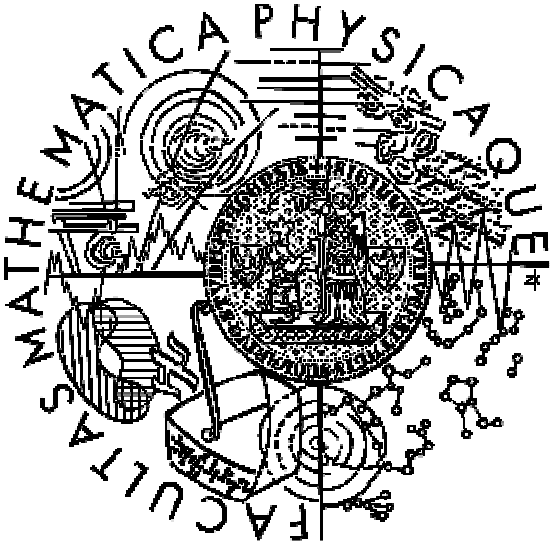
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Hampl

Webový systém pro správu projektů

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Martin Trčka

Studijní program: Informatika, obor Programování

2006

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním

V Praze dne Michal Hampl

Obsah

Obsah **3**

1 Cíl práce **5**
1.1 Důležitost systému pro správu projektů v praxi 5
1.2 Analýza podobných produktů 6

2 Volba způsobu realizace **9**
2.1 Prostředí aplikace 9
2.2 Konkrétní implementační technologie 10
2.3 Kde uchovávat data 11

3 Návrh struktury aplikace **13**
3.1 Evidované informace 13
3.2 Pevná nebo pružná hierarchie informací 17
3.3 Procesy v rámci aplikace 17
3.4 Systém uživatelských práv 19

4 Datový model **20**
4.1 Implementace hiearchické struktury 20
4.2 Různé typy uzlů 21
4.3 Základní struktura databáze 22
4.4 Implementace kalendáře 24
4.5 Způsob přístupu k databázi z aplikace 24

5 Struktura webové aplikace **26**
5.1 Standardní vzhled stránek 26
5.2 Pomocné třídy 27
5.3 Použité knihovny 28

6 Možnosti budoucího rozšiřování aplikace **29**
6.1 Optimalizace 29

7 Co obsahuje CD **30**

Název práce: Webový systém pro správu projektů
Autor: Michal Hampl
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Martin Trčka
e-mail vedoucího: martin.trcka@mff.cuni.cz

Abstrakt: Cílem práce je návrh a implementace internetové aplikace, která poskytne nástroje pro správu projektů. Aplikace bude primárně určena pro projekty vývoje software, díky obecnému návrhu však umožní využití i v jiných oblastech.
Aplikace bude nabízet následující funkce:

- správa uživatelských účtů a rolí, včetně kontroly přístupu k jednotlivým částem a funkcím systému
- správa struktury projektů (fáze, podprojekty)
- propojení s kalendářem, plánování a kontrola termínů
- podpora textové komunikace mezi uživateli
- sklad dokumentů a jiných souborů
- znalostní báze (knowledge base)

Aplikace bude naprogramována s použitím technologie ASP.NET v jazyce C#.

Klíčová slova: webový systém, správa projektu, ASP.NET

Title: Web system for project management
Author: Michal Hampl
Department: Department of Software Engineering
Supervisor: Mgr. Martin Trčka
Supervisor’s e-mail address: martin.trcka@mff.cuni.cz

Abstract: Aim of this work is to design and implement internet application, which offers tools for project management. Application will be intended for software development projects in particular, but regarding universal design, it will be possible to use it in other branches as well.
Application will offer these features:

- user accounts and roles management, including access control for individual parts and system features
- projects structure management (phases, sub-projects)
- connected with calendar, planning and deadline supervision
- support for text communication of users
- storage of documents and other files
- knowledge base

Application will be implemented using ASP.NET technology, C# language.

Keywords: web system, project management, ASP.NET

Kapitola 1

Cíl práce

Cílem bakalářské práce je navrhnout a implementovat systém pro správu a sledování všech potřebných údajů týkajících se managementu projektu. Aplikace je primárně určená pro projekty vývoje software, ale je navrhnuta co nejobecněji, aby mohla zachytit průběh projektu i v jiných odvětvích. Ke splnění tohoto požadavku bude umožněno dělení projektů na časové úseky (fáze) a tématické části (složky).

Základním požadavkem je navrhnout aplikaci, která na jednom místě poskytne rozhraní pro všechny členy projektového týmu – **zákazníka (zadavatele), vedoucího projektu, další managery, fakturaci, pracovníky (implementátory)** a nepostradatelnou skupinu osob **testery (kontrolory).** Samozřejmostí je poskytnout určité skupině personifikované rozhraní, které bude obsahovat pouze prvky týkající se její role v projektu.

Snaha jde ale ještě o něco dále - vytvořit aplikaci, která může sloužit jako jednoduchý **firemní intranet**, který bude na jednom místě obsahovat nejen informace o jednotlivých projektech, ale i dostupné informace o know-how používaném v rámci organizace. K tomuto účelu bude sloužit strukturovaná **znalostní báze (knowledge base)** a také **sklad dokumentů a specifikací.**

Nedílnou součástí při plnění jakéhokoliv projektu je nastavení termínů a kontrola jejich splnění prostřednictvím **kalendáře,** jako i možnost **komunikace** mezi jednotlivými osobami podílejícími se na projektu.

1.1 Důležitost systému pro správu projektů v praxi

Vybrané téma bakalářské práce (tedy *Webový systém pro správu projektů*) mi připadá nejen zajímavé, ale i užitečné. Během bakalářského studia spolupracuji se společností, která se zabývá vývojem software. Ačkoliv vývojový tým čítá pouze několik členů, vidím nasazení a používání podobného systému jako klíčové. Bohužel takový systém se ve zmíněné společnosti v tuto chvíli nepoužívá. Tím jsou způsobeny problémy v koordinaci a komunikaci během realizace projektů, které bych chtěl řešit mnou vyvíjenou aplikací v rámci této práce.

Nasazení systému pro správu projektů nemusí nutně znamenat žádný drastický zásah do existujících zvyklostí a nastavených procesů ve společnosti. V každém případě ale znamená zjednodušení podpůrné agendy (podobně jako systém pro správu verzí, který ale není obsažen v tématu této práce) a tím umožní členům týmu věnovat se více vlastnímu předmětu podnikání místo vytváření různých přehledů a výstupů pro vedoucí projektu nebo zákazníka. Vzhledem k tomu, že jsou všechny relevantní informace obsaženy na jednom místě, je umožněno jejich snazší vyhledávání a zpracovávání.

Mimo zvýšení efektivity při předávání informací v rámci vývojového týmu (např. zadávání, specifikování a potvrzování úkolů případně dohledávání požadovaných

informací v rámci znalostní báze nebo skladu dokumentů) je pravděpodobné také zjednodušení a zrychlení komunikace se zákazníkem (zadavatelem) projektu. Tím, že i zákazník (zadavatel) je uživatelem vytvářeného systému, jsou mu okamžitě zpřístupněny informace, které by mu jinak bylo nutné předávat prostřednictvím e-mailové nebo jiné komunikace. Podobně zde najde i informace o aktuálním stavu prací, které by si musel jinak vyžádat od odpovědných osob. Mezi důležité vlastnosti patří také možnost zasílání a specifikace vlastních požadavků.

1.2 Analýza podobných produktů

Před a během implementace *Webového systému pro správu projektů* byly analyzovány i podobné volně dostupné aplikace. U některých aplikací je na Internetu k dispozici testovací (demonstrační) účet. U ostatních aplikací byla analyzována dostupná dokumentace nebo snímky obrazovek.

Mezi analyzovanými aplikacemi se nacházely různě komplexní aplikace od velmi jednoduchých evidenčních systémů až po velké a rozsáhlé aplikace s napojením na systémy pro realizaci technické podpory. I mezi podobně rozsáhlými aplikacemi existovala velmi vysoká míra variability a nedá se říci, že by určitý způsob implementace mohl být označen jako standardní. Systémy sice obsahují podobné možnosti, ale jejich pojetí se liší.

Je nutné také poznamenat, že zvolené téma (tedy *„Systém pro správu projektů“*) je velmi široké a aplikace je možné neustále vylepšovat a doplňovat o nové funkce a vylepšení, provázání na jiné technologie, apod.

1.2.1: 37signals, LLC – Basecamp

Webová aplikace **Basecamp** (http://www.basecamphq.com/) společnosti **37signals** (http://www.37signals.com/) nese ocenění BusinessWeek Best of the Web 2005 a dle informací na webové stránce produktu ji používá více jak 250 tisíc uživatelů. Na téže stránce je možné také najít mnoho referencí od významných osobností.

Po přihlášení do aplikace je uživateli zobrazena stránka nazvaná *Dashboard*. Na tomto místě získá přehled o všech důležitých záležitostech týkajících se jeho projektů. Důraz je kladen na zvýraznění zmeškaných událostí, uživatel také vidí výhled na události pro dalších 14 dní.

Z *Dashboard* je možné jednoduše přeskočit na stránky jednotlivých projektů, které zobrazují podobné informace týkající se jednoho projektu.

Hlavním elementem aplikace je zpráva (*Message*). Ta může být vázána buď přímo k projektu nebo ke konkrétnímu termínu (*Milestone*) v rámci projektu. Ke zprávám mohou být přiloženy soubory, zprávy je možné seskupovat do kategorií a diskutovat o nich.

Další funkce:

- *Milestones* – Evidence termínů projektů, zvýrazňování těchto termínů v kalendáři. U termínů mohou být přiloženy zprávy (*Messages*).
- *To-Dos* – Seznamy úkolů, které mohou být veřejné nebo privátní. Je možné definovat osobu odpovědnou za splnění.
- *Time-Tracking* – Jednoduchá evidence práce v rámci projektu.
- Jednoduché nastavení práv na úrovni projektů
- Export do XML, veřejné API

Zajímavou myšlenkou aplikace je zpřístupnění informací pomocí technologií RSS nebo iCalendar a tedy umožnění sledování aktivity a přistupování k datům offline z desktopových aplikací.

1.2.2: Login Software Inc. – Easy Projects .NET

Aplikace **Easy Projects .NET** (http://www.easyprojects.net/) společnosti **Logic Software Inc.** (http://www.logicsoftware.net/) oproti produktu, který byl popsán v předchozí kapitole, zapojuje do systému také zákazníky, kteří mohou vkládat své požadavky a komunikovat s ostatními členy týmu.

Práce uživatelů začíná opět na stránce nazvané *Dashboard*. Tato stránka shrnuje aktuální informace o úkolech naplánovaných na aktuální týden, zpožděných úkolech, neopravených chybách v projektu a posledních zprávách od jiných uživatelů.

Stejně jako v předchozí aplikaci jsou informace členěny do projektů, které mají lineární strukturu – nejsou členěny do složek. Aplikace umožňuje import projektů z Microsoft Project. U projektů je možné definovat vlastní pole pro informace, které nejsou standardně evidovány. Je možné přikládat soubory, přiřazovat členy týmu. Členům týmu je možné přiřazovat předdefinované role nebo vytvořit vlastní roli s podrobnou definicí přístupových práv na jednotlivé elementy aplikace (projekty, bugy, úkoly, požadavky). Zajímavou funkcí je automatické přidělování nových bugů a požadavků definovaným členům týmu.

V rámci projektů jsou definované položky (*Tasks, Requests, Bugs)*, u kterých se eviduje název, priorita, popis a je možné opět přikládat soubory. Položky je možné dělit na podpoložky do libovolné hloubky.

Další funkce:

- *E-mail Notification* – Je možné vkládat pravidla definující události, o kterých je zasílána zpráva e-mailem.
- *Time Log* – Umožňuje evidovat práci strávenou členy týmu na jednotlivých projektech. Dobu práce je možné sčítat podle různých kritérií.
- *Billing Report* – Pro jednoduchou evidenci účtovaných položek.
- *Gantt Chart* – Z vložených dat je možné generovat standardní diagram běžný pro zobrazování průběhu projektů.
- Další přehledy a diagramy

Mezi užitečné vlastnosti aplikace patří členění položek na podpoložky a také možnost definice vlastních polí pro hodnoty, které standardně evidovat nejdou.

1.2.3: Intersource Solutions Group, LLC – PRMPRO

Project Relations Manager Professional (PRM PRO, http://www.prmpro.com/) společnosti **Intersource Solutions Group, LLC** (http://www.isgroup.com/) v rámci demonstrační animace slibuje ušetření nákladů, zamezení ztráty důležitých informací týkajících se projektu a umožnění sdílení informací v rámci virtuálních vývojových týmů.

Všechny informace se opět evidují v rámci projektů. Stejně jako v předchozím případě mají do aplikace personalizovaný přístup i zákazníci.

Každý uživatel je zařazen do jedné ze čtyř rolí:

- *Administrator* – má neomezený přístup do aplikace a ke správcovským funkcím
- *Customer* – vkládá své požadavky do systému a kontroluje jejich splnění
- *Project Leader* – má na starosti vývojový tým, komunikaci se zákazníkem, rozděluje úkoly
- *Team Members* – plní přidělené úkoly

Uživatel po přihlášení vidí *Home page* s aktuálními informacemi týkajícími se jeho role v systému. U jednotlivých elementů systému je možné vkládat dokumenty a členové týmu mohou diskutovat. Při plnění úkolů je evidována doba práce, která může sloužit jako podklad pro určení částky fakturované zákazníkovi.

Jako rozšiřující moduly je možné využívat:

- *Billing* – fakturace placených požadavků zákazníkovi
- *Community* – centrální místo pro komunikaci uživatelů systému
- *Library* – centrální sklad dokumentů
- *Team* – informace o jednotlivých členech týmu včetně kontaktu

1.2.4 Shrnutí

Tři vybrané aplikace naznačují různé směry, kterými by se mohl vytvářený *Webový systém pro správu projektů* ubírat. Bakalářskou práci bude provázet snaha vybrat ty nejdůležitější vlastnosti a ty dále rozpracovat tak, aby byl umožněn provoz systému i ve velkých společnostech.

Vzhledem k tomu, že zvolené téma je velmi obsáhlé, bude dáván menší důraz na reportovací služby. Možnost získání různých přehledů je samozřejmě velmi důležitou funkcí aplikace, ale v naprosté většině netvoří vlastní vnitřní jádro aplikace. Kvalitní jádro umožňuje snadné rozšíření systému například právě o reportovací služby.

Kapitola 2

Volba způsobu realizace

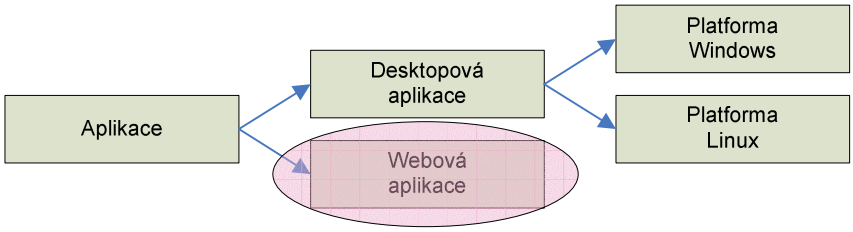
Na počátku navrhování všech aplikací je nutné učinit spoustu významných rozhodnutí, které velkou měrou ovlivní výslednou použitelnost aplikace a také její oblíbenost u uživatelů.

Při navrhování systému pro správu projektů je potřeba mít na paměti, že uživatelé budou aplikaci používat každodenně, nebude to nicméně hlavní náplň jejich práce ale pouze prostředek, který jim práci usnadní a urychlí. Tomu je potřeba podřídit veškeré uvažování a rozhodování.

Aplikace tedy musí být za každou cenu intuitivní, aby uživatel nemusel před každou akcí strávit dlouhou dobu hledáním příslušného tlačítka nebo vstupního pole. Na druhou stranu musí být komplexní, aby uživatele v jeho práci neomezovala.

2.1 Prostředí aplikace

Prvním zásadním rozhodnutím, které je potřeba učinit před zahájením dalších prací, je volba prostředí, ve kterém bude aplikace fungovat.



Obrázek 1: Volba prostředí aplikace

Jak je vidět na obrázku 1, bylo rozhodováno mezi třemi možnými variantami – desktopovou aplikací na platformě Windows nebo Linux nebo webovou aplikací.

V případě volby desktopové aplikace by bylo nutné se rozhodnout mezi konkrétní platformou. Toto by nebylo úplně přímočaré rozhodnutí, protože v cílové skupině uživatelů aplikace mohou existovat uživatelé na obou platformách. Vzhledem ke snaze o obecnější charakter aplikace by byla pravděpodobně zvolena platforma Windows. Tato platforma je na desktopových počítačích stále nejrozšířenější. Popsané dilema volby platformy desktopové aplikace by bylo možné řešit i volbou technologie umožňující vývoj multiplatformních aplikací (např. Java). Nicméně byla zvolena jiná cesta – systém pro správu projektů byl realizován jako webová aplikace.

Nevýhodou složitějších webových aplikací může být jejich těžkopádnost – ovládání nemusí být tak příjemné jako v případě desktopových aplikací. Toto se dá nicméně řešit zvýšeným úsilím při návrhu konkrétního vzhledu a funkce uživatelského rozhraní. Druhou významnou nevýhodou je nemožnost přistupovat k datům offline – tj. v době, kdy z nějakého důvodu není dostupný server poskytující aplikaci. U

desktopových aplikací je tuto situaci možné řešit uchováním lokální kopie dat u klienta. Vzhledem k tomu, že k webovým aplikacím se přistupuje pomocí libovolného internetového prohlížeče, není možné tento problém úplně odbourat. Protože se ale v tomto případě předpokládá umístění aplikačního serveru v místě uživatelů (tedy v rámci organizace užívající implementovaný systém), a také protože rozšířenost internetového přístupu je již na dobré úrovni, je možné tento problém odsunout do pozadí.

Naproti zmíněným nevýhodám mají webové aplikace spoustu výhod:

- **Nezávislost na platformě** – Jak bylo řečeno, pro přístup k aplikaci se používá libovolný internetový prohlížeč. Toto řeší výše zmíněný problém při volbě platformy desktopové aplikace.

- **Odpadá nutnost instalace specializovaného klientského programu** – Aplikace může být zpřístupněna celému Internetu. Potom je možné přistupovat k informacím z libovolného počítače bez nutnosti jeho speciální konfigurace (stačí běžný internetový prohlížeč). Je tedy možný přístup například i z internetových kaváren po celém světě.

- **Standardní prostředí** – Prostředí internetu a webových aplikací je v dnešní době velmi rozšířené. Uživatelům aplikace tedy odpadá nutnost učit se ovládat nový program. Použijí pouze své zkušenosti s ovládáním webových stránek.

2.2 Konkrétní implementační technologie

Tvorbu webových aplikací je v dnešní době možné realizovat mnoha různými prostředky. Nejrozšířenější jsou skriptovací jazyky – konkrétně PHP, ASP, Perl a další. Webové aplikace je možné vytvářet i s použitím jazyka Java (pomocí servletů nebo Java Server Pages). Mimo to je možné prostřednictvím CGI použít program napsaný téměř v jakémkoliv jazyce.

Z popsaných kategorií vybočuje technologie ASP.NET z dílny společnosti Microsoft. Jedná se o náhradu dřívější skriptovací technologie ASP, ale je postavena na úplně jiných základech. Princip fungování ASP.NET vychází z příslušnosti této technologie k Microsoft .NET Framework. Každá stránka je reprezentována třídou, která je při požadavku klienta instanciována a má na starosti vygenerování výstupu a jeho odeslání klientovi těsně před dokončením jejího zpracování.

Stránka se všemi jejími elementy je reprezentována objektově, což umožňuje kdykoliv během zpracování měnit vlastnosti těchto elementů nebo dokonce dynamicky některé elementy vytvářet nebo rušit. Mimo to je možné úplné oddělení aplikačního kódu od definice vzhledu stránky, což umožňuje snazší orientaci a správu kódu. Pro lepší výkonnost aplikace je možné její předkompilování do mezikódu.

V rámci ASP.NET je možné použít celou řadu programovacích jazyků. Mezi nejrozšířenější patří C#.NET a VB.NET. Protože všechny jazyky pracují nad stejnou technologií, mají tím pádem i hodně podobnou sílu, takže volba konkrétního jazyka je spíš jen otázkou preferencí vývojáře.

Tato aplikace je realizována technologií ASP.NET v jazyce C#.

2.3 Kde uchovávat data

Obdobně jako v předchozím případě, i pro uchovávání dat existuje mnoho rozdílných produktů mnoha různých společností. V zásadě můžeme tyto produkty rozdělit do dvou hlavních skupin – souborové databáze a databázové servery.

- **Souborové databáze** jsou postaveny na *přímém přístupu aplikace k jednomu nebo více souborům* uchovávajícím data a další podpůrné informace. Z důvodu snazšího a univerzálnějšího vývoje je možné použít některé standardní databázové technologie (např. ADO, ODBC a další), které ale prostřednictvím specializovaných ovladačů opět přistupují přímo k souborům.

  Tato databázová technologie je vhodná zvláště při přístupu k datům pouze jedním uživatelem (nebo programem) v jednom okamžiku. Jak se ukázalo v praxi, technologie naopak není vhodná pro víceuživatelský přístup, kdy se může stát takzvaným úzkým místem aplikace.

  Z tohoto důvodu není tato skupina technologií vhodná pro webové aplikace, kdy je většinou žádoucí, aby byla data sdílena v jednom okamžiku více uživateli.

- **Databázové servery** fungují na principu služby, která je servery poskytována jednotlivým uživatelským programům (klientům). V tomto případě *má nad datovými soubory kontrolu pouze databázový server* a klienti přistupují k datům jeho prostřednictvím. Server má tak přehled o všech požadavcích a implementovanými mechanismy zabraňuje konfliktům, které by mohly vést k poškození dat. V tomto případě je tedy možné bez problémů obsluhovat několik souběžně pracujících uživatelů.

  K této základní funkcionalitě jsou většinou přidány další doplňkové služby jako možnost transakčního zpracování, replikace dat nebo automatické zálohování.

  Zejména z důvodu bezproblémového víceuživatelského přístupu, který je žádoucí pro implementovanou aplikaci, je tato skupina produktů vyhovující.

V současnosti je na trhu spousta různých databázových serverů. Svůj produkt nabízí velké společnosti jako Oracle, IBM nebo Microsoft, ale k dispozici je i celá řada dalších produktů, které jsou určeny pro volné použití – MySQL, PostgreSQL, Firebird a další. Výrobce každého významnějšího produktu se snaží o jeho podporu v různých vývojářských technologiích (včetně .NET Framework a ASP.NET). Tato

podpora je na různých úrovních – nativní, prostřednictvím OLE DB nebo ODBC, apod.

Pro použití v rámci bakalářské práce byl nakonec vybrán databázový produkt Microsoft SQL Server 2000. Vzhledem k tomu, že tento produkt je vytvořen stejnou společností jako .NET Framework, je očekávána jeho bezproblémová podpora v rámci .NET Framework a ASP.NET.

Existuje také odlehčená verze Microsoft SQL Server (edice Microsoft SQL Server 2000 Desktop Edition známá také jako MSDE 2000), která je k dispozici zdarma. Tato edice je omezena v některých výkonnostních parametrech, ale je jinak plně kompatibilní se serverovými edicemi Microsoft SQL Serveru a je tedy možné ji nasadit v případě, že je očekáváno nižší využití aplikace. V případě následného rozšíření počtu uživatelů a zvýšení požadavků na výkon databázového serveru je možné přejít na serverovou edici Microsoft SQL Serveru při zachování uložených dat.

Kapitola 3

Návrh struktury aplikace

Před začátkem implementačních prací je nutné nejdříve rozvrhnout funkcionalitu aplikace. Této části vývoje software je nutné se věnovat dost podrobně. Veškeré nejasnosti a málo detailní popis funkcí s největší pravděpodobností povede k následnému předělávání menší či větší části aplikace.

3.1 Evidované informace

3.1.1 Definice pojmů

Pro pochopení vztahů v aplikaci popisovaných v dalších částech je nutné definovat použité pojmy. Každý tento pojem reprezentuje skutečnost v reálném prostředí.

- **Projekt** je základní informační element v aplikaci, ke kterému se vážou další informace. Reprezentuje oddělenou množinu informací týkající se jednoho obchodního případu nebo jedné logicky uzavřené akce.

- **Fáze projektu** slouží ke strukturování projektu podle jeho časového průběhu, neboli pro oddělení částí projektu, které jsou časově omezeny – například fáze přípravy, fáze návrhu, fáze implementace a fáze testování. Fáze nemohou být členěny na podfáze.

  Fáze neslouží k dělení projektu na tématické části, které se mohou vyskytovat v libovolném časovém okamžiku projektu – například uživatelské rozhraní, tiskové sestavy, apod. – obojím se bude tým pravděpodobně zabývat jak v rámci návrhu, tak v rámci implementace

- **Podsložky** oproti fázím slouží k dělení projektu na tématické části – uživatelské rozhraní, tiskové sestavy, apod. Podsložky mohou tvořit hierarchickou strukturu.

- **Požadavek** je zadání týkající se projektu, které pochází buď od zákazníka nebo manažera projektu. Většinou neobsahuje technické informace, pouze je podnětem pro bližší analýzu a další zpracování.

- **Úkol** je zadání týkající se projektu, které je adresováno jednotlivým pracovníkům. Vzniká na základě požadavků a často obsahuje již bližší technické informace.

3.1.2 Projekty a fáze projektů

Navrhovaná aplikace slouží pro evidenci informací o projektech v rámci jedné organizace. Z tohoto tedy vyplývá hlavní součást aplikace – **evidence projektů.**

V různých organizacích jsou u projektů evidovány různé informace. Pro zachování volnosti při vkládání informací bylo rozhodnuto, že by mohlo být užitečné zahrnout pole neomezené délky pro vložení textu libovolného obsahu a struktury. Aby bylo možné oddělit veřejné a interní informace, bylo přidáno ještě jedno podobné pole. Tato koncepce dvojice neomezeně dlouhých textových polí (veřejný a interní popis) byla použita i u dalších částí aplikace.

Ke každému projektu jsou přiřazeny následující údaje:

- Hlavní položky
  - Název projektu
  - Uživatel, který projekt založil
  - Datum a čas založení projektu
- Další položky
  - Vedoucí projektu – pouze informativní pole
  - Popis projektu (veřejný a interní)

Přidání dalších polí do evidence projektů je otázkou budoucího rozšiřování aplikace na základě zpětných informací od uživatelů.

Každý netriviální projekt je členěn na několik fází, v případě projektu vývoje software jsou to například analýza, implementace a testování. Toto členění je vhodné zahrnout i do aplikace. Pro každý projekt je tedy možné definovat libovolný počet **fází projektu.** Každá tato fáze má stanoveny:

- Hlavní položky
  - Název fáze
  - Projekt, ke kterému se fáze váže
- Další položky
  - Předpokládané datum započetí a dokončení, datum skutečného dokončení
  - Seznam zúčastněných pracovníků – z tohoto seznamu je možné vybírat pracovníky při vytváření úkolu
  - Popis projektu (veřejný a interní)

3.1.3 Požadavky

V rámci každého projektu jsou vždy definovány nějaké **požadavky** vycházející ze specifikace projektu. Během realizace projektu může další požadavky specifikovat také zákazník. Tyto dodatečné požadavky mohou být i zpoplatněny a tudíž je vhodné evidovat tuto informaci v rámci aplikace. Cena za požadavek může být určena třemi způsoby:

- Požadavek není zpoplatněn
- Požadavek je zpoplatněn dohodnutou fixní částkou
- Požadavek je zpoplatněn na základě skutečného množství odpracovaných hodin podle dohodnuté hodinové sazby

Požadavky by mohlo být vhodné členit podle nějakého typu a určovat jejich prioritu. Z důvodu univerzálnosti aplikace není předem možné stanovit, jaké typy a případně priority bude potřeba přiřadit jednotlivým úkolům. Proto je vhodné umožnit editaci seznamů typů a priorit. Seznamy je možné evidovat na různých úrovních (celá aplikace, projekt, fáze projektu, složka). Pro dosažení dostatečné univerzálnosti a uživatelské přívětivosti jsou typy a priority nakonec navázány na fáze projektu.

Informace evidované u požadavků tedy jsou:

- Hlavní položky
  - Název požadavku
  - Uživatel, který založil požadavek
  - Datum a čas založení požadavku
  - Fáze projektu, ke které požadavek přísluší
  - Stav zpracování požadavku – vychází z procesu popsaného v kapitole 3.3
- Další položky
  - Typ a priorita požadavku na základě výběru ze seznamu
  - Předpokládaná délka práce na požadavku a předpokládané datum dokončení
  - Informace pro určení ceny požadavku
  - Popis požadavku (veřejný a interní)

3.1.4 Úkoly

Na realizaci jednotlivých požadavků se podílí jeden nebo více zaměstnanců. Těmto jsou přiděleny různé **úkoly**, které je rovněž nutné evidovat v aplikaci. Každý úkol je opět přiřazen k nějaké fázi (pokud je úkol vytvořen v kontextu požadavku, je stejná jako fáze požadavku) a podle ní je možné také zvolit typ a prioritu. Z důvodu výpočtu ceny požadavku na základě skutečně odpracované doby je u každého úkolu nutné tuto dobu práce evidovat.

Pro každý úkol by měla existovat možnost otestování jeho správného splnění. Tento test úkolu bude reprezentován rovněž jako úkol. Nebude tak nutné zbytečně duplikovat podobnou funkcionalitu standardního úkolu pro testy úkolu. Pro tento typ bude tedy nutné uvést odkaz na příslušný testovaný úkol.

Informace evidované v rámci úkolu jsou:

- Hlavní položky
  - Název úkolu
  - Uživatel, který založil úkol
  - Datum a čas založení úkolu
  - Požadavek, ke kterému úkol přísluší (pouze při vytvoření úkolu v rámci požadavku; přiřazení je dáno zařazením úkolu v hierarchii)
  - Fáze, ke které úkol přísluší
  - Pracovník, kterému úkol přísluší
  - Nadřazený úkol (v případě, že se jedná o test úkolu)

    - Stav zpracování úkolu – vychází z procesu popsaného v kapitole 3.3
- Další položky
    - Typ a priorita úkolu na základě výběru ze seznamu
    - Předpokládaná délka práce na úkolu a předpokládané datum dokončení
    - Aktuální celková doba strávená prací na úkolu
    - Popis úkolu (veřejný a interní)

3.1.5 Znalostní báze

Znalostní báze slouží jako centrum znalostí a postupů. Základem jsou tedy textové informace.

Různé aplikace volí různý způsob implementace znalostní báze. U některých aplikací jsou k dispozici předem definovaná pole pro zadání různých informací (například pole *Příznaky*, *Příčina*, *Řešení*, *Další informace* pro znalostní bázi evidující řešené problémy). Vzhledem k požadované univerzálnosti implementované aplikace není pevné členění vhodné. Místo toho aplikace nabízí dvojici polí (veřejné a interní) pro zadávání libovolně dlouhých textů.

3.1.6 Komentáře a datový sklad

Aplikace umožňuje komunikaci jednotlivých členů týmů. Komunikační systém může být implementován nezávisle na struktuře, jako lepší řešení se však jeví umožnit komunikaci v rámci již zmíněných částí aplikace (projekty, požadavky, úkoly a znalostní báze). Takto bude možné diskutovat na místě, které je diskuzi nejblíže.

Obdobný přístup byl zvolen pro funkci datového skladu (místě pro uložení specifikací a dokumentů). Možnost úplně odděleného skladu se nejeví jako uživatelsky schůdná. Při implementaci v rámci existujících částí bude požadovaný soubor snáze dostupný a nebude nutné přecházet na jinou stránku.

3.1.7 Kalendář

Kalendář má sloužit hlavně ke kontrole a evidenci termínů v rámci požadavků a úkolů. Neméně důležitá je ale také možnost vkládání vlastních záznamů týkajících se například termínů různých jednání. Pro uživatelský komfort je nutné, aby sledování termínů bylo jednoduché, tedy nejlépe, aby byly termíny evidované u požadavků a úkolů zobrazovány v kalendáři automaticky.

Aplikace umožňuje také evidenci vícedenních událostí (například pro evidenci dovolených, vícedenních seminářů, …). Informace uložené u těchto položek tedy jsou:

- Identifikace uživatele vlastnícího záznam kalendáře
- Datum a čas započetí a ukončení záznamu kalendáře
- Název a popis záznamu kalendáře

3.2 Pevná nebo pružná hierarchie informací

Další důležitou věcí, kterou bylo nutné rozhodnout před započetím návrhu datového modelu je, zda hloubka hierarchie sledovaných informací bude pevná nebo pružná.

Pevná hierarchie informací

- Jednotlivé úrovně:
  - Seznam projektů
  - Seznam požadavků
  - Seznam úkolů
- Výhodou tohoto způsobu ukládání informací je jednoduchost implementace. Nicméně žádná variabilita znamená velké omezení požadované univerzálnosti aplikace. Vzhledem k tomu, že všechny evidované projekty jsou na jedné úrovni, znamenalo by to v praxi i omezení počtu projektů a nemožnost členění dat podle příslušnosti k oddělením velké organizace.

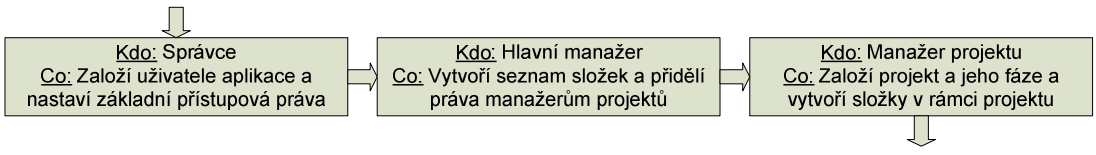
Pružná hierarchie informací

- Pravidla:
  - Projekty je možné zařazovat do složek
  - Projekty obsahují požadavky, které je možné také zařazovat do složek
  - Úkoly jsou vytvářeny buď v rámci požadavků nebo mimo ně (přímo v rámci projektů)
  - Úkoly mohou být dále členěny na podúkoly
- S pomocí složek bude možné tříúrovňovou hierarchii upravit podle speciálních požadavků různých organizací. Pomocí složek projektů bude možné seskupovat projekty podle příslušnosti k oddělením větších organizací. Pomocí složek požadavků bude možné dělit projekt na různé tématické části.

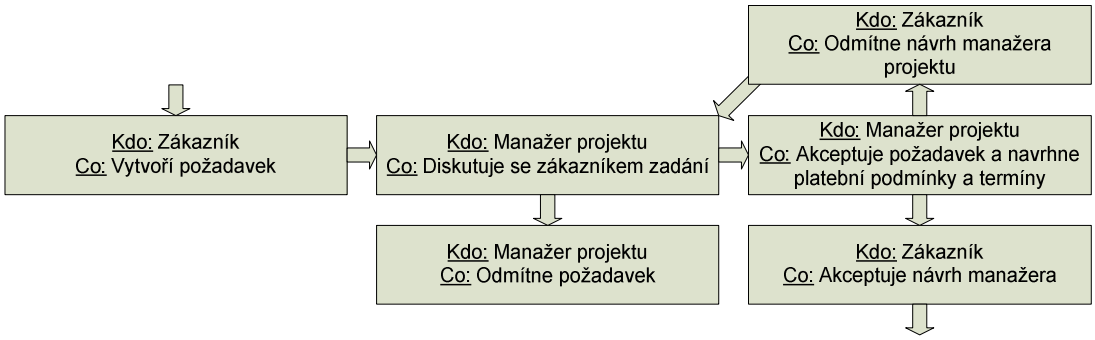
Pro splnění požadavků kladených na aplikaci byla zvolena implementace pružné hierarchie informací.

3.3 Procesy v rámci aplikace

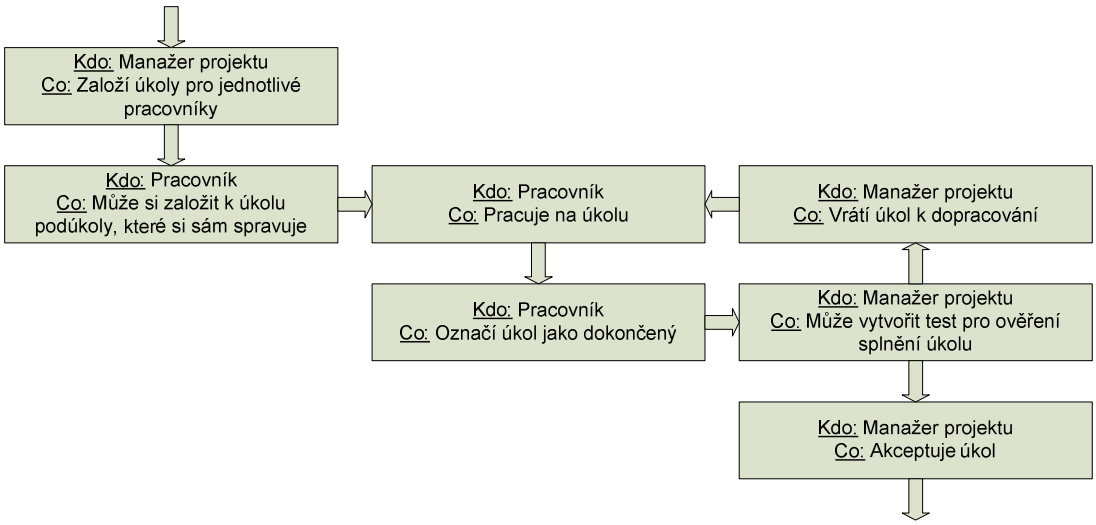
Následující schémata obsahují životní proces aplikace od nasazení až po dokončení projektu.



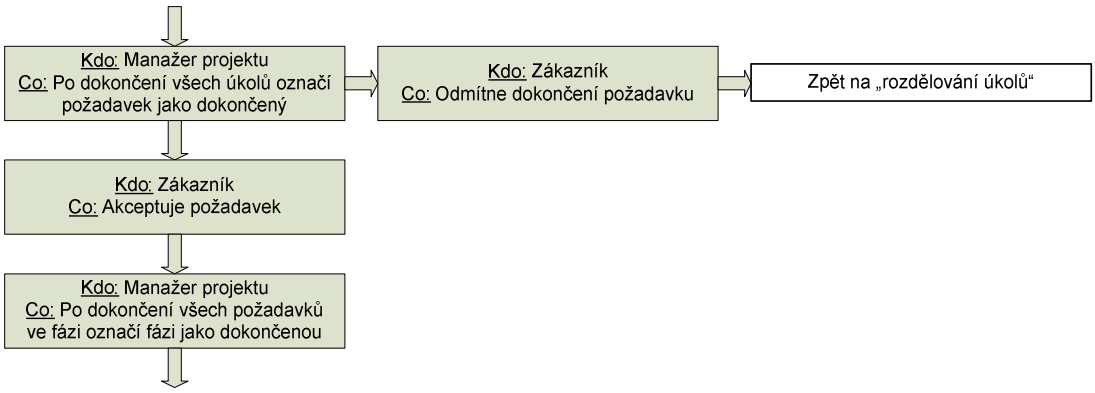
Obrázek 2a: Procesy v rámci aplikace – příprava



Obrázek 2b: Procesy v rámci aplikace – schvalování požadavků



Obrázek 2c: Procesy v rámci aplikace – rozdělování úkolů



Obrázek 2d: Procesy v rámci aplikace – kontrola zákazníkem a dokončení projektu

3.4 Systém uživatelských práv

Vzhledem k tomu, že uživatelé, kteří budou pracovat s aplikací, budou mít různé role, je možnost definice uživatelských práv v podobných aplikacích nutností. Ne každý uživatel má možnost přistupovat ke všem evidovaným údajům nebo provádět s těmito údaji všechny úkony.

Existuje spousta různých způsobů implementace přístupových práv. Vzhledem k volbě implementace pružné hierarchie ukládání dat se datový model *Webového systému pro správu projektů* v určitých ohledech podobá souborovým systémům. Z toho důvodu je možné pro výběr způsobu implementace uživatelských práv vzít inspiraci právě v implementaci uživatelských práv v souborových systémech – konkrétně model ACL.

Na každém uzlu (složka projektů, projekt a složka požadavků) v hierarchii je tedy možné definovat seznam explicitních práv pro některé uživatele. Aby bylo možné jednoduše nastavit práva uživateli pro rozsáhlejší část hierarchie, dědí se tato práva i do poduzlů. Uživatel má tedy na uzel určité právo v případě, že má toto právo na daném uzlu explicitně přiřazeno nebo jej má přiřazeno na některém uzlu v hierarchii směrem ke kořeni.

Konkrétní typy uživatelských práv reprezentují uživatelovu roli na projektu:

- manažer – spravuje projekt, požadavky zákazníků a úkoly
- pracovník – má možnost pracovat s úkoly, které jsou mu přiděleny
- fakturace – eviduje fakturaci požadavků, na které má právo
- zákazník – sleduje stav projektu a zakládá požadavky

Systém nabízí také dvě doplňková práva (role):

- administrace složek – uživatel má možnost v podhierarchii spravovat složky
- přidělovat práva – uživatel má možnost přidělovat práva v podhierarchii jiným uživatelům

Kapitola 4

Datový model

Datový model každé aplikace musí být navrhnut velmi přesně s ohledem na detailní zadání tak, aby při vývoji aplikace neomezoval, ale byl dost efektivní a optimalizovatelný. Důležité je také pamatovat na možnost budoucího rozšíření aplikace.

Z těchto důvodů je nutné ujasnit si veškerou funkcionalitu, která bude po aplikaci požadována ještě před započetím návrhu jednotlivých tabulek.

4.1 Implementace hiearchické struktury

V kapitole 3.2 byla pro uložení dat vybrána pružná hierarchie. Z definice na stejném místě vyplývá, že tato struktura je definována rekurzivně. Implementace rekurzivních struktur není v SQL obecně jednoduchá. Nativní podpora se objevuje až v posledních verzích databázových produktů. Databázový server Microsoft SQL Server 2000 bohužel tuto podporu zabudovanou zatím nemá. Proto je nutné implementovat rekurzi svépomocí.

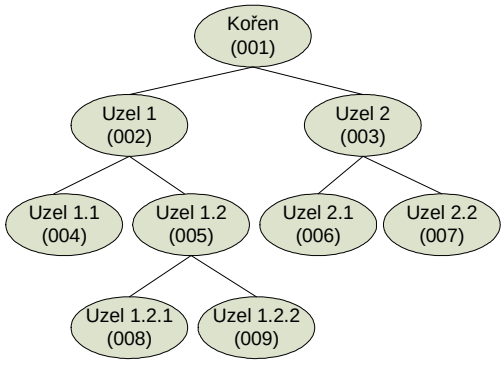
Jedním z běžných způsobů implementace pomocí standardních prostředků je využití dvojice tabulek:

- **hlavní tabulka (Nodes)** – obsahuje vlastní rekurzivní data včetně primárního klíče a odkazu na nadřazený uzel.
- **pomocná tabulka (NodeTree)** – obsahuje linearizaci stromu definovaného v hlavní tabulce. Odkazy na jednotlivé uzly jsou seřazeny podle infixového průchodu stromem. Navíc, pro jednoduché získání celé hierarchie poduzlů, je přidán sloupec obsahující zřetězení názvů uzlů v pořadí průchodu od kořene k vybranému uzlu. Pomocná tabulka umožňuje efektivnější vyhodnocování některých typů dotazů (například získání všech uzlů nebo části stromu v infixovém pořadí nebo porovnání, zda je jeden uzel předkem nebo potomkem jiného uzlu v libovolné hloubce)

Příklad jednoduchého stromu a jeho linearizace v pomocné tabulce.

Hlavní tabulka:

ID uzlu	Nadřazený uzel	Název uzlu
001		Kořen
002	001	Uzel 1
003	001	Uzel 2
004	002	Uzel 1.1
005	002	Uzel 1.2
006	003	Uzel 2.1
007	003	Uzel 2.2
008	005	Uzel 1.2.1
009	005	Uzel 1.2.2



Obrázek 3: Příklad stromu

Pomocná tabulka:

Pořadí	ID uzlu	Cesta k uzlu
1	001	Kořen
2	002	Kořen / Uzel 1
3	004	Kořen / Uzel 1 / Uzel 1.1
4	005	Kořen / Uzel 1 / Uzel 1.2
5	008	Kořen / Uzel 1 / Uzel 1.1 / Uzel 1.2.1
6	009	Kořen / Uzel 1 / Uzel 1.1 / Uzel 1.2.2
7	003	Kořen / Uzel 2
8	006	Kořen / Uzel 2 / Uzel 2.1
9	007	Kořen / Uzel 2 / Uzel 2.2

Důležitou částí implementace je zajištění správného provázání hlavní a pomocné tabulky. K tomuto účelu slouží trigger spouštěný při jakékoliv změně hlavní tabulky. Tento trigger spouští uloženou proceduru (sp_BuildNodeTree). Ta pomocí zásobníku ve formě dočasné tabulky prochází infixově strom a generuje data v pomocné tabulce.

Toto je pravděpodobně úzké místo aplikace, které by stálo za zvážení optimalizovat. Proceduru generující aktuální data do pomocné tabulky není nutné provádět při každé změně v hlavní tabulce (aktualizaci pomocné tabulky je možné provádět dávkově). Také není nutné generovat pokaždé pomocná data pro celý strom, pravděpodobně by stačilo generovat data jen pro určitý podstrom, který je ovlivněn provedenou změnou v hlavní tabulce.

4.2 Různé typy uzlů

Dalším problémem, který přináší zvolený způsob uložení dat, je nutnost vypořádat se s faktem, že uzly stromu jsou různého typu – složka, projekt, požadavek, úkol a zápis ve znalostní bázi.

Pro uložení v SQL databázi je možné použít dvě základní metody:

- Sjednocení množin polí všech typů uzlů umístit do jedné velké tabulky
- V hlavní tabulce umístit jen základní pole a typ uzlu a další informace uložit do jednotlivých tabulek příslušejících jednotlivým typům uzlů

Jako vhodný způsob uložení byla vybrána druhá metoda a to hlavně ze dvou důvodů:

- Vzhledem k tomu, že pole evidované u jednotlivých typů uzlů se hodně liší, bylo by nepraktické a nepřehledné umístit všechny rozmanité pole v jedné tabulce

- Z principu fungování programů pracujících se záznamy dat je vhodné dosáhnout co největšího blokovacího faktoru (co největšího počtu záznamů v jednom bloku dat). Při jednom načtení bloku dat do paměti je tak načteno více záznamů a je tak možné rychleji provést operaci nad větší množinou

záznamů ve stejném čase. Umístění všech různých polí do jedné tabulky – tedy i jednoho záznamu – není z tohoto hlediska tedy výhodné.

4.3 Základní struktura databáze

4.3.1 Identifikace záznamů

Velmi důležitou úlohu u relačních databází má systém klíčů (hlavně primární a cizí klíče). Hodnoty těchto klíčů je možné určovat několika způsoby:

- Jako pole tvořící klíč se použije některé unikátní **významové pole identifikující daný objekt** (často například rodné číslo nebo IČ). I když se zvolené pole může jevit jako vhodné, může se stát, že unikátnost hodnoty je narušena – například jedno rodné číslo dostane přiděleno více osob. U některých významových polí může dojít ke změně identifikační hodnoty a tím nastane nutnost změnit hodnoty klíčů i u podřízených tabulek, což s sebou nese nezanedbatelnou režii. Od použití významových polí jako hodnot pro klíče se v databázových aplikacích z těchto důvodů upouští.

- Často se jako hodnota klíče používá **posloupnost celých čísel**. Většina databázových produktů nabízí snadný způsob, jak generování hodnot této posloupnosti docílit. V Microsoft SQL Serveru 2000 je u celočíselných polí k dispozici vlastnost *IDENTITY*, která způsobí automatické přidělování čísel aritmetické posloupnosti daných parametrů.

- Jiným způsobem pro přidělování hodnot databázových klíčů je použití generátorů hodnot vracejících unikátní hodnotu. Microsoft SQL Server 2000 nabízí tuto vlastnost ve formě pole typu *uniqueidentifier* spolu s funkcí *NEWID* vracející hodnotu *GUID* tohoto typu, u které je podle dokumentace zaručena celosvětová unikátnost. Tento způsob určení hodnoty klíčů oproti použití posloupnosti celých čísel zjednodušuje případnou replikaci dat mezi několika databázemi. Vzhledem k tomu, že u hodnot *GUID* je zaručena celosvětová unikátnost, nevznikne tak konflikt mezi identifikátory ve dvou databázích a tyto databáze je možné jednoduše spojit do jedné.
  Tato varianta je použita i ve *Webovém systému pro správu projektů*.

4.3.2 Názvy polí

Při návrhu databáze je také vhodné zvolit si nějakou koncepci pojmenování polí.

Vzhledem k tomu, že jednotlivé tabulky se v dotazech do databáze často spojují, vznikají tak konflikty ve jménech polí různých tabulek. Tyto pole je tak nutné prefixovat názvem tabulky následovaným tečkou. Délka složitějších dotazů tak může významně narůst a dotaz se stane hůře čitelným. Pro odbourání této nepříjemnosti je vhodné jako předponu názvu každého pole uvést prefix, který charakterizuje příslušnost pole k určité tabulce. V rámci aplikace byly použity čtyř-písmenné předpony vycházející z názvu tabulky.

Užitečné je také zvýraznění polí, které slouží jako primární nebo cizí klíč. V předchozí kapitole bylo rozhodnuto o použití klíčů typu *uniqueidentifier*. Hodnoty tohoto typu jsou označovány také zkratkou *GUID*. Tohoto bylo využito při pojmenování polí tvořících klíče – za čtyř-písmennou předponou určující příslušnost pole k tabulce je uvedeno označení *GUID*.

4.3.3 Základní schéma databáze

Na obrázku 4 je zobrazeno základní schéma databáze. Podrobné schéma je spolu s popisem jednotlivých tabulek obsaženo v *programátorské dokumentaci*.

Jak je možné vidět, nejvýznamnější je tabulka *Nodes*. Obsahuje seznam jednotlivých uzlů. S ohledem na rozhodnutí učiněná v kapitolách 4.1 a 4.2 tato tabulka obsahuje pouze základní informace o jednotlivých uzlech spolu s odkazem na rodičovský uzel. Linearizace stromu uzlů je uložena v tabulce *NodeTree*.

Na tabulku *Nodes* se váže čtveřice tabulek s detailními informacemi pro jednotlivé typy uzlů – jedná se o tabulky *KnowledgeBase, Projects, Requests* a *Tasks*. Primární klíč těchto tabulek je ekvivalentní primárnímu klíči v hlavní tabulce *Nodes*. Tabulka *Nodes* obsahuje také informaci o typu uzlu, podle které se hledají detailní informace v jedné ze čtyř zmíněných podřízených tabulek.

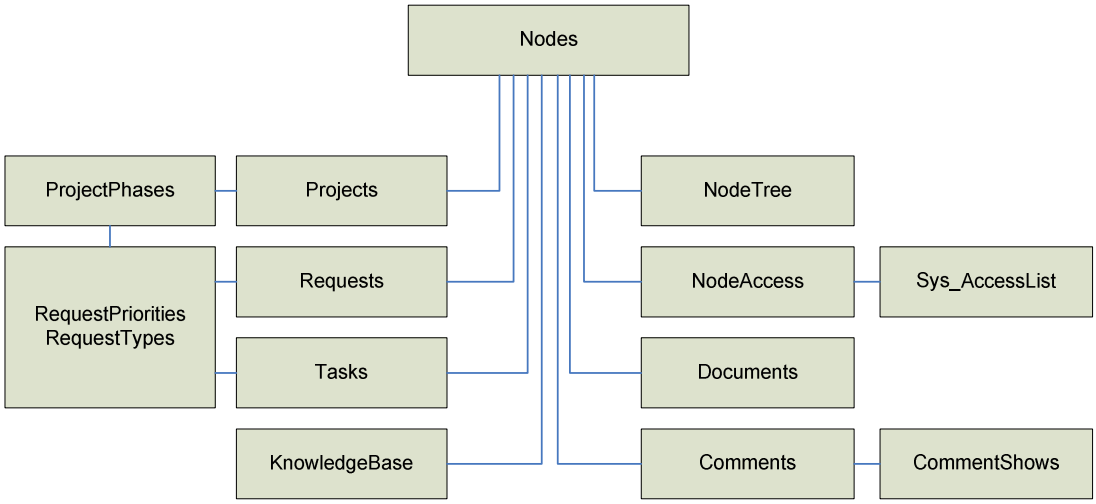
Dalšími tabulkami, které se vážou na centrální tabulku *Nodes* jsou:

- *NodeAccess* – obsahuje záznamy o přidělených právech uživatelů k danému uzlu; tabulka *Sys_AccessList* obsahuje definici jednotlivých typů práv
- *Documents* – obsahuje seznam dokumentů vázaných k danému uzlu (viz kapitola 3.1.6)
- *Comments* – obsahuje diskuzi vztahující se k danému uzlu; tabulka *CommentShows* obsahuje příznak, zda daný uživatel komentář již viděl

Velmi významnou je i tabulka *Users* obsahující informace o uživatelích, která je vázána na spoustu tabulek. Pro přehlednost proto není zahrnuta do schématu.

Na tabulku *Projects* je dále navázána tabulka se seznam fází daného projektu (*ProjectPhases)* a na tu dále tabulka se seznamem členů dané fáze projektu (*ProjectPhaseMembers)*, seznam priorit (*RequestPriorities)* a seznam typů požadavků a úkolů (*RequestTypes)*. Pro poslední 2 tabulky existují 2 pomocné tabulky (*RequestPrioritiesDef* a *RequestTypesDef)*, které obsahují výchozí seznam priorit a typů, který je zkopírován vždy při vytvoření nové fáze projektu.

K tabulce úkolů (*Tasks)* se vztahuje tabulka *TasksTrack*, ve které je evidován seznam úkolů, na kterých se právě pracuje. Do této tabulky je při započetí práce uloženo datum a čas, pomocí kterého je vypočítána celková doba práce.



Obrázek 4: Základní schéma databáze

4.4 Implementace kalendáře

Jak bylo popsáno v kapitole 3.1.7, kalendář by měl umožňovat zároveň evidenci vlastních termínů i zobrazení automaticky generovaných termínů o požadavcích a úkolech. Vzhledem k tomu, že se jedná o údaje, které jsou v databázi uloženy na různých místech, bylo nutné vymyslet řešení, jak tato data co nejjednodušeji získat.

Každý oddělený dotaz do databáze s sebou nese určitou režii, je tedy snahou spojovat více dílčích dotazů do jednoho komplexního. Z toho důvodu byl dotaz implementovaný jako *pohled* (konkrétně pojmenovaný *v_CalendarItems*), který provádí pomocí operátoru *UNION* spojení těchto dílčích výsledků:

- Uživatelem vložené termíny
- Termíny dokončení požadavků
- Termíny dokončení úkolů

Termíny dokončení jsou zahrnuty v celkovém výsledku pro každého uživatele majícího právo na daný uzel. Z konečného výsledku pohledu je tedy možné filtrovat jak termíny pro určitý den, tak termíny pro určité uživatele.

4.5 Způsob přístupu k databázi z aplikace

Přistupovat k datům uloženým na databázových serverech z aplikace je možné několika způsoby. Technologie .NET Framework obsahuje implementované rozhraní pro přístup s použitím rozhraní OLE DB. Mimo to obsahuje nativní podporu pro Microsoft SQL Server, která je optimalizovaná právě pro přístup k tomuto databázovému serveru.

Manipulace s daty na serveru je možné dělat buď přímo – aplikace přímo sestavuje konkrétní SQL dotazy, které posílá na server. Druhou metodou je využití určitého

způsobu abstrakce pomocí uložených procedur SQL serveru. Aplikace tak nemusí mít úplně přesnou představu o jednotlivých databázových polích a tabulkách, ale stačí definovat rozhraní uložených procedur, které provedou získání výsledků a také jejich vrácení v očekávané podobě.

Toto řešení má 2 výhody:

- Umožňuje provádět určité dílčí změny ve struktuře databáze (třeba z důvodu optimalizace nejčastějších databázových operací) bez nutnosti úpravy kódu aplikace. Nutné změny v SQL dotazech jsou provedeny pouze v uložených procedurách a aplikace dostane stejně formátovaná data jako před změnou.

- Umožňuje případnou změnu databázového serveru na jiný produkt. Uložené procedury patří ke standardní součásti databázových serverů. Stačí tedy, aby tyto procedury měly stejné rozhraní a stejné výstupy jako procedury na původně použitém Microsoft SQL Serveru 2000. Konkrétní problémy vzniklé neúplnou kompatibilitou jazyka SQL bude možné řešit pouze na jednom místě (v uložených procedurách) a nebude tak nutné procházet celou aplikaci a vyhledávat nefunkční SQL dotazy.

Kapitola 5

Struktura webové aplikace

Tato kapitola popisuje aspekty návrhu prezentační části aplikace *Webový systém pro správu projektů*.

Jak již bylo zmíněno dříve, aplikace je realizována s použitím technologie ASP.NET. Výhodou této technologie je možnost oddělení vzhledu jednotlivých stránek od aplikačního kódu. Toto je v terminologii ASP.NET nazýváno *Code behind*. Každá stránka je ve skutečnosti zastoupena třemi soubory:

- **.aspx** soubor – jedná se o soubor s rozšířeným HTML kódem. V tomto kódu mohou být speciální direktivy (např. `<%@ Page … %>`, `<%@ Register …%>`) a speciální elementy instancující webové ovládací prvky ASP.NET. Tyto prvky je možné poznat podle atributu `runat="server"`. Takto definované prvky jsou k dispozici za běhu aplikačního kódu a je možné upravovat jejich vzhled a funkci.

- **.aspx.cs** soubor – obsahuje aplikační kód. V tomto souboru je definována třída odvozená standardně od `System.Web.UI.Page`. Při obdržení požadavku na stránku zastoupenou touto třídou je vytvořena instance třídy a pomocí různých událostí je možné reagovat na různé situace, které při zpracování požadavku vzniknou.

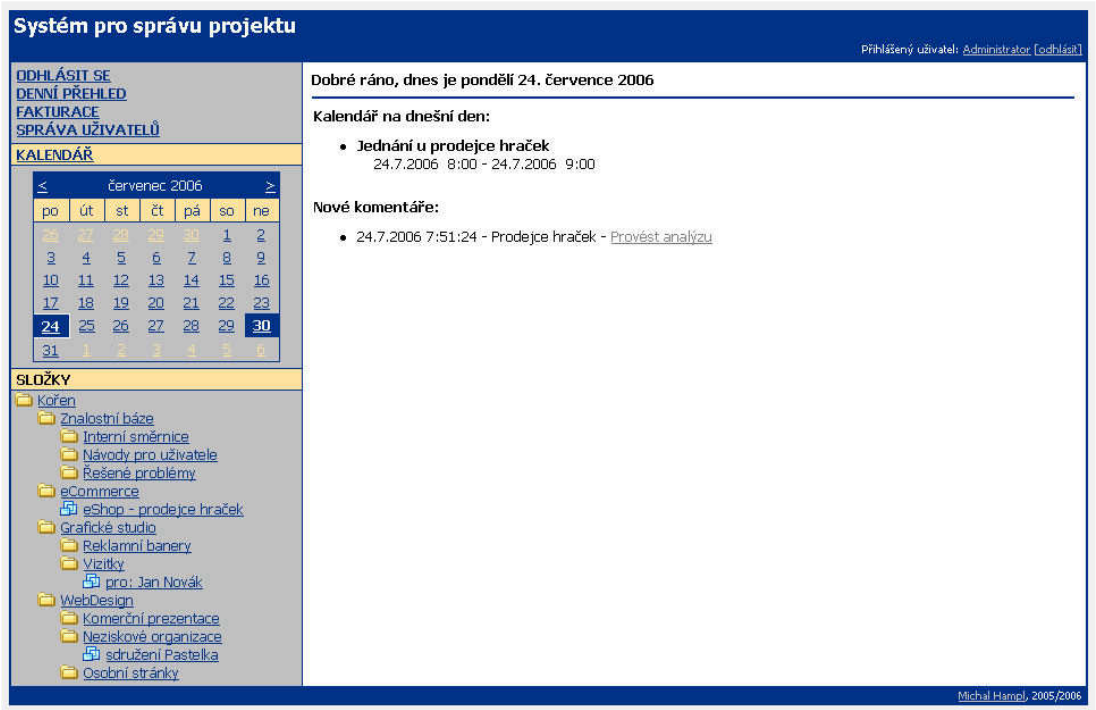
- **.aspx.resx** soubor – umožňuje definovat zdroje, které jsou k dispozici za běhu stránky

Aplikační kód a zdroje jsou před umístěním na server zkompilovány jako mezikód do DLL souboru (*Assembly*), který se umístí do adresáře *BIN*. Z původních souborů je na server nutné zkopírovat pouze soubor s HTML kódem. Aplikační kód je tak chráněn proti zneužití.

Před prvním přístupem do aplikace je prováděna kompilace z mezikódu do konečného kódu pro danou platformu. To způsobuje o něco delší zdržení prvního načítání aplikace. Pokud se s aplikací delší dobu nepracuje, může .NET Framework provést uvolnění aplikace z paměti, a proto i první načítání po delší době nečinnosti zabere o něco delší dobu.

5.1 Standardní vzhled stránek

Od všech aplikací je vyžadován jejich konzistentní vzhled a konzistentní uživatelské rozhraní. U webových aplikací to znamená univerzální hlavičku a patičku většiny stránek, ze kterých se aplikace skládá. Pro aplikaci *Webový systém pro správu projektů* tomu není jinak a je očekáváno, že všechny stránky po přihlášení uživatele do systému budou mít stejný vzhled a chování (viz obrázek). Z tohoto důvodu není možné mít v každé stránce vložený kód hlavičky a patičky - každou změnu v tomto kódu by bylo nutné replikovat do všech stránek.



Obrázek 5: Standardní vzhled stránek

Řešením tohoto problému jsou uživatelské prvky ASP.NET. Do těchto speciálních prvků je možné vyčlenit opakující se vzhled a funkčnost a na jednotlivá místa je potom vložit standardním způsobem. V aplikaci *Webový systém pro správu projektů* jsou uživatelské prvky použity na několika místech – je pomocí nich definována hlavička a patička běžné stránky, levé navigační menu a výpis uzlů daného typu včetně společných funkcí.

5.2 Pomocné třídy

Obdobně jako vyčlenění společného vzhledu je rozumné také vyčlenit opakující se části kódu do tříd. V technologii ASP.NET jsou jednotlivé třídy definovány v souborech s příponou *.cs* a stejně jako aplikační kód jednotlivých stránek je i tento kód zkompilován do společného DLL souboru.

Aplikace *Webový systém pro správu projektů* pomocí takových tříd například doplňuje standardní rodičovskou třídu všech stránek (`System.Web.UI.Page`) o inicializaci objektu pro práci s databází a metody užitečné na více místech aplikace nebo definuje třídy mající na starost autentizaci a autorizaci uživatelů, načítání informací o uživatelích nebo načítání uživatelských práv pro zvoleného uživatele a uzel.

5.3 Použité knihovny

Ačkoliv .NET Framework nabízí rozsáhlé knihovny funkcí a ovládacích prvků ASP.NET, bylo pro některé funkce efektivnější použít jiné knihovny tříd a ovládacích prvků. V aplikaci jsou použity tyto knihovny

- **FCKEditor –** open source WYSIWYG HTML editor obsahující implementaci i pro ASP.NET vytvořený v rámci komunity SourceForge.net
  URL: http://www.fckeditor.net/

- **Obout Calendar –** ovládací prvek kalendáře, který je možné použít také v módu `DateTimePicker` pro uživatelsky příjemnější zadávání data
  URL: http://www.obout.com/calendar/

  Společnost *obout inc.* poskytla pro účely bakalářské práce licenci tohoto ovládacího prvku zdarma. Order ID poskytnuté licence je 11225.

- **GIGANT.Common –** vlastní knihovna jednoduchých funkcí (např. konverzní, zaokrouhlovací, testovací), funkce pro jednodušší komunikaci s databází

Kapitola 6

Možnosti budoucího rozšiřování aplikace

V rámci analýzy podobných produktů (viz kapitola 1.2) byla nalezena inspirace pro možné budoucí rozšiřování aplikace. Tato kapitola shrnuje základní směry, kterými by se vývoj *Webového systému pro správu projektů* mohl v budoucnu ubírat.

- **Reporting** – většina analyzovaných aplikací obsahovala velmi propracované možnosti reportingu. Jako zajímavé vypadají hlavně různé výstupy ve formě grafů a tabulek určených různým uživatelům aplikace.

- **Exporty –** zajímavou možností pro uživatele může být i export vybraných dat do různých formátů – Excel, TXT, CSV, apod.

- **Napojení na elektronickou poštu –** v tuto chvíli *Webový systém pro správu projektů* umožňuje rozesílání denních informačních e-mailů obsahující výpis z kalendáře pro daný den a informace o nových komentářích. Rozšířením této funkcionality by mohla být možnost výběru, zda bude rozesílán pouze denní přehled nebo také upozornění na různé události přímo v době jejich vzniku. Tohoto by se dalo s úspěchem využít pro upozorňování formou SMS zpráv.

- **API** – v dnešní době je kladem velký důraz na komunikaci různých aplikací mezi sebou. Za zvážení by tedy stálo zpřístupnit dokumentované rozhraní a umožnit tak volání jednotlivých funkcí aplikace z externích systémů a tím pádem umožnit začlenění *Webového systému pro správu projektů* do širšího kontextu. Jako vhodná technologie pro vzdálené volání funkcí webové aplikace mohou sloužit *Webové služby (Web services)*. Microsoft .NET Framework a ASP.NET nabízí nativní podporu této technologie a rozšíření o rozhraní webové služby by tak mohlo být relativně jednoduché.

6.1 Optimalizace

Při běžném používání budou s největší pravděpodobností objevena úzká místa aplikace. Z toho důvodu bude nutné jednotlivé části optimalizovat, aby nedocházelo k omezování uživatelů během jejich práce například pomalou odezvou.

Jedno problematické místo, které se pravděpodobně projeví při větším množství dat, bylo již popsáno v kapitole 4.1.

Mimo to by stálo za úvahu výstupy některých složitějších databázových dotazů ukládat do keše. ASP.NET nabízí propracovanou podporu kešování celých stránek, jejich částí nebo některých objektů (např. objektů obsahující výstupy z databáze). Pro každý prvek uložený v keši je možné definovat dobu, za kterou jeho platnost vyprší, případně je možné vynutit jeho uvolnění manuálně.

Kapitola 7

Co obsahuje CD

Součástí bakalářské práce je CD obsahující vše potřebné pro instalaci aplikace a také zdrojové kódy a dokumentaci.

Konkrétně CD obsahuje:

- Instalační soubory pro Microsoft .NET Framework verze 1.1 – *v adresáři „NETFramework“*
- Instalační soubory Microsoft SQL Server 2000 Desktop Edition (známé také jako MSDE 2000) – *v adresáři „MSDE2000“*
- Programátorskou dokumentaci – *v adresáři „Documentation“*
- Uživatelskou dokumentaci – obsahující mimo jiné i postup instalace aplikace – *v adresáři „Documentation“*
- Zdrojové soubory aplikace včetně projektu pro Microsoft Visual Studio 2003 – *v adresáři „Source“*
- Soubory aplikace pro instalaci na aplikačním serveru – *v adresáři „Application“*
- Text vlastní bakalářské práce včetně originálních souborů s obrázky a diagramy (pro Microsoft Visio 2003) – *v adresáři „Text“*