwaru také není dostačující ani nezbytnou podmínkou pro určení kvality software. V každém trochu složitějším softwaru pozorujeme větší či menší výskyt chyb. Kvalitu dodaného software dokumentuje také množství dodatečných nákladů, které je potřeba vynaložit na odstranění chyb v softwaru. Poroste-li počet chyb v softwaru, porostou i náklady na jejich odstranění. Tato závislost je znázorněna v grafu rostoucí křivkou.



Obrázek 4.2: Optimální kvalita softwaru. [14]

Celkové náklady na tvorbu softwaru jsou dány náklady na jeho vývoj a také náklady na odstraňování chyb ze softwaru.

Kvalitou výrobků (i softwaru) v evropských zemích se zabývá mezinárodní soustava norem ISO 9000. Obecně však lze říct, čím má firma kvalitnější proces řízení projektů, tím bude jejich výsledný produkt (software) vyšší kvality.

4.2 Plánování projektu

Plánování projektu je další důležitou částí vývoje softwaru. Tuto část vývoje má standardně na starost projektový manažer. Jeho úkolem je rozdělit projekt na množinu pod-úkolů, u kterých odhadne čas jejich doby zpracování. Dále musí přiřadit ke každému úkolu zodpovědnou osobu, která se bude starat o bezproblémový chod plnění úkolu.

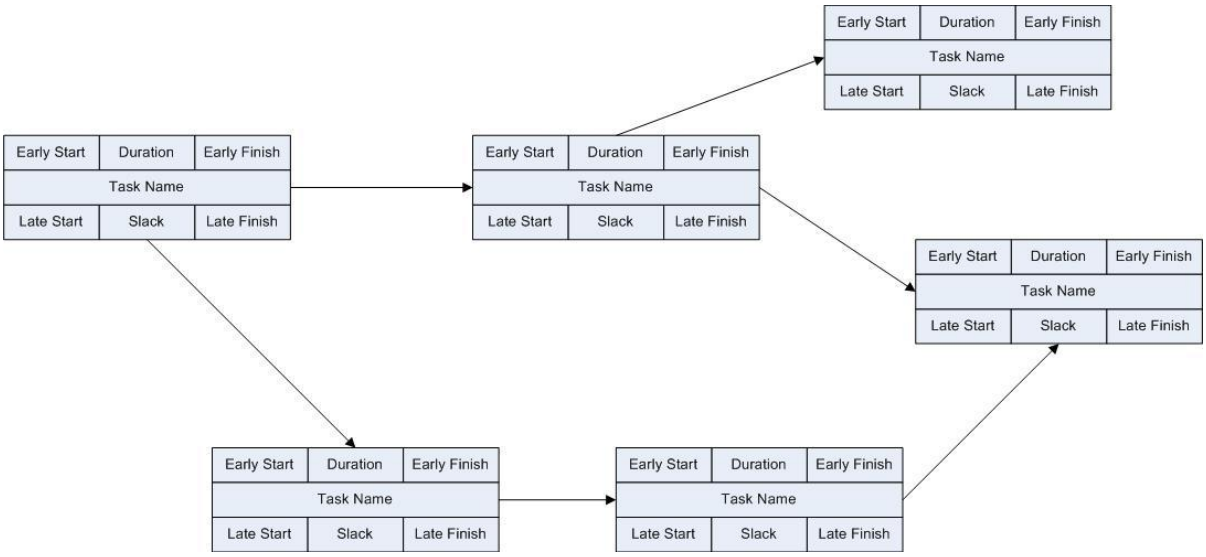
K modelování závislostí nebo délky trvání nebo struktury projektů slouží řada diagramů. Lze je rozdělit mezi síťové, sloupcové.

4.2.1 Síťový diagram

Síťový diagram (také nazývaný diagram závislosti) modeluje úkoly tak, jak na sobě závisí. Řada úkolů nemůže začít, aniž by skončil jiný. Zpravidla definuje i čas, který je potřeba vynaložit na splnění daného cíle. Lze tak například určit celkovou dobu trvání celého projektu.

4.2.1.1 PERT diagram

Typickým představitelem diagramu závislosti je diagram PERT (Program Evalution and Review Technique). Aktivity jsou zde zobrazeny jako uzly a vztahy mezi nimi jsou znázorněny jako spojnice. Následující obrázek představuje ukázku PERT diagramu.



Obrázek 4.3: Ukázka PERT diagramu.

Cílem PERT diagramu je rozfázování rozsáhlého projektu na jednotlivé elementy, u kterých můžeme zkoumat časový průběh. Lze vypočítat optimistickou, pesimistickou a nejpravděpodobnější dobu trvání.

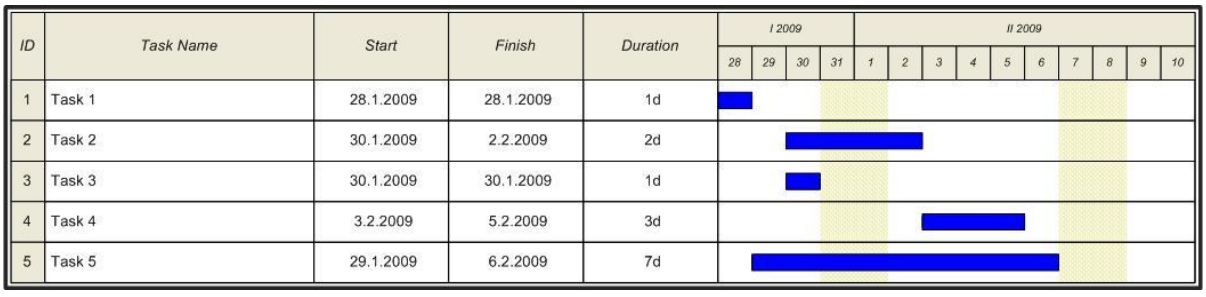
4.2.2 Sloupcový diagram

Hlavním představitelem tohoto typu je Ganttův diagram, který se využívá pro grafické naplánování posloupnosti činnosti v čase.

4.2.2.1 Ganttův diagram

Tento diagram byl jeden z prvních, který se vryl do podvědomí projektových vedoucích. H. L. Gantt, podle kterého se tento diagram nazývá, je používal za první světové války. Byl jeho průkopníkem.

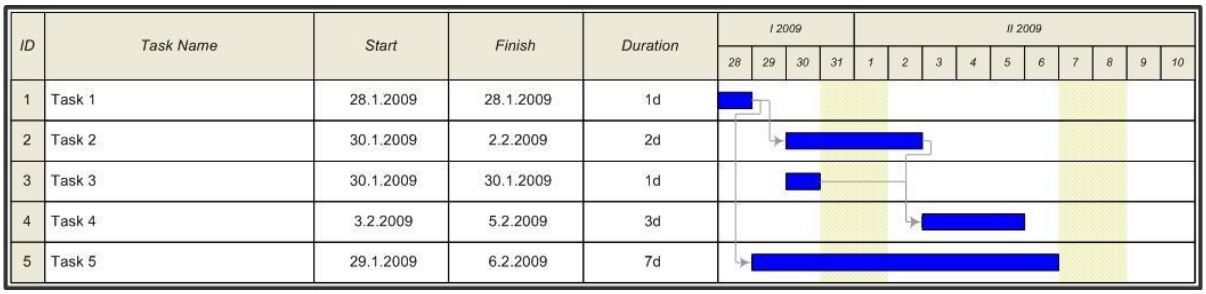
Pomocí vodorovných čar znázorňuje časovou dobu trvání. Jejími jednotkami mohou být dny, týdny, měsíce. Každý jeho řádek představuje prováděnou činnost, která je znázorněna obdélníky, které navíc zobrazují aktuální stav a časový průběh.



Obrázek 4.4: Základní Ganttův diagram.

4.2.2.2 Rozšířený Ganttův diagram

Ganttův diagram v původní podobě neobsahoval závislosti mezi jednotlivými elementy, avšak moderní softwarové produkty určené k plánování tuto modifikaci již mají.



Obrázek 4.5: Rozšířený Ganttův diagram.

V tomto typu diagramu lze propojovat jednotlivé dílčí úkoly a vytvářet tak návaznosti mezi nimi.

5 Analýza systému pro správu projektů

Jednou z nejdůležitějších částí tvorby softwaru je jeho analýza. Je nutné si ujasnit, co se od systému očekává. Prvním zdrojem informací o funkčnosti systému jsou požadavky zákazníka. Ty jsou mnohdy velmi obecné a je nutné je dále blíže analyzovat a specifikovat. V této fázi se ještě nezabýváme, jakým způsobem je požadavků docíleno, ale soustředíme se na požadovanou funkcionalitu. V další části je tyto neformální požadavky nutno transformovat do přesnější formy, která představuje zadání projektu. Na základě těchto informací lze vytvořit diagram případu užití (*use case)*, který zachycuje funkčnost systém vykonávanou jedním z účastníků.

Protože systém nebyl zadáván žádnou firmou ani jiným přesně definovaným zadáním, mohl jsem požadavky na funkčnost systému navrhnout sám za pomoci mého vedoucího pana Ing. Ladislava Ruttkaye a za pomoci menšího vzorku spoluobčanů. Rozhodl jsem se proto dotázat skupiny lidí, kteří jsou bezpochyby schopni poskytnout cenné informace a nápady odlišné od těch mých.

5.1 Získání informací o možných požadavcích na systém

Jednou z možností, jak získat informace o požadavcích na daný software, je interakce s budoucím zákazníkem. Lze využít řadu prostředků, mezi které patří například interview při osobním kontaktu, rozbor již existujícího podobného systému nebo také dotazník.

Abychom splnili co nejlépe požadavky uživatelů, je vhodné ještě před zahájením procesu tvorby zakázky použít systém dotazování. Je vhodné ho také použít v průběhu vytváření a po vytvoření úkolu.

Před návrhem dotazníku je nutné si ujasnit několik věcí. Dotazník musí být výstižný, stručný, jednoznačný a jednoduchý. Lidé velmi neradi čtou rozsáhlé texty a ještě méně ochotně rozsáhlé texty

vypisují. Proto je potřeba jim nabídnout výběr z různých možností a „fulltextových“ odpovědí používat co nejméně.

5.1.1 Prvotní představa dotazovaných o softwaru pro Správu projektů

První otázka pro potencionální zákazníky byla: „Co požadujete od systému ke správě projektů z pohledu zaměstnance?“ Druhá otázka byla: „Co požadujete od systému ke správě projektů z pohledu zaměstnavatele?“

Hned na první pohled je patrné, že otázky jsou velmi obecné. Z toho důvodu byly odpovědi velmi různorodé. Tato skutečnost ale vyhovovala mně, protože jsem se potřeboval rozhodnout, na kterou část systému se zaměřit.

Dotazoval jsem se lidí, kteří již přišli do styku s podobnou tématikou. Názory na obě otázky se radikálně lišily. Rozdílně odpovídal člověk v roli vedoucího a jinak člověk v roli zaměstnance.

V roli zaměstnance zazněly nejčastěji tyto odpovědi:

- Pokud bych musel takový software používat, tak bych požadoval ulehčení práce a ne jen povinnost někomu sdělovat informaci o mé práci.
- Jednoduchým způsobem zjistit, co mám dělat, v jakém rozsahu, co se ode mě očekává. Kolik mám na daný problém času.
- Chtěl bych mít po ruce veškeré organizační záležitosti. (kalendář, poštu, dokumenty)
- Chtěla bych mít přehled o kontaktech na lidi, s kterými potřebuji komunikovat.
- Chtěl bych, abych mohl se systémem pracovat, i když nejsem připojený na internet.
- Hlavně aby to nebylo složité a abych se neuklikal.
- Vhodná velikost písma a přehledné rozložení prvků na obrazovce.
- Přítomnost nápovědy.

V roli zaměstnavatele zazněly nejčastěji tyto odpovědi:

- Chci mít přehled o svém čase a o čase svých podřízených.
- Software nemá být vhodný jen pro mě, ale i také pro mé pracovníky. Bylo by k ničemu, kdybych já měl z takového systému radost, ale ostatní by nadávali.
- Hned na první pohled vidět, jak se věci mají a kde je potřeba zasáhnout.
- Mít představu o vytíženosti svých zaměstnanců
- Mít představu o rozpočtu.
- Jednoduchým způsobem navrhovat strukturu projektů.
- Chtěl bych mít možnost nastavovat práva, všichni nemusí vidět všechno.
- Mít přehledně uložené kontakty na své zaměstnance i na zákazníky.
- Líbí se mi grafy. Takže hodně statistik.
- Kvalitní vyhledávání všeho možného.

5.1.1.1 Zhodnocení prvotních představ

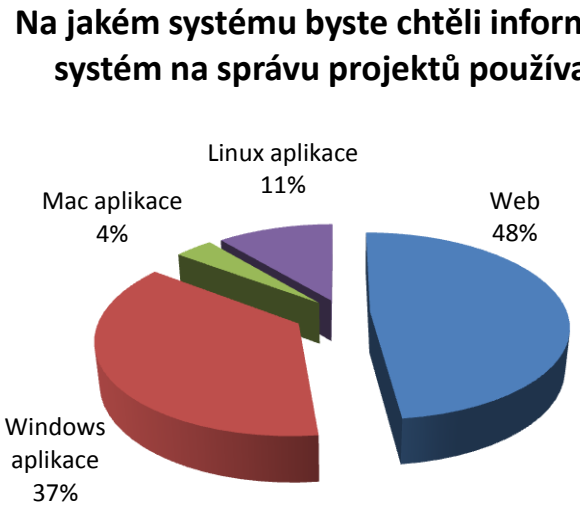
Po položení výše zmíněných dvou otázek lidem, kteří s PC téměř nepřišli do styku, jsem se rozhodl oslovovat pouze lidi, u kterých jsem předpokládal aspoň malou zkušenost s danou problematikou.

Zazněly dosti odlišné názory. Někteří chtěli aplikace, které jsou běžně dostupné na internetu. Například kalendář, poznámkový blok, chat. Tyto aplikace se jistě hodí, jistě by mohly být časem součástí systému. Ale se správou projektů nejsou zcela spjaté.

Z odpovědí jsem usoudil, že skloubit vše je téměř nemožné. Vyhovět navíc současně manažerovi a zaměstnanci bude také problém. Dotazovaní požadují systém, který jim ulehčí práci, je pro ně zdrojem cenných informací, které chtějí mít pohromadě.

5.1.2 Systém, na kterém se bude aplikace provozovat

I když systém a platforma byly dány v zadání bakalářské práce, rozhodl jsem se ze zajímavosti položit otázku, na jakém systému by si přáli uživatelé provozovat informační systém. Na výběr měli z operačních systémů: Windows, Mac OS, Linux nebo provoz přes webové rozhraní.

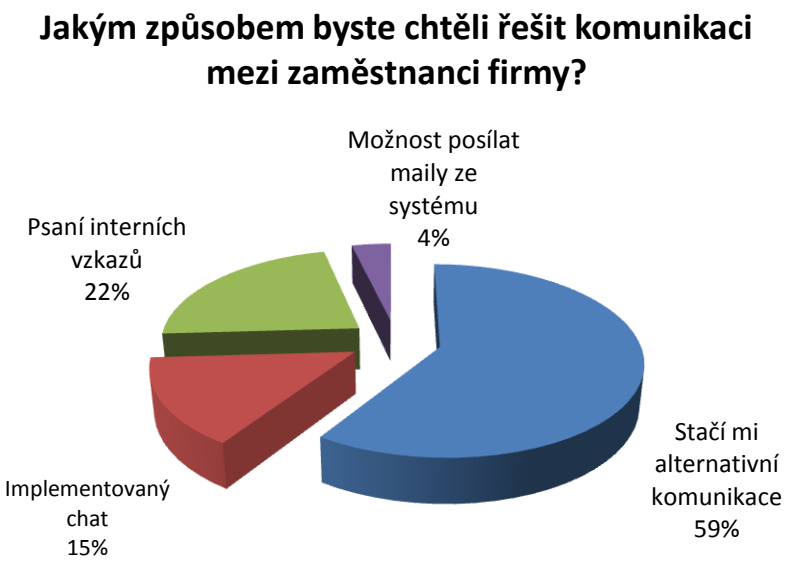


Obrázek 5.1: Graf použití na jednotlivých systémech.

Nejvíc se dotazujícím zamlouvala služba, na kterou mohou přistupovat přes internet pomocí webového prohlížeče. Není totiž potřeba instalovat žádný software. Téměř každý dnes má na svém počítači internetový prohlížeč s připojením na internet. Avšak od systému se požaduje práce offline, která by šla implementovat přes webové rozhraní jen stěží. Proto jsem se rozhodl pro vytvoření aplikace pracující pod operačním systémem Windows za pomoci .NET Frameworku 3.5.

5.1.3 Způsob komunikace

Přemýšlel jsem, jakým způsobem by se mohli spolupracovníci mezi sebou dorozumívat. Proto jsem dodatečně zjišťoval, jakým způsobem by chtěli komunikovat.



Obrázek 5.2: Graf rozložení komunikace.

I když většina lidí nepotřebuje komunikovat přes tento systém, rozhodl jsem se přidat do aplikace jednoduché zasílání zpráv, které usnadní komunikaci mezi lidmi ve firmě.

5.2 Shrnutí požadavků

- **Jednoduchost**. Aplikace se bude skládat jen z nejnutnějších prvků, popsaných níže. Cílem je vytvořit program, se kterým se uživatel naučí velmi rychle a jednoduše pracovat.
- **Přehlednost**. Jednotlivé elementy aplikace musí mít logickou stavbu a logické uspořádání. Text musí být čitelný, nejlépe nastavitelný. Obrázky jasné a zřetelné.
- **Intuitivnost**. Systém by měl rozumět uživateli a rozumět jeho požadavkům. V každém kroku by měl uživatel vědět, jakých možností lze v systému dosáhnout.
- **Autentizace**. Zaměstnanec se bude moci přihlásit. A podle role, kterou přistupuje k jednotlivým úkolům, se budou řídit jeho práva.
- **Registrace**. Do systému bude moci administrátor přidávat/mazat nové zaměstnance.
- **Zobrazení statistik.**
- **Grafické vytváření struktury projektu a její editace**. Blíže specifikováno v kapitole grafický návrh.
- **Promítnutí délky trvání úkolu do grafického zobrazení.**
- **Vkládání informací k úkolům**. Zaměstnanec, který bude pracovat na úkolu, bude moci vkládat nové informace o stavu zakázky.

- **Vkládání dokumentů**. Do aplikace se budou moci vkládat dokumenty k jednotlivým úkolům.
- **Zasílání zpráv**. V aplikaci bude umožněno zasílat zprávy jednotlivým uživatelům a jednotlivým úkolům.
- **Práce offline**. Systém bude moci pracovat offline, kdy není nutné být připojený k databázi. Práce bude omezena na níže specifikované úkony.
- **Pomocí webových služeb poskytnout základní informace**. Např.: zobrazení zaneprázdněnosti pracovníků. Zobrazení problémových projektů.

5.3 Use Case (diagram případů užití)

Na základě předchozí kapitoly lze vytvořit use case diagram, který blíže specifikuje akce, kterými budou disponovat jednotliví účastníci systému. Znázorněny jsou dva diagramy. Jeden modeluje situaci, kdy má lokální aplikace přístup k databázovému systému, druhý znázorňuje režim offline, kdy spojení aplikace s databází nelze navázat.

5.3.1 Online mód pro práci s projektem

Tento diagram případové studie představuje lokální aplikaci, která má přístup k serveru s dostupnou databází. V diagramu se vyskytuje pojem element. Ten slouží pouze k zobecnění pojmu projekt a úkol. Aplikace bude graficky modelovat strukturu úkolů a projekt je jen zastřešení do jejich logické hierarchie. V systému bude projekt stejného typu jako úkol a rozlišit půjde podle toho, že nebude mít žádné předchůdce. Jedná se tak o první úkol, který vznikne při založení nového projektu. Smazáním tohoto úkolu (představující projekt) se smaže celý projekt.

Obrázek 5.3: Use case diagram režimu online.

Aktéři:

V use case diagramu režimu online jsou dva aktéři. Jeden představuje uživatele a druhý představuje synchronizačního klienta, který je zodpovědný za uložení offline změn na server.

- **Uživatel** – představuje uživatele systému.
- **Synchronizační agent** je proces, který se aktivuje při přechodu do stavu online, kdy je možné komunikovat se serverem. Přenese na něj změny, které uživatel zadal v režimu offline (viz. níže).

Případy užití:

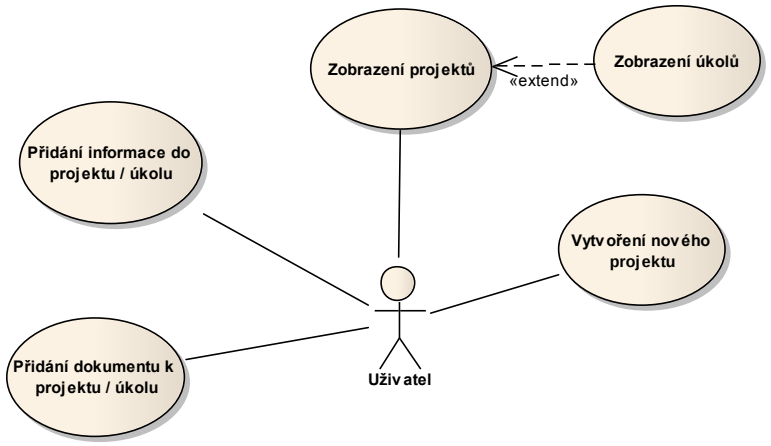
Z důvodu velkého počtu případů užití, uvedu pouze pár vybraných.

- **Editace struktury projektu –** slouží k vytváření grafické struktury projektu. Konkrétněji umožňuje vložení nového, navazujícího a vložení předchozího úkolu. Podrobněji bude tato problematika probrána v následujících kapitolách.

- **Smazání úkolu – s**louží i ke smazání projektu. Rozšiřuje ho situace, kdy má daný úkol nějaké potomky. V takovém případě je nutné rekurzivně smazat i tyto pod-úkoly.
- **Smazání projektu –** viz. *Smazání úkolu.*
- **Skrytí stromu úkolů** – skryje strukturu podřízeného úkolu. Lze si jej představit jako skrytí souborů nebo adresářů v procházení stromové struktury na disku.

5.3.2 Offline mód pro práci s projektem

V následujícím diagramu případové studie je nastíněna lokální aplikace, která nemá přístup k serveru. V tomto případě je data třeba načítat ze záložního XML souboru, který byl vytvořen a aktualizován na lokální stanici, když byla aplikace v režimu online. Pokud však tento soubor neexistuje (aplikace nebyla v režimu online), je uživatel upozorněn, že je nutné aktualizovat data v režimu online. Podrobnější procesní schéma je níže v kapitole 6.2.



Obrázek 5.4: Use case diagram režimu offline.

Aktéři:

- **Uživatel** – představuje uživatele systému.

Případy užití:

- **Zobrazení projektů** – pokud má projekt i nějaké pod-úlohy, zobrazí se i ony.
- **Vytvoření nového projektu –** uživatel muže zakládat nové úkoly v režimu offline.
- **Přidání informace do úkolu –** uživatelé mohou zasílat aktuality k jednotlivým úkolům.
- **Přidání dokumentu –** systém umožní uživatelům ukládat data k jednotlivým úkolům.

5.4 Struktura projektu

Je řada způsobů, jakými lze plánovat, udržovat a zobrazovat průběh projektů. Dva příklady jsem uvedl v kapitole 4.2. Jednalo se o PERT diagram a Ganttův diagram. Mým úkolem bylo navrhnout co možná nejpřívětivější rozhraní, které by dokázalo plánovat a organizovat projekt jednoduše a

přehledně. Na první pohled uživatel uvidí, která část je kritická, kolik na ni zbývá času a kolik zbývá financí. U jednotlivých úkolů nalezne veškeré potřebné informace, které jsou nezbytné pro dokončení daného problému.

Struktura se bude skládat z prvků, které budeme nazývat úkoly. Ty budou představeny grafickým obrázkem, který bude mít takovou velikost, jakou představuje délka jeho trvání v reálné situaci. Každý úkol může mít maximálně jednoho následovníka. To znamená, že až po splnění předcházejícího cíle je možné pokračovat v navazujícím cíli. Jedná se tedy o návaznost časů. Abychom mohli strukturu projektů členit na podřízené prvky, je nutné zavést hierarchii do navrhovaného diagramu.

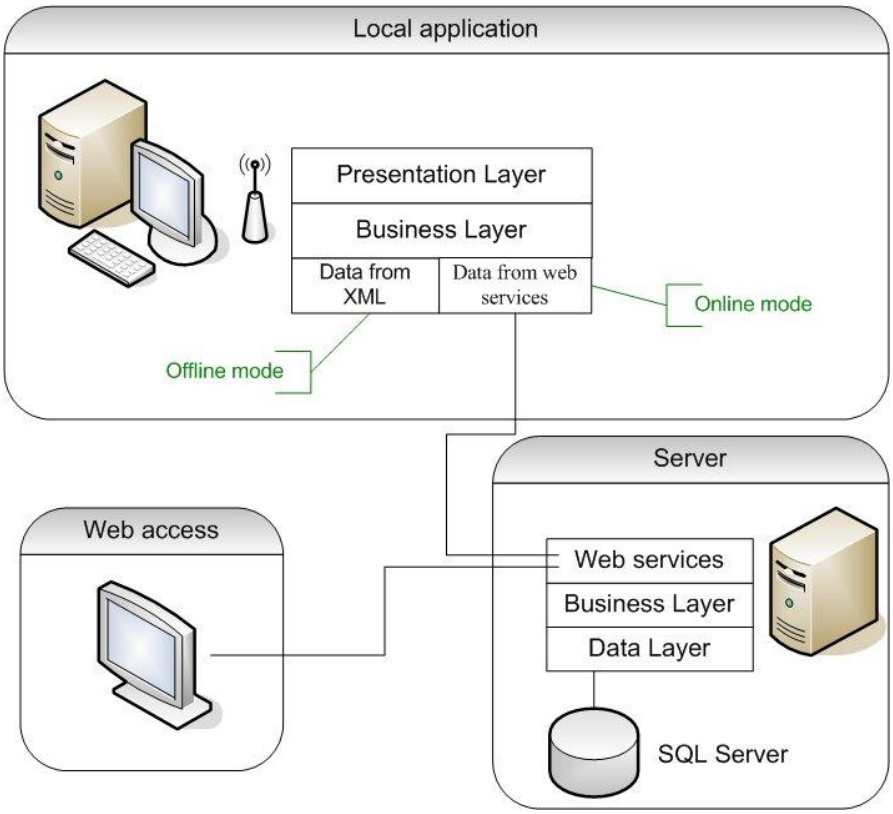
Při návrhu databáze bude nutné tyto specifikace důkladněji promyslet a kvalitně navrhnout. Již teď je zřejmé, že tabulka, která uchovává údaje o elementech projektů, bude obsahovat dvě rekurzivní vazby sama na sebe.

6 Návrh systému

Celý systém se bude skládat z několika částí. Bude zde serverová část, poskytující webové služby, ke kterým se bude připojovat Windows aplikace a aplikace běžící prostřednictvím webového prohlížeče. Webové rozhraní bude sloužit pouze k zobrazování základních statistik. Windows aplikace bude sloužit jako tenký klient, který bude prostřednictvím webových služeb umožňovat správu projektů a vše co ke správě patří.

Pro efektivní a praktické programování je zvolena troj-vrstvá architektura, která nám zaručí přehlednost a pozdější jednodušší úpravy systému. Pokud se například změní databáze, stačí pouze přeprogramovat Datovou vrstvu (Data layer). Stejně to platí o jakékoli jiné části. Při změně kterékoli vrstvy je však nutné zachovat rozhraní, pomocí kterého vrstva komunikuje s okolím.

Na následujícím obrázku je znázorněno fyzické rozložení celého systému. Databáze poskytuje data, ke kterým přistupuje Internet Information Services (IIS). Ten dále pomocí webových služeb umožňuje lokální aplikaci přijímat data. Stejným způsobem zprostředkovávají své služby i webovému klientu.

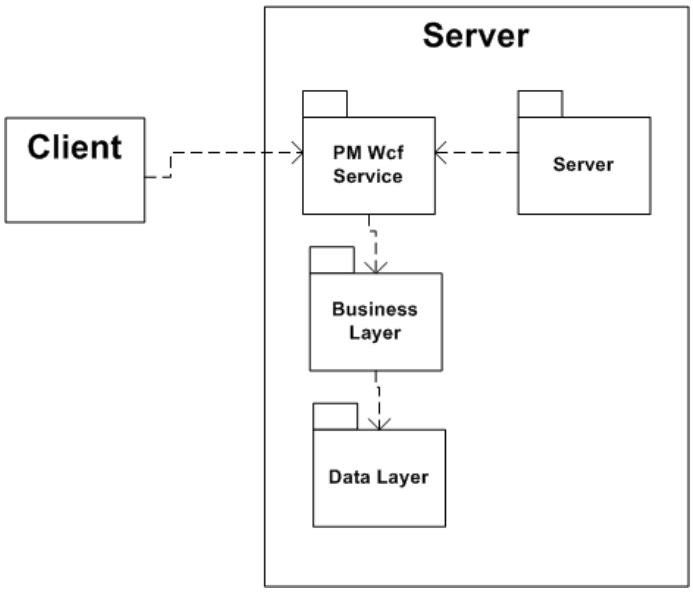


Obrázek 6.1: Celkové schéma informačního systému.

6.1 Serverová část

Serverová část systému slouží ke komunikaci s klienty na úrovni webových služeb a pro komunikaci s databází. Datová vrstva pracuje nad databázovým systémem MSSQL2008. Komunikační vrstva je implementována pomocí Windows Communication Foundation, který je obsažen v .NET Frameworku 3.5 a vyšším.

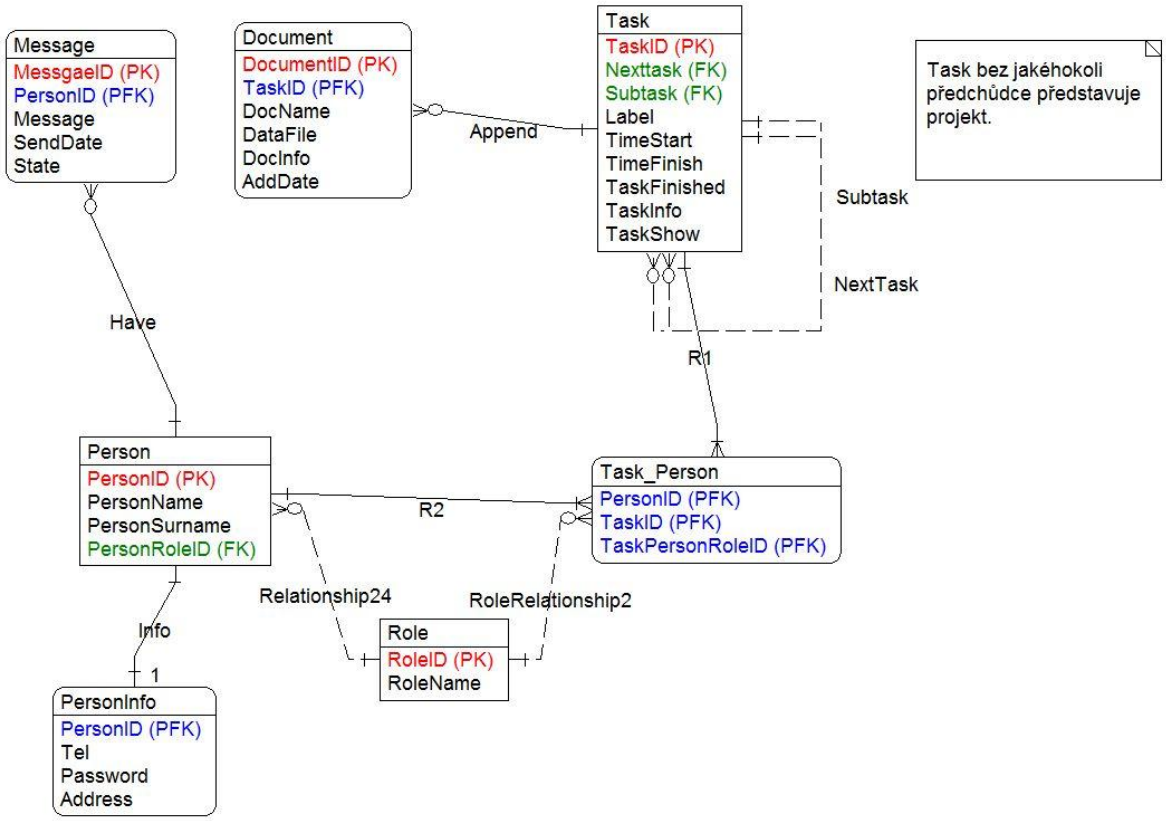
Strukturu serveru lze přehledně vidět na následujícím diagramu balíčků, který odpovídá předchozímu návrhu z Obrázek 6.1: Celkové schéma informačního systému. Kde Datová vrstva slouží pro komunikaci s databází. Obsahuje konkrétní SQL příkazy pro komunikaci s databází. Datovou vrstvu používá výpočetní vrstva (Business layer), ve které je obsažena logika serverové části. Jejím úkolem je zpracovávat požadavky od nadřazené vrstvy, připravit potřebná data pro odeslání klientovi a pomocí Datové vrstvy obsluhovat databázi. Komunikační vrstva (PM Wcf Service) slouží ke komunikaci s připojeným klientem. Poskytuje funkce, které je možné přes tcp protokol vzdáleně volat. Aplikace Server lokálně spouští celou serverovou část na daném portu.



Obrázek 6.2: Diagram balíčku serverové části.

6.1.1 Návrh databáze

Pro perzistentní uložení dat v relační databázi je potřeba navrhnout strukturu, kterou popíši E-R diagramem.



Obrázek 6.3: E-R diagram databáze.

Základem je entita *Task* (úkol). Slouží pro uchování informací o úkolech. Rekurzivní vazbou je sám na sebe napojen dvakrát. Jednou jako *Subtask* (pod úkol) a podruhé jako *Nextask* (navazující úkol). Toto propojení nám umožní vytvořit navrženou grafickou strukturu, která je hlavní částí celého systému. Jednotlivé projekty uložené v systému jsou ve skutečnosti také úkoly, avšak s tím rozdílem, že projekt nemá žádného předchůdce. Proto je možné je v tabulce úkolů rozlišit.

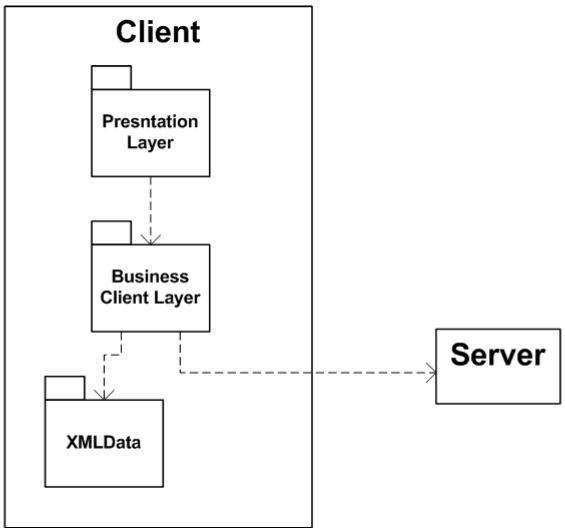
Na jednotlivé úkoly je napojena tabulka se soubory.

Další entitou je tabulka *Person* (uživatel). K uživateli jsou ve vazbě 0..N navázány zprávy. Dále jednomu uživateli odpovídá jeden záznam z tabulky *PersonInfo* (informace o osobě). Osoba ve firmě je označena určitou rolí ve firmě přes entitu *Role*.

Poslední entitou je tabulka *Task_Person*, která spojuje osobu systému s konkrétním úkolem v projektu s určitou přiřazenou rolí. Jako roli v úkolu si můžeme představit například: zodpovědnou osobu, osobu spolupracující na úkolu a podobně.

6.2 Lokální klientská část

Klientská část aplikace slouží ke komunikaci systému s uživatelem, k obstarávání dat ze serveru a také k udržování aktuálních dat na lokálním počítači pro režim offline. Jako u serverové části je vhodné navrhnout diagram balíčků, který bude sloužit jako hrubá kostra k vývoji programu. Diagram je nastíněn na obrázku Obrázek 6.4: Diagram balíčků klientské části.



Obrázek 6.4: Diagram balíčků klientské části.

Xml data layer (Xml datová vrstva) slouží k práci s offline xml soubory. Je to „náhradní“ zdroj dat, kdy není dostupný server. *Business client layer* (Logická klientská vrstva) se stará o spojení datových zdrojů s prezentační vrstvou. Jako nejbližší vrstvou k uživateli je *Presentation layer*

(prezentační vrstva), která tvoří GUI (Graphic user interface) celého systému. Bude tvořena za pomoci Windows presentation foundation (WPF).

Klienta je možné rozdělit na dvě části. Jednu, kdy aplikace má přístup k internetu a druhou, kdy program přístup k internetu nemá. Pro tyto účely jsou použity dva XML soubory, které se vytváří podle následujícího procesního diagramu.

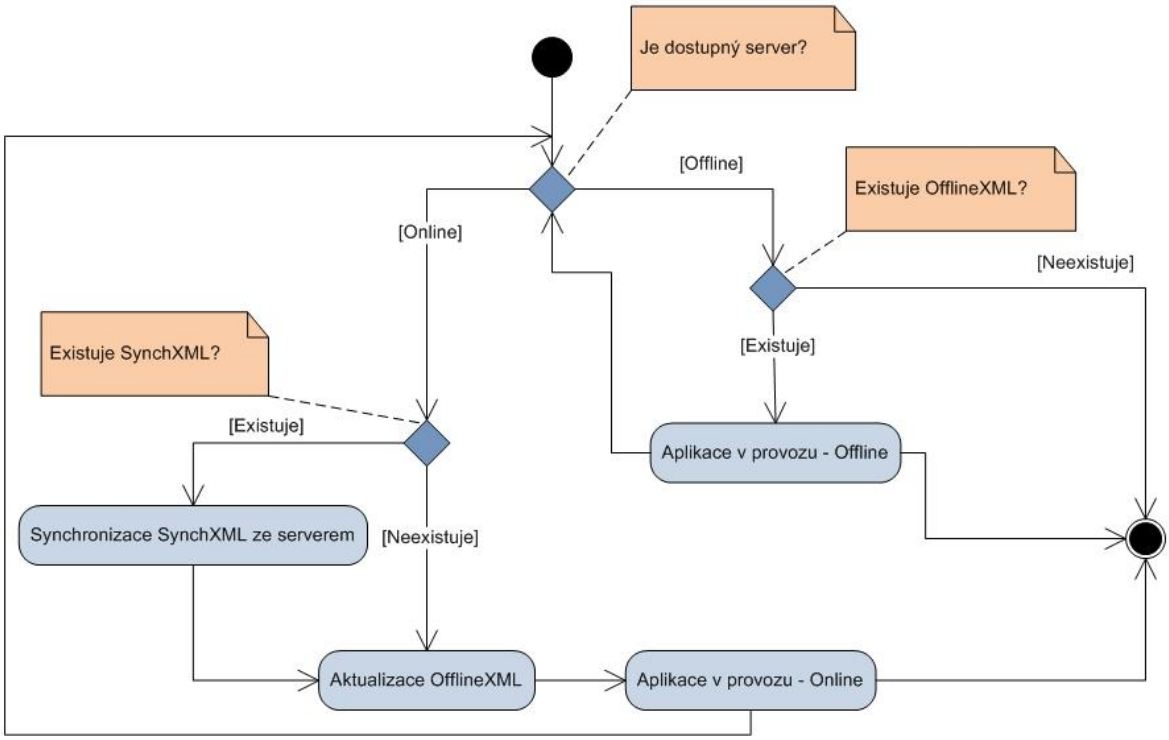
OfflineXML slouží k uchovávání dat, které se uloží do xml při připojení aplikace k serverové části. Ukládají se informace o zaměstnancích a také základní informace o veškerých úkolech, které lze poté měnit a synchronizovat se serverem.

Dalším xml souborem je SynchXML, který slouží pro uchování změn provedených v režimu offline. V aplikaci jsou možné jen základní úpravy z důvodu, kdy by si jednotliví zaměstnanci mohli přepisovat údaje. Proto jsou povoleny jen operace, kdy jsou data přidávána do databáze.

6.2.1 Proces řešící režim offline a online

Při startu aplikace je nutné zjistit, jestli je možné připojit se na server a získávat aktuální informace. Pokud ano, musí zkontrolovat, zda existuje soubor se synchronizačními údaji. Když existuje, tak jsou změny odeslány na server a ten je dále zpracovává. V opačném případě se tento krok přeskočí a můžou být aktualizována offline data uložená lokálně na počítači. Následně je aplikace v režimu online a uživatel s ní pracuje. Když aplikace nemůže navázat spojení se serverem, je nutné zjistit, jestli jsou v počítači uložena offline data z minulého spuštění, která jsou poté využita, jako podklad pro režim offline. Jestli nejsou, běh aplikace nemá smysl a dokud se uživatel nespojí se serverem, není mu umožněna jakákoli práce. Pokud však offline data jsou k dispozici, je možné provádět úpravy, které jsou zaznamenávány do synchronizačního souboru.

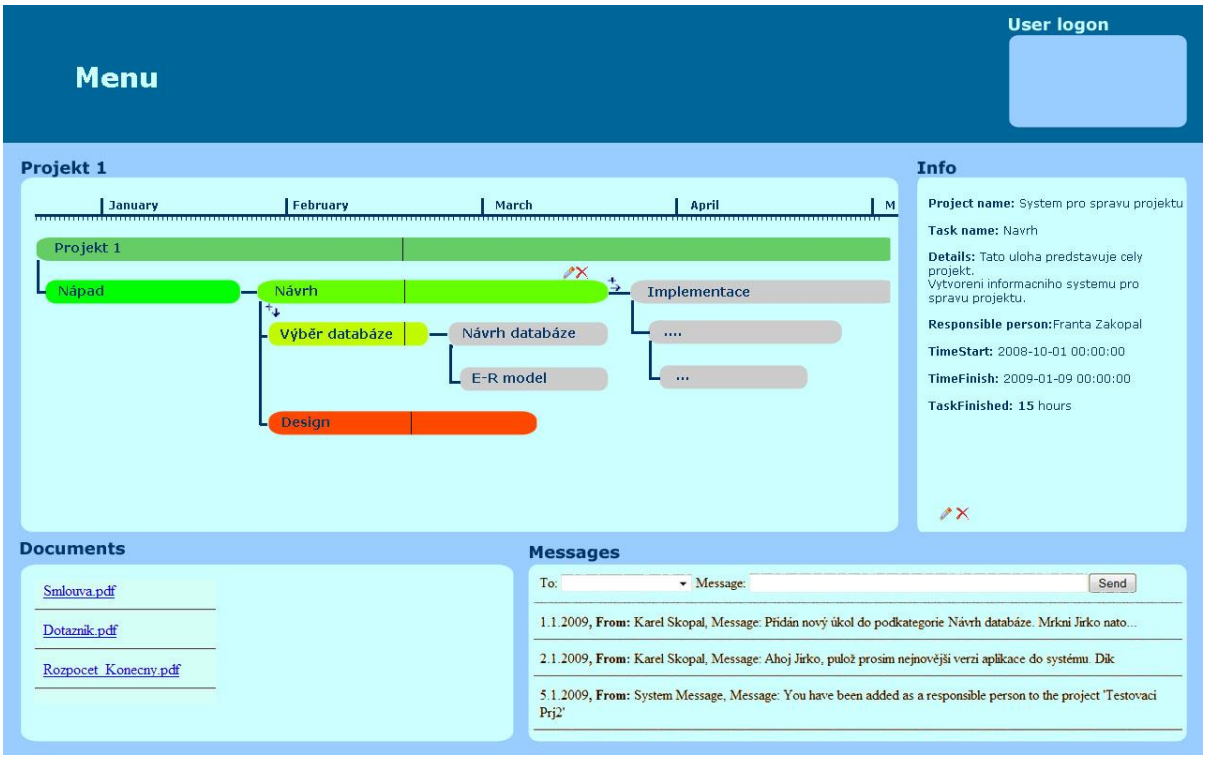
Pokud aplikace v jakémkoli ze dvou režimů zjistí, že se změnila dostupnost serveru, celý proces se opakuje.



Obrázek 6.5: Proces zavádění lokální aplikace.

6.2.2 Grafický návrh

Grafický návrh je vhodné vytvořit co možná nejpoutavěji a co možná nejpřehledněji, aby se uživatel mohl bez problému orientovat v aplikaci. Je tedy nutné zřetelně oddělit menu od zbylé části. Je potřeba, aby menu bylo vždy dostupné z jakéhokoli stavu aplikace. Návrh ukazuje rozčlenění klientské aplikace na několik částí. V horní části je nastíněno umístění menu, které bude zpracováno pomocí obrázků. V pravém horním menu je místo pro přihlášení uživatele. Zbylé části se budou měnit podle aktuálního stavu aplikace. Prezentační vrstva bude programována pomocí WPF, které je určeno právě pro tvorbu „moderního“ uživatelského rozhraní, kde vývojáře omezuje pouze jeho „grafické cítění“. Je možné i oddělit práci vývojářskou s prací grafického návrháře, kdy grafik pomocí značkovacího jazyka XAML navrhne vzhled a vývojář naprogramuje chod aplikace. V případě mé práce obě tyto role zastanu já.



Obrázek 6.6: Grafický návrh klientské aplikace.

7 Realizace systému

V této kapitole jsou stručně nastíněny změny a postupy, které se v průběhu tvorby systému objevovaly. Málokdy se stane, že realizovaný systém přesně odpovídá návrhu. V zásadě však mohu konstatovat, že k žádným vážným změnám oproti návrhu při realizaci nedošlo.

Drobné změny nastaly z podnětu dotazovaných z kapitoly 5.1.1., kdy jsem dal přednost svému názoru zabudovat části aplikace, které uživatel nevyžadoval.

Úpravou prošel původní návrh režimu offline, kdy jsem chtěl, aby tento režim měl plnou funkčnost jako ve stavu, kdy je dostupný server. To však nebylo možné z toho důvodu, že lokální aplikace byla koncipována jako tenký klient, kde data používaná v klientu jsou zpracovávána na serveru, ze kterého jsou také odesílána. Dalším důvodem byl problém se synchronizací, kde by si jednotliví uživatelé mohli přepisovat strukturu projektu pod rukama.

Pomocí WPF jsem mohl grafický návrh zdokonalit a vytvořit tak lepší grafické uživatelské rozhraní celého systému. Netradičním způsobem jsem vytvořil systém převedení číselně počítaných údajů do grafického vyjádření. Například vizuální zobrazení délky trvání projektu, atd.

8 Odezva na vytvořený systém

K odezvě od dotazovaných jsem bohužel musel použít ne úplně dokončenou verzi systému. Proto výsledky nemají vysokou vypovídající hodnotu. Ale jsou prospěšné pro konečné doladění systému a pro následující rozšiřující vývoj aplikace.

Při dotazování na vytvořený systém jsem použil textový dotazník s konkrétně položenými otázkami. První čtyři otázky měli dotazovaní oznámkovat jako ve škole.

- **Jak se Vám zdá být systém přehledný?**
  Průměr: **1,9**
- **Pochopil systém, co po něm v daném okamžiku chcete?**
  Průměr: **2,3**
- **Líbí se Vám grafický design?**
  Průměr: **1,4**
- **Reagoval systém dostatečně rychle?**
  Průměr: **2,5**

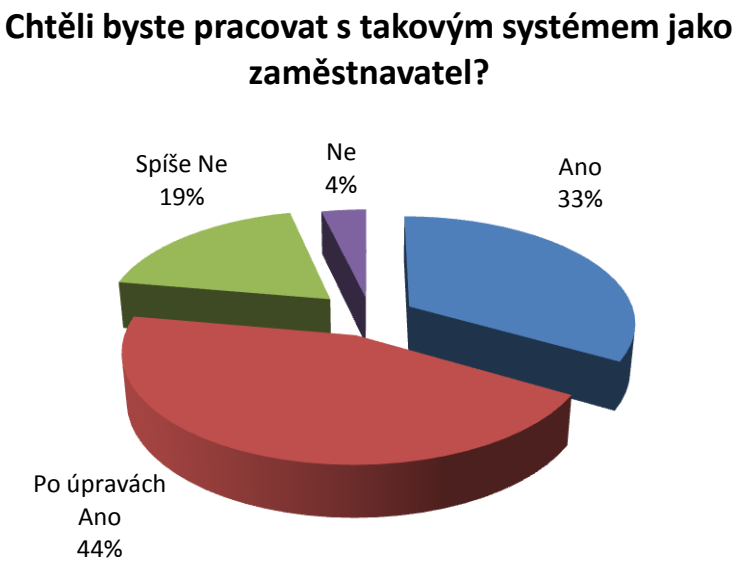
Ze známkovacího hodnocení lze vyčíst, v jakém směru by se měl systém v dalších úpravách změnit. Z průměrných hodnot je patrné, že lidem se líbil grafický obal. Také se jim líba intuitivnost systému a jeho přehlednost. Odezva systému se dotazovaným zdála přijatelná, avšak ne optimální. Zdůvodňuji si to prací přes internet, kde hraje velkou roli rychlost připojení.

U dalších otázek dotazovaní odpovídali pomocí možností.

Obrázek 8.1: Graf odpovědi 3.

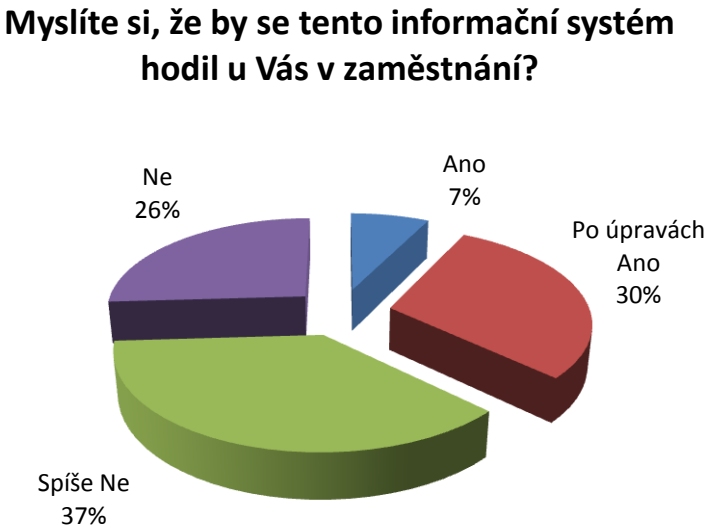
Do funkce uživatele, jako zaměstnance, se mnoho dotazovaných nehrnulo. Většina si vystačí s emailovou schránkou, Outlookem, popřípadě s jinými webovými službami, s kterými se „roztrhl

pytel“. Někteří však uznali, že při více úkolech, by si dokázali představit práci s touto aplikací. To bezpochyby autora potěší.



Obrázek 8.2: Graf odpovědi 4.

V roli vedoucího pracovníka si tento produkt dovedlo představit více dotázaných. Námitky byly spíše kvůli malému spektru užití. Usuzuji z toho, že lidé chtějí komplexní programy „na vše“. Zlomyslná je jistě myšlenka, že takový software bude právě kupovat zaměstnanec z vedení, a pokud se mu systém líbí, tak se musí zaměstnanci podřídit. Taková úvaha je ale jistě špatná a uvědomělý vedoucí si je toho vědom.



Obrázek 8.3: Graf odpovědi 5.

Na otázku, zda si dotazovaní myslí, že by se systém uživil u nich v zaměstnání, odpověděli spíše negativně. Někteří si ho však dokázali představit jako svůj organizátor času a úkolů.

A jako poslední položky byly otázky rozepisovací.

Jaké byste chtěli rozšíření nad rámec implementovaných?

- Kalendář,
- poznámkový blok,
- vícejazyčnost,
- detailnější nastavení,
- stejnou funkčnost i přes webové rozhraní.

Co se Vám líbí?

- Aplikace je přehledná.
- Grafický vzhled je moderní.
- Když najedu myší na úkol, hned vidím možnosti, které s úkolem lze dělat.
- Nápad grafického zobrazení.

Co se Vám nelíbí?

- Problikávání stránky.
- Hodně věcí nedokončeno.
- Mně stačí Outlook.
- Zadávání času ručně - vypisováním.

Výše zmíněné dotazy lze použít jak k vytvoření nové a dokonalejší verze programu tak i k doladění některých detailů. Změna některých požadovaných funkcí již však možná není. Proto je vhodnější dotazovat se potencionálních uživatelů v průběhu vývoje.

9 Závěr

Tvorba informačního systému pro řízení projektů mě utvrdila v tom, že kvalitní návrh a jeho důkladné zpracování je velmi důležitou součástí vývoje softwaru. Při psaní této bakalářské práce jsem se naučil a dozvěděl nové informace o tvorbě návrhu zejména za pomocí jazyka UML. Aplikaci jsem programoval v dosud pro mě neznámém programovacím jazyku C# s použitím technologie .NET společnosti Microsoft. Zvláště jsem využil novinky z .NET Frameworku 3.5. V průběhu učení se nového jazyka jsem narážel velmi často na různé problémy a úskalí, jejichž řešením jsem pronikl do těchto technologií poměrně hluboko. Nyní si troufnu konstatovat, že bych dokázal tyto znalosti prodat v profesním životě. Zajímavým poznatkem pro mě je to, že nakoupené knihy o programovacím jazyku se nevyrovnají originální dokumentaci a poznatkům jiných vývojářů, kteří své zkušenosti a problémy sdílejí na internetu.

Důležitou částí práce je komunikace se zadavatelem, nejlépe zákazníkem, abychom jeho požadavky mohli splnit co možná nejlépe. V mém případě jsem si vystačil s názory mého vedoucího, Ing. Ladislava Ruttkaye, několika dotázaných spolužáků, rodinných příslušníků, spolupracovníků a kamarádů. Byl jsem překvapen, že jejich názory se v některých ohledech výrazně lišily od těch mých. Díky jejich postřehům jsem zjistil, že vymyslet systém, který se bude líbit všem, je nemožné. Proto jsem byl nucen dělat kompromisy a implementovat části tak, abych nepřekračoval rámec bakalářské práce, abych splnil zadání, a abych alespoň částečně docílil požadavků dotazovaných.

Informační systém pro management projektů je velmi rozsáhlá úloha a má práce jistě nepokrývá všechny možnosti. Naskýtá se zde proto prostor pro mnoho rozšíření a vylepšení. Vylepšit by bylo vhodné správu uživatelů a jejich práva. Dále pak udělat další přístupové rozhraní přes web. Jistě by se v aplikaci hodil osobní kalendář. Manažerům a vedoucím pracovníkům by přišlo vhod více statistik pro monitorování zaměstnanců a průběh projektů.

Na závěr bych chtěl zkonstatovat, že se mi podle mého názoru a názorů dotazovaných podařilo v přijatelné míře naplnit představy mé i jejich. Předpokládám, že vytvořený systém může pomoci při plánování a řízení projektů u dalších uživatelů.

Použité zdroje

HRUŠKA, T., KŘIVKA, Z. *Informační systémy (IIS, PIS) – studijní opora* [online]. Listopad 2008. 147 s. Dostupné z <https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaIIS2PISPojemDataProcesyTransakce.pdf>.

HRUŠKA, T., MÁČEL, M. *Pokročilé informační systémy – studijní opora* [online]. Říjen 2008. 239 s. Dostupné z < https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaPISAnalyzaNavrhImplementace.pdf>.

JONES, M. P. *Základy objektově orientovaného návrhu v UML*. Vyd. 1. Praha: Grada, 2001. 367 s. ISBN 80-247-0210-X.

KŘENA, B., KOČÍ, R. *Úvod do softwarového inženýrství – studijní opora* [online]. 6. 12. 2006. 100 s. Dostupné z <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/texts/opora.pdf>.

KŘIVKA, Z., KOLÁŘ, D. *Principy programovacích jazyků a objektově orientovaného programování IPP – II – studijní opora* [online]. Ver. 0.8. Únor 2006. 70 s. Dostupné z <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IPP-IT/texts/IPP-II-ESF-1_1_printable.pdf>.

PETZOLD, Ch. *Mistrovství ve Windows Presentation Foundation*. Vyd. 1. Brno: Computer Press, 2008. 928 s. ISBN 978-80-251-2141-2.

RUTTKAY, L. *Systém pro ekonomické analýzy v IS školicího střediska*. Brno: Vysoké učení technické, Fakulta podnikatelská, 2008.

SHARP, J. *Microsoft Visual C# 2008: krok za krokem.* Vyd. 1. Brno: Computer Press, 2008. 592 s. ISBN 978-80-251-2027-9.

Wikipedie [online]. Dostupné z <http://cs.wikipedia.org/wiki/Hlavní_strana>.

Seznam citací

[1] PUŠ, P. 1.díl. *Poznáváme C# a Microsoft .NET* [online]. 23. 11. 2004. [cit. 25. 1. 2009]. Dostupné z <http://www.zive.cz/Clanky/Poznavame-C-a-Microsoft-NET--1dil/sc-3-a-120978/default.aspx>.

[2] *Visual Studio vs. Eclipse IDEs* [online]. 11. 1. 2009. [cit. 25. 1. 2009]. Dostupné z <http://www.wilsonmar.com/msdotnet.htm>.

[3] *.NET* [online]. 21. 3. 2009. [cit. 25. 3. 2009]. Dostupné z <http://cs.wikipedia.org/wiki/.NET>.

[4] FIORATO, J. *Deciphering the .NET Framework Versions* [online]. 8. 2. 2008. [cit. 25. 1. 2009]. Dostupné z <http://www.hubbardone.com/hubbardperspective/blog.aspx?entry=22>.

[5] *Microsoft Visual Studio* [online]. 20. 1. 2009. [cit. 25. 1. 2009]. Dostupné z <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.

[6] PETZOLD, Ch. *Mistrovství ve Windows Presentation Foundation*. Vyd. 1. Brno: Computer Press, 2008. s. 7 – 9. ISBN 978-80-251-2141-2.

[7] *Windows Presentation Foundation I – základy a architektura* [online]. 26. 9. 2006. [cit. 17. 11. 2008]. Multimediální prezentace. <http://www.microsoft.com/cze/msdn/webcasts/default.mspx>.

[8] *Case Studio* [online]. [cit. 25. 1. 2009]. Dostupné z <http://www.casestudio.com/csy/default.aspx>.

[9] KŘENA, B., KOČÍ, R. *Úvod do softwarového inženýrství – studijní opora.* 6.12.2006. s. 85 - 92. Dostupné z <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/texts/opora.pdf>.

[10] Ibid. s. 85.

[11] Ibid. s. 86.

[12] Ibid. obr. 9.2, s. 87.

[13] Ibid. s. 88.

[14] Ibid. obr. 9.3, s. 89.

Seznam použitých zkratek

API	Application Programming Interface
CLS	Common Language Specification
MSSQL	Microsoft SQL Server
PERT	Program Evaluation and Review Technique
PHP	Hypertext Preprocessor
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
WSDL	Web Services Description Language
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Seznam příloh

Příloha 1. DVD

- Zdrojové kódy.
- Uživatelská příručka.