Elektronická třídní kniha

Electronic class register

Bc. Tomáš Dudek

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš DUDEK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Elektronická třídní kniha**

Zásady pro vypracování:

1. Proveďte analýzu stávajících přístupů k vedení třídní knihy a vám známých elektronických variant.
2. Navrhněte metodický model třídní knihy jako podklad pro elektronické řešení.
3. Realizujte elektronickou třídní knihu a prověřte ji ve zkušebním režimu.
4. Prezentujte svou metodu návrhu, realizace a praktické výsledky.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Luboslav Lacko – SQL Hotová řešení, Computer Press Brno 2003
2. Martin Gurtler, Pavel Kocich – Visual Basic .NET Hotová řešení, Computer Press Brno 2005
3. František Šíma, David Vilínek – Visual Studio .NET praktické programování krok za krokem, Grada 2006
4. Luboslav Lacko – ASP.NET a ADO.NET 2.0 Hotová řešení, Computer Press Brno 2006
5. Hana Janusová, Miroslav Muller – UML srozumitelně, Computer Press Brno 2005
6. Miroslav Kučera, Jiří Peterka – Programování na webu, Knihy iDnes 1997

Vedoucí diplomové práce: **Ing. Petr Neumann, Ph.D.**
Ústav elektrotechniky a měření

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **28. května 2007**

Ve Zlíně dne 13. února 2007

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Dokument popisuje využití výpočetní techniky k vedení třídní knihy, nikoliv její nahrazení. Důvodem je snaha tento proces co nejvíce zjednodušit a zefektivnit. Práce je rozdělena do několika částí. První z nich je věnována popisu použití klasické třídní knihy. Další část se zabývá porovnáním klasické třídní knihy s existujícími elektronickými podobami tohoto dokumentu a návrhem vlastního systému. Obsahem práce je vytvoření uživatelského rozhraní a toho, co vše je nutné vytvořit, aby výsledná aplikace splnila obecně platná kritéria na vzhled a použitelnost. Největší důraz je věnován speciálně návrhu webových aplikací, které mají svá specifika a popisu jednotlivých technologií pro vývoj webových aplikací. Toto prostředí je zvoleno záměrně z důvodu obliby mezi počítačovými uživateli. Ti se s ním mohou setkat takřka na každém kroku nejen v podobě internetových stránek, ale například i v různých formulářích, hrách i v podobě rozhraní pro konfiguraci síťového hardwaru, jako jsou routery, ADSL modemy a jiná nejen síťová zařízení. Dokument popisuje použité technologie, zejména ASP .NET, Framework a jeho specifika.

Klíčová slova:

ASP, PHP, server, .NET, Framework, webová aplikace, třídní kniha

ABSTRACT

This document describes how computers can be used to manage a class register but not to replace it. The reason is to make this process simpler and more effective. This paper is divided into several parts. The first part describes the procedure of using the standard class register. The next part compares the standard class register with existing electronic registers and my proposed system. This paper elaborates the design of a user interface and all the conditions that must be met in order for this application to meet the generally accepted appearance and usability criteria. The biggest emphasis is on the design of web applications and their specifics as well as the description of technologies used in developing web applications. I have selected this tool because of its great popularity among computer users. They can encounter this tool virtually everywhere; not only as web pages but also as various web forms, games and configuration interfaces for network hardware such as routers, ADSL modems and other network equipment. This document describes used technologies, particularly ASP .NET, Framework and its specifics.

Keywords:

ASP, PHP, server, .NET, Framework, web application, class register

Poděkování:

Děkuji ing. Petrovi Neumannovi Ph.D. za odborné vedení, cenné rady a připomínky, které jsem uplatnil při psaní diplomové práce.

Motto:

Tajemství úspěchu člověka spočívá ve zvyku dělat to, co neúspěšní lidé dělají neradi.

A.Jackson King

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

V dnešní době se stále více mluví o rozmachu počítačů. Nejedná se jen o technické vybavení počítače, ale především o programové vybavení určeném snad již pro všechny odvětví lidského života. Již dnes existují lidé, kteří si neumějí představit třeba jen jediný den bez počítače.

Většina uživatelů používá své zařízení jako komunikační prostředek, na práci, pro zábavu a jako zdroj informací. Ale asi největším důvodem, proč tak velký rozmach počítačových systémů je jejich vzájemné propojení pomocí počítačové sítě. Což uživateli dává obrovské možnosti využitelnosti a zapojení výpočetní techniky do různých odvětví lidské činnosti, tedy i do školských dokumentů.

Tato práce je zaměřena na software určený pro pedagogickou činnost a především vytvoření aplikace pro elektronickou třídní knihu pomocí webové aplikace. Vývoj samotné aplikace jde rozfázovat do několika částí.

Zaprvé musíme stanovit požadavky na aplikaci. Toto provedeme na základě analýzy dostupných dokumentů o klasické „papírové“ třídní knize, její tvorbě, pravidlech vedení a možnostech moderního přístupu k ní. Také je nutné porovnat klasické třídní knihy a již existující podoby elektronických třídních knih. V neposlední řadě získání informací od svých kolegů učitelů, kteří budou vhodný zdrojem námětů a případné kritiky.

Zadruhé musíme nalézt vhodný nástroj pro tvorbu aplikace, aby její použitelnost byla co nejhodnější. Tato část práce zahrnuje seznámení se s technologií pro vývoj aplikací, s jednotlivými programovacími jazyky, přístupu k tvorbě databázi a práci s ní. Tato část je velice důležitá. Musíme porovnat klady a zápory.

Další krok obsahuje postupy vývoje aplikace, tvorba jednotlivých částí a jejich propojení. Závěrečným krokem je zavedení aplikace do praxe, včetně popisu funkcí a instalace.

I. TEORETICKÁ ČÁST

1 TŘÍDNÍ KNIHA

Snad každý si ze svých mladých let pamatuje třídní knihu. Někteří si lépe či hůře vybaví, jak vypadá a čemu slouží.

Nejprve trochu teorie, popíšeme si, jak taková kniha vypadá. Každé školské zařízení musí dle zákona, dokonce i mateřské školy vést třídní knihu. Studenti školského zařízení jsou rozděleni do skupin, kterým se říká třídy. Třída školského zařízení má svou třídní knihu, která slouží pro zaznamenávání různých dat. Každá třídní kniha je jednoznačně identifikovaná jménem třídy, školním rokem.

1.1 Klasická podoba třídní knihy

Třídní kniha je důležitou součástí školních tiskopisů. Její vyplňování se řídí určitými pravidly. Mezi nejhlavnější pravidla patří, že se v dokumentu se nic neškrtá ani nepřepisuje. Další pravidla řídí, kdo do třídní knihy zapisuje a jakým způsobem [1.].

Třídní kniha se skládá z těchto částí:

1. Obal
2. Přední strana
3. Seznam předmětů
4. Hospitace a inspekce ve třídě
5. Seznam žáků
6. Týdenní zápis
7. Zasedací pořádek a rozvrh hodin

1. Obal

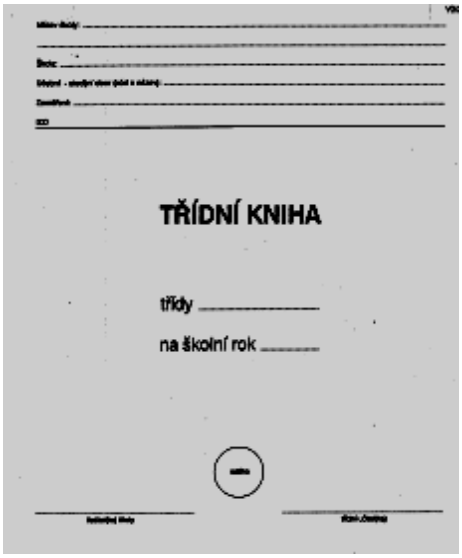
Obsahuje jméno třídy a školní rok. Slouží pouze k informativním účelům, které vypíše třídní učitel na začátku školního roku. Na této straně se v průběhu roku nevpisují ani neupravují stávající údaje (obr. 1).

Obr. 1. Obal třídní knihy

2. Přední strana

Obsahuje úplný název školy s kódem školy, zkrácený název školy, název studijního oboru s jeho kódem, zaměření oboru, IZO, jméno třídy, školní rok, podpis třídního učitele a ředitele školy, kulaté razítko školy.

Tato strana opět slouží k informačním účelů a vypisuje je třídní učitel. Velice důležité je zde kulaté razítko školy a podpisy (obr. č. 2).



Obr. 2. Přední strana třídní knihy

3. Seznam předmětů

Strana je rozdělena na dvě části.V první části jsou vypisovány povinné předměty, v druhé části jsou nepovinné předměty. U každého předmětu je vypsán jeho celý název, zkratka a podpis vyučujícího s údajem, od kdy předmět ve třídě vyučuje.

Název předmětů a zkratku vypisuje třídní učitel na začátku roku. Každý vyučující má za povinnost se podepsat, zapsat datum na začátku vyučování. Pokud dojde během roku k výměně vyučujících, také další vyučují má za povinnost se podepsat a napsat datum zahájení výuky (obr. 3).

Obr. 3. Seznam předmětů

4. Hospitace a inspekce ve třídě

Tato strana slouží k informaci o hospitaci nebo inspekci provedené ve třídě.

Osoba na inspekci či hospitaci je povinna vypsat datum, název předmětu, jméno učitele a také se podepsat (obr. 4).

Obr. 4. Hospitace a inspekce ve třídě

5. Přehled docházky

Seznam se skládá ze dvou stran.

Na první straně existuje seznam žáků dle abecedy, jejich jméno i příjmení, informace o dojíždění do školy a týdenní součet absence. Tato strana slouží

pro první pololetí. V poslední kolonce je celkový součet jak omluvených, tak i neomluvených hodin žáka.

Na druhé straně již není seznam žáků, ovšem pokračuje zde tabulka pro součet zameškaných hodin v jednotlivých týdnech. Opět je zde celkový souhrn omluvených i neomluvených hodin za druhé pololetí. Dále zde nalezneme místo pro poznámku o nepovinných předmětech jednotlivých žáků, rozdělení do skupin a další poznámky.

Tyto informace zapisuje třídní učitel či zástupce třídního učitele

(obr. 5. a 6.).

Číslo	Seznam žáků (Příjmení a jméno)	Dojíždí denně čím? kdy?	Počet zameškaných hodin v prvním pololetí – Týden: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	Celkem 1. pol. c*)	n*)
1					

Obr. 5. Docházka list 1.

Číslo	Počet zameškaných hodin ve druhém pololetí – Týden: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	Celkem 2. pol. c*)	n*)	Nepovinné předměty	Rozdělení do skupin	Poznámky
1						
2						

Obr. 6 Docházka list 2.

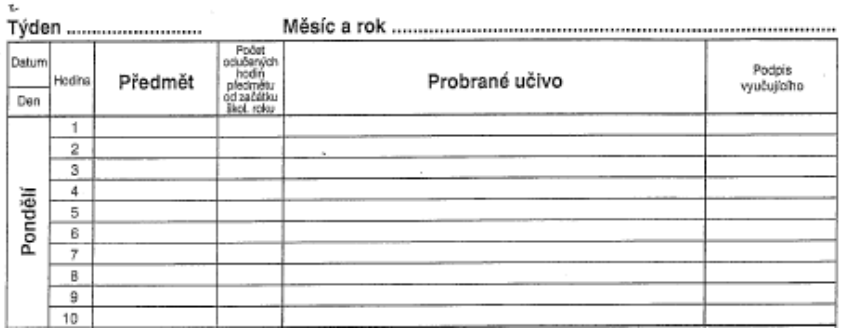
6. Týdenní zápis

V třídní knize obsahuje 41 týdenních výpisů. Každý výpis se skládá ze dvou listů. V záhlaví stánek nalezneme číslo týdne, měsíc, školní rok a pořádkovou službu z řad žáků třídy. List je rozdělen na 5 hlavních tabulek – jednotlivé dny s datem. Dny jsou rozděleny do hodin.

Každý vyučující je povinen na začátku hodiny provést zápis názvu předmětu ve zkratce, pořadové číslo hodiny předmětu, probírané učivo a podepsat se. Druhou důležitou část tvoří seznam nepřítomných žáků v jednotlivých hodinách. Jména se píší do sloupce pod sebou, při větší absenci se mohou napsat dva žáci na jeden

řádek. Nepřítomnost tohoto žáka se úhlopříčně vyškrtne. V neposlední řadě je zde také místo pro poznámky vyučujících jak k chování jednotlivých žáků, tak i k sdělením.

Třídní učitel má za povinnost vypisovat číslo týdne vypsat číslo týdne, měsíc a rok, datum u jednotlivých dnů a jména žáků pověřených službou. Postupně musí vyplňovat kolonky u absence žáků. Omluvení nepřítomnosti vyznačí přeškrtnutím úhlopříčky. Také musí zapsat součet omluvených a neomluvených hodin s důvodem absence (obr. 7 a 8).

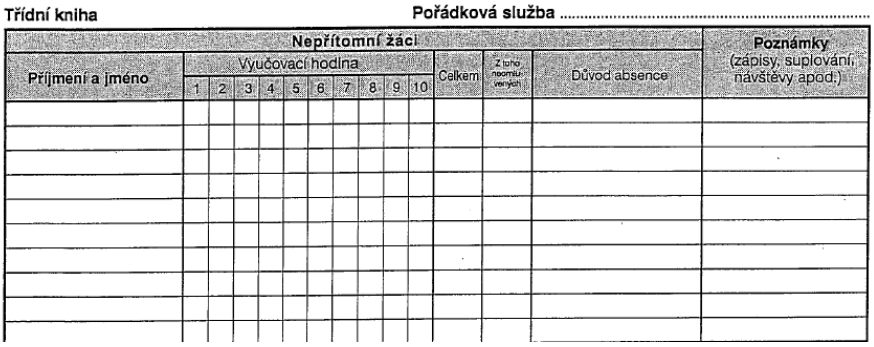


Týden ..... Měsíc a rok .....

Datum / Den	Hodina	Předmět	Počet omluvených hodin předmětu od začátku škol. roku	Probrané učivo	Podpis vyučujícího
Pondělí	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
	9				
	10				

Obr. 7. Týdenní zápis 1. Lis

t



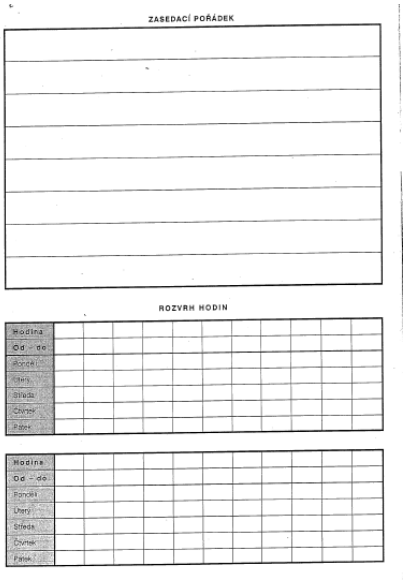
Třídní kniha Pořádková služba .....

Příjmení a jméno	Nepřítomní žáci: Vyučovací hodina 1	2	3	4	5	6	7	8	9	10	Celkem	Z toho neomluvených	Důvod absence	Poznámky (zápisy, suplování, návštěvy apod.)

Obr. 8. Týdenní zápis 2. list

7. Zasedací pořádek a rozvrh hodin

Tato strana slouží k informaci pro žáky i pro učitele. V první části jsou zapsány jména žáků dle jednotlivých lavic ve třídě. V druhé části je prostor pro rozvrh hodin na 1. i 2. pololetí (obr. 9).



Obr. 9. Zasedací pořádek a rozvrh hodin

1.2 Třídní kniha jako povinný dokument

Třídní kniha patří mezi povinnou dokumentaci školy. Pravidla její podoby jsou v zákoně č. 561/2004 Sb. O předškolním , základním, středním, vyšším odborném a jiném vzdělávání (školský zákon) § 28 odst. 1 Dokumentace škol a školských zařízení a ve znění pozdějších předpisů. Je zařazena do druhé kategorie tiskopisů : *„Tiskopisy vzor SEVT - alternativní formou schválených tiskopisů SEVT jsou výstupy z počítače na kancelářský papír a tiskopisy ostatních vydavatelství za předpokladu, že bude dodržen schválený vzor SEVT. Platnost od školního roku 1989/1990 (třídní knihy, třídní výkazy, protokoly..).“* [1.] Z těchto zákonů vyplývá:

Třídní kniha patří mezi povinné dokumentace. seznam dokumentace pro školy je každoročně aktualizován a uveřejňován na Věstníku MŠMT ČR. Změny v textových částích tiskopisů jsou každoročně uváděny v Seznamu povinné dokumentace škol a v Kupních smlouvách SEVT.

Počítačové zpracování třídních výkazů je možné za předpokladu, že bude současně vytvářena listinná kopie dokumentů a dodržena věcná schoda počítačového a ručního zpracování tiskopisů. Pro usnadnění tisku třídních výkazů je možné vytvořit každé pololetí souhrnou tabulku hodnocení prospěchu a chování třídy a na závěr studia nebo při přestupu žáka na jinou školu doplnit jednotlivým žákům hodnocení prospěchu

a chování. Tabulka hodnocení třídy se stanou součástí dokumentace a budou rovněž archivovány. Poznámky , které nebudou psány ručně, opatří třídní učitel parafou. Tato podmínka vstoupila v platnost od školního roku 1999/2000.

Dle zákona o osobních údajích č. č. 256/1999 Sb., o ochraně osobních údajů v informačních systémech a § 8 zákona č. 106/1999 Sb., o osobním přístupu k informacím a zákonu č. 101/2000 Sb., o ochraně osobních údajů a o změně některých zákonů od školního roku 2001/2002 nebudou školní tiskopisy obsahovat kolonku o národnosti.

Vzory tiskopisů povinné dokumentace škol a školských zařízení jsou uveřejněny na internetových stránkách MŠMT v oddílech předškolní, speciální, základní, střední a vyšší odborné školství.

1.3 Elektonická podoba třídní knihy

V předchozí kapitole si můžeme přečíst větu o tom, že je možné používat elektronickou podobu písemností, jestliže existuje věcná shoda s originálními písemnostmi. K čemu tedy mít klasickou třídní knihu a ještě mít elektronickou podobu těch samých informací? Odpověď je snadná – i v tak „jednoduchém systému“ jako je evidence třídní knihy, lze nalézt spoustu důležitých informací. Například který vyučující co, kde a kdy učí, zda student byl přítomen výuce, co která třída právě probírá, kolik času věnuje opakování, kdy vyučující naposledy psal s třídou písemnou práci, a tak dále.

Podle mého názoru nejvíce možností skýtá evidence absence, kde je možné jednoduše zjistit, kdo a kdy ze studentů třídy chyběl. Je možné vysledovat a vytvořit si jasný pohled na studenta a jeho docházku. A to zda je jeho absence nahodilá nebo cílená. Dochází-li ze strany studenta k záškoláctví a vyhýbání se výuce nebo jen určitému předmětu. Pokud vyučující bude mít nástroje na jasnou analýzu studentovi docházky, je možné předejít různým prospěchovým problémům studenta a vzájemným konfliktům mezi ním a vyučujícím.

V dnešní době se již na trhu vyskytuje několik podob elektronické třídní knihy. Většinou se jedná o přídavný modul k evidenci žáků a jejich klasifikace. Ovšem stále se jedná jen o první vlaštovky. Jejich uplatnění v praxi se ukáže až časem. Je tu stále mnoho překážek, které tyto programy budou muset překovat, než se dostanou do každodenní součásti učitelova života.

Jedná se především o tyto překážky:

- Špatná vybavenost škol technickým vybavením
- Neochota zavádět a učit se nové věci z řad „starších“ učitelů
- Špatná informovanost o novinkách

Ovšem tyto překážky se již na některých školách překonávají a i výuka dostává nový pohled. K překonání první překážky velice pomáhá ochota MŠMT ČR, které se snaží zavést do škol nové inovativní prvky jak do výuky, tak i do řešení problémů s dokumentací.

V dalších kapitolách se seznámíme se třemi programy, které umožňují vést elektronickou třídní knihu. Jsou velice rozlišné, jak pro náročnost finanční, tak i typem ovládáním. Jedná se o programy:

- Škola OnLine
- Bakalář
- Záškolák

1.3.1 Bakalář – systém pro administrativu školy

Jedná se o program, který pokrývá prakticky všechny oblasti školní administrativy [2.]. Program se skládá z několika modulů. Mezi základní moduly patří:

- Evidence žáků a zaměstnanců
- Přijímací zkoušky, Knihovna, Inventarizace, ...
- Grafické zpracování klasifikace
- Rozvrh hodin, Suplování, Plán akcí školy
- Rozpis maturit
- Tematické plány
- Web - informační modul pro rodiče a žáky
- Web - učitelé
- Třídní kniha

V dnešní době je tento program velice rozšířen do škol. Mezi jeho výhody patří lehké rozšíření dokumentace o potřebné údaje a také potřebné průběžné aktualizace novinek.

Mezi novinky patří modul třídní kniha, který je dnes jen v beta verzi a s jeho zařazení do prodeje se počítá od příštího školního roku . Ta nám umožňuje zadávat nepřítomnosti žáků v jednotlivých hodinách, dnech a týdnech, její rozlišení v teoretické a odborné výuce. Lehké sledování absence v jednotlivých předmětech. Tisk přehledů absence v jednotlivých obdobích, včetně překročení nastaveného limitu v jednotlivých předmětech. Zadaná absence je zobrazována ve webové aplikaci Informační modul pro rodiče. Je propojena nejen s modulem Web – informační modul pro rodiče a žáky, ale také s modulem Rozvrh a Suplování.

1.3.2 Projekt „Škola OnLine“

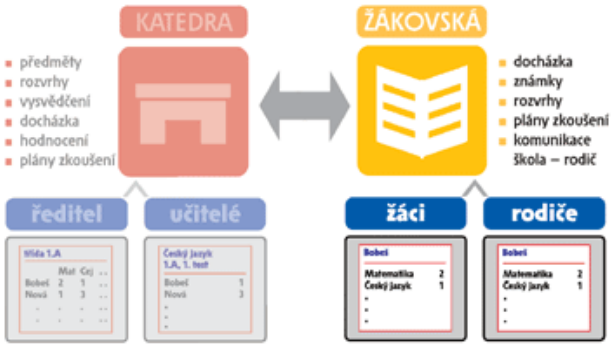
Projekt je tvořen pod záštitou MŠMT [3.]. Škola OnLine je nejen informačním zdrojem pro školy, je také komunikačním nástrojem on.line mezi školou a rodiči. Je rozdělen na několik sekcí.

Způsob přístupu k informacím a jejich zabezpečení:

Každý rodič nebo student, který přistupuje k žákovské, má své osobní heslo, pomocí kterého má přístup pouze k těm informacím, které jsou pro něj určeny. Díky tomu jsou pro žáka dostupné pouze jeho vlastní výsledky a pro rodiče pouze výsledky jeho dětí.

Možnosti žákovské sekce (obr. 10):

- Docházka - přehled docházky žáka v celém školním roce
- Hodnocení - výsledky žáka v předmětech
- Plány hodnocení - přehled zkoušení, která žáka v nejbližší době čekají
- Rozvrhy - aktuální rozvrh žáka i celé jeho třídy
- Komunikace - prostřednictvím e-mailu nebo SMS zpráv automatické odesílání informací o známkách a docházce pomocí e-mailu nebo na váš mobilní



Obr. 10. Princip žákovské sekce

1.3.3 Záškolák – žákovský informační a docházkový systém

Systém vyrábí plzeňská firma NetPro systems, spol. s r.o. Je tvořen v rámci projektu *„Metla na záškoláky“* [4.]. Systém umožňuje kontrolu nad docházkou, studijními aktivitami a prospěchem žáků. Poskytuje okamžitý přehled rodičů o chování jejich dětí. K systému je nutné vybavit školu snímacími terminály a počítačovou sítí.

Projekt je rozdělen do tří sekcí – pro učitele, pro rodiče a pro žáky.

- Rodičům umožňuje aktivně sledovat studijní aktivity svých dětí.
- Učitelům dává nástroj pro plánovité uplatňování učebních osnov.
- Žákům nabízí prostor pro aktivní vzdělávání bez zbytečného stresu s podporou informovaných rodičů.

Celý systéme je rozdělen do dvou částí:

1. **Žákovský docházkový systém**

   Žák-student získá při nástupu do školy registrační průkazku určenou k jeho bezdotykové identifikaci v rámci docházkového systému školy. Po příchodu do budovy školy se žák-student zaregistruje u instalovaných bezdotykových terminálů. Tím je do systému zanesena informace o jeho přítomnosti, která se zakládá do docházkového listu každého žáka-studenta.

   Vyhodnocování docházky žáků je stěžejním kritériem pro navazující funkce jak ve sféře škola-učitel tak ve sféře rodič Výsledky statistik vedených nad docházkovými listy žáků z jedné třídy nebo skupiny jsou vedeny jako „Elektronická třídní kniha“.

2. **Školský informační systém.**

   Informační systém školy představuje soubor intranetových služeb přístupných pomocí běžného internetového prohlížeče.

1.4 Elektronická třídní kniha v praxi

Hlavním záměrem mé diplomové práce je vytvořit program – třídní kniha. Tuto aplikaci jsem vytvořil na podkladě kladených požadavků na školské dokumenty.

Díky tomu, že vyučuji na střední škole, a ochotě mých spolupracovníků a vedení školy jsem měl i možnost svou aplikaci předvést svým kolegům a vyzkoušet ji v praxi.

Naše škola je rozdělena na několik budov. V naší budově si žijeme z velké části vlastním životem, máme zde jen 8 tříd, proto zde může vyučovat z velké části stálá skupina vyučujících a je vytvořen vlastní rozvrh hodin.

Po technické stránce jsme docela dobře vybaveni. V budově máme učebny výpočetní techniky a projektory, přenosné projektory s notebooky a samozřejmě zde nechybí internetová síť s wifi.

Velká část učitelů začíná využívat moderních výukových prostředků ve výuce, tedy i notebook s projektorem a připojením na internet. Proto nebyl tak velký problém zaučit je pro práci s elektronickou třídní knihou. Tento nápad se části vyučujícím zamlouval, ovšem stále se vyskytují jedinci, kteří nechtějí zavádět nové věci a především u nich chybí ochota něčemu novému se učti, především, pokud se to týká výpočetní techniky.

Část spolupracovníků využívalo několik dní můj program. Jejich odezva byla většinou kladná. Ovšem i problémy se vyskytly. Některé chyby programu se odhalili až při dlouhodobějším využívání. Informace od spolupracovníků o nalezených chybách a problémem při práci však jsou velice důležité a zakládá se na nich další konfigurace programu. Doufám, že v budoucnu bude naše škola moci využívat více informačních systémů nejen k vnitřním záležitostem, ale i ke komunikaci s žáky a rodiči pomocí internetu.

2 VÝBĚR VHODNÉ TECHNOLOGIE PRO RELIZACI PROJEKTU

Výběr vhodného technologie pro realizaci projektu, je jedna z klíčových operací, kterou nelze podcenit. Výběr nevhodné technologie může znamenat velké problémy jak při vývoji, implementaci, tak i provozu systému.

Vzhledem k mé orientaci na produkty firmy Microsoft, mám do jisté míry omezený výběr. Nicméně i v tom to segmentu se dá najít vhodné prostředky pro vývoj takovéto aplikace. Vzhledem k dlouhodobé zkušenosti s vývojovým prostředím Visual Studio, jsem volil cestu nejmenšího zla a zůstal jsem jí věrný i nadále. Nicméně měl jsem nutkání poznat něco nového a nakonec jsem si vybral vývojové prostředí postavené na technologii .NET a to konkrétně Visual Studio 2005. Dalším poměrně velkým dilematem bylo rozhodnout se pro vytvoření klasické desktopové aplikace, nebo rovnou vytvořit webovou aplikaci. Odpověď se zprvu nezdála zcela jednoznačná, bylo zde hodně plusů – nová zaběhnutá technologie, silná podpora atd., ale na druhé straně bylo několik mínusů, kterých jsem se při vývoji obával – přece jenom webová aplikace se trochu jinak chová, složitější vývoj, implementace. Nakonec převládly klady nad zápory a vybral jsem webovou verzi aplikace. Ještě než se pustíme do realizace konkrétní aplikace, nejprve v další části popíši některé důležité pojmy.

2.1 Webová aplikace

Webová aplikace je podle definice klient/server software, který komunikuje s uživatelem nebo jiným systémem prostřednictvím protokolu HTTP. Za klienta používají uživatelé nejčastěji webové prohlížeče, jako je Internet Explorer, Mozilla nebo Opera. Automatizované systémy, například roboti fulltextových vyhledávačů, pak přistupují s pomocí HTTP agentů.

Klient na základě interakce s uživatelem zasílá serveru jednotlivé požadavky a následně zobrazuje obdržené webové stránky zapsané zpravidla v jazyce (X)HTML. Rozsah aplikací poskytovaných služeb se může pohybovat od jednoduchých úloh, jakými je například kniha hostů, až po složitější a programy typu elektronické obchody, nebo bankovní aplikace.

Je možné vymezit skupinu funkčních požadavků, které jsou společné pro téměř všechny pokročilejší webové aplikace. Jejich podpora přímo v používaných programovacích jazycích ale často nativně neexistuje vůbec, nebo je pouze částečná či nedostatečná. Potřebné funkce si musí vývojář doprogramovat sám. Naštěstí pro naprostou většinu takových úloh existují již hotové knihovny, které bývají i volně k dispozici. Ucelenější soubor takových knihoven, zajišťující komplexní funkcionalitu nutnou pro vývoj webové aplikace, tvoří framework. Neexistuje žádná ustálení a všeobecně přijímaná definice výrazu framework. Nejčastěji se však objevují dvě základní pojetí tohoto výrazu.

První přístup chápe framework jako balík užitečných kódů a knihoven, které uceleně a standardně pokrývají dané společné požadavky aplikací. Jsou uloženy v nějaké dostupné repository a programátor si do aplikace načítá jen ty vybrané knihovny, které zrovna potřebuje, v závislosti na charakteru dané aplikace. V komplexní aplikaci tak pro ošetření nějakého problému použije pokročilé a sofistikované knihovny, zatímco pro jednoduchou úlohu stačí knihovna se základní funkčností.

Druhé pojetí je užší v tom smyslu, že vše výše popsané považuje jen za balík relativně samostatných navzájem nesouvisejících knihoven, nikoliv za framework. Framework vzniká právě až vhodným poskládáním těchto jednotlivých knihoven dohromady, jejich účelným provázáním. Vzájemnými interakcemi mezi nimi často získáme dodatečné funkčnosti, které se u izolovaných knihoven nemohou objevit. Tímto postupem se tedy vytvoří pevný rámec, framework, uvnitř kterého se pak vyvíjí výsledná aplikace.

Druhé pojetí má jednu zřejmou nevýhodu. V takovém frameworku je pevně dán rozsah a komplexnost poskytovaných funkcí. Programátor aplikace má jen omezenou volnost výběru knihoven, kvůli jejich vzájemné provázanosti se vždy musí použít naprostá většina z nich, ať už jsou právě potřeba, nebo ne. Jeden konkrétní takto pojatý framework se tak z hlediska složitosti a šíře poskytovaných služeb hodí jen pro určitou skupinu aplikací, které jej využijí. Například tvořit primitivní knihu hostů v komplexním frameworku, který poskytuje podporu pro moduly, registrované uživatele, uživatelské skupiny, přístupová práva a ošetření timeoutu je jako jít s kanónem na vrabce. Naopak, při výběru vhodného a odpovídajícího frameworku je tvorba výsledné aplikace zpravidla efektivnější a systémovější, než při použití jednotlivých oddělených knihoven.

Nyní zkusme definovat základní funkční požadavky, které jsou společné a typické pro většinu pokročilejších webových aplikací, a které by tedy měl komplexní framework nějakým způsobem zajišťovat:

- vhodná architektura aplikace;
- modularita;
- zabezpečení aplikace;
- validace vstupních parametrů;
- uživatelské autentizace a správa uživatelů;
- sessions a entitní autentizace;
- autorizace a přístupová práva;
- jednotná databázová vrstva;
- logování událostí a akcí;
- validita výstupních stánek;
- maskování URL.

2.2 Výpočetní modely

Server je v informatice obecné označení pro proces nebo systém, který poskytuje nějakou službu. Služba je obvykle realizována některým aplikačním síťovým protokolem, jako je například http (web server) nebo LPD (tiskový server). Proces, který službu využívá, se nazývá klient, architektura, která používá tento princip, se nazývá klient – server.

2.2.1 Klient – server

Výpočetní model klient – server je vhodné chápat především jako dělbu práce mezi dvěma složkami a to serverem a jeho klientem. V dnešní době, plné různorodých aplikací, přitom mají obě složky velmi mnoho prostoru pro to, jak konkrétně si mezi sebou rozdělit jednotlivé úkoly. V důsledku toho je pak možné se setkat s více různými variantami modelu

klient – server, které se liší právě v tom, kolik toho nese na svých bedrech každá z obou složek. A kromě toho, musí se vždy jednat právě a pouze o dvě složky?

Až doposud jsme za hlavní motivaci celého modelu klient – server považovali snahu minimalizovat objem dat přenášených po sítí mezi klientem a serverem. Imperativnost tohoto kritéria se v čase může měnit, zejména v důsledku stále větší dostupnosti přenosových kapacit, a ke slovu se mohou významnějším způsobem dostávat i další kritéria a motivace. Například různá implementační hlediska související s tím, jak obtížné je prakticky implementovat tu kterou složku. Vždyť pořídit si výkonnější hardware, či propustnější přenosové cesty, je dnes do značné míry již jen otázkou dostatku peněz, zatímco zajištění všeho potřebného pro vývoj i následný provoz aplikací může být mnohem náročnější. Ať již jde o potřebný knot – how, vhodné vývojové nástroje, personální zajištění i schopnost vše kočírovat. No a to samozdřejmně také musí ovlivnit konkrétní formu dělby práce mezi servery a jejich klienty.

2.2.1.1 Tři druhy činností

Výpočetní modely klient – server je dnes nejčastěji využívanými aplikacemi, které spadají do široké, velmi vágně definované kategorie Informačních systémů. Představit si pod tím můžeme například aplikace pro podporu nejrůznějších agend různých firem, institucí a dalších orgánů. Od účetnictví, skladového hospodářství, až třeba po systémy na podporu rozhodování. Všechny takové aplikace přitom mají některé společné rysy. Například se v nich vždy vyskytují nějaká základní data, která musí být vhodným způsobem uskladněna a současně k nim musí být zajištěn takový přístup, jaký vyhovuje povaze a charakteru samotné aplikace.

Další nezbytnou součástí je pak nějaká forma komunikace s uživatelem, spočívající ve sběru dotazů a jiných příkazů a povelů od uživatele směrem k aplikaci, a na druhé straně i nezbytné zobrazování (či honosnější prezentaci) získaných výsledků. Dále zde musí být ještě třetí část, která zajišťuje všechno to, co je pro danou aplikaci specifické a konkrétní, neboli realizuje vlastní logiku celé aplikace. Například půjde-li o účetnictví, jsou právě zde implementována všechna pravidla způsobující správné promítnutí jedné účetní operace do všech účtů, podúčtů, knih, listů, kterých se to týká.

Zmíněné tři druhy činností, které jsme si vymezili, jsou v různých odborných pramenech označovány různě. Naznačme si zde alespoň jednu terminologii, zavedenou prestižní konzultační firnou The Partner Group:

- **Presentation** – uživatelské rozhraní a komunikace s uživatelem (prezentační činnost)
- **Application Function** – vlastní logika aplikace (aplikační činnost)
- **Data Management** – správa dat

Volba jedné konkrétní terminologie v závěru předchozího odstavce přitom nebyla náhodná, protože právě od zmíněné konzultační firmy Garten Group pochází zřejmě nepoužívanější klasifikace možných způsobů dělby práce mezi klientem a serverem. Patří se například:

- **Distributed Presentation**

  Zde jsou prezentační činnosti rozděleny mezi klientem a serverem, který navíc sám vykonává i všechny zbývající činnosti. Na straně serveru se nejčastěji jedná o generování výstupů ve znakovém režimu do textového okna, které ale ve skutečnosti není nikde zobrazováno. Jeho obsah je místo toho přenášen klientovi, a teprve ten jej zobrazuje uživateli, obvykle již v grafickém režimu emulujícím znakový režim textového okna. Analogicky pak pro obrácený směr, týkající se vstupů od uživatele. Hlavní výhodou tohoto řešení, které se asi nejvíce blíží modelu host – terminál je skutečnost, že serverová část aplikace může používat standardní systémové prostředky pro zobrazování na znakových výstupních zařízeních.

- **Repote Presentation**

  Veškeré prezentační funkce ponechány na klientovi, zatímco veškeré aplikační funkce a funkce spojené se správou dat zajišťuje server.

- **Distributed Funkction**

  Všechny prezentační funkce jsou na klientovi, veškerá správa dat na serveru a o aplikační funkce se obě složky dělí. Některé činnosti související s vlastní logikou aplikace tedy zajišťuje sám klient, zatímco zbývající obstarává server.

- **Repote Data Management**

Veškeré prezentační i aplikační činnosti zajišťuje klient, zatímco server se věnuje pouze správě dat.

- **Distributed Data Base**

  Většinu činností zajišťuje klient (veškeré prezentační činnosti, aplikační činnosti), zatímco o udržení a správu dat se dělí se serverem. Jak již název této varianty napovídá, je tato varianta určena zejména pro podporu distribuovaných databází.

2.2.2 Dvou versus tříúrovňová architektura

Základní premisou, kterou jsme až dosud u celého modelu klient – server předpokládali, je dělba práce mezi dva disjunktní subjekty, server a klient. Jak se ale tato dvousložkovost slučuje s tím, že většina aplikací, které z tohoto modelu vychází, musí zajistit tři hlavní okruhy činností? Nebylo by přirozenějším řešením používat složky tři? Pak by rozdělení kompetencí mezi ně mohlo být velmi jednoduché a přirozené a nemuselo by vést někdy k poněkud krkolomným způsobům rozdělování tří věcí mezi dva subjekty. Nebylo by tedy koncepčnějším řešením zavést tříúrovňovou architekturu klient – server místo stávající dvouúrovňové?

Pravdou je, že se tak dnes mnohdy již děje, především u rozsáhlejších a náročnějších aplikací povahy velkých informačních systémů. Dochází zde totiž stále více k osamostatňování databází a systémů řízení bází dat a k jejich přesunu na pozadí nebo na samostatné počítače, které jsou optimalizovány pro provozování různých databází. Výhodou je pak i skutečnost, že aplikační činnosti, ve smyslu výše uvedené tříúrovňové klasifikace, mohou být stále méně závislé na konkrétním databázovém stroji, díky pokrokům ve standardizaci způsobů komunikace s databázemi, například díky jazyku SQL.

Zajímavé a velmi perspektivní možnosti se přitom objevují i v oblasti prezentačních služeb. Zde se totiž doslova vnucuje myšlenka využití služby Word Wide Web, ze které se stále více stává univerzální klientská platforma využitelná především pro prezentační účely. S pomocí služby WWW je vcelku jednoduché postarat se o zobrazení výstupů na uživatelově počítači a o sběr jeho vstupů určených samotné aplikaci. Jediné, co je k tomu potřeba, je zajistit realizace vhodné brány mezi aplikační částí samotné aplikace a WWW serverem, která bude zajišťovat potřebné konverze. Velkou výhodou přitom bude jednotné uživatelské prostředí, neboť koncový uživatel bude přistupovat

k různým službám a systémům prostřednictvím jednoho jediného uživatelského rozhraní, respektive prostřednictvím jediného klientského programu WWW prohlížeče (browseru).

2.2.3 Tencí bohatí klienti, technologie Java a .NET

Zatímco v architektuře klient/server běží programy (přesněji aplikační logika, business logika) na bohatých, dobře vybavených grafických klientech a s daty uloženými na serverech v centrálních databázích, na přelomu tisíciletí se začala prosazovat centralizovaná koncepce s označením vícevrstvá architektura.

2.2.3.1 Výhody vícevrstvé architektury

Zjednodušeně lze říci, že jde o návrat po vývojové spirále k terminálovému provozu, jen o poznání dokonalejšímu, než jak jej znali pamětníci starých sálových počítačů. Ve vícevrstvé architektuře program běží na centrálních systémech vytvořených ze vzájemně spolupracujících aplikačních serverů. Na klientských stanicích se pouze zobrazují data a ovládací prvky pro styk uživatele s aplikací.

Základní výhodou vícevrstvé architektury při údržbě programu je proto snadná údržba aplikací, daná již zmíněnou centralizací. V případě (nového) tenkého bohatého klienta se totiž kromě výměny části centrální aplikace na aplikačním serveru čas od času provede - plně automaticky a pro uživatele skrytě - jeho jednoduchá aktualizace (update) po síti. Opravy chyb, změny verzí a drobná vylepšení programů se již nemusejí složitě provádět na tisících počítačů, stačí zde pouhá výměna příslušné části centrální aplikace na **aplikačních serverech**.

Vzhledem k centralizaci tedy odpadají problémy spojené s distribuovaným zpracováním dat, jako je sehrávání datových souborů s následnou synchronizací a ztotožňováním. S podporou nového technického vybavení (clustery, externí disková pole, grid-technologie a blade-servery) se snadno řeší i problémy svázané se škálovatelností a se zálohováním informačního systému.

2.2.3.2 Tenký bohatý klient

Uživateli se údaje často zobrazují na koncovém terminálu (stanici), v univerzálním a snadno dostupném internetovém prohlížeči (HTML), který nemá přílišné nároky na zdroje. Taková klientská stanice se proto někdy také označuje jako "tenký klient"

HTML. Současný vývoj však ukazuje, že snížení komfortu ovládání aplikací, které je vynucené omezenými možnostmi jazyka HTML, je příliš znatelné.

Začíná se proto čím dál více objevovat komfortnější koncové prostředí. Jedná se o klientské programy využívající plně grafické možnosti operačních systémů k tomu, aby uživateli nabídly všechen ovládací komfort, na který je zvyklý ze starších systémů klient/server.

Aby se odlišili "noví" klienti z vícevrstvé architektury od "starých" (tlustých) klientů z architektury klient/server, zavedl se ve vícevrstvé architektuře také pojem **tenký bohatý klient**. Tento druh terminálových programů lze vytvářet jak v prostředí .NET společnosti Microsoft, tak i jako aplikace nebo applety v jazyce Java s využitím grafických knihoven Swing.

Jedním z možných klientů jsou i informační kiosky, jaké jsou umístěny například na Ministerstvu práce a sociálních věcí.

2.2.3.3 Technologie Java a .NET

Převládající technologie pro tvorbu vícevrstvých aplikací jsou dvě, a sice **.NET** společnosti Microsoft a **Java 2 Enterprise Edition** (J2EE) společnosti Sun Microsystems. Technologie .NET se uplatňuje zejména v rámci operačního systému Microsoft Windows, technologii J2EE zase podporují firmy jako IBM, Oracle, Sun a další směrem k systémům Linux, Unix a Windows. Ačkoli jsou rozdíly mezi oběma technologiemi značné, existuje naštěstí společná platforma, na níž se mohou technologie .NET a J2EE domluvit. Jedná se o webové služby (Web Services) a s nimi spojené standardy XML (eXtended Markup Language), SOAP (Simple Object Access Protocol), UDDI (Universal Description Discovery and Integration) a WSDL (Web Service Definition Language).

2.2.3.4 Další klientská prostředí

Ve vícevrstvé architektuře se vedle tenkých klientů na bázi osobních počítačů a notebooků stále více uplatňují PDA (Personal Digital Assistant) a mobilní telefony. Malá mobilní zařízení se mohou připojovat k informačním systémům buď bezdrátově prostřednictvím služeb mobilních operátorů, nebo přes klientskou stanici. návrhu systému elektronické třídní knihy

3 SPECIFIKACE POŽADAVKŮ

Specifikace požadavků znamenají popis jisté funkčnosti systému nebo jeho vlastností, které budou v systému implementovány, tedy požadavky, přání uživatelů systému.

Rozlišujeme přitom dva druhy základní typy požadavků:

- funkčnost – specifické požadavky na funkčnost systému,
- ne-funkční – specifikují jisté vlastnosti systému, případně podmínky omezující fungování systému.

Specifikace požadavků musí vyjadřovat, co by systém měl poskytovat, nikoliv jakým způsobem se to bude provádět.

Tato část je asi nejdůležitější z celého návrhu a realizace projektu, také nejvíce podceňovanou částí návrhu. Je to mu tak z jednoho prostého důvodu, a to že přání zadavatele nebo uživatele se často velice liší od výsledného díla programátora.

3.1 Zdroje požadavků

Proces získávání požadavků od budoucích uživatelů je poměrně náročný. Existují uživatelé, kteří mají již konkrétné představu o funkčnosti systému a jsou schopni přesně specifikovat své požadavky a s programátorem se přesně domluvit jaké služby má systém poskytovat. Na druhé straně jsou uživatelé, kteří vývojáři svou práci nijak neulehčují a nejsou schopni jasně formulovat svá přání. Tito uživatelé nechávají vývojáři určitou volnost při návrhu a požadují dodání systému, který uspokojí všechny jeho nároky, aniž by spolupracovali na jejich návrzích.

Zdroje požadavků:

- **legislativa** – jsou situace, při kterých se musíme řídit některými normami, které určují právní mantinely, jako je například ochrana osobních informací
- **požadavky zákazníka** – tato skupina je asi největším zdrojem informací
- **existující systémy uživatelů** – do jisté míry slouží jako vhodný náhled na problematiku, kterou někdo vytvořil před vámi
- **pracovní procesy uživatelů** – uživatel provádí stále stejné operace, které se dají z automatizovat a ulehčit tím pracovníkovi čas na jiné důležité úkony

- **vlastní know-how pro danou problematiku** – jsou specifické obory, kdy programátor nejprve musí nastudovat a získat informace, aby sám byl schopen navrhnou to nejlepší řešení pro zákazníka, aby systém po jeho zavedení splňoval specifické požadavky
- **prostředí zákazníka a jeho softwarové a hardwarové vybavení** – programátor se musí rozhodnout, jakým způsobem bude prováděna implementace systému a zda bude moci využít prostředků, které má uživatel k dispozici.

Pro úplnost si ještě uvedeme nefunkční požadavky, které sice neovlivňují funkčnost systému, jsou důležité je brát na vědomí, protože tvoří okrajové podmínky řešení projektu.

- dodržení určitých standardů,
- využití určitých komponent,
- rychlost odezvy systému na požadavek,
- nároky na výkonnost,
- zabezpečení systému,
- použitá architektura,
- atd.

3.1.1 Požadavky na systém elektronické třídní knihy

Z takto jasně definovaných pravidel pro specifikaci požadavků vyplývají následující očekávání od systému.

Je nutné evidovat základní informace o těchto skupinách a to:

- **Informace o škole** – sem zejména patří název školy, který je jednoznačným identifikátorem školního zařízení, specifikace a struktura školního roku, která je klíčová pro celou datovou strukturu aplikace. Hlavní závislost je v tom, že aplikace musí být schopná jednoduše znovu použitelná pro další zachycení školního období nejen pro současné použití.

- **Informace o studentech** – zde se jedná o data související s existencí studenta, jako je jeho jméno a příjmení, rodné číslo pro jeho jednoznačnou identifikaci v celém systému, samozřejmě je zde kladem důraz na ochranu osobních dat, a proto by se rodné číslo nemělo vyskytovat jen tam, kde je to nutné. Dále pak adresa studenta, popřípadě další kontaktní informace.
- **Informace o rodičích studenta** – je nutné vytvořit u každého studenta informace o jeho rodičích, které mají mít informační charakter v případě potřeby komunikace s rodiči studenta.
- **Informace o zaměstnancích** – zde se bude jednat pouze údaje sloužících pro interní potřeby organizace. Každá zaměstnanec musí být jednoznačné identifikovatelný.
- **Informace o předmětech** – základní údaje o tom, které vyučovací předměty se v dané organizaci vyučují.
- **Třídy** – budou sloužit pro vytváření skupin studentů daného oboru. Informace, které bude nutno evidovat, pro vedení třídí knihy. Mezi tyto data patří obor vzdělávání, označení střídy a aktuální ročník třídy. Mezi důležité informace bude nutné rozdělení studentů do skupin podle nutnosti vyučování některých předmětů nebo z důvodu kapacity odborných učeben.
- **Místnosti** – slouží jako údaj, pro organizaci výuky, bude tedy nutné vytvořit systém identifikátorů každé místnosti.
- **Plánování rozvrhu školy** – tyto informace budou asi nejzásadnější pro celou aplikaci. Zde je nutné podchytit všechny časové sledy výuky, využití učitelů a obsazenost třídy. Tato evidence tvoří nejdůležitější část návrhu a jeho konečná podoba ovlivní podobou a funkčnost celého systému.
- **Evidence třídní knihy** – pro každou třídu bude vytvořena třídní kniha. Třídí kniha bude obsahovat údaje o tom, kdy byla odučena jaká vyučovací hodina, kdo ji vedl, co bylo obsahem. U každé vyučovací hodině bude uvedena absence studenta.

Nefunkční požadavky

- Aplikace bude muset být přístupná odkudkoliv z interní sítě školy, tak aby vyučující mohl pohodlně jednoduše provést zápis do třídní knihy.
- Je nutné vytvořit jasné pravidla přístupu pro jednotlivé skupiny uživatelů a zamezit tím neoprávněnému vstupu nežádoucích osob.
- Je nutné vytvořit aplikaci, která bude jednoduše ovladatelná a uživatelsky přívětivá, tak aby uživatel neměl problémy s ovládáním systému.

Při své práci na projektu jsem nechal do jisté míry ovlivnit vlastními názory a požadavky na evidenci třídní knihy. Postupem práce jsme zjistil, že samotný návrh systému nebude , až tak jednoduchý a budu muset zakomponovat mnoho věcí a požadavků, s kterými jsem při mém prvotním návrhu vůbec nepočítal.

3.2 Případy užití

Případy užití někdy také označované jako užitné případy. Snahou je přesně zachytit funkčnost, která bude budoucím informačním systémem pokryta a vymezují tak jednoznačně rozsah prací. Každý případ popisuje jeden z použití systému, tedy jednu funkčnost systému.

3.2.1 Aktéři

Pojem aktér představuje roli, ve které vystupuje uživatel v rámci jeho komunikace se systémem. Aktérem se nemyslí pouze jeden konkrétní člověk, ale celá skupina uživatelů, kteří budou provádět popisovanou roly. Na druhé straně jeden fyzický uživatel může vystupovat ve více různých rolích. K jednomu aktéru můžeme asociovat více případů užití. Při návrhu si jednoznačně musíme uvědomit, že aktérem musí být člověk nikoliv část systému.

3.3 Navržené případy užití

Vzhled systému a jeho rozmanitosti, je nutné brát v potaz, že systém bude muset být nějakým způsobem chráněn. Uživatelé, kteří budou aplikaci využívat, musí mít omezené pravomoci a omezený přístup k jednotlivým operacím.

V zásadě budou v systému použity tyto typy uživatelů:

- **Admin** –neomezený vládce celé aplikace. Bude mít za úkol zprávu celého systému. Bude mít nestarosti zprávu uživatelů, tedy jejich vytváření, mazaní, editaci. Dále také definovat rozvrh.
- **Učitel** – bude moci editovat stávající data o studentech a třídách, zápis do třídní knihy, omlouvat absenci.
- **Student** – bude se jednat o nepřihlášeného uživatele, který bude moci například pouze prohlížet rozvrh.
- **Rodič** – tento aktér bude moci po zadání ověřovacího údaje prohlížet absenci svého dítěte.

Use Case (obr. 11) názorně popisuje uvedené případy užití a vztahy mezi uživatelskými rolemi.

Obr. 11. Use Case – případy užití

3.4 Grafické uživatelské rozhraní

Grafický vzhlede a způsob komunikace s uživatelem jsou základními faktory, které ovlivňují použitelnost a kvalitu programu. Grafický návrh vzhledu je kreativní věc, která se neustále vyvíjí a musí se do jisté míry přizpůsobit novým trendům, které jej ovlivňují. Pěkný vhled nesmí být na úkor použitelnosti celého aplikace. Největším z problém je zajistit kompromis mezi kvalitním zhledem a funkčností. Není nikde definovány hranice, které určují, jak co má vypadat.

Každý dnes vytvořený program používá pro interakci s uživatelem nějaké grafické prostředí (Graphical User Interface). Pryč jsou ty doby, kdy uživatel se musel smířit s černou obrazovkou a celá interakce se probíhala pomocí příkazů v stupu a výstupu.

Grafické uživatelské prostředí umožňuje jednoduše a přehledně shromažďovat informace od uživatele a opět je uživateli zobrazit.

Základem GUI jsou standardní grafické prvky, které tvoří celé uživatelské rozhraní. Jednotlivé prvky jsou tvořeny tlačítky, textovými boxy, nabídkami, kontextové menu, seznamy atd. Grafické uživatelské prostředí do jisté míry přináší do aplikací jakýsi stereotyp, protože většina aplikací (asi 2 / 3) má stejný vzhled. Nicméně pro uživatele to je výhoda. Uživatel nemusí dlouho přemýšlet o tom, jak který prvek má vypadat a co má za úkol.

II. PRAKTICKÁ ČÁST

4 POUŽITÉ TECHNOLOGIE PŘI REALIZACI PROJEKTU

V této kapitole se podrobněji podíváme na technologie, které jsem použil pro vývoj aplikace.

4.1 .NET

.NET je zastřešující název pro soubor technologií v softwarových produktech tvořících celou novou platformu, která je dostupná pro Web, Windows i Pocket PC.

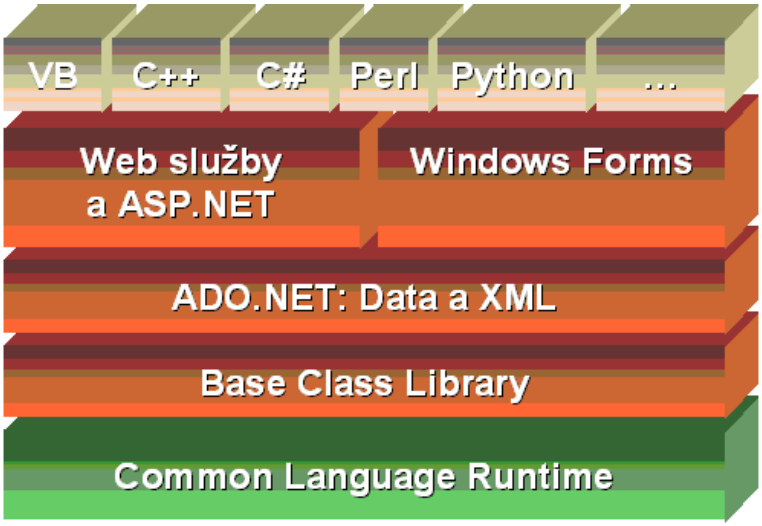
Pro tvorbu aplikací splňujících tyto myšlenky vydal Microsoft Visual Studio .NET, které bylo oproti předchozí verzi rozšířeno o snadné návrhy webových XML služeb .NET, a .NET Framework, zajišťující prostředí potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak potřebné knihovny, jako Java. Visual studio se skládá z jazyků Visual Basic, J#, C# a Managed C++. Těmito jazyky napsanou aplikaci pak můžete bez problémů převést do ASP.NET(Web) nebo pro pocket PC (.NET Compact Framework).

4.1.1 .NET framework

Pro činnost webových stránek v ASP.NET 2.0 je třeba komponenta zvaná *Microsoft .NET Framework 2.0*. Framework se stará o zabezpečující záležitosti, které dříve museli vývojáři často řešit a dnes je považují za tak samozřejmé a mnohdy si je ani neuvědomují. NET Framework se stará o nízkoúrovňové a nezáživné povinností jakými jsou:

- správa paměti, vytváření a rušení objektů
- spouštění a zastavování vláken kódu
- bezpečnost kódu a kontrola oprávnění k prováděným operacím
- natahování potřebných knihoven a komponent do paměti apod.

O tyto téměř neviditelné, ale velmi důležité operace se stará část zvaná *Common Language Runtime* (CLR) – viz obr. 12



Obr. 12. Vrstvy Framework

Base Class Library (BCL) je knihovna obsahující nejčastější pomocné funkce – práci se soubory, třídění, diagnostiku, síťovou komunikaci apod. ADO.NET je knihovna pro prácis daty s možností jejich XML reprezentace. Dále jsou na obrázku dvě knihovny pro vývoj uživatelského rozhraní – Windows Forms pro desktopové aplikace a ASP.NET pro webové uživatelské rozhraní [5.] .

Velmi důležitou částí .NET frameworku jsou podporované jazyky. .NET framework je jazykově nezávislý, pro libovolnou úlohu lze v podstatě použít jakýkoliv z podporovaných jazyků. Asi nejčastěji používanými jazyk jsou C# a Visual Basic.NET, které jsou s dílny Microsoft.

Je úplně jedno, který jazyk si vybereme, výsledek by měl být stejný, pokud jde o funkčnost i výkonnost.

4.1.2 Operační systém a Framework

.NET Framework je primárně určen pro majitele operačního systému Windows, vyžaduje se minimálně verze Windows 98. Jedná se komponentu, která se do systému doinstaluje. Komponenta se dá získat například pomocí služby Windows Update, nebo stáhnout ze stánek firmy Microsoft.

K dispozici je i verze .NET Compact Framework (.NET CF) pro Pocket PC s operačním systémem Windows Mobile, která je s „klasickou“ verzí kompatibilní a tak není nutné kompilovat různé aplikace pro PC a PDA - na obou systémech bude aplikace

fungovat stejně. Jak již název napovídá, .NET CF neobsahuje všechny objekty, funkce a metody z původního .NET Frameworku

GNU obdoba .NET se nazývá DotGNU; její část nazývaná DotGNU Portable.NET umožňuje spouštět všechny .NET aplikace na unixových platformách (Linuxu, BSD, Mac OS X, Solarisu, AIX) a dokonce pomocí nástrojů Cygwin a Mingw32 i na Windows.

V prostředí operačních systémů Linux, UNIX, Mac OS X je ovšem k dispozici i sada nástrojů kompatibilní přímo s Microsoft .NET pod názvem Mono. Tuto sadu nástrojů však nevyvíjí firma Microsoft, ale v obvyklém duchu opensource skupina dobrovolných vývojářů.

4.2 ASP .NET 2.0

ASP – Active Serve Page – jedná se o stránky vykonávané na straně serveru. V současné době se používá verze 2.0. Z pohledu komunikace mezi klientem a serverem je funguje ASP .NET stejně jako PHP. Požadavky na zobrazení webu jsou poslané na server, ten posléze vygeneruje kód HTML, který potom zašle klientovi. Když se na problém podíváme podrobněji, najdeme odlišnosti od typického serverového skriptovacího jazyka, protože ASP .NET a v něm vytvořené stránky se označují jako webové aplikace. Po přijetí požadavku od klienta server zkontroluje požadovanou aplikaci a vytvoří relaci pro práci aplikací. Aplikací se dá jednoduše naprogramovat, co má v daném okamžiku provádět. Vytvoří potřebnou flexibilitu a úplnou kontrolu na tím, co se s aplikací děje.

Prostředí pro vývoj ASP .NET aplikací poskytuje programátoru mnoho zjednodušení. Základním principem je technologie formulářů. Kde je celý projekt ASP .NET postavený na formulářích a komunikace prostřednictvím nich. Formulář potom obsahuje jednotlivé ovládací prvky a ty mají své události. K události prvku můžeme přidat jednoduše kód, který se má provést při vyvolání události. Vznikem událostí, například stiskem tlačítka, se provede odeslání požadavku na server a ten pak vrátí výsledek v podobě HTML.

4.2.1 Výhody ASP.NET oproti předchozí verzi ASP

- Díky kompilovanému kódu běží aplikace rychleji a více chyb je zachyceno už při vývoji
- Uživatelsky definované ovládací prvky lze použít jako šablony, čímž se významně redukuje duplicitní kód
- Podobný přístup jako k aplikacím pro Windows zjednodušuje přechod od jednoho prostředí k druhému
- Bohatý výběr ovládacích prvků a knihoven tříd velmi zrychluje vývoj aplikací
- Programátoři mají na výběr velké množství programovacích jazyků
- Schopnost cachovat celou stránku nebo pouze její části podstatně zvyšuje výkon serveru
- Lze jej provozovat na různých operačních systémech i webových serverech, např. IIS (Windows), Apache (Windows, Linux s open source implementací .NETu Monem)
- Počínaje verzí 2 generuje ASP.NET validní HTML 4.0 / XHTML 1.0 / XHTML 1.1 kód a JavaScript

4.2.2 Stavové prostředí nad bezstavovým protokolem

Ačkoliv webový protokol HTTP je sám o sobě bezstavový, událostmi řízené programování zachování stavu (uchování kontextu mezi jednotlivými požadavky) vyžaduje. ASP.NET tento problém řeší kombinací HTML a JavaScriptu pomocí dvou základních technik:

ViewState uchovává informace mezi postbacky (opakovaným odesíláním formuláře na server) v zakódovaném tvaru ve skrytých formulářových polích. Jeho výhodou je, že využívá pouze HTML a nevyžaduje žádnou speciální podporu na straně serveru ani klienta. Nevýhodou je, že se mezi serverem a klientem přenáší větší objem dat, zejména je-li ViewState využíváno nesprávně.

Session State oproti tomu ukládá veškeré informace na straně serveru a předává (typicky jako cookie nebo součást URL) pouze jednoznačný identifikátor. To sice zmenšuje objem přenášených dat, ale klade vyšší nároky na výkon serveru. Pokud se sessions

používají nesprávně, může být server náchylný i k Denial of Service útokům. Oproti ASP umožňuje ASP.NET ukládání session state do samostatného procesu nebo na SQL server. To zjednodušuje použití session we webových farmách, zvyšuje výkon a umožňuje stav zachovat i při restartu serveru.

4.3 Ukázky kódu asp

Na tom to místě chci ukázat některé možnosti a schopnosti jazyka ASP 2.0, které byly použity při návrhu programu.

4.3.1 C# nebo Visual Basic

Uživatel má při psaní webové aplikace na výběr ze dvou podporovaných programovacích jazyků. Oba dva představují stejné výhody a je vlastně jedno, který si z nich programátor vybere. Rozdíl je pouze v použité syntaxi zápisu jednotlivých příkazů. Dokonce lze oba dva jazyky ve webové aplikaci kombinovat a to tak, že část stránek mohu programovat v jazyce C# a další část programovat v jazyce Visual Basic. Oba dva jazyky nelze použít v rámci jedné stránky. Použitý jazyk je rozlišen v direktivě stránky a to následovně:

<% Page Language = "VB" %>

4.3.2 Dva oddělené přístupy psaní kódu

ASP .NET umožňuje uživateli na výběr zda bude ASP kód psát přímo do kódu HTML tak jak byl zvyklí u předchozích verzí ASP, tomuto přístupu se říká Code Besidea, v takovém případě je zdrojový kód vepsán mezi značky <% %>. Nebo kód bude umístěný v samostatném souboru, „za kódem“ HTML – Code Behind, což uživateli umožňuje striktní oddělení aplikačního kódu od vzhledu a grafického návrhu. Fyzicky to potom vypadá tak, že vpodstatě máme dva soubory, kde jeden má příponu *.aspx* a obsahuje kód HTML, druhý má příponu .vb a obsahuje zdrojový kód jazyka Visual Basic nebo příponu .cs a je použit jazyk C#.

V obou případech je výsledek a funkčnost stejná. Podle mého osobního názoru je druhý přístup mnohem lepší, a také jsme ho při vývoji aplikace použil.

4.3.3 Stavební články webové aplikace

Jsou to jednoduché ovládací prvky, známé například z internetových stránek, které obsahuje formulář nebo z klasických desktopových aplikací. Takovými ovládacími prvky jsou např. tlačítka, textová okna, popisky, listboxy, atd.

Tyto stavební prvky aplikace jsou označovány jako serverové ovládací prvky. Tvoří základní pilíř pro komunikaci s uživatele s aplikací. Takový ovládací prvek obsahuje klauzuli runat = "server", tím zabezpečíme, že se kód vykoná na straně serveru. Serverové prvky nám v principu umožní programovou manipulaci s elementy HTLM. V podstatě se jedná o objekty, které mají definovány své vlastnosti, metody a události. Vlastnosti prvků určují např. velikost, vzhled atd. Metody říkají, co se má s daným prvkem udělat – třeba vymaž svůj obsah. Událostí může být například kliknutí uživatele, načtení stránky apod.

Příklad zápisu serverového ovládacího prvku Tlačítko v kódu ASP.

```
<asp:LinkButton ID="linkNew" runat="server" Width="66px">Nový</asp:LinkButton>
```

Ukázka části kódu události click v programovacím jazyce Visual Basic:

```
Protected Sub linkNew_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles linkNew.Click
    Me.teRC.Enabled = True
    Cisti()
End Sub
```

4.3.3.1 Ovládací prvek PlaceHolder

Na tomto ovládacím prvku si můžeme ukázat pravou sílu ASP a pomocí něj tvořených aplikací. Jedná se takový kontejner pro ukládaní jednotlivých elementů webové stránky. Umožňuje takto nashromážděné ovládací prvky zobrazit na daném místě stránky.

To, co chci pomocí přikladu demonstrovat představuje dynamické vytvoření ovládacích prvků a zobrazení prvků na stránce jedním kliknutím na tlačítko formuláře.

```
Protected Sub Napln_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles linkNew.Click
    Dim Jmeno as new TextBox
    Dim Prijmeni as New TextBox
    Jmeno.Text = "Karel"
    Prijmeni.Text = "Novotný"
    PlaceHolder1.controls.add(jmeno)
    PlaceHolder1.controls.add(Prijmeni)
End Sub
```

Na příkladě je vidět, že ovládací prvky můžeme dynamicky vytvářet, nastavovat jim jejich vlastnosti a jednoduše přidávat do kontejnerů.

4.3.4 Předávání hodnot z předchozí stránky

Možnosti, jak předávat hodnoty mezi dvěma stránkami je více. Ten, kdo někdy vytvářel formuláře pomocí PHP, jistě zná metody pro posílání hodnot POST a GET. Tyto metody jsou do jisté míry nepraktické, zvláště metoda POST, kde je přímo v drese načítané stránky vidět seznam proměnných a jejich hodnoty. Takovým to způsobem, kdybychom předávali citlivá data, jenom bychom tím nahráli útočníkům a naše aplikace by se nedala označit jako „bezpečná aplikace".

ASP :NET tento problém řeší velice jednoduše, a to tím způsobem, že u předchozí stránky, z které chceme čerpat data, vytvoříme property – odkazy na vnitřní data objektu, které nám potřebná data získají. Princip je použit z objektově orientovaného objevování. V podstatě jde o analogii získávaní vnitřních hodnot objektu.

Příklad takovéto property:

```
Public ReadOnly Property pMistnost() As String
    Get
        Return Me.ddlMistnosti.SelectedValue
    End Get
End Property
```

Název property je pMistnost a odkazuje na ovládací prvek DropDownList a vrací z něj aktuálně vybranou položku.

Získání hodnoty z předchozí stránky provedeme pomocí příkazu PreviousPage, kde se jednoduše odkážeme na jméno property pMistnost.

```
If PreviousPage IsNot Nothing Then
  If PreviousPage.IsCrossPagePostBack = True Then
      Mistnost = Me.PreviousPage.pMistnost
  End If
End If
```

Přiklad ještě obsahuje dvě jednoduché podmínky, první představuje test, zda vůbec existuje předchozí stránka, a druhá podmínka říká, zda byl předán PostBack na tuto stránku.

5 REALIZACE DATOVÉHO MODELU

S orientací na .NET a technologii ASP 2.0 jsem za datové skladiště zvolil SQL Server. Jedná se o verzi Express, která je firmou Microsoft nabízen pro nekomerční projekty zdarma. Pro jednoduché ověření funkčnosti bude zcela vyhovovat.

SQL Server nabízí jednoduchou tvorbu databází, jejich správu, tvorbu tabulek, správu uložených dat, tvorbu a údržbu tabulek.

5.1 ER-Diagram Elekronické třídní knihy

Celý datový model aplikace je popsán na obrázku 13 v podobě ER-Diagramu. ER-Digram znázorňuje jednotlivé tabulky, jejich atributy a vzájemné vazby mezi tabulkami.

Na tom to digramu se pokusím nastínit problematiku ukládání dat Elektotronické třídní knihy. Celá databáze je tvořena několika tabulkami, které představují jednotlivé celky. Tyto celky jsou navzájem propojeny a tvoří ucelenou datovou základnu pro celou aplikaci.

Z pohledu datového modelu je nejdůležitější tabulka rozvrh, která tvoří důležitý obsah pro další tabulky. Ze vztahu s vyučujícím má za úkol zachytit informace o tom kdo, kdy, co učí a v jaké místnosti.

Další poměrně klíčovou tabulkou je *SkolniRok*. Její funkce je nezastupitelná z časového pohledu, a to proto, že umožňuje celý systém znovu použít každý rok znovu a znovu, jen jednoduchou klíčového atributu *Rok*.

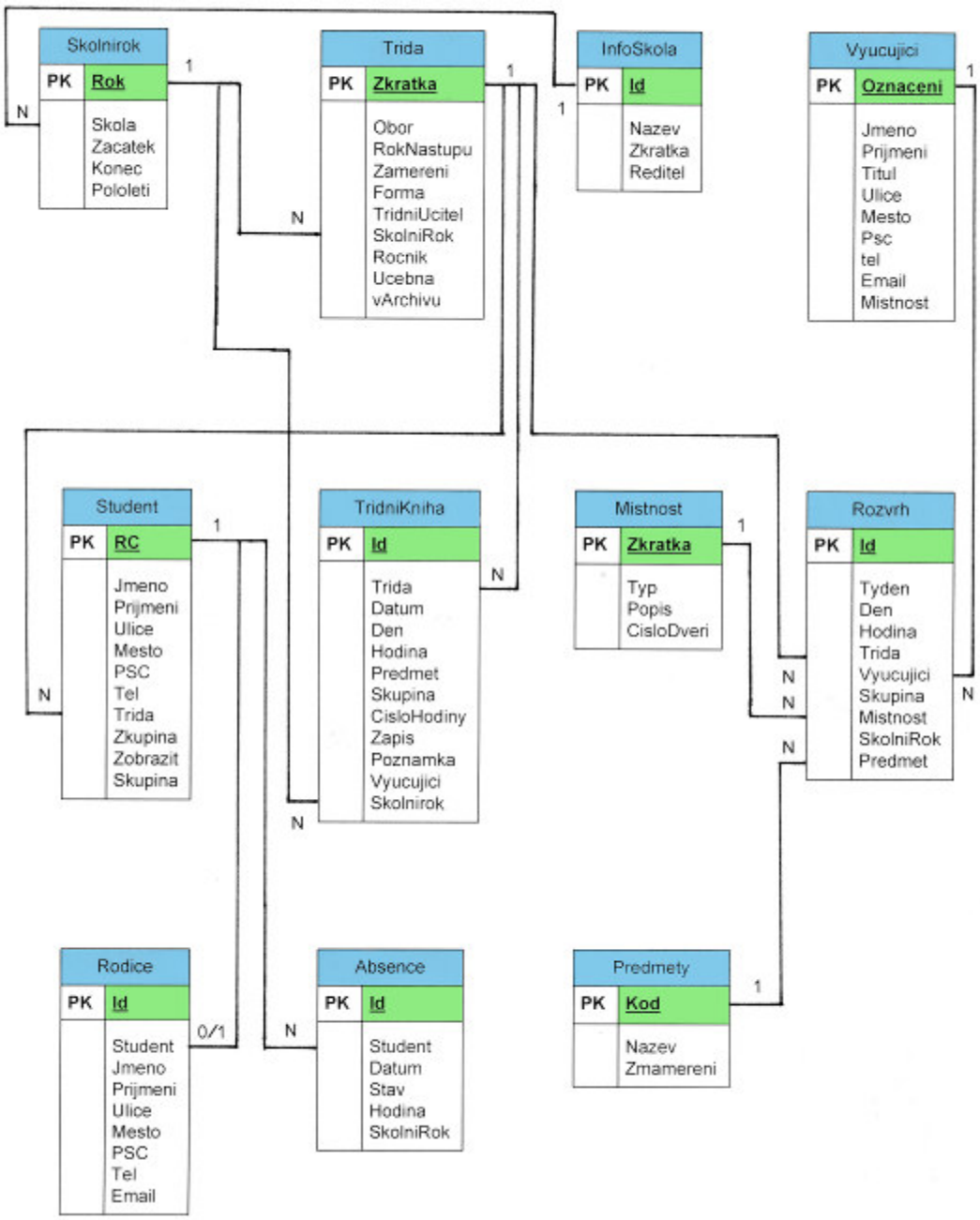
Tabulka InfoSkola, jak již název napovídá má jen informační charakter. Je do systému dodána z důvodu změny názvu školského zařízení (což se na naší škole stalo už několikrát). Dlouho jsem uvažoval, že tuto tabulku nahradím pouze XML souborem, protože ve skutečnosti obsahuje a vlastně bude stále obsahovat jen jeden řádek, ale nakonec jsem došel k názoru, že jako o tabulce v databázi k ní bude jednodušší přístup, a navíc nebudu muset řešit další technické problémy, které s tím souvisejí.

Pro uchování informací o odučených hodin slouží tabulka *TridniKniha.* Zachycuje takové informace, jako například který den, jakou vyučovací hodinu kdo a co učil.

Informace o třídě jsou uloženy v tabulce *Trida*. Ta má funkci zachytit data o jednotlivých třídách, zkratku třídy, obor, rok nástupu, formu studia, ročník. K této tabulce je asociována tabulka Studenti.

Data o studentovi jsou uložena v tabulce Studenti, obsahuje základní informace o studentovi jeho bydlišti a také to tom, jakou třídu navštěvuje. Na tuto tabulku se váží další dvě tabulky, které jsou vytvořeny proto, aby v nich byly uloženy informace o rodičích studenta a jeho absenci.

Tabulka Absence slouží pro ukládání informací o absenci studenta. Zachycuje který den, jakou vyučovací hodinu absence vznikla. Atribut stav vypovídá zda je absence omluvená či nikoliv.



Obr. 13. ER-Diagram

6 POPIS PROGRAMU

Nyní se dostáváme k samotné aplikaci. Součástí této kapitoly bude ukázat některé funkcí s jejich popisem. Ještě než se dáme do samotného popisu, přidám pár vět ohledně spuštění aplikace.

6.1 Spuštění aplikace

Na začátku je nutné upozornit na okolnost, kterou je nutné provést před spuštěním aplikace, a tak vytvořit určité podmínky pro aplikaci a její chod. Potřebné informace naleznete v kapitole 7 nazvané *„Minimální instalace“*. Zdrojový kód se samostatnou aplikací naleznete na přiloženém CD.

Pokud byly vytvořeny potřebné podmínky, byla korektně provedena instalace, můžeme provést spuštění aplikace. Jelikož se jedná o webovou aplikaci, bude nutné nejprve spustit některý z internetových prohlížečů, jako je Internet explorer nebo FireFox. Do adresy prohlížeče je nutné zadat jméno serveru například:

http://duokmotr/cb/default.apsx

Slovo *duokmotr* – představuje název serveru, *cb* – jméno virtuálního adresáře. Vyraz *default.aspx* je názvem úvodní stránky aplikace.

Pokud jsou údaje adresy vyplněny správně, měla by se v internetovém okně objevit úvodní stránka aplikace (obr. 14).

6.2 Úvodní stránka aplikace

Celá aplikace je rozdělena do několika částí, podle potřeb uživatele. Podle specifikace požadavků, popsané v kapitole 3.1.1 je plánováno, že návštěvníky budou tvořit celkem čtyři skupiny osob (*admin, vučující, studenti, rodiče*). Těmto potřebám je přizpůsobena celá funkčnost aplikace. Různé části jsou zabezpečeny a pouze vybraní uživatelé mají k nim přístup. Na druhé straně i nepřihlášení uživatelé si mohou prohlížet rozvrhy jednotlivých tříd.

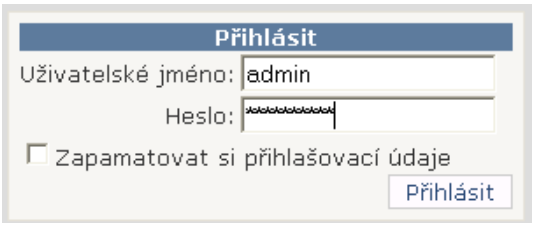
Po spuštění aplikace se uživateli zobrazí úvodní stránka (obr. 14), obsahuje celkem tři samostatné částí:

1. **Horní pruh** – má informativní charakter, uživatel má přehled o názvu školy, o aktuálním školním roce a v pravé části o stavu přihlášení.
2. **Menu** – slouží jako navigace celou aplikací.
3. **Tělo aplikace** – zobrazuje obsah dané stránky.

Obr. 14. Úvodní obrazovka aplikace

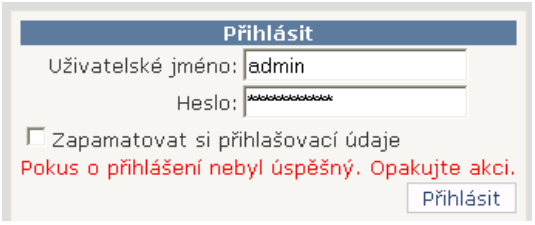
6.3 Přihlášení do systému

Přihlášení je možné více způsoby. A to buď kliknutím v Horním pruhu na odkaz *Přihlásit,* anebo kliknout v menu na některý pododkazů záložky Správa. V obou dvou případech je efekt stejný. Dojde k otevření stránky *../prihlaseni.aspx* s dialogem přihlášení (obr. 15).

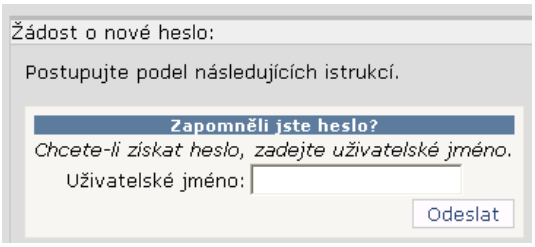


Obr. 15. Přihlášení do systému

Návštěvník nejprve musí vyplnit jméno uživatele a jeho heslo. Pokud je provedena autorizace, korektně se zobrazí v horním pruhu jméno přihlášeného uživatele, v našem případě jméno: *admin*, heslo: *admin.kolo*. Pokud naopak nedojde k ověření uživatelského jména nebo hesla, zobrazí se informace o této skutečnosti (obr. 16). Návštěvník má možnost celou operaci zopakovat nebo vyplnit formulář pro odeslání nového hesla (obr. 17) na Email. Žadatel o nové heslo musí zadat Email a odpověď na kontrolní otázku.



Obr. 16. Chybné přihlášení



Obr. 17. Žádost o nové heslo

V následujících kapitolách se budeme věnovat nastavení programu a vkládání dat.

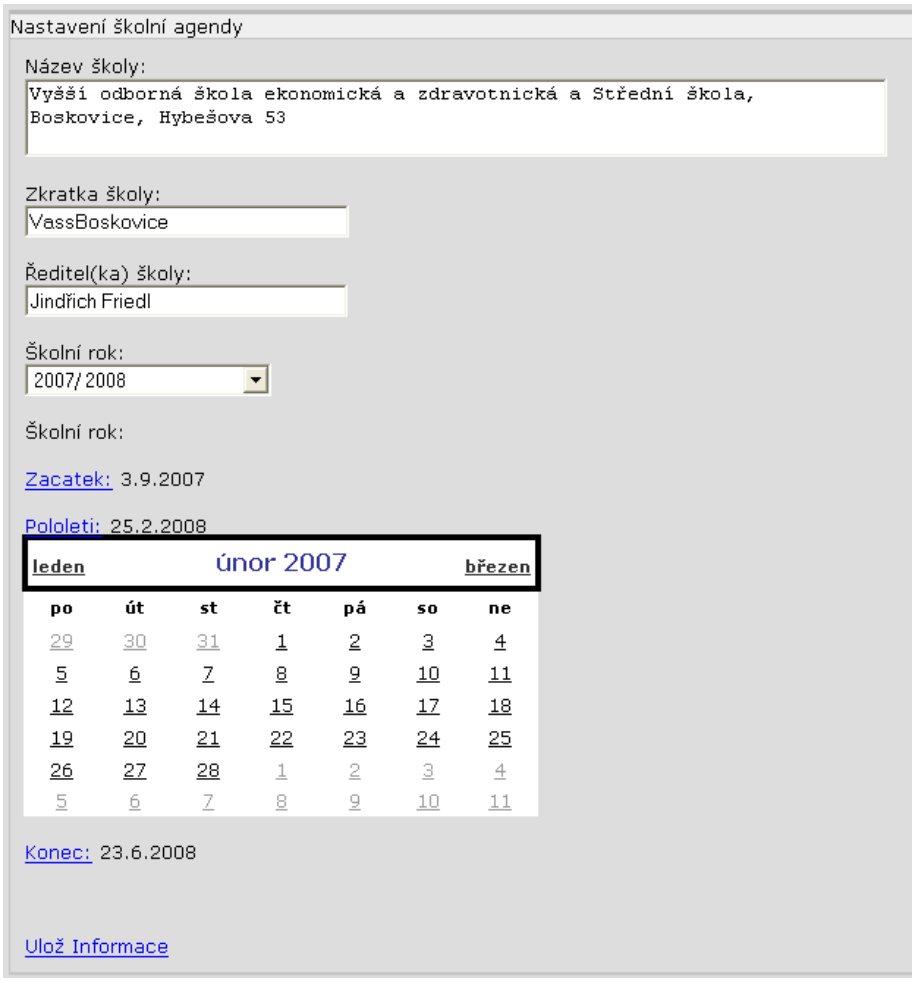
6.4 Administrátorské činnosti

Nejprve je pro administrátorskou práci nutné se do systému přihlásit. Proces přihlášení je popsán v přecházející kapitole.

Administrátorské činnosti jsou úkony související se správou a údržbou systému. Uživatel *admin* má v systému neomezenou pravomoc a z důvodu zabezpečení, správy systému by měl být v celém systému pouze jeden uživatel s takovými právy.

6.4.1 Nastavení školního roku

Položka se nachází v položce menu – *Správa, Agenda školy*. Odkaz slouží pro zadání jednotlivých informací o školském zařízení, jako je název, zkratka, jméno ředitele, k definici aktuálního školního roku a jeho parametrů (orb. 18).



Obr. 18. Nastavení informací o škole

Tento proces je velice důležitý, nastavuje celý program s pohledu času. Nastavení uložíme volbou *Uložit informace*.

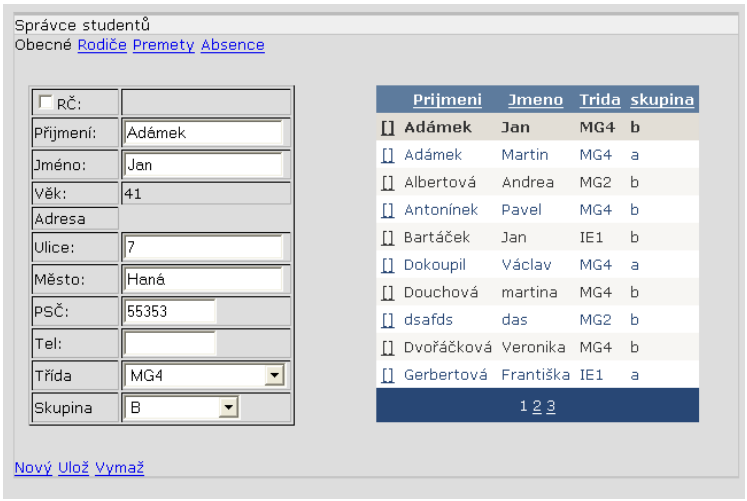
6.4.2 Evidence studentů

Evidenci nalezneme v menu *Správa, Agenda školy – Studenti*. Jedná se o formulář pro správu studentů (obr. 19). Zde možné studenty, editovat jejich data případně studenta odstranit.

Formulář obsahuje celkem čtyři záložky:

- **obecné** – obecné data o studentovi jako je jméno, příjmení, adresa, označení třídy, kterou navštěvuje,

- **rodiče** – informace o rodičích,
- **předmět** – pokud je student zařazen do třídy, zobrazí se v této záložce seznam předmětů,
- **absence** – informace o docházce studenta.



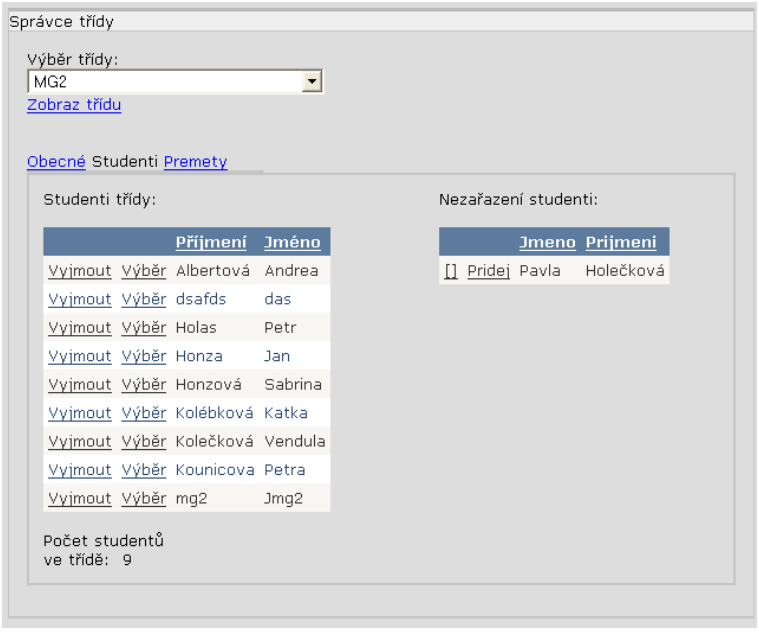
Obr. 19. Formulář „studenti“

6.4.3 Třídy

Složí pro správu tříd školy. Umožňuje základní editační operace, přidávaní a také odebíraní studentů do třídy (obr. 20). Eviduje základní informace o třídě, označení, obor, forma studia, třídní učitel.

Pomocí položky: *Odstranit do archívu.* Je možné uschovat třídu do archivu pro další budoucí práci. Posléze třídu opět obnovit z archivu.

Formulář *Třídy* obsahu je následující záložky: obecné, studenti, předměty.



Obr. 20. Formulář „Třídy“, záložka „studenti“

6.4.4 Předměty

Odkaz v menu slouží pro vytváření, editaci předmětů, které se na škole učí. Umožňuje zaznamenat: zkratku, popis a typ předmětu.

6.4.5 Místnosti

Definuje místnosti, v rámci školského zařízení. Evidují se informace jako, je: zkratka, popis, typ místnosti, číslo dveří.

6.4.6 Zaměstnanci

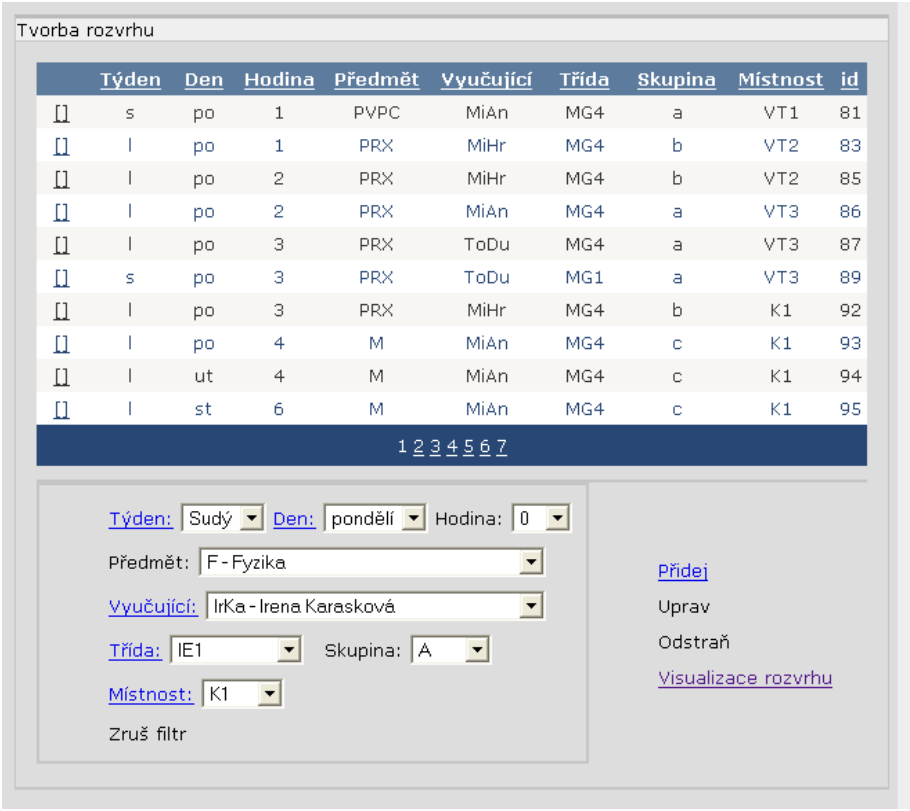
Jedná se o správu zaměstnanců školy, zejména pak učitelů. Zaznamenávají se základní informace: jméno, příjmení zaměstnance, adresa, kontakt.

6.4.7 Uživatelské účty

Složí pro správu účtů systému. Formulář umožňuje vytváření nových účtů, jejich mazaní, editaci nebo zablokování účtu. Tato funkce je výhodná zejména pokud správce potřebuje provést údržbu, zálohu systému.

6.4.8 Správa rozvrhu

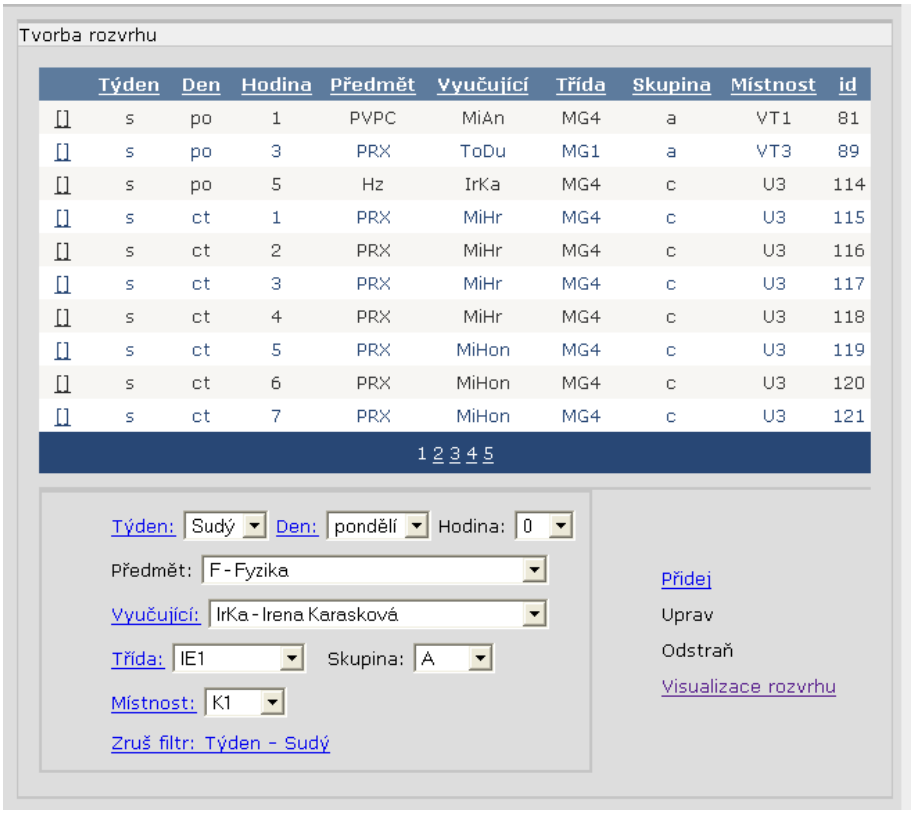
Správa rozvrhu (obr. 21) slouží pro vytváření rozvrhu celého školského zařízení. Definuje na jedné straně rozvrh třídy, třídy, vyučující a tak i obsazenost místností.



Obr. 21. Formulář „Správa rozvrhu“

Celé hlavní okno je rozděleno do tří částí, tabulka slouží pro výpis jednotlivých záznamů rozvrhu. Vždy jeden řádek v tabulce představuje jedno „políčko rozvrhu“. Symbol „[]“ slouží pro výběr řádku tabulky. Takto vybraný řádek lze editovat. Tabulka umožňuje také třídění, a to kliknutím na jméno sloupce v záhlaví tabulky.

Zadávací část formuláře má více funkcí. Jednak definování nových záznamů rozvrhu, editování vybraného záznamů, ale i selekci obsahu tabulky. Pomocí formuláře se provede natavení potřebných informací a posléze se provede patřičná operace Přidání nebo upravení vybraného záznamu. Selekce se provádí tak, že se nejprve provede výběr selektivní informace, např. vybereme sudý týden kliknutím na modrou popisku *Týden*. Výsledkem pak je to, že v tabulce se zobrazí pouze záznamy o v sudém týdnu (obr. 20). Pokud chceme filtr zrušit, vybereme volku „*Zruš filtr*“ ve spodní části okna. Vždy se dá filtrovat pouze podle jedné filtrovací informace.



Obr. 22. Ukázka selekce podle týdne

Ovládací číst slouží pro přidávání, úpravu, odstranění jednotlivých záznamů. Pro editaci nebo úpravu musí být vybrán záznam.

Tlačítko *Vizualizace rozvrhu* slouží k zobrazení rozvrhu podle zadaných kritérií, tuto funkci si popíšeme v kapitole 6.7.

6.5 Třídní kniha

Pracovat s třídní knihou mohou pouze přihlášení uživatelé, kteří náleží do role – *vyučující*.

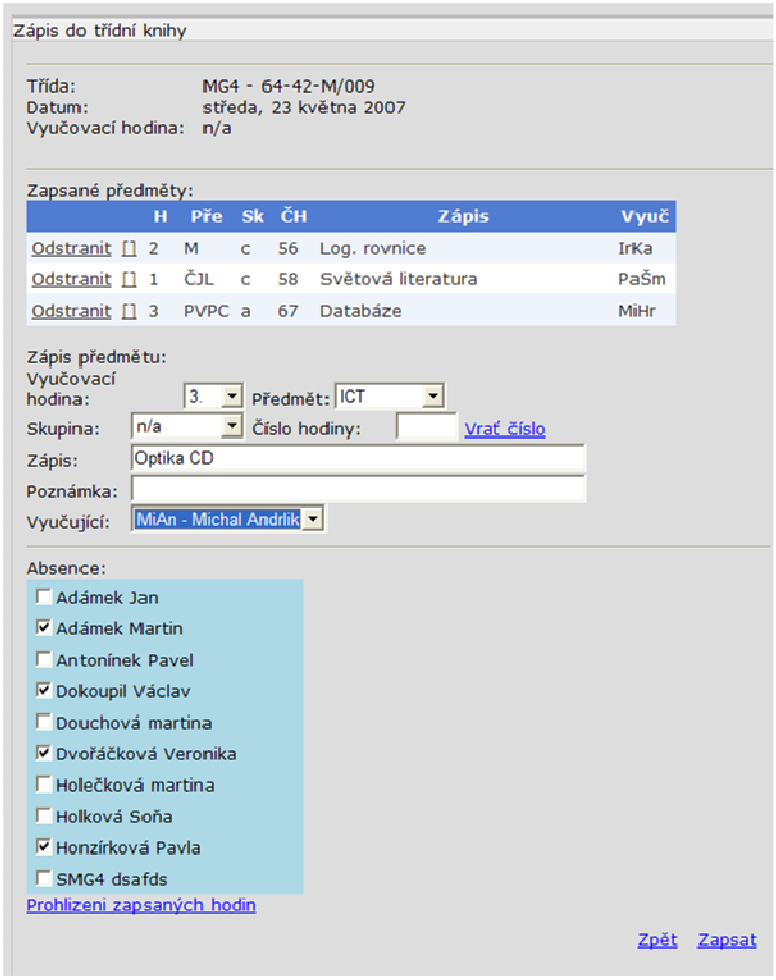
6.5.1 Zápis do třídní knihy

Zápis se provádí pomocí formuláře *Zápis*, který najdeme v menu, sekce *Třídní kniha.*

Zápis se provádí v několika krocích.

1. Výběr třídy.
2. Výběr data pro zápis.
3. Výběr vyučovací hodiny z rozvrhu třídy – uživatel může tento krok přeskočit.
4. Provedení vlastního zápisu – tento krok se skládá z dvou úkonů (obr. 21):
    1. provedení zápisu hodiny,
    2. výběru studentů, kteří se nezúčastnili vyučovací hodiny.

Mezi jednotlivými kroky se přechází pomocí tlačítek *zpět* a *další.* Tlačítkem *Vrať číslo* provede uživateli zobrazení čísla hodiny pro zápis nové vyučovací hodiny. Celý proces zápisu se provede tlačítkem *zapsat*. Vyučující má možnost zobrazit si zapsané hodiny a absenci studentů.



Obr. 23. Zápis do třídní knihy

6.6 Kontrola absence

Rodiče mohou pomocí internetového formuláře kontrovat, docházkou svého dítěte. Správce systému má možnost vygenerovat rodiči, na jeho žádost, číselný kód. Číselný kód, spolu s rodným číslem, slouží k prohlížení školní docházky svého dítěte.

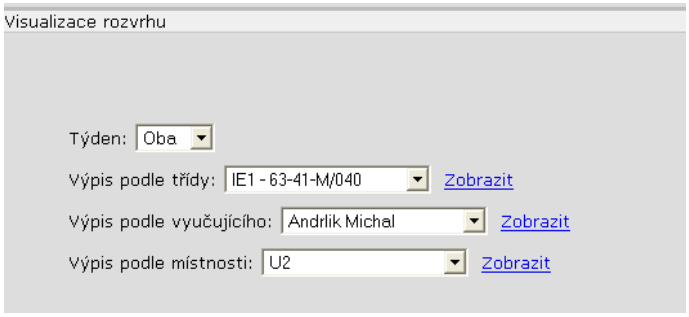
6.7 Vizualizace rozvrhu

Umožňuje zobrazení rozvrhu podle následujících kritérií (obr. 24):

- *Týden* – položka slouží k výběru, zde chcete zobrazit jen sudý, lichý nebo obě dvě možnosti,

- *vyučujícího* – volba pro zobrazení rozvrhu vybraného učitele,
- *třídy* – pro zobrazení rozvrhu konkrétní třídy,
- *místnosti* – výběr pro zobrazení rozvrhu učeben.

Funkce je přístupná pro každého návštěvníka, webu přímo z menu, nebo také ze Správy rozvrhu. Pro zobrazení daného rozvrhu slouží volba *Zobrazit.* Z informací se dá jednou duše vyčíst základní informace, kde, kdo učí, jaký předmět a v jaké místnosti.



Obr. 24 Výběr pro zobrazení rozvrhu.

Jako příklad jsem vybrat zobrazení rozvrhu v sudém týdnu třídy MG4 (obr. 25). Výsledná stránka zároveň slouží jako tiskový výstup, který jde jednoduše přímo vytisknou z internetového prohlížeče.

Třída: IE1 - 63-41-M/040

Sudý týden

	0.h	1.h	2.h	3.h	4.h	5.h	6.h	7.h	8.h	9.h	10.h	11.h	h/den
po	- - -	Ze-c IrKa U5	M-c IrKa U4	M-c IrKa U4	M-c MiHo U4	M-c MiHon U4	F-c SvKu K1	M-c MiHo U4	- - -	- - -	- - -	- - -	7
út	- - -	Hz-c SvKu U5	M-c MiHo U4	TeVy-c ToDu U4	TeVy-c ToDu U4	Ze-c IrKa U5	F-c SvKu K1	F-c SvKu K1	- - -	- - -	- - -	- - -	7
st	- - -	Hz-c SvKu U5	F-c MiHo U4	M-c MiHo U4	MPS-c ToDu U4	Ze-c IrKa U5	F-c MiHo U4	M-c MiAn U4	F-c SvKu K1	- - -	- - -	- - -	8
čt	- - -	TeVy-c ToDu U4	TeVy-c ToDu U4	MPS-c ToDu U4	MPS-c ToDu U4	M-c MiHo U4	- - -	- - -	- - -	- - -	- - -	- - -	5
pá	- - -	PRX ToDu U5 PRX ZdMi VT1	PRX ToDu U5 PRX ZdMi VT1	PRX ToDu U5 PRX ZdMi VT1	Ú-c MiHon U4	Ú-c MiHon U4	- - -	- - -	- - -	- - -	- - -	- - -	5

Obr. 25 Ukázka rozvrhu třídy IE v sudém týdnu.

7 MINIMÁLNÍ INSTLACE A DMINISTRACE WEBOVÉHO SÍDLA

Aplikace ASP.NET vyžadují pro svůj bezproblémový chod jakési zázemí. Právě tím to zázemím je správně nakonfigurovaný operační systém s technologickou platformu Microsoft .Net Framework alespoň ve verzi 2.0, popřípadě vyšší. Jako operační systém pro chod jednoduchých aplikací poslouží asi nejpoužívanější operační systém od firmy Microsoft a to Windows XP. Proto se v této kapitole budu věnovat právě tomuto systému.

7.1 Výběr operačního systému

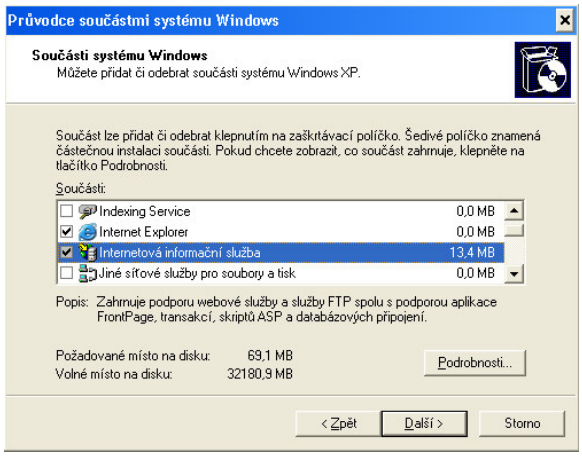
Technologie ASP.NET 2.0 je přes technologickou platformu .NET Framework úzce svázaný právě s operačním systémem Windows. Všechny operační systémy Microsoft se dají rozdělit do dvou skupin podle toho, pro jakou práci jsou primárně určeny. První skupinou jsou Windows určené pro hry a domácnost – jedná se o verze série Windows 95, 98, Millenium, Windows XP Home. Druhou skupinou jsou Windows určené pro podnikovou sféru. Sem patří klientské verze Windows (Windows 2000, Windows XP Professional) a serverové verze (Windows 2000 Server, Windows 2003). Verze postavené na starém jádře 4.0 již nemá cenu zmiňovat, protože jejich podpora byla již výrobcem ukončena a očekává se, že se již dnes nepoužívá. Převážně se bude očekávat, že aplikace ASP .NET 2.0 budou běžet na serverových verzích Windows, ale pro jejich odzkoušení a vývoj postačí i klientské verze.

7.2 Instalace webového serveru

Webová technologie ASP.NET je úzce svázaná z Windows, bude jako nutný předpoklad využití technologie serveru nebo klientského počítače právě na také platformě. Nejnovější verze serverového operačního systému Windows 2003 Server nemá implicitně instalovanou sužbu IIS – Internet Information Server. Která je nutná pro chod aplikací ASP .NET. Hlavním důvodem proč tato služba není součástí instalace serveru, je bezpečnost. Server připojený k síti se špatnou administrací může stát snadným cílem útoku. Navíc ne vždy server neslouží jako webový server. Pro nasazení na ostrém severu doporučuji administraci a zabezpečení přenechat zkušenému administrátoru. Pro lokální vývojářské počítače a testovací servery si uvedeme minimální postupy konfigurace pro bezchybný chod aplikací.

7.2.1 Instalace IIS pro Windows 2000/XP

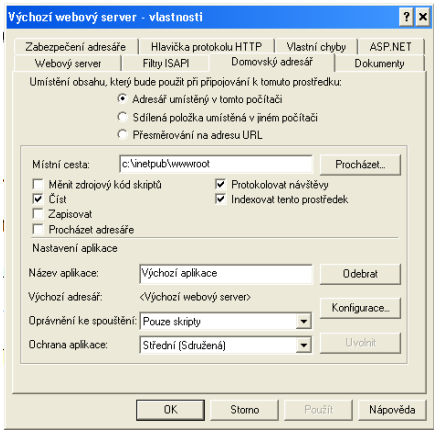
Nejprve budu popisovat inflaci služby IIS pro systémy s operačním systémem Windows 2000 a Windows XP. Pro instalaci bude potřebovat CD operačního systému. V nabídce Start vybereme volbu Nastavení → Ovládací panely → Přidat nebo odebrat programy. Zde v levé části dialogového okna vybere volbu Přidat nebo odebrat součásti operačního systému. Instalace spočívá v zaškrtnutí volby Internetová informační služba (obr. 26).



Obr. 26. Instalace služby Internetová informační služba

Veškerou nutnou konfiguraci provedeme pomocí položky v *Ovládacích panelech, Nástroje pro správu*, zde vybereme volbu Internetová informační služba. V dialogovém okně najdeme položku *Výchozí webový server*, pomocí prvého tlačítka myši vyvoláme vlastnosti pro konfiguraci webového serveru (obr. 27). V záložce Domovský adresář najdete umístění domovského adresáře. Výchozí cesta domovského adresáře je *C:\inetpub\wwwroot*. Obsah webového serveru můžeme zobrazit pomocí internetového prohlížeče a to tak, že do adresy zadáme buď URL *http://localhost* a nebo místo slova localhost může použít jméno počítače nebo IP adresu počítače.

Tím to dokončena první část instalace. Dále je nutné ještě provést registraci ASP.NET. Podrobný popis najde v kapitole – Registrace ASP.NET pro Internet Information Server.



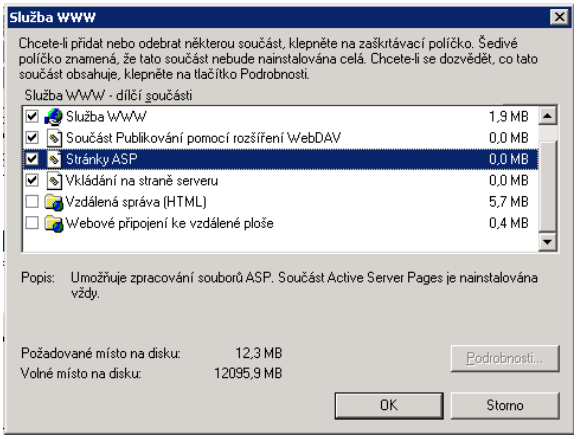
Obr. 27. Vlastnosti webového serveru

7.2.2 Instalace IIS pro Windows Server 2003

Postup instalace IIS pro Windows Server 2003 je v některých situacích trochu odlišný. Nejprve se musíme dostat k dialogovému oknu Přidat nebo Odebrat součásti systému a to provedeme stejným způsobem, jako je popsáno v předchozí kapitole při popisu Windows XP.

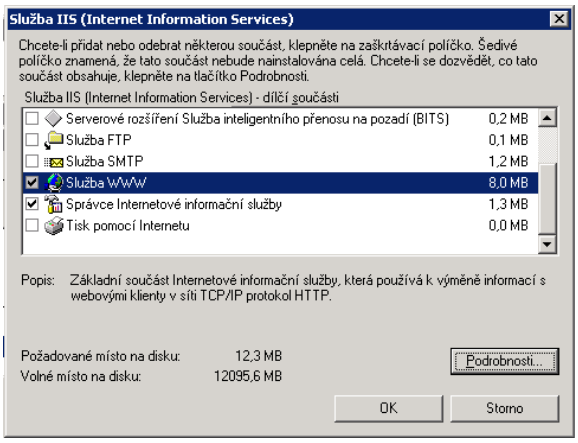
Zde nastává zmíněná drobná odlišnost. V dialogovém okně vybere položku Aplikační server a pomocí tlačítka podrobnosti vstoupíme do natavení aplikačního serveru.

V prvním mezikroku dříve než začneme samotnou instalaci IIS, povolíme sub komponentu ASP.NET (obr. 28). Až potom se vnoříme pomocí tlačítka podrobnosti do další úrovně Internet Information Sevices.



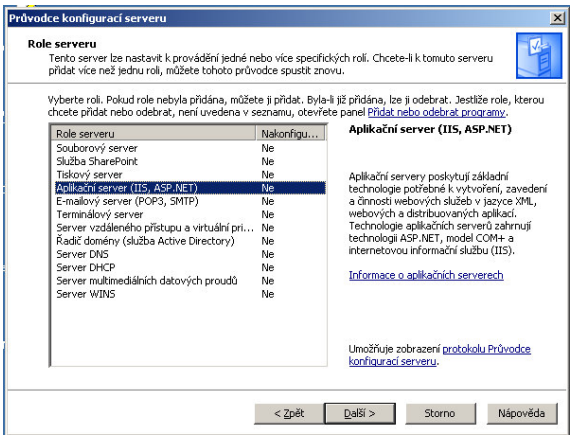
Obr. 28. Služba WWW

Na této úrovni je nutné stisknou volbu Podrobnosti a v plném rozsahu povolit službu WWW (obr. 29). Stiskem tlačítka Ok se provede instalace vybraných služeb.



Obr. 29. Služba IIS

Celý proces se dá také provést pomocí průvodce Správa serveru a to pomocí volby Přidat nebo odebrat roli. V možnostech konfigurace zatrhneme Vlastní konfigurace. Ze seznamu rolí serveru vybereme Aplikační server (IIS, ASP.NET) a v dalším kroku zaškrtneme volbu Povolit technologii ASP.NET (obr. 30).



Obr. 30. Průvodce konfigurace serveru

7.2.3 Virtuální adresáře

7.2.3.1 Pojem virtuální adresář

Virtuální adresář je popisný název (také se mu říká alias) buď pro fyzický adresář na pevném disku serveru, který není umístněn v domovském adresáři (mimo složku wwwroot), nebo adresář umístěný v jiném počítači. Aliasy je výhodné používat vzhledem k tomu, že jsou poměrně krátké a uživatel si nemusí pamatovat celu cestu k umístění, ale stačí si jen zapamatovat alias. Používání alisů je také mnohem bezpečnější, protože uživatel nemusí znát fyzické umístění dat, ale pracuje s daty pomocí konkrétního alisu. Aliasy usnadňují přemisťování adresářů v rámci webového serveru. Další výhoda je v tom, že pokud u nějakého aliasu změníme fyzické umístění a provede se jeho přemapování, nemusí se měnit URL adresa.

Virtuální adresář je třeba vytvořit také v případech, kdy chcete publikovat z kteréhokoli adresáře, který není obsažen v domovském adresáři. Příkladem může být vytváření webového serveru pro oddělení marketingu v podnikové síti intranet. V následující tabulce je uvedeno mapování mezi fyzickým umístěním souborů a adresou URL pro přístup k těmto souborům (tab. č. 1).

Tabulka 1 Příklady použití alisů a jejich fyzické umístění a URL odkaz.

Fyzické umístění	Alias	URL
C:\Inetpub\wwwroot	*domovský adresář* (žádný alias)	http://WebovyServer
\\Server2\Data_prodeje	Zakazky	http://WebovyServer/ Zakazky
D:\Inetpub\wwwroot\Objednávky	Žádné	http://WebovyServer/ Objednavky
D:\Marketing\PublicRel	PR	http://WebovyServer/PR

7.2.3.2 Vytvoření virtuálního adresáře pro aplikaci ASP.NET

Konfigurace webového serveru se provádí ve dvou úrovních:

- na úrovni webového serveru,
- na úrovni konkrétního webového sídla.

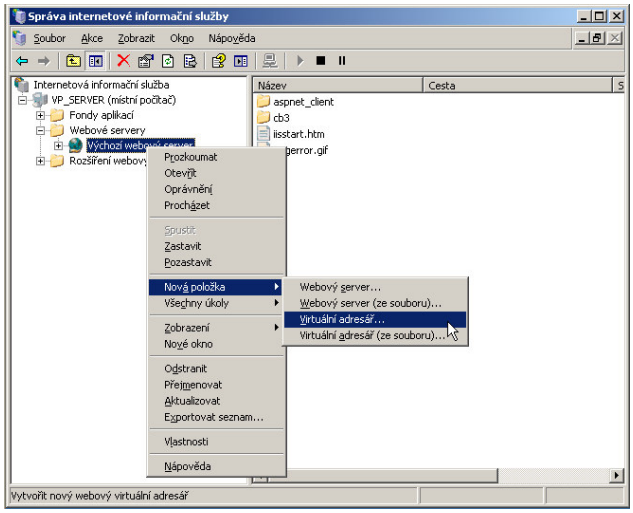
Konfiguraci na úrovni webového serveru máme již dokončenu a může se pustit do konfigurace konkrétního webového sídla.

Stránky ASP.NET a aplikační kód je uložen na serveru nebo na vývojářském počítači a to v nějakém složce uložené na disku. Tuto fyzickou složku potřebujeme namapovat pomocí virtuálního adresáře, čím dosáhneme důsledného oddělení fyzické ho souborového systému od logické souborové struktury.

Nyní si popíšeme postup vytváření a nakonfigurování virtuálního adresáře na vývojářském počítači s operačním systémem Windows XP.

Nejdříve musíme vytvořit na disku serveru nebo vývojářského počítače složku například *C:\elekronickakniha*. Na tento adresář v dalším kroku namapujeme virtuální adresář. Složka může obsahovat i podadresáře.

Virtuální adresář vytvoříme následujícím způsobem. Pomocí nabídky Start si otevřeme Ovládací panely a zde vybereme položku Nástroje pro zprávu a zvolíme Internetová informační služba. V kontextové nabídce zvolíme Nový a Virtuální adresář (obr. 31) a necháme se vést průvodcem.



Obr. 31 Správa internetové informační služby

Při vytváření virtuálního adresáře určeného pro webovou aplikaci zadáme vhodný alias, například „ekniha“. Nově vytvořený virtuální adresář následně namapujeme na fyzický adresář. V následujícím kroku můžeme nastavit přístupová práva pro virtuální adresář. Pokud vytváříme pouze cvičný virtuální adresář, zabezpečení nás až tak moc netrápí, ale pokud bychom totu operaci prováděli na ostrém webovém serveru, platí pravidlo čím méně tím lépe.

Pro každý virtuální adresář, lze jednotlivě nastavit stupeň zabezpečení. V kontextovém menu každého virtuálního adresáře nastavíme zabezpečení změnou parametru Ochrana aplikace na požadovanou hodnotu – Nízká (Proces IIS), Střední (sdružená), případně Vysoká (Izolovaná).

7.2.4 Registrace ASP :NET pro Internet information service

Máme již nainstalován a nakonfigurován webový server a technologickou platformu .NET Framework. Při instalaci .NET Frameworku v případě, že už máme nainstalovaný Internet information service, se automaticky zaregistruje engine pro stránky ASP .NET. U některých konfigurací operačního systému je nutné registrovat ASP .NET explicitně. Pokud není zaregistrovaný, tento stav diagnostikujeme pomocí hlášení „The XML page cannot be displayed“, v tomto případě je nutné provést registraci a to následujícím způsobem.

Pro registrování ASP .NET slouží nástroj aspnet_regiis.exe, který najdete v adresáři *%WindowsDir%\Microsoft.NET\Framework\vx.y.zzzz*. Kde *vx.y.zzzz* představuje verzi nainstalovaného Frameworku.

V tom adresáři spustíme program s parametrem –i:

aspnet_regiis.exe –i

Posledním krokem je výběr správné verze ASP :NET. V dialogovém okně Vlastnosti Webového serveru v záložce ASP. NET, položka parametru ASP .NET version: na verzi 2.0 [10.].

7.3 Instalace datového skladu

Pro chod aplikace je nutné aby na serveru, kde bude umístěn virtuální adresář, byl nainstalován SQL Server. Můžeme použít i verzi SQL Server Express Edition, která je pro nekomerční organizace zadarmo. Instalační balíček lze nalézt na stránkách firmy Microsoft a to na adrese *http://www.microsoft.com/downloads/*. Z této adresy je nutné nejprve stáhnout instalační balíček na disk počítače, posléze jej spustit a věnovat pozornost instalačním pokynům.

ZÁVĚR

Cílem diplomové práce bylo vytvoření elektronické třídní knihy a seznámením se s principy tvorby webových aplikací. Na základě analýzy získaných dokumentů a porovnání existujících aplikací. Jsem se zaměřil na funkčnost a uživatelský komfort ovládání aplikace, což se do jisté míry povedlo.

Během vývoje jsem celý proces rozfázoval do jednotlivých úseků, které bylo nutné řešit odděleně. Jako například přístup k databázi a práci s ní. Na těchto jednoduchých celcích jsem si ověřil funkčnost zamýšlených kroků, což bylo odrazovým můstkem pro tvorbu celé aplikace.

V textu jsou také popsány technologie pro vývoj aplikací, zejména pak ASP, která jasně určuje směr vývoje nových aplikací pro webové prostředí. Jelikož nemám přímou zkušenost s konkurenční platformou pro Linux, nejsem schopen posoudit výhody a nevýhody použití jiných technologií pro vývoj takovýchto systémů na jiné platformě než Windows, jako je třeba vývoj webových aplikací pomocí PHP. Obě technologie mají své zaryté příznivce i odpůrce, ale ve finále na tom nejvíce vydělají vývojáři, kteří mají možnost volby svého vývojového nástroje.

V závěru své práce jsem provedl ověření programu i v praxi. Tento krok sloužil k ověření aplikace a nalezení případných nedostatků. Tyto nedostatky jsem se snažil průběžně odstranit. Pro plné nasazení mé aplikace do provozu by vyžadovalo delší časový úsek ověřování v praxi. Systém nebyl nasazen na žádném z veřejných serverů, testování jsem prováděl výhradně na lokálním serveru nebo na serveru naší školy.

Systém obsahuje mnoho rezerv pro možná rozšíření o další moduly, například o kartotéku žáků a jejich klasifikaci, archív pro ukládání starých dat. Ale i přesto v této fázi je aplikace životaschopná a splňuje počáteční požadavky.

THE CONCLUSION

The objective of my senior paper is to create an electronic class register and familiarize myself with the principles of developing web applications. Based on the analysis of acquired documents and the comparison of existing applications, I focused on the functionality and user comfort in controling the application. To a great degree, I succeeded to do so.

I divided my development into several phases, which needed to be addressed separately. One such phase is the database access and working with it. These phases helped me to verify the functionality of planned steps, which was the essential start in the development of the entire application.

The text also describes technologies for application development, especially ASP, which clearly sets the way of application development nowadays. Since I don't have a profound experience with the Linux OS, I'm not able to judge the advantages and disadvantages of using other development technologies on different platforms other than Windows, for instance web application development using PHP. Both technologies have their stark supporters and opponents but the most benefit then comes to the developers, who have the luxury of choosing the most suitable development tool.

In conclusion of my paper, I tested the program in real life. This step was to test the application and detect possible errors. I have been working continuously to remove the errors. In order for my application to be put into regular use, real-life testing would require more time. The system was not installed on any public servers. I did all testing exclusively on a local server or our school server.

The system poses much room for feature expansion and new modules such as a student record database, classification of the records or archivation of old data. In spite of that, the application is very usable at the present state and it fulfills the initial requirements.

SEZNAM POUŽITÉ LITERATURY

Internetové zdroje:

[1.] WWW stránky. Informace o školských dokumentech
www.msmt.cz

[2.] WWW stránky. Bakaláři – programy pro školní administrativu.
www.bakalari.cz

[3.] WWW stránky. Škola OnLine – informace o programech
http://zakovska.skolaonline.cz

[4.] WWW stránky. Záškolaci .cz – stránky o projektu „Metla na záškoláky“
www.zaskolaci.cz

[5.] WWW stránky. Živě.cz
www.zive.cz

[6.] WWW stránky. Wikipedie – otevřená encyklopedie.
http://cs.wikipedia.org

Monografie:

[7.] LACKO, Luboslav. SQL Hotová řešení.
1. vyd. Brno: Computer Press, 2003. 298 s. ISBN 80-7226-975-5

[8.] GURTLER, Martin., KOCICH, Pavel. Visual Basic .NET Hotová řešení.
1. vyd. Brno: Computer Press, 2005. 312 s. ISBN 80-251-0367-6

[9.] ŠÍMA, František., VILÍNEK, David. Visual Studio .NET praktické programování krok za krokem.
1. vyd. Praha: Grada, 2006. 256 s. ISBN 80-247-1418-3

[10.] LACKO, Luboslav. ASP.NET a ADO.NET 2.0 Hotová řešení.
1. vyd Brno: Computer Press 2006. 385 s. ISBN 80-251-1028-1

[11.] KANISOVÁ, Hana., MULLER, Miroslav. UML srozumitelně

1. vyd Brno: Computer Press 2004. 158 s. ISBN 80-251-0231-9

[12.] KUČERA, Miroslav., PETERKA, Jiří., Programování na webu.

1. vyd Praha: iDnes 2004. 600 s. ISBN 80-865-9336-4

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASP	Active Server Pages – aktivní serverové stránky.
ČR	Česká Republika.
HTTP	Hypertext Transfer Protocol – Hypertextový transportní protokol
IIS	Internetová informační služba
IZO	Indetifikační číslo školského zařízení.
MŠMT	Ministerstvo školství mládežě a tělovýhovy.
Např.	Například.
Obr.	Obrázek.
PHP	Personal Home Page – Osobní domácí stránky .
Sb	Sbírka.
SEVT	Statistické a evidenční vydavatelství tiskopisů
SQL	Structured Query Language – Strukturovaný dotazovací jazyk.
URL	Uniform Resource Locator.
Wi-Fi	Wireless Fidelity - Označení bezdrátové sítě
WWW	Word Wide Web – Celo světová pavučina.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK