VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

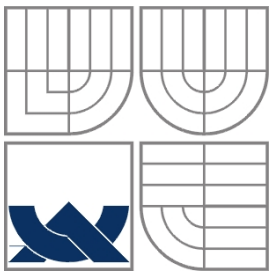
INFORMAČNÍ SYSTÉM PRO ŠKOLY S AUTOMATICKOU TVORBOU ROZVRHŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

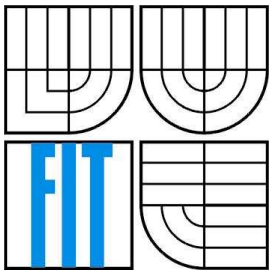
AUTOR PRÁCE
AUTHOR

BC. JIŘÍ ŠVADLENKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ŠKOLY S AUTOMATICKOU TVORBOU ROZVRHŮ

INFORMATION SYSTEM FOR SCHOOLS WITH AUTOMATIZED TIMETABLING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. JIŘÍ ŠVADLENKA
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR CHMELAŘ
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce se věnuje použitím informačního systému pro správu školní agendy. Školy jsou nuceny spravovat velké množství informací a to nejenom o svých studentech. Samotná problematika je velmi rozsáhlá a různorodá. Proto jsou uvedeny nejběžnější typy dat a požadavků škol na provoz školního informačního systému.

Součástí školního informačního systému je systém pro automatické generování rozvrhů. Nejdříve jsou definovány základní pojmy z oblasti rozvrhování, na které navazují metody a algoritmy pro řešení problému vytvoření školních rozvrhů. Školní rozvrhování je problém naplánování výuky, za určitých omezujících podmínek.

Dále se práce věnuje návrhu školního informačního systému, organizování dat v nich a řešením problémů při jeho návrhu. Navrhovaný informační systém klade důraz na jednoduchou rozšiřitelnost a širokou možnost využití. V této části práce je také uveden navrhovaný algoritmus pro řešení definovaného školního rozvrhování.

Klíčová slova

Informační systém, školní agenda, generování rozvrhů, školní rozvrhování, genetické algoritmy, barvení grafu, heuristika, lokální hledání, simulované žíhání, zakázané prohledávání, grafy, XML.

Citace

Švadlenka Jiří: Informační systém pro školy s automatickou tvorbou rozvrhů. Brno, 2008, diplomová práce, FIT VUT v Brně.

Abstract

This thesis devote itself to use of information system for school agenda administration. Schools are forced to administer big amounts of informations, not only referred to their students. Broad issue is very extensive and disparate, so the most common types of data and demands on school information system operation are stated.

The system for automatic generation of timetables is part of the school information system. At the first, basic conceptions of scheduling scope are defined and tied together with them are methods and algorithms for timetable creation problem solving. School timetabling is problem of scheduling lessons with certain limitative conditions.

Further, thesis is engaged in design of school information system, data organization in such system and solving of system design problems. Designed information system accentuates on easy expandability and wide range of usage possibilities. Also suggested algorithm for solving of defined school timetabling is stated in this part of thesis.

Keywords

Information system, automated timetabling, school timetabling, genetic algorithm, graph coloring, heuristic, local search technique, simulated annealing, tabu search, graphs, XML.

Informační systém pro školy s automatickou tvorbou rozvrhů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Chmelaře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Švadlenka
19.5. 2008

Poděkování

V této části bych rád poděkoval vedoucímu práce Ing. Petru Chmelařovi, za poskytnuté rady a věcné vedení.

Obsah

Obsah 1
1 Úvod 3
2 Školní agenda 4
2.1 Informace o studentech 4
2.2 Informace o zaměstnancích 4
2.3 Další nároky 4
2.4 Současné implementace 4
2.4.1 Bakaláři 5
2.4.2 eŠkola 5
2.4.3 iŠkola 5
2.4.4 SAS – systém agendy pro školy 6
2.4.5 Další systémy 6
3 Analýza nároků školy 7
4 Generování rozvrhů 9
4.1 Rozvrh 9
4.2 Rozvrhování 9
4.3 Školní rozvrhování 10
4.3.1 Omezení 11
4.3.2 Možnosti řešení 12
5 Požadavky na informační systém 17
5.1 Webová aplikace 17
5.2 SEO 17
5.3 Adaptabilita 18
5.4 Systémové role 18
5.5 Popis modulů a jejich funkcionalita 18
5.5.1 Evidence studentů 18
5.5.2 Evidence zaměstnanců 18
5.5.3 Seznam tříd 19
5.5.4 Seznam předmětů 19
5.5.5 Klasifikace 19
5.5.6 Rozvrhy 19
6 Grafy 20
6.1.1 Barvení grafu 21
7 XML 24

7.1.1 Základní struktura 24
7.1.2 DTD 25
7.1.3 XML schéma 27
8 Návrh informačního systému 28
8.1 Modularita 28
8.2 Architektura aplikace 29
8.2.1 Přímý model 29
8.2.2 Model s řízením 30
8.2.3 Model–View–Controller 30
8.3 Autorizace a přístupová práva 32
8.4 Řízení přístupu založené na rolích 33
8.5 Zachycení událostí 33
8.6 Návrh architektury 34
8.7 Jádro aplikace 35
8.7.1 Diagram případů užití 35
8.7.2 Konceptuální datový model 36
8.7.3 Konfigurace modulu 38
8.7.4 Validace vstupních dat 40
8.7.5 Reprezentace modulu v databázi 40
8.7.6 Zpracování výstupu 42
8.8 Návrh školního informačního systému 42
8.8.1 Případy užití 43
8.8.2 Struktura modulů 45
8.8.3 Rozvrhovací algoritmus 48
8.8.4 Grafický návrh aplikace 52
9 Implementace 54
9.1 Adresářová struktura 54
9.2 Konfigurace systému 55
9.3 Implementace jádra 55
9.4 Implementace modulů 56
9.5 Rozvrhování 56
9.6 Nároky na konfiguraci serveru 57
10 Nasazení školního informačního systému 58
10.1 Návrh na vylepšení 58
11 Závěr 59
Literatura 60

1 Úvod

Vzhledem k neustále se zvyšujícímu požadavku zpracovat a organizovat informace, je nutné tyto informace třídit, systematicky ukládat a zobrazovat. Právě k tomu slouží informační systémy. Použití takových systémů je dnes velmi časté, využití najdou prakticky všude, kde se nějakým způsobem pracuje s informacemi. Oblastmi mohou být například veřejná správa, doprava, firemní provoz nebo zdravotnictví.

Díky nástupu internetu a jeho rozšíření do škol vyvstává poptávka po elektronické správě informací ve školách. Jelikož škola pracuje s velkým množstvím dokumentů, je důležité najít nástroj, díky kterému se ulehčí školní agenda. Velmi silným nástrojem v této oblasti se uplatnily informační systémy.

Školní informační systém by měl především ulehčit práci a organizaci školní agendy. Mnohé školy již nyní využívají komerční systémy, nebo systémy, které samy implementovaly. Takovéto systémy se používají na základních, středních i vysokých školách. Systém by měl usnadňovat práci, zlepšovat dostupnost informací, urychlovat komunikaci mezi školou a studenty, školou a rodiči. Požadavek urychlení komunikace splňuje internetový provoz informačního systému. Protože je školství velmi dynamickou oblastí a rozdíly mezi jednotlivými školami ve způsobu a možnostech vyučování jsou velké, je kladen důraz na jednoduchou modifikaci a rozšiřitelnost sytému.

Používání internetového informačního systému je podmíněno alespoň základní znalostí práce s počítačem a samozřejmě přístupem k internetu. To je vyváženo možností přistupovat k systému z celého světa, rodiče mohou kontrolovat docházku a prospěch svých dětí třeba z práce, učitelé mohou vypisovat známky z testů z domova, nebo studenti mají aktuální přehled o vyučování.

Tento text se věnuje použitím informačních systémů především na základních a středních školách. V úvodní části se věnuji analýze požadavků, které jsou spojeny s problémem správy školní agendy. Jedním z požadavků školy na školní informační systém je možnost automatické tvorby rozvrhů. Oblast využití rozvrhování je velmi široká, v této práci se zabývám pouze problémem školního rozvrhování. Vytvoření školního rozvrhu je dáno požadavky, které jsou kladeny školami. V dalších kapitolách je popsán problém automatického generování rozvrhů a metody jeho řešení.

Cílem práce je analyzovat požadavky škol, pomocí kterých je vytvořen návrh školního informačního systému. Od návrhu systému se očekává navržení metodik a postupů, které se využijí při samotné implementaci informačního systému. Jedny z posledních kapitol jsou věnovány popisu implementace a řešení konkrétních problémů.

V práci se snažím používat většinou české termíny. Ale kvůli nejednoznačnosti překladu a doposud neustálené české terminologii uvádím v závorkách termín v angličtině.

2 Školní agenda

Škola je nucena pro svůj provoz spravovat velké množství informací. Některé z těchto informací jsou dány školským zákonem České republiky, jiné mají pouze statistický charakter. S množstvím informací roste požadavek na jejich archivaci a to buď v elektronické podobě nebo podobě papírové. To dává prostor vzniku systémům, které si dávají za cíl ulehčit nebo alespoň organizovat práci spojenou se školní agendou.

2.1 Informace o studentech

Nejdůležitějším úkolem je vedení informací o každém studentovi. Počínaje informacemi o jeho bydlišti, či kontaktu na jeho rodiče, přes informace o docházce až po studentův prospěch. Škola je povinna vést o studentech spoustu dat po celou dobu jeho studia, ale i po té co student školu opustil.

2.2 Informace o zaměstnancích

Pro účetní potřeby je uchovávat informace o odpracovaných hodinách každého zaměstnance. Samozřejmostí je vedení záznamů o práci učitelů, kdy se musí evidovat jaké předměty učitel učí, v jakých třídách a kdy proběhlo vyučování

2.3 Další nároky

S rostoucí potřebou uchování velkého objemu informací se uměřeně zvyšují nároky na jejich manipulaci. Ve školní agendě se dále musí počítat s řešením problémů suplování, rozvrhování, plánování zkoušek, přijímacích řízení atd.

2.4 Současné implementace

V současné době existuje již několik více či méně zdařilých projektů, zaměřených na správu školní agendy. Některé výrazně převyšují konkurenci a právě přehled jejich funkcí uvedu v dalším textu. Ve většině systémů je již samozřejmostí a požadovaným minimem:

- Vedení informací o studentech.
- Úprava rozvrhů (nikoliv tvorba).
- Správa zaměstnanců.
- Přístup rodičů ke klasifikaci svých dětí.

Vedením záznamů o studentech bývá tvořeno strukturovanými seznamy, ve kterých mají oprávnění uživatelé systému (obvykle učitelé) možnost provádět klasifikaci jednotlivých studentů. Systém pouze pro potřeby školy uchovává i soukromé informace o studentech.

Většina pokročilých projektů umožňuje manuální manipulaci se školním rozvrhem. Vytvoření nového rozvrhu je pak příliš zdlouhavé a zabere příliš mnoho hodin práce.

Obdobným způsobem, jako systémy pracují se záznamy o studentech, bývají vedeny i záznamy o zaměstnancích školy. Tyto záznamy jsou ve většině projektů pouze informativními pro ostatní uživatele, například pro vyhledání kontaktu na učitele. Ale existují i možnosti zpracování informací o odpracovaných hodinách a výstup této operace použít v jiné účetní aplikaci.

2.4.1 Bakaláři

Podle mého průzkumu a získaných statistik je tento program [1] jedním z nejpoužívanějším na základních školách u nás. Program poskytuje práci s evidencí žáků a zaměstnanců, klasifikací, tisk vysvědčení, výkazy suplování, automatické generování rozvrhů a další. Rozsah projektu je velmi široký a od svých konkurentů nabízí i modul pro evidenci školní knihovny a rozpočtů školy. Aplikace je vytvořena jako GUI pro operační systém Windows a jako webová aplikace je vedena pouze klasifikace studentů a jejich docházka.

2.4.2 eŠkola

Projekt firmy M2000 s.r.o. [2] je také velmi rozsáhlá webová aplikace, určená především pro základní a střední školy. Nabízí také některé zajímavé funkcionality jako systém nástěnek, pro každého žáka je vytvořen emailový účet či prostor pro vlastní webovou stránku. Všechny virtuální školy jsou spravovány jednou webovou aplikací, tím pracují všechny zřízené školy na jednom serveru. Na tomto serveru si může každý založit a provozovat vlastní „virtuální školu“. Systém neposkytuje možnost vytvářet rozvrhy.

2.4.3 iŠkola

Projekt společnosti Computer Media s.r.o. [3] je webový informační systém, svým rozsahem a funkcemi se velmi podobá projektu eŠkola. Systém správy školních účtů je také centralizovaný a jeho provoz je zajištěn na serveru společnosti. Zajímavými funkcemi je on-line zasílání omluvenek, sms centrum. Nejzajímavější funkcí je systém pro správu testů, pomocí jednoduchého návrhového prostředí lze vytvořit test, který pak jakýkoliv žák může vypracovat.

2.4.4 SAS – systém agendy pro školy

Dalším úspěšným informačním systémem je Systém agend pro školy [4] od firmy MP-Soft, a.s. Se zhruba 900 zakoupenými licencemi se řadí k nejpoužívanějším u nás. Klientská aplikace je nainstalována na počítače ve škole, server, který řídí celý provoz aplikace, je také zřízen v rámci školy.

2.4.5 Další systémy

Byla vyvinuta spousta školních informačních systémů, ale nemalé procento z nich nestačilo konkurenci. Jejich provoz byl zastaven nebo nebyl dále rozvíjen. Některé ze systémů, které jsou svým zpracováním kvalitní a mezi školami mají stále své zájemce uvedu níže.

- Škola OnLine
- aSc Agenda

3 Analýza nároků školy

Pro analýzu problematiky správy informací ve školách jsem se osobně setkal se zástupci tří škol, Základní škola Sladkovského Chrudim, Základní škola Chrudim, Dr. J. Malíka a Gymnázium Brno – Řečkovice. Některé další školy jsem kontaktoval pomocí elektronické pošty. Na všech školách jsem byl příjemně překvapen velmi pozitivním přístupem a ochotou mi vše vysvětlit a objasnit.

Nejpřínosnějším pro mne bylo setkání s ředitelem ZŠ Sladkovského panem Vladimírem Janečkem. Rád mi vše ukázal a vysvětlil s jakými problémy se setkávají v informačních systémech školní agendy. Tato škola již v minulosti vyzkoušela mnoho systémů a nyní používá aplikaci Bakaláři. Dle slov pana Janečka, ani tato aplikace plně nevyhovuje požadavkům, které má základní škola. Ale změnu tohoto systému v dohledné době nechystají, protože do systému Bakaláři již škola vložila nemalé finanční prostředky a zaměstnanci školy již prošli mnohými školeními.

Spokojenost ze strany školy je s částí aplikace spravující evidenci zaměstnanců a školní aktivity, především je škola spokojena s jednoduchou formou tisku. Ale problém ve škole vidí v generování rozvrhů. Na přípravě rozvrhů pro nový školní rok, musí zaměstnanci školy začít pracovat již v květnu a během měsíců července a srpna ladí drobnosti do finální formy. Celá práce na vytvoření nového rozvrhu pro celou školu, na již postavených základech z minulých let, podle pana Janečka a jeho zástupce pana Rudolfa Dytrta trvá přes 150 hodin. Toto je dle mého soudu velmi nepříjemné a pro školu nevyhovující.

Škola se snaží svým studentům nabídnout co možná nejlepší variantu rozvrhu a nespokojí se jen s variantou, která by byla funkční, ale pro studenty méně výhodná. Jedním z požadavků této školy je vytvořit vyrovnaně zatížený rozvrh pro každou třídu. Předměty jsou proto rozděleny do skupin podle náročnosti a v ideálním případě jsou v denním rozvrhu zastoupeny jen některé předměty z každé kategorie. Dalším požadavkem je zkrátit studentům dobu strávenou ve škole bez zbytečných hodin volna. Při delším vyučování je potřeba do rozvrhu vložit hodinovou přestávku na oběd, kterou určuje i vyhláška o základním vzdělávání a některých náležitostech plnění povinné školní docházky školského zákona [14]. Jelikož škola provozuje i vlastní jídelnu, musí být vytížení jídelny rovnoměrné. Toto jsou jen některé z požadavků ZŠ Sladkovského v Chrudimi, musím ale říci, že škola se je snaží poctivě plnit a proto věnuje do tvorby rozvrhů nemalé úsilí.

Při dotazu, zda-li by mi škola nepředvedla práci se systémem Bakaláři, mi ochotně vytvořili uživatele v testovacím módu a já mohl vyzkoušet všechny funkce systému v reálném prostředí bez vlivu na pečlivě spravovaný systém školy. Tato aplikace na mne nepůsobila příliš dobrým dojmem, řekl bych, že nastavení není vůbec intuitivní a nezkušeného uživatele spíše odrazuje. Vyzkoušel jsem také generování rozvrhů, kde jsem jen modifikoval jednu z podmínek (přesně jsem definoval

přestávku na oběd u jedné třídy) a spustil vytváření. Program vytvořil rozvrh jen pro jeden den v týdnu a skončil s chybovou hláškou, že nemůže pokračovat.

Další školy mi také vyšli vstříc. Zhodnocením všech odpovědí jsem zjistil, že většina škol používá podobné systémy jen pro evidenci studentů a průběhů vyučování, málokterá pro vedení klasifikace přístupné rodičům nebo k tvorbě rozvrhů. K vytváření rozvrhů školy většinou používají již léta zaběhnutý systém, nebo upravují vygenerované rozvrhy z různých programů.

4 Generování rozvrhů

Tato kapitola je věnována tvorbě rozvrhů, od definování základních pojmů až po metody, které jsou používány při řešení této problematiky.

4.1 Rozvrh

Rozvrh je definován jako přiřazení zdrojů a času k aktivitám tak, aby byla splněna všechna omezení, přiřazené k jednotlivým aktivitám a zdrojům [5].

Rozvrh je složen z množiny aktivit, každá aktivita má předem definovanou délku trvání a zdroje, které bude využívat. Zdroj je chápán jako stroj, na kterém se daná aktivita provádí. Zdroj pak má určenou kapacitu nebo časový úsek, v kterém se může nacházet. Zdrojem může být například učitel. Dále musíme uvažovat závislosti mezi aktivitami [9]. Musíme být například schopni vyjádřit, že nějaký úkon musí danému úkonu předcházet atd.

4.2 Rozvrhování

Problém rozvrhování (timetabling) je složen z naplánování sekvencí mezi učitelem a studentem, třídou (class-teacher timetable problem, CTTP) v určitý časový úsek (typicky týden) a zároveň splnění podmínek mnoha rozdílných typů. Při manuální tvorbě rozvrhů se velmi často stává, že výsledek je neuspokojivý. Tato situace je dána velkým množstvím variant, kterými je možné výsledný rozvrh složit, proto se využívají výpočetní modely a algoritmy pro automatickou tvorbu rozvrhů.

Automatické tvorbě rozvrhů se vědci a matematici věnují již přes 30 let, kdy Gotlieb, C.C. uveřejnil první publikaci týkající se problému rozvrhování [5, 7]. Tato problematika je stále aktuální, o čem svědčí i konference z posledních let konající se například v České republice [12].

Problémy rozvrhování se řeší převedením problému na jeden ze tří základních.

I. Školní rozvrhování

Vytvoření týdenního rozvrhu pro všechny třídy školy a vyvarovat se situace, kdy by jeden učitel učil více tříd a jedna třída byla vyučována více učiteli v jeden časový úsek. Zde se předpokládá, že jsou vytvořeny třídy (studijní skupiny), kde je studentu přiřazena jedna třída se stejným rozvrhem. Tato podmínka omezuje vytváření rozvrhů, na rozdíl od dalších problémů, na jednotlivé třídy, nikoliv na jednotlivé studenty. V ideálním případě jsou pro jednotlivé předměty přiřazeny místnosti. To je však v reálném případě málokdy splněno, protože místnosti mohou být omezeny svojí výbavou.

Tato situace je typická pro tvorbu rozvrhů pro základní a střední školy v České republice. Další omezení, která mohou být školou vyžadována jsou uvedena v kapitole analýza na školách.

II. Univerzitní rozvrhování

S řešením tohoto problému se nejčastěji setkáváme na akademické půdě, kdy je potřeba vytvořit rozvrh vysoké školy nebo univerzity. Studenti zde mají větší volnost a mohou si zapsat předměty podle svého uvážení. To ve výsledku znamená, že každý předmět, má zapsaná rozlišná skladba studentů. Tím se musí při vytváření rozvrhů přihlížet na to, aby předměty zapsané studentem neprobíhaly ve stejný časový úsek, docházelo by tím ke kolizím. Vytvořit rozvrh bez kolizí zpravidla nelze, takže se hledání omezuje na hledání rozvrhu s co nejmenším počtem konfliktů, ideálně minimálním. Hledáme tedy rozvrh, který většině studentů umožní vyučování bez kolizí. Tento problém je NP-úplný [5].

Důležité je také řešit problém přiřazování místností jednotlivým předmětům v daný časový úsek. Pro každý předmět je nutné vybrat místnost s potřebným vybavením a kapacitou. Na akademické půdě je také nutné řešit dostupnost místností, aby student stihl přesun na další vyučování.

III. Rozvrhování zkoušek

Tento problém řeší rozvržení zkoušek z jednotlivých předmětů během určitého časového období. Předpokládá se, že každý předmět bude mít pouze jednu zkoušku. Tento problém netoleruje žádné kolize, každý student má možnost se v zadaném časovém období dostavit na termín zkoušky, která je mu přiřazena. Tato situace se stejně jako univerzitní rozvrhování řeší na univerzitách a vysokých školách, ale algoritmus může využit například při organizaci maturitních zkoušek na středních školách.

Řešení problému rozvrhování zkoušek se řeší například použitím genetických algoritmů, simulovaného žíhání, zakázaného prohledávání nebo programovaní s omezujícími podmínkami [5].

Na základě těchto klasifikací základních problémů školního rozvrhování se řeší všechny ostatní rozvrhovací problémy. Nejdříve je nutná podrobná analýza problému, aby se mohlo správně určit, na jaký problém bude ten řešený převeden.

4.3 Školní rozvrhování

V této kapitole podrobněji rozeberu problém školního rozvrhování, neboli modelu třída-učitel. Nejprve problém ukáži na jeho jednoduché variantě a problému řešitelného v polynomiálním čase. Poté uvedu další omezení, na které je potřeba brát zřetel při reálném rozvrhování.

4.3.1 Omezení

Velmi důležitou součástí problému rozvrhování je správná formulace omezujících podmínek. Nejprve zapíšeme problém omezení, ve kterém žádný učitel neučí více než 1 třídu a žádná třída nemá vyučování v jednom časovém úseku. Tuto matematickou reprezentaci formuloval prof. Dr. Dominique de Werra již v roce 1985 [5, 7].

Předpokládejme, že $c_1 \ldots c_m$ je konečná množina m tříd, $t_1 \ldots t_n$ je konečná množina n učitelů a $1 \ldots p$ je p časových úseků. Dále je dána matice R_{mxn} nazývaná matice požadavků [7], kde její prvky $r_{i,j}$ jsou dány jako t_i a c_j.

Pak hledáme x_{ijk}, kde $i \geq 1, j \geq 1, k \geq 1 \wedge i, j, k \in N$

$$\sum_{k=1}^{p} x_{ijk} = r_{ij} \left(i = 1..\mathrm{m}, j = 1..\mathrm{n}\right) \quad (1)$$

$$\sum_{j=1}^{n} x_{ijk} \leq 1 \left(i = 1..\mathrm{m}, k = 1..\mathrm{p}\right) \quad (2)$$

$$\sum_{i=1}^{m} x_{ijk} \leq 1 \left(j = 1..\mathrm{n}, k = 1..\mathrm{p}\right) \quad (3)$$

$$x_{ijk} = 0 \text{ nebo } x_{ijk} = 1 \left(i = 1..\mathrm{m}, j = 1..\mathrm{n}, k = 1..\mathrm{p}\right) \quad (4)$$

kde $x_{ijk} = 1$ pokud učitel t_j má vyučování ve třídě c_i v časovém úseku k, jinak $x_{ijk} = 0$.

Omezení (1) zaručuje, že odpovídá počet odučených hodin jednoho učitele s počtem hodin, která jsou vyučována tímto učitelem ve všech třídách. Omezení (2), resp. omezení (3), zaručuje, že každý učitel, resp. třída, má nejvýše jedno vyučování v jednom časovém úseku.

Výše uvedená omezení jsou základními pro vytvoření rozvrhů. Pokud nebudeme uvažovat žádná další omezení, tak je tento problém řešitelný v polynomiálním čase [10].V reálném prostředí školy ovšem tyto omezení nestačí a je potřeba jich nadefinovat mnohem více. V dalším textu uvedu příklady dalších omezení, která jsem získal průzkumem ve škole.

4.3.1.1 Souběžné vyučování

V reálném provozu školy je obvyklé, že některý předmět je vyučován jedním učitelem souběžně ve více třídách. To znamená, že třídy musejí být spojeny, to klade dále nároky například i na možnosti učeben, kde musí být patřičné vybavení a prostory. Díky moderním technologiím již není problém přenášet přednes vyučujícího do dalších místností, tím se složitost automatického rozvrhování zvyšuje.

4.3.1.2 Vztah učitel - předmět

Dále je potřeba nadefinovat vztah mezi učitelem a předmětem. Učitel musí mít přiřazeny předměty, které může učit.

4.3.1.3 Průběh vyučování

Zvláště na základních a středních školách je důležité nadefinovat průběh vyučování. Ideálně by mělo vyučování probíhat kontinuálně od ranních hodin a bez zbytečných hodin volna. Nutností je, ale hodinová přestávka na oběd. Tyto přestávky na oběd jsou definovány státem, dle ročníku, ve vyhlášce školského zákona [14].

4.3.1.4 Speciální místnosti

Při vytváření rozvrhů se musí také přihlížet na zvláště vybavené místnosti, typickým příkladem jsou místnosti, kde se vyučuje chemie a fyzika. Proto bude zapotřebí nadefinovat vztah místnost – předmět, kde ke každému předmětu budou přiřazeny místnosti, ve kterých mohou být vyučovány.

4.3.2 Možnosti řešení

V literatuře o automatické tvorbě rozvrhů se uvádí různé techniky a přístupy. Technikou je myšlen algoritmus potřebný k řešení problému, může jít například o genetické algoritmy. Přístupy se myslí způsoby zpracování informací a postupy vedoucí k nalezení řešení, může se jednat například o logické programování.

Nyní uvedu nejpoužívanější techniky pro řešení problému školního rozvrhování.

4.3.2.1 Barvení grafu

Grafy a jejich barvením se zabývá celá kapitola 6.

4.3.2.2 Genetické algoritmy

Začátkem 60. let vzniká nový směr v rozvoji informatiky, genetické algoritmy. Tyto algoritmy vycházejí z Darwinovy teorie o vývoji druhů. Úspěšně se používají k řešení optimalizačních problémů.

Z biologie známe způsob, jakým živé organismy předávají své genetické informace svým potomkům. Tyto informace jsou uloženy v chromozómech jednotlivých organismů. Obsah chromozómu se dá chápat jako jistý kód informace o jedinci. Podle teorie Charlese Darwina v přírodě přežívají silnější jedinci a tito pak mají šanci předat svoji genetickou informaci do další generace. Důležité přitom je, že každý organismus je potomkem dvou rodičů a tudíž se v něm mísí genetické informace obou rodičů. Neboli informace uložené v jeho chromozómech jsou zčásti převzaté od

jednoho z rodičů a z části od rodiče druhého. Na těchto základních principech pracuje i genetický algoritmus.

Genetické programování [9] je evoluční technika, kdy jedinci jsou reprezentováni programy pro řešení dané úlohy.

Obrázek 1: Cyklus genetických algoritmů

Algoritmus 1: Princip genetických algoritmů

1. Návrh struktury – je nutné správně navrhnout strukturu jedince, aby bylo možné křížení a dosáhli jsme požadovaných výsledků.
2. Inicializace – přiřazení genetických informací k počáteční populaci, toto přiřazení většinou probíhá náhodně.
3. Ohodnocení – na základě stanovených kritérií probíhá výpočet kvality jedince, každému jedinci je přiřazeno ohodnocení.
4. Selekce – výběr jedinců do nové populace, probíhá na základě ohodnocení, kde šanci mají především průměrní a nadprůměrní jedinci.
5. Křížení - během tohoto procesu vzniká ze dvou nebo více zvolených jedinců nový jedinec. Vlastnosti nového jedince vzniknou kombinací vlastností rodičů.

6. Mutace – je náhodná změna genetické informace zvolených jedinců, umožňuje rozšíření stavového prostoru.
7. Reprodukce – je promítnutí výsledků do nové populace.
8. Cyklicky se opakují kroky 3.- 7. až do splnění podmínky nalezení řešení.

Nyní je potřeba navrhnout převedení problému rozvrhování, aby se mohl zpracovat pomocí genetického programování. Každý jedinec populace bude představovat jeden z variant rozvrhů. Rozvrh bude u jedince určen pomocí genů, které určují kdy a kým je předmět vyučován.

Počáteční populace se může náhodně vygenerovat, ale užitečnější je i první populaci navrhnout pomocí alespoň jednoduché heuristiky, aby byl zaručen kvalitnější vývoj populace. Další jedinci populace se vytvoří křížením dvou rozvrhů, tím vznikne úplně nový rozvrh. Návrh křížení genů je nejobtížnější částí genetického programování, protože je obtížné najít na sobě nezávislé geny, které by se mohly prohodit. Někdy se může stát, že se tyto geny nenajdou, tím není křížení vždy zaručeno.

Po vytvoření nového jedince (rozvrhu) může nastat proces mutace, jenž náhodně změní některé geny a tím se tento jedinec bude odlišovat od jedinců, z kterých byl vytvořen.

Po vytvoření celé populace, tedy nových rozvrhů, jsou jednotlivé rozvrhy ohodnoceny. Ohodnocující funkce přiřadí každému rozvrhu například procentuální úspěšnost, s jakou jsou splněna všechna omezení kladena na výsledný rozvrh.

Tento cyklus se opakuje tak dlouho, dokud výsledkem není rozvrh splňující omezení, buď nad nějakou procentuální hranici, nebo dokud nejsou splněna alespoň základní omezení [5, 9].

4.3.2.3 Heuristické metody

Heuristické metody byly vyvinuty pro řešení optimalizačních úloh. Tyto metody nám sice nezaručují nalezení globálního optimálního řešení, ale poskytují řešení, které je vzhledem k obtížnosti problému uspokojivé, tzv. lokální řešení.

Princip prohledávání stavového prostoru potenciálních řešení vyžaduje rovnováhu mezi:

- Nalezení lokálního optima v blízkém okolí výchozího bodu.
- Co největší prohledání stavového prostoru.

Dále zde uvedu některé z heuristických metod, které jsou nejčastěji používány k řešení generování rozvrhů.

Metoda lokálního hledání

Jedna z nejsnadněji implementovaných heuristik. Svou jednoduchostí je tato metoda vhodná na problémy jednoduššího řádu, neboť na složitější problémy s více parametry tato metoda už nestačí.

Algoritmus 2: Lokálního vyhledávání

1. Úvodní řešení je náhodně vygenerováno.
2. Procházení sousedních řešení a jejich ohodnocení.
3. Jako nové řešení zvolíme to s nejlepším ohodnocením.
4. Dokud není splněna podmínka pro dokončení algoritmu, nebo v okolí není lepší řešení, tak se pokračuje bodem 2.

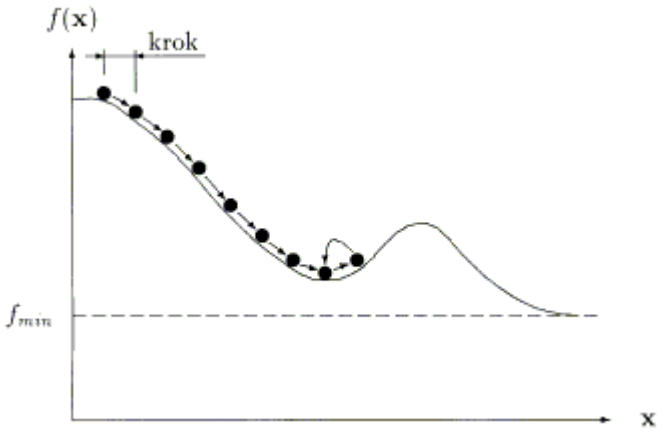
Tento postup spolehlivě hledá pouze lokální řešení, při hledání globálního řešení může pomoci vícenásobné spuštění metody lokálního hledání s různě vygenerovanými úvodním řešením. Z výsledných řešení vybereme to nejlepší řešení, podle daných podmínek. Ale i tak nemáme zaručeno nalezení globálního řešení, jedná se pouze o jedno z vylepšeních.

Horolezecký algoritmus

Tento algoritmus patří do skupiny heuristických algoritmů, kde je povolen i krok směřující k horším prozatímním řešením. Algoritmus je velmi podobný lokálnímu hledání, kde vybíráme jedno řešení z okolí toho aktuálního. Cyklus algoritmu je ukončen po předem daném počtu iterací.

Algoritmus 3: Horolezecký algoritmus

1. Úvodní řešení je náhodně vygenerováno.
2. Procházení sousedních řešení a jejich ohodnocení.
3. Jako nové řešení zvolíme to s nejlepším ohodnocením.
4. Nové řešení si zapamatujeme.
5. Dokud nebylo dosaženo počtu iterací pokračuje bodem 2.



Obrázek 2: Horolezecký algoritmus

Zapamatováním si historie všech dosažených řešení máme jistotu, že po provedení všech iterací se algoritmus vrátí zpět k nejlepšímu řešení. Problémem u tohoto řešení může být zacyklení, kdy se algoritmus stále vrací k již nalezenému řešení.

Simulované žíhání

Tato metoda vychází z fyzikálního procesu odstraňování defektů krystalické mřížky. Představme si výsledné řešení jako materiál, jehož kvality vyzkoušíme postupným zahříváním a ochlazováním při jeho chladnutí. Na počátku krystal zahřejeme na vysokou teplotu, při jeho chladnutí mají defekty krystalové mřížky pravděpodobnost zániku. Pravděpodobnost je větší, čím je větší teplota. Krystal se vždy snaží dostat do stavu, kdy je jeho energie minimální, krystal bez defektů. Tento způsob nám pomáhá určit nejen lokální řešení, ale díky větším změnám můžeme zjistit i globální řešení.

V terminologii rozvrhování, nejdříve je náhodně vybrán jeden rozvrh. Dále jsou znovu náhodně vybrány rozvrhy v jeho okolí, pokud má vybraný rozvrh lepší ohodnocení je automaticky vybrán. Může být vybrán i rozvrh s horším ohodnocením a to právě podle pravděpodobností funkce

$$\exp(-\delta / t)\delta \geq 0, t > 0 \qquad (5)$$

kde δ vyjadřuje kvalitu řešení a t současnou teplotu.

S klesající teplotou, klesá i pravděpodobnost výběru rozvrhu s horším ohodnocením. Výsledkem algoritmu je přinejmenším jedno z výrazně kvalitnějších lokálních řešení v porovnání s předchozími algoritmy [11].

Metoda zakázaného prohledávání (Tabu search)

Při běhu horolezeckého algoritmu může dojít k zacyklení, po určitém počtu iterací se algoritmus vrací zpět k lokálnímu řešení. Metoda zakázaného prohledávání se snaží tomuto cyklu zabránit a tím zvýšit globální obsáhlost. Jedná se tedy o vylepšení horolezeckého algoritmu.

Zabránit zacyklení, kde se algoritmus vrací po již prošlém řešení (rozvrhu), můžeme postupným vytvářením seznamu zakázaných řešení (tabu list). Když jsou procházena sousední řešení a hledá se to s nejlepším ohodnocením, může být vybráno pouze to, které není obsaženo v seznamu zakázaných řešení. Velikost takového seznamu je volitelná a je jedním ze vstupních dat tohoto algoritmu, protože pro různé velikosti seznamu většinou získáme různá řešení. Při větší délce seznamu zabráníme delším cyklům, ale s rostoucí délkou roste i složitost algoritmu [12].

5 Požadavky na informační systém

Po analýze požadavků školy a školní agendy, způsobu jejich implementací a nastínění problému rozvrhování nyní mohu definovat požadavky na výsledný projekt.

Cílem je navrhnout a implementovat informační systém školy s funkcí automatického generování rozvrhu. Systém bude zaměřený především na základní a střední školy, vzhledem k rozdílnému přístupu na vysokých školách.

5.1 Webová aplikace

Požadovaný informační systém by měl být implementován jako webová aplikace. Toto řešení přináší výhody, především díky velké dostupnosti. K přístupu do informačního systému je potřeba pouze internetové připojení a webový prohlížeč.

Vzhledem k rozdílné implementaci CSS (Cascading Style Sheets) a Javascriptu v různých internetových prohlížečích, je nutné stanovit si jako cíl, ve kterých prohlížečích bude možný bezproblémový běh vyvíjené webové aplikace. Abychom zaručili co největší dostupnost systému, je požadováno, aby byl informační systém plně funkční ve 4 dnes nejpoužívanějších [23] internetových prohlížečích:

- Internet Explorer (6, 7) od firmy Microsoft
- Mozilla Firefox
- Opera od firmy Opera ASA
- Safari vyvíjený Apple Inc.

5.2 SEO

SEO (Search Engine Optimization, optimalizace pro vyhledávače) je metodologie navrhování a vytváření webových stránek, takovým způsobem, aby výsledek vyhledávání určité fráze v internetových vyhledávačích umisťoval webovou aplikaci na co nejlepší pozici.

Tato funkcionalita je v našem případě považována za nefunkční požadavek. Prvním důvodem je nutnost přihlášení do informačního systému, žádná z optimalizací by tedy neměla požadovaný účinek (vyhledávací robot by se nedostal k požadovaným informacím). Druhým důvodem je citlivost některých informací, například ve zkoumaném školním informačním systému to mohou být adresy všech uživatelů.

5.3 Adaptabilita

Různé odlišnosti zavedených postupů na jednotlivých školách je důvodem proto, aby byl systém co možná nejjednodušší a snadno rozšiřitelný. Informační systém by se měl přizpůsobit požadavkům školy.

V případě, že škola vede nějaké informace, které nebudou součástí tohoto navrhovaného školního informačního systému, měl by systém klást malé nároky na jeho rozšíření a provázání s již fungujícími částmi aplikace.

5.4 Systémové role

Předpokládá se, že se systémem pracují zaměstnanci školy, studenti, jejich rodiče a administrátor, který má nad celým systémem dohled a jeho práva nejsou žádným způsobem omezena. Proto je nutné definovat rozsah a možnosti jejich účtů.

- Zaměstnanec – přehled o všech informacích týkajících se školy, možnost přidání např. aktuality. Pokud se bude jednat o učitele, bude moci spravovat předměty a třídy, které vyučuje.
- Student – přístup ke své klasifikaci, všem předmětům, rozvrhu a dalším obecným informacím.
- Rodič – stejná práva jako student a navíc s možností komunikace s učiteli.
- Správce systému – bude mít přístup ke všem informacím v systému, včetně vytváření nových uživatelských účtů. Správcem systému ve škole může být například správce sítě nebo ředitel.

5.5 Popis modulů a jejich funkcionalita

Popisem jednotlivých částí školního informačního systému se určí jejich rozsah a účel v aplikaci.

5.5.1 Evidence studentů

Systém spravuje všechny záznamy o studentech školy. Oprávněným uživatelům poskytuje možnost vložení nového, úpravu nebo smazání záznamu studenta. O studentech jsou vedeny informace jako datum narození, zdravotní pojišťovna, u které je žák pojištěn, kontaktní informace, kontakt na rodiče.

5.5.2 Evidence zaměstnanců

Systém také eviduje informace o všech zaměstnancích školy. Systém umožňuje přidání nových zaměstnanců, pokud je zaměstnanec učitel je možné mu definovat předměty, které vyučuje. Přístup k těmto datům má pouze správce systému, aby nemohlo dojít k zneužití soukromých dat.

5.5.3 Seznam tříd

V modulu tříd je nutné při inicializaci systému každý školní rok nadefinovat všechny třídy ve škole (možnost přidání/editace/smazání). O třídě je evidováno do jakého ročníku patří a její popis. Dále systém umožňuje zařazení studentů do tříd.

5.5.4 Seznam předmětů

V seznamu předmětů je zapotřebí definovat i ročník, ve kterém je předmět vyučován, aby se jednoznačně identifikoval předmět nazvaný například matematika v 5. ročníku od toho v 8. ročníku. Každému předmětu je možné přiřadit popis a jeho osnovu. U každého z předmětů je evidováno, kteří učitelé jej vyučují.

5.5.5 Klasifikace

Pro každého studenta je vedena jeho klasifikace za celou dobu jeho studia na dané škole. Hodnocení studentů je nejdůležitějším modulem. Z průzkumu na školách vyplývá, že se používá více způsobů ohodnocení studenta. Systém umí zpracovat tzv. klasickým hodnocením studentů. Tím je nejčastěji používané ohodnocení výkonu studenta pomocí čísel 1-5.

U každého záznamu o klasifikaci je nutné uvádět, kdy byla známka přidělena, za jakou formu zkoušení (ústní zkouška, test, písemná práce), kým byla známka udělena a samozřejmě v jakém předmětu.

Vkládání a editaci známek mohou provádět pouze učitelé a to pouze těm žákům, které vyučují. Systém neumožňuje ohodnocení studenta učitelem, který ho nevyučuje.

Student nebo jeho zástupce (rodič) sleduje známky (ohodnocení) v modulu žákovská knížka.

5.5.6 Rozvrhy

Hlavní předností celého projektu je možnost automatické tvorby rozvrhů tříd. Generátor rozvrhů by měl přihlížet na uživatelem definované omezující podmínky. Uživatel, který má přidělena práva vyvářet rozvrh, má možnost nastavení těchto podmínek:

- Učební plán, v kterém nadefinuje vztahy mezi ročníkem a předmětem, bude tedy určovat kolik hodin týdně má daný ročník v rozvrhu daných předmětů.
- Každé třídě přiřadit, v kterém časovém úseku bude mít povinnou přestávku na oběd.
- Požadavky učitelů na to, kdy nemohou učit, tedy jaké hodiny v týdnu nechtějí mít v rozvrhu vyučování.
- Speciální místnosti, tj. učebny ve kterých může probíhat výuka pouze nějakého předmětu, takovým příkladem může být laboratoř určena pouze pro výuku chemie.

6 Grafy

Grafem G se rozumí objekty popsané množinou uzlů a množinou hran [15]. Objekty jsou v grafu znázorněny jako uzly U, také označovány jako vrcholy grafu. Skutečnost, že dva objekty jsou ve vzájemném vztahu se zakresluje do grafu jako spojení dvou uzlů, reprezentující tyto objekty. Toto spojení se nazývá hranou H.

Graf definujeme [16] jako trojici G = (U, H, I), kde U je množina uzlů, jak už bylo řečeno výše, H je množina hran a I je incidenční relace, pro kterou platí

$$I : H \rightarrow \binom{U}{2} \tag{6}$$

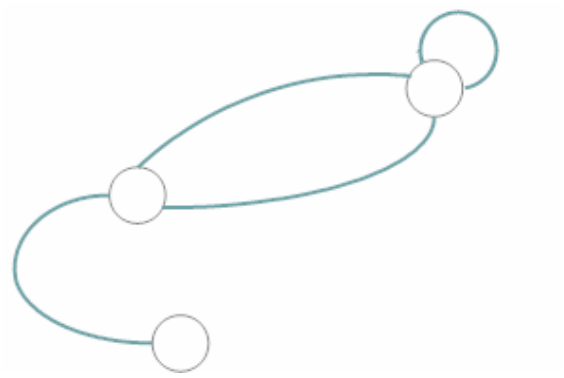
pokud se jedná o zobrazení prosté, pak zapisujeme G = (U, H), tzv. prostý graf.

Stupeň uzlu je dán počtem hran, které jsou s tímto uzlem spojeny. Říkáme, že hrana s tímto uzlem inciduje. Pro označení stupně uzlu $u \in U$ používáme zápis deg(u). Nechť G = (U, H) je obyčejný graf, kde |H| = m. Pak platí [16]

$$\sum \deg(u) = 2m \text{, kde } u \in U \tag{7}$$

Existují ještě jiné typy grafů [28]. Obecné grafy připouštějí i smyčky, tedy také $H \subseteq U$. Orientované grafy mají všechny hrany orientované, jsou reprezentovány uspořádanou dvojicí $H \subseteq U \times U$.

Pro multigrafy platí obecná relace $H \rightarrow U \times U$ a připouštějí tedy i násobné hrany, příklad multigrafu je uveden na obrázku 3.



Obrázek 3: Konstrukce grafu o 3 uzlech

6.1.1 Barvení grafu

Barvení grafu je jednou z disciplín teorie grafů, která se zabývá přiřazováním barev, reprezentovaným přirozeným číslem, různým objektům v grafu - vrcholům, hranám, stěnám atd. Nejčastěji jde o barvení vrcholů, ostatní případy (například barvení sousedících ploch) lze na tento jednoduše převést.

Problém barvení grafu je jedním ze známých NP-úplných problémů [5]. Zadáním je graf G = (U, H), kde U je množina vrcholů a H množina hran. A úkolem je přiřadit každému vrcholu jednu barvu tak, aby žádné dva vrcholy spojeny jednou hranou neměly stejnou barvu. Samozřejmě můžeme každému vrcholu přiřadit jinou barvu, ale takové řešení by nám bylo k ničemu, chceme najít takové řešení, kde máme graf obarven co nejmenším počtem barev. Takové číslo potom nazýváme chromatickým číslem grafu G. Graf G nazýváme *k*-chromatickým, pokud lze jeho uzly obarvit s použitím *k* různých barev [15].

Pro obarvení grafu existuje řada heuristických postupů, které jsou deterministické a pro větší grafy nedávají jistotu nalezení optimálního řešení [5].

6.1.1.1 Uvedení problému

Nalezení co nejmenší hodnoty *k*, pomocí kterého lze obarvit zadaný graf G. G je pro zjednodušení obyčejný graf. Snažíme se obarvit, tj. nalézt číselnou hodnotu, uzly u patřící do U(G) tak, že dva uzly spojené hranou nemají stejnou barvu.

Sekvenční barvení

Tento algoritmus se snaží vyřešit problém přímou metodou, kdy obarvuje jeden vrchol za druhým. Touto jednoduchou metodou dosahuje velice přijatelných výsledků, typicky je počet barev *k* jen o 5% vyšší než je chromatické číslo grafu [15].

Algoritmus 4: Sekvenční barvení

1. Nastavení $k = 0$ a všech uzlům nastavíme barvu 0.
2. Zvolení dosud neobarveného vrcholu (obarveného hodnotou 0).
3. Nalezení nejmenšího přirozeného čísla *p* takové, že neexistuje vrchol spojený hranou s aktuálním uzlem a obarvený hodnotou *p*.
4. Aktuálnímu uzlu nastavíme jako barvu hodnotu *p* a pokud je $p > k$, pak nastavíme $k = p$.
5. Pokud nejsou obarveny všechny uzly, tak se pokračuje bodem 2.

Pro upřesnění kroku 2, existuje více způsobů jak zvolit dosud neobarvený uzel.

- Náhodné vybrání jednoho z množiny neobarvených uzlů (stejnou variací je v kroku 1 náhodně seřadit uzly a v kroku 2 je potom volit postupně).
- V kroku 1 uzly uspořádáme do nerostoucí posloupnosti, podle velikosti jejich stupňů deg(u). Poté v kroku 2 vybíráme uzly postupně, od začátku. Tím docílíme toho, že uzly s vysokým stupněm jsou obarvovány nejdříve a až nakonec ty s nejmenším stupněm.
- V kroku 2 spočítáme pro každý uzel hodnotu N(u), která určuje počet barev, které byly potřeba k již zpracovaným sousedům. Vybíráme vždy ten uzel, jehož hodnota N(u) je největší. Je-li takových hodnot více zvolíme ten, který má nejvíce nezpracovaných sousedů.

Barvení pomocí nezávislých množin

Tento algoritmus je založen na skutečnosti, že množina uzlů, které jsou obarveny stejnou barvou tvoří nezávislou podmnožinu množiny uzlů grafu. Důležitým krokem je nalezení maximální nezávislé množiny uzlů, tato úloha ovšem také patří do skupiny NP-úplných problémů [16].

Nezávislá množina uzlů je taková množina $A \in U(G)$, že žádná hrana $h \in H(G)$ nespojuje uzly množiny A. Počet prvků největší nezávislé množiny se označuje jako α(G).

Algoritmus 5: Nalezení maximální nezávislé množiny v grafu

1. Nastavení výsledné množiny V = {}.
2. Vybrání jednoho uzlu z množiny uzlů grafu takový, který ještě nebyl vybrán.
3. Pokud množina V obsahuje sousedy vybraného uzlu, tak se z množiny odstraní.
4. Vložení vybraného uzlu do množiny V.
5. Pokud nebyly vybrány všechny uzly, tak se pokračuje bodem 2.

Jiným způsob výběru uzlu z grafu může být:

- Zvolení uzlu s nejmenším stupněm deg(u).
- Zvolení uzlu, který v daném okamžiku minimalizuje stupeň deg(u), kde $u \in U(G)$.

Algoritmus 6: Barvení pomocí nezávislých množin

1. Nastavení hodnoty k = 1, kde k určuje barvu.
2. Nalezení maximální nezávislé množiny uzlů v grafu G, použitím algoritmu 5.
3. Uzly v této množině jsou obarveny barvou k.
4. Odstranění vybrané množiny uzlů z grafu G. Také se odstraní hranami, které jsou s uzly incidentní.
5. Pokud graf G obsahuje alespoň jeden uzel, zvýšení hodnoty barvy k = k + 1 a pokračuje se krokem 2.

Barvení slepováním uzlů

Tento algoritmus provádí spojování uzlů, které lze obarvit jednou barvou, tedy uzlů, které spolu nejsou spojeny hranou. Výsledkem je úplný graf, který lze velmi snadno obarvit.

Algoritmus 7: Barvení grafu slepováním uzlů

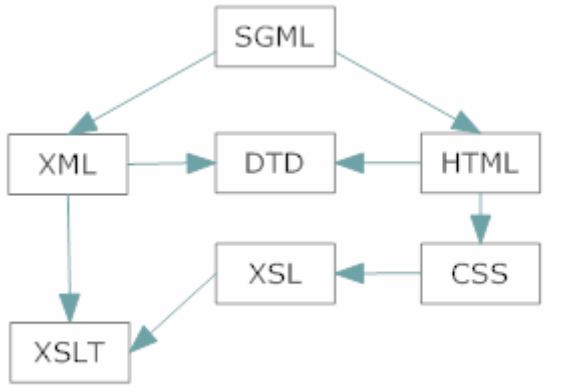
1. V grafu se náhodně vyberou dva uzly u_1 a u_2, které nejsou spojeny hranou.
2. Tyto uzly se nahradí jedním, ten se spojí hranou se všemi uzly grafu G, se kterými byl spojen alespoň jeden uzel u_1 nebo u_2.
3. Pokud není graf G úplným grafem pokračuje se krokem 1.
4. Obarvení úplného grafu, kdy se každému uzlu přiřadí jiná barva.
5. Uzly původního grafu jsou obarveny barvou uzlu, do kterého přešly.

7 XML

XML (Extended Markup Language) je, jak už název napovídá, určený pro dokumenty obsahující strukturované informace. Pojem „markup language“ znamená, že jazyk obsahuje nějakým způsobem zavedený mechanismus pro identifikaci nejenom samotných hodnot v dokumentu.

XML patří také do rodiny tzv. meta-jazyků. Jako meta-jazyk deklaruje možnou syntaxi pomocí tzv. tagů. Protože XML umožňuje vytvořit (podle svých pravidel [17]) libovolný tag a tím dává možnost vytvářet libovolné struktury, je chápán jako metajazyk.

XML dokument je uložen jako textový dokument a lze jej přímo číst libovolným textovým editorem. Struktura dat je v tomto dokumentu velmi dobře čitelná a lidsky srozumitelná. V XML se význam dat označuje pomocí tagů. Tagy známe z jazyka HTML, kde mají stejný význam, oba jazyky jsou si totiž ze sémantického hlediska velmi podobné. Oba vycházejí z jazyka SGML, jak je vidět na obrázku 4. Významným rozdílem je ovšem fakt, že HTML je jazykem a XML je metajazykem. To znamená, že HTML má omezenou množinu tagů, oproti tomu XML nemá množinu tagů omezenou, kde o významu tagů rozhoduje jejich interpretace.



Obrázek 4: Hierarchie jazyků založených na SGML

7.1.1 Základní struktura

Při psaní XML dokumentu se musí dodržet určitá velmi jednoduchá struktura. Než zde uvedu příklady zápisu samotného dokumentu musím zmínit, že povinným prvkem je uvedení deklarace na prvním řádku XML dokumentu. V deklaraci je uvedena verze jazyka a způsob kódování, ten je sice nepovinný, ale doporučuje se jej uvádět.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Základním prvkem struktury XML dokumentu je element. Ten je tvořen pomocí počátečního a koncového tagu a uvnitř mezi tagy nese svůj obsah. Pro správný zápis, je nutné stejné pojmenování počátečního a koncového tagu, jazyk XML rozlišuje malá a velká písmena. Ukázka zápisu elementu v XML dokumentu je uvedena na příkladu 1.

Dále je možné vytvářet hierarchickou strukturu pomocí zanořování elementů. Je však možné vkládat pouze celý element, tj. s počátečním i koncovým tagem. Korektní zanoření elementů obsahuje příklad 2.

Dalším důležitým prvek XML dokumentu je atribut. Atributy mohou nést další informace, kterým říkáme metadata, spojené s elementem. Atribut je dvojicí název a hodnoty, které jsou umístěny v počátečním tagu elementu s použitím rovnítka. Označení atribut je libovolné, stejně jako v případě elementů, ale hodnota musí být uzavřena v dvojitých uvozovkách. Ukázka je uvedena na příkladu 3.

Základní strukturu XML dokumentu tvoří i další prvky, některé z nich jsou uvedeny v navazující kapitole.

Příklad 1:

```
<význam> obsah elementu </význam>
```

Příklad 2:

```
<význam> obsah elementu
     <vnitřní>se svým obsahem</vnitřní>
</význam>
```

Příklad 3:

```
<význam atribut=“hodnota atributu“> obsah elementu </význam>
```

7.1.2 DTD

DTD (Document Type Definition, definice typu dokumentu) je nepovinnou součástí XML dokumentů [17], [24]. DTD je přesná definice přípustné struktury dokumentu, tj. souhrn pravidel definující tagy a jejich vzájemné vztahy. Podle této struktury se potom řídí samotný XML dokument.

Výhodou je rozpoznání správnosti dokumentu už při jeho načítání, kdy je nesprávně strukturovaný dokument odmítnut.

DTD rozdělujeme na dva typy:

1. Externí, kdy je DTD umístěn mimo samotný obsah XML dokumentu a dokument se na něj odkazuje.
2. Interní, který je DTD součástí dokumentu.

DTD zapisujeme do souboru hned na začátek, ale samozřejmě až za deklaraci XML. Pro zápis používáme speciální element `<!DOCTYPE kořenový_element SYSTEM "URL">` pro deklaraci externího DTD, kde kořenový_element obsahuje entitu, kterou je XML dokument obalen [24] a URL cestu k externímu DTD.

Pro deklaraci interních DTD má zápis tvar:

```
<!DOCTYPE "kořenový_element" [
   <!-- Samotné DTD -->
]>
```

kde kořenový_element obsahuje název elementu, v kterém je obsažen celý obsah dokumentu XML.

Definice DTD obsahuje čtyři typy různých prvků [24], jedná se o element, atribut, entitu a notaci. Definice elementu stanovuje jaký element se v dokumentu může objevit a jaký musí mít obsah. Zapisuje se ve tvaru:

```
<!ELEMENT název_elementu obsah_elementu>
```

Pro obsah elementu může být použito klíčových slov EMPTY a ANY, kde EMPTY říká, že obsah elementu musí být prázdný a ANY pro obsah elementu s libovolnými daty. Pro přesnější deklaraci obsahu elementu se použijí tzv. modelové skupiny. Ta musí být uzavřena v kulatých závorkách a deklaruje jaké další elementy jsou vkládány do obsahu, čímž se elementy zanořují do sebe. Pro přesné určení vnořených elementů použijeme operátorů ?, +, *. Operátor ? zapsaný za jedním elementem, nebo celou modulovou skupinou, značí nepovinný výskyt tohoto údaje. Naopak operátorem + značí alespoň jeden výskyt. A operátor * deklaruje libovolný počet opakování. Pro oddělení elementů se používá čárka, nebo znak |, ve významu nebo. Pokud je obsahem elementu text, použije se identifikátoru `#PCDATA`.

Příklad 4: Definice elementu

```
<!ELEMENT význam (subelement1?, subelement2 | subelement3)*>
```

která uvádí, že element *význam* má skupinu vnořených elementů subelement1, který nemusí být uveden, následně subelement2 nebo subelement3 v tomto pořadí. A celá tato skupina se může libovolně krát opakovat.

Dále deklarujeme atributy, deklarace se potom skládá z názvu elementu, kterému je atribut přiřazen, názvu samotného atributu, jeho typu a určení standardní hodnoty. Má-li element více atributů, lze je zapsat v rámci jedné deklarace.

Příklad 5: Definice atributu elementu *text*, u kterého uvádíme důležitost textu výčtem možností a autora, který je dán libovolným textem.

```
<!ATTLIST text důležitost (malá | normální | velká) autor CDATA>
```

Dále může být v DTD XML dokumentu deklarována entita. Entitou se v XML myslí zkratka pro určitý obsah. Podle typu obsahu se dělí entity na binární a textové. Binární entita může být pouze externí, uvádíme u ní parametr URL [24]. Entity obsahují libovolný obsah, který jim je deklarován a pomocí odkazu na entitu můžeme v XML dokumentu tento obsah využít. Samotné použití entit je velmi široké, pro pochopení označení entita zde uvedu jen ukázku na příkladě 6.

Příklad 6: Entita podpis

```
<!ENTITY podpis "Moje jméno">
```

Takto se definuje entita *podpis*, která má obsah "Moje jméno", při použití odkazu na tuto entitu, ve tvaru &podpis, bude zápis nahrazen obsahem entity.

Posledním typem deklarace v DTD je deklarace notace. Ta určuje pro daný typ dat aplikaci, která je schopna tento typ dat zpracovat [24]. Syntaxe zápisu je

```
<!NOTATION název_pro_typ_dat identifikátor_aplikace>
```

7.1.3 XML schéma

XML Schéma je alternativou k DTD (Document Type Definition). Pro definici syntaxe XML dokumentu používá několik rezervovaných názvů elementů. Hlavní výhodou je podpora datových typů. Díky ním lze snadno vymezit obsah elementů, a tím určit správnost obsahu těchto elementů.

Primární význam schémat leží v jejich použití pro formální definici značkovacích jazyků [17] založených na XML. Vzhledem k tomu, že schéma jednoznačně určuje jak má XML dokument vypadat, můžeme pro jeho ověření použít validaci.

Schéma mnohem přesněji určuje strukturu dokumentu a význam jednotlivých elementů či atributů. Popis takto vytvořeného dokumentu je tak přesnější na rozdíl od pouhého DTD. Zato zápis specifikace XML schémat vyžaduje mnohem větší úsilí..

8 Návrh informačního systému

Po seznámení se s požadavky kladenými na výsledný informační systém, ve kterých je definováno co se od očekává, se tato kapitola věnuje samotnému návrhu. Návrhem takového informačního systému rozumíme návrh postupů a schémat, která se musí důsledně dodržovat při implementaci.

Nejdříve je nutné specifikovat jaký systém budeme navrhovat a co od systému očekáváme.

8.1 Modularita

Po pečlivém prostudování analýzy školního informačního systému stanovujeme, jak bude výsledný systém navržen. Jedním z požadavků je jednoduchost a snadná rozšiřitelnost, proto jsem se rozhodl návrh systému zobecnit.

Má-li být vytvořen informační systém, na kterém může být nasazeno několik rozdílných aplikací složených z určitých bloků, pak se jedná o modulově založenou aplikaci. Modulem se rozumí část aplikace plnicí určitou funkcionalitu. S každým modulem je možné pracovat jako s ucelenou jednotkou, ale jednotlivé moduly jsou mezi sebou provázány.

Modularita přináší výhody, především ze strany vývoje a implementace, uživatel systému by ovšem neměl pozorovat rozdíly mezi modulově založenou aplikací a jinou. Logické a strukturované rozdělení aplikace na jednotlivé moduly zvyšuje přehlednost. Z toho plyne možnost snadného rozšíření systému, kdy se přidáním nového modulu rozšíří výsledná aplikaci o nové možnosti. Tak přizpůsobivost, kdy se vhodnou kombinací různých modulů získá aplikace zdánlivě rozdílné.

Jednou z dalších výhod modularity je znovupoužitelnost kódu, kdy po vytvoření jednoho modulu je tento modul použit i k řešení podobného problému.

Příkladem takového modulu je například modul studenti. Od kterého očekáváme možnost přidání nového studenta do datového úložiště, výpis všech studentů, nebo smazání záznamu o studentovi. Funkce takového modulu může být mnohem širší, ale už na tomto příkladu je vidět, že lze tento modul použit i pro vytvoření seznamu učitelů nebo předmětů. Moduly tříd, předmětů, či učitelů jsou si velmi podobné a jedná se pouze o práci se seznamem položek, kde mají položky pouze rozdílné atributy. Atributy se v tomto příkladě rozumí údaje, s kterými modul pracuje. U modulu student jsou jimi jméno studenta, datum narození nebo bydliště.

Pro použití jednoho modulu ke zpracování různých atribut, je nutné mu nějakým způsobem nastavit s jakými daty a jakým způsobem pracovat. Jako ideální pro konfiguraci modulu je soubor XML, který přináší velikou výhodu v tom, že modul je potom ovládán obecně nějakým skriptem s výstupem ve formátu XML, nebo jej může sestavit člověk, který není programátor. Samozřejmě není

možné, aby konfigurační soubor byl jakýmkoliv soubor ve formátu XML, ale je nutné stanovit určitá pravidla. Tomuto se věnuji hlouběji v dalším textu.

8.2 Architektura aplikace

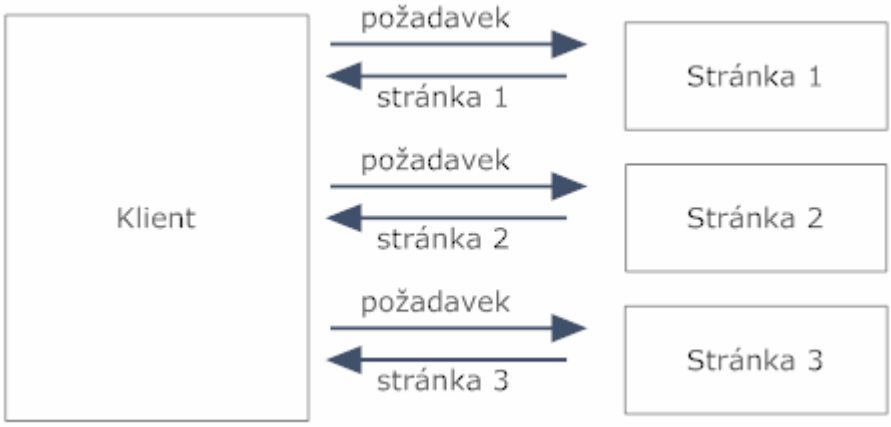
Při návrhu takto složitého informačního systému je zapotřebí věnovat pozornost detailnímu návrhu architektury, kterým se bude aplikace řídit. Je důležité navrhnout správnou a přehlednou strukturu, která bude natolik obecná, aby pokryla celý návrh informačního systému. K volbě architektury se musí přistupovat na úplném počátku vývoje aplikace.

Architektura aplikace řeší rozdělení aplikace, aplikačních dat, procesů i datových toků do logických celků, které jsou co možná nejvíce elementární.

8.2.1 Přímý model

Velmi jednoduchá architektura [18], určená především pro statické webové stránky. Přímý model z důvodu, že zobrazované stránky se otevírají ve webovém prohlížeči pomocí URL, která přímo odkazuje na soubor. Na webové stránce jsou pak odkazy na další stránky určeny absolutně, tj. další URL odkazující na další soubor. Řízení je tak založeno na přesně zapsaném odkazu a parametrech GET a POST zaslaném od klienta. Při použití této architektury je důležitým prvkem stanovení přehledné adresářové struktury.

Schéma přímého modelu:



Obrázek 5: Přímý model

8.2.2 Model s řízením

Pokročilejší architektura zavádí řízení (Controller) [18], které na základě přijatých dat od klienta (aktuálního URL, parametrů GET a POST a aktuálního stavu aplikace) rozhodne, který soubor se bude zobrazovat. Tím centralizuje logiku, která může využít funkcí, nebo zde vzniká prostor na řešení přístupových práv. Použití tohoto modulu už nabízí mnohem větší možnosti a nabízí se jako řešení pro interaktivní webové stránky.

Schéma modelu s řízením:

Obrázek 6: Model s řízením

8.2.3 Model–View–Controller

Architektura Model–View–Controller (MVC) je jednou z nejpoužívanějších návrhových architektur, obzvláště pak u webových aplikací. Využívá výhod dvou předchozích modelů a přidává řadu dalších.

Koncept MVC byl prvně využit v knihovnách tříd programovacího jazyka Smalltalk [19]. Typicky se využívá při programování uživatelského rozhraní, kde se používá klasického postupu vstup – zpracování – výstup. Rozšíření MVC můžeme připisovat jazyku Java, kde je Model obvykle realizován třídami definovanými v JavaBeans komponentám [20]. View jsou generovány díky JavaServer Pages a Controller jako množina servletů.

Rysem MVC je koncept, který se skládá ze tří logických částí Model, View a Controller. Každá z nich je v rámci aplikaci odpovědná za jinou přesně definovanou činnost. Vztah všech částí je zobrazen na obrázku 7.

8.2.3.1 Model

Reprezentuje data, nad kterými aplikace pracuje, poskytuje prostředky k datovému úložišti a operace s ním. Model také pracuje s aktuálním stavem aplikace a jeho budoucím posuvem, rozhoduje co a jak se bude vykonávat.

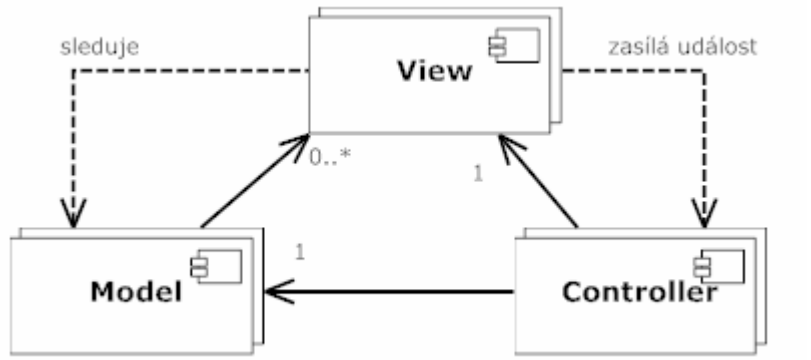
8.2.3.2 View

Jak už z názvu vyplývá, View má roli zobrazování. Zajišťuje grafický výstup aplikace. Přes Model přistupuje k aktuálním datům a stavu systému, který se má zobrazit.

Při změně stavu Modelu zajistí zobrazení aktuálního stavu v grafické podobě. U webových aplikací View generuje HTML kód, který se má zobrazit ve webovém prohlížeči.

8.2.3.3 Controller

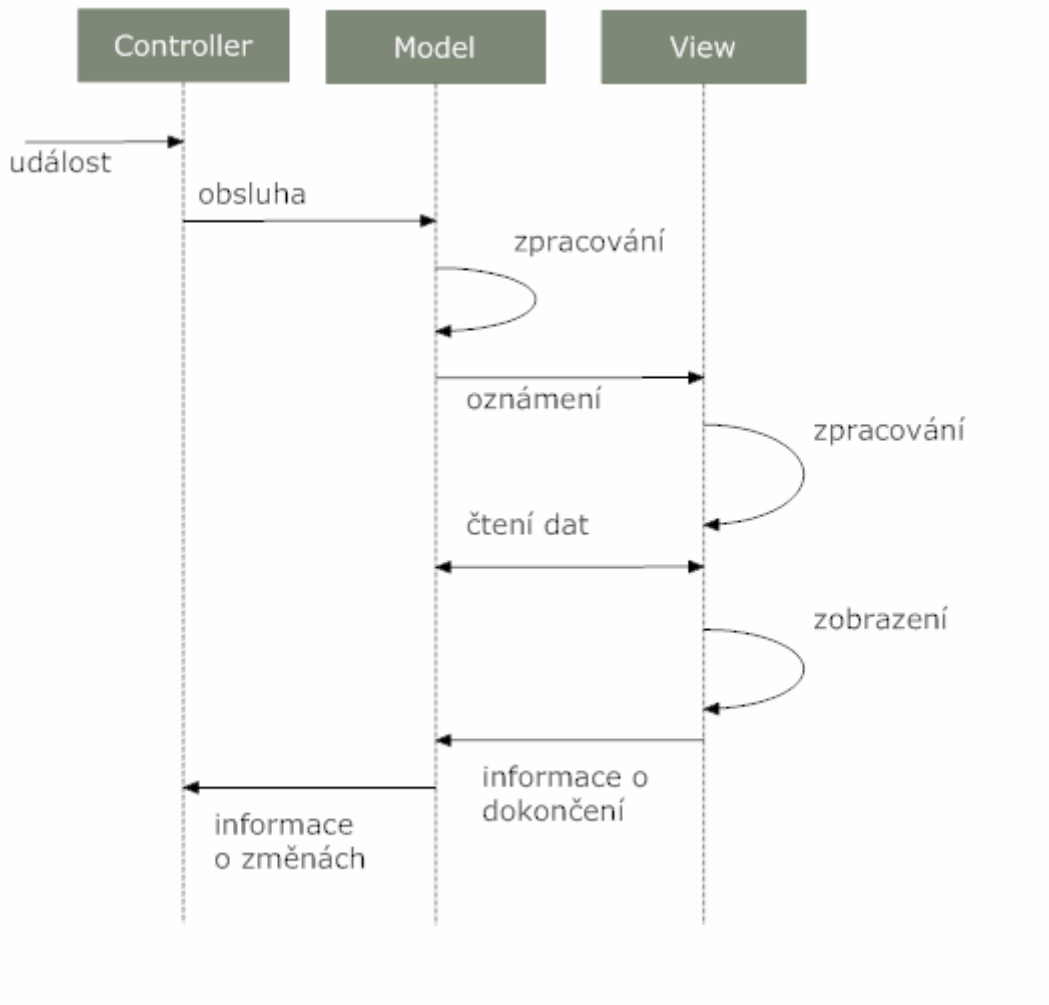
Zajišťuje logiku chování celé aplikace a v určité míře komunikaci mezi Modelem a View. Controller reaguje na události vyvolané uživatelem, přijímá všechny vstupy. Po přijetí jakékoliv události (např. stisk tlačítka) ji zpracuje a upraví výstupní objekty View. Samozřejmě podle potřeby spolupracuje s Modelem, kterému předává požadavky na změny, nebo si od něj vyžádá získání nových informací z datového úložiště. Ve webových aplikacích je hlavním vstupem přijímaným Controllerem parametr GET, nebo POST.



Obrázek 7: Vztah mezi Model, View a Controller

Činnost MVC zachycuje scénář na obrázku 8, na kterém je vidět šíření informací o nové události a reakce na ni. Činnosti probíhají v tomto pořadí:

1. Komponenta Controller reaguje na uživatelský vstup, který zpracuje.
2. Controller zavolá Model s žádostí o obsloužení zpracovaných vstupních dat.
3. Komponenta Model vykoná požadovanou službu. Výsledkem je změna interních dat.
4. Komponenta Model uvědomí všechny komponenty View (pohledy) o změnách.
5. Komponenta View pak požaduje změněná data v komponentě model a zobrazí je na výstup.
6. Nakonec se navzájem uvědomí všechny komponenty o výsledku zpracování události.



Obrázek 8: Scénář obsloužení události pomocí diagramu sekvencí UML

8.3 Autorizace a přístupová práva

Řídící jednotka aplikace zodpovědná za řízení přístupu, kterou je v architektuře MVC Controller, zaručuje přístup oprávněných uživatelů k jednotlivým datům, funkcím a dalším prostředkům autentizovaného uživatele. Řídící jednotka chrání celý systém před zásahem neautorizovaných uživatelů.

Chápání pojmů:

- Autentizace [22] je proces pro jednoznačné ověření uživatele, obecně subjektu, který vstupuje do informačního systému. Typickým řešením v oblasti webových aplikací je přidělení uživatelského jména a hesla každému uživateli, pomocí kterých je provedena kontrola.
- Autorizace [21], [22] je ověření práv uživatele, zda-li má dostatečné oprávnění přistupovat k požadovanému modulu, nebo činnosti.

8.4 Řízení přístupu založené na rolích

Přístup jednotlivých uživatelů systému je určen jejich rolí. Role a jejich práva definuje administrátor systému, který má plná oprávnění přistupovat a pracovat se všemi daty v systému. Administrátor musí být schopen rozlišit odpovědnosti každé role. Každá role by měla mít přístup pouze k těm částem systém, se kterými skutečně potřebuje pracovat. Dále by měla mít nadefinovanou co možná nejmenší úroveň práv. Práva rolí v systému se vztahují ke každému modulu zvlášť, je proto nutné definovat práva pro vztah role-modul.

Práva jsou rozdělena na úrovně:

- Číst – umožňuje číst data v modulu, bez možnosti jejich modifikace. Pokud role nemá právo číst data vztahující se k některému z modulu je zbytečné, aby měla právo vyšší úrovně jako například smazat.
- Vložit – vkládání nových záznamů do informačního systému.
- Editovat – aktualizace dat.
- Smazat – odstranění záznamů.

8.5 Zachycení událostí

Zachytáváním událostí se myslí, vytváření záznamů o činnostech systému a každého z uživatelů. Vytváření takových to záznamů slouží především jako bezpečnostní mechanizmus.

Samotné ukládání je velice jednoduché, vytvářejí se tzv. logovací soubory, kde zpravidla každý řádek znamená záznam o jedné události.

Hlavními důvody pro zachytávání událostí jsou:

- Detekce chyb v systému – zpětnou analýzou záznamů v souboru snadno odhalíme místo, kde a kdy dochází k chybě.
- Odhalení útoku – díky logovacím souborům můžeme odhalit slabá místa v systému. Nebo díky speciálních analytických aplikací, které pracují v reálném čase lze získat informace o právě probíhaném útoku.
- Informace o uživatelích – pokud bude systém pravidelně ukládat informace o pohybu uživatele v systému, je například možné zjistit co a jak by danému uživateli více vyhovovalo a podle těchto poznatků systém zdokonalit.

Další otázkou je, jaké události odchytávat a pořizovat o nich záznamy, alespoň základním minimem je vytváření záznamu o:

- Chybách, kde a za jakých okolností vznikly.
- Smazání záznamu z databáze, kdy a jaký uživatel mazal data.
- Přihlášení uživatele.
- Nepodařený pokus o přihlášení do systému

Zachytávání událostí by mělo být v systému implementováno na místě, kde se všechny tyto informace zpracovávají. U použitého návrhu MVC má činnost implementován Controller.

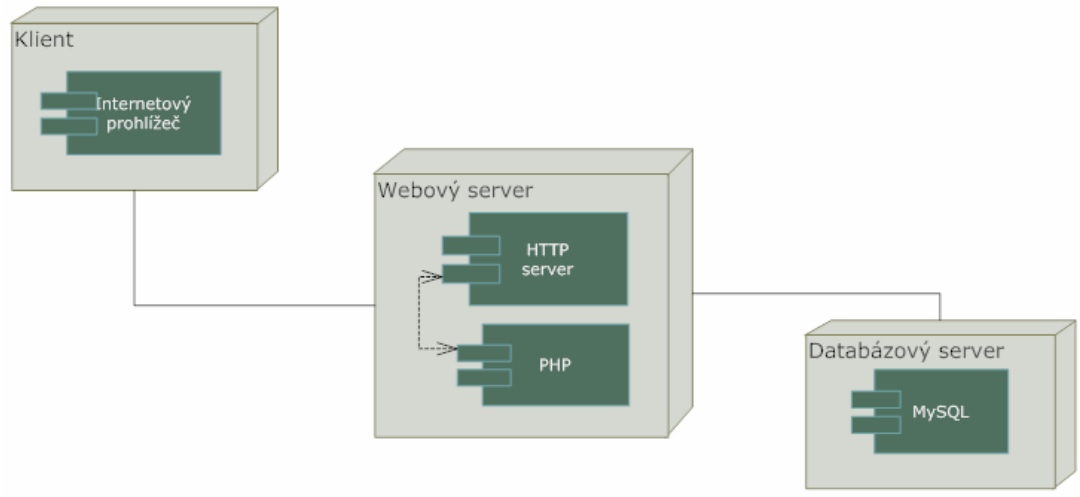
8.6 Návrh architektury

V našem případě se jedná o portálové řešení [22], tedy o aplikaci na postavenou na architektuře klient-server. Serverem se rozumí webový server (např. Apache) a klientem internetový prohlížeč.

Internetový prohlížeč na straně klienta zobrazuje data ze strany webového serveru. Jak je již obsaženo v požadavcích na aplikaci, je důležité, aby byl program plně funkční ve většině dnes používaných prohlížečích [23].

Pro implementace informačního systému jsem zvolil skriptovací programovací jazyk PHP. PHP 5 je objektově orientovaným programovacím jazykem určený především pro vývoj webových aplikací. PHP bude zajišťovat veškerou aplikační logiku informačního systému. Díky velmi dobré podpoře práce s relačními databázemi MySQL je jazyk PHP ideálním pro řešení portálového informačního systému.

Prezentační vrstvou je na obrázku 9 klient, tedy jeho internetový prohlížeč. Při požadavku na webový server je tento požadavek zpracován samotným PHP, pokud je zapotřebí, tak PHP komunikuje s databázovým serverem. PHP skripty jsou prováděny na straně serveru a klientovi je formou jazyka HTML zobrazen až výsledek těchto skriptů.



Obrázek 9: Diagram nasazení použitý pro návrh architektury

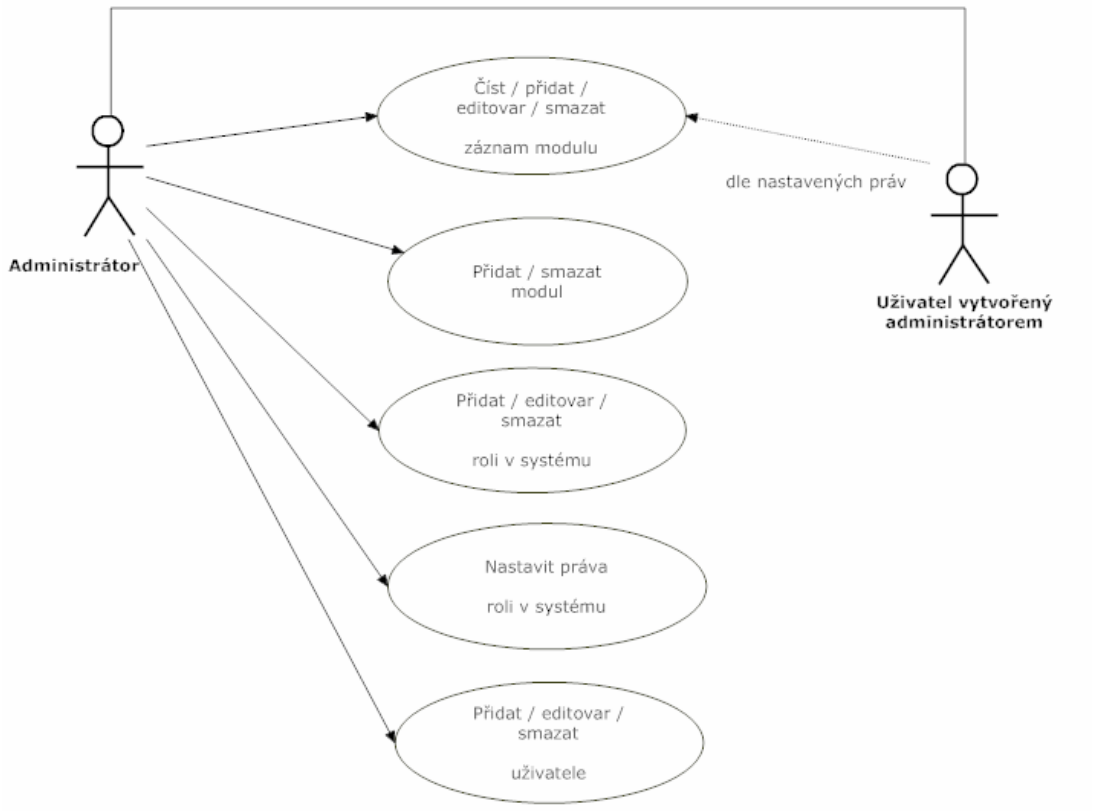
8.7 Jádro aplikace

Jádro aplikace je hlavní částí celého systému. Na tomto jádře je pak vhodnou kombinací modulů sestaven jakýkoliv informační systém. Jádro aplikace je proto postavené na obecném základu a jeho funkce je zprostředkování dat, která se čerpají z modulů. Samotné jádro řídí chod aplikace, včetně správy uživatelů, uživatelských rolí a správy nad moduly.

8.7.1 Diagram případů užití

Na obrázku 10 jsou znázorněny případy užití (Use Case) jádra aplikace. Hlavní rolí v systému je administrátor, který má neomezené možnosti přístupu ke všem datům. Pouze administrátor definuje role ostatních uživatelů v systému a rolím určuje úrovně práv k modulům. Administrátor dále přidává nové moduly do systému. Po přidání nového modulu je potřeba dodefinovat práva k tomuto modulu, protože s nově vloženým modulem nemá žádná role právo s ním pracovat. Administrátor také vytváří nové uživatele a těmto uživatelům přiřazuje roli v systému.

Administrátorem vytvořený uživatel má podle práv své role omezen přístup k modulům v systému.

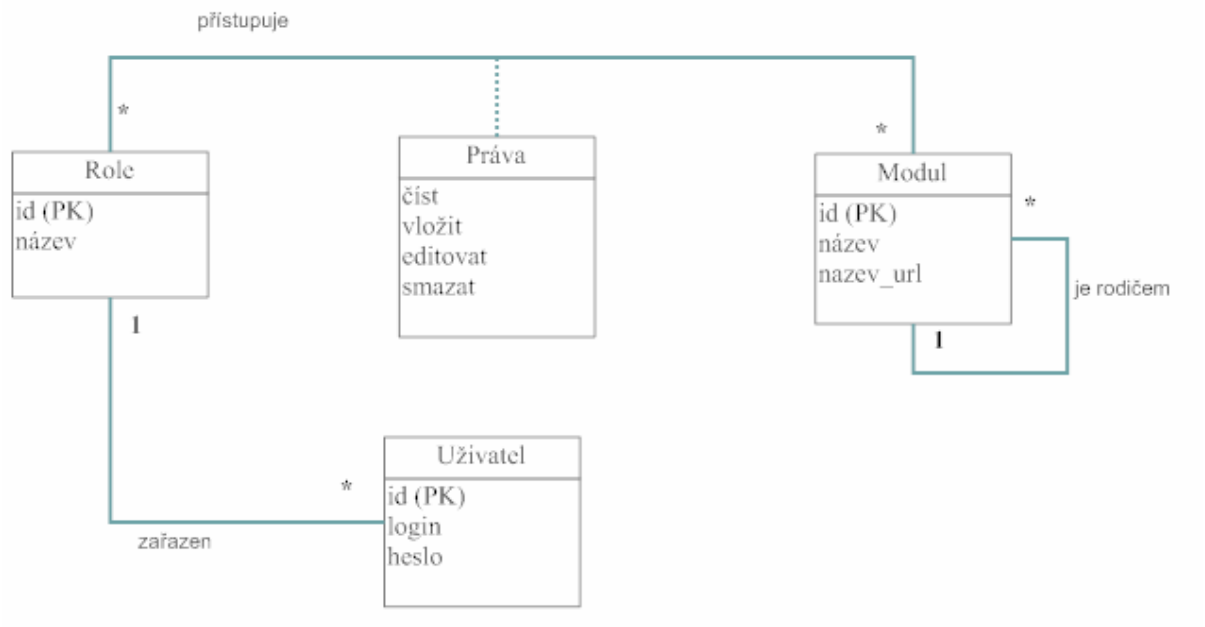


Obrázek 10: Případy užití jádra aplikace

8.7.2 Konceptuální datový model

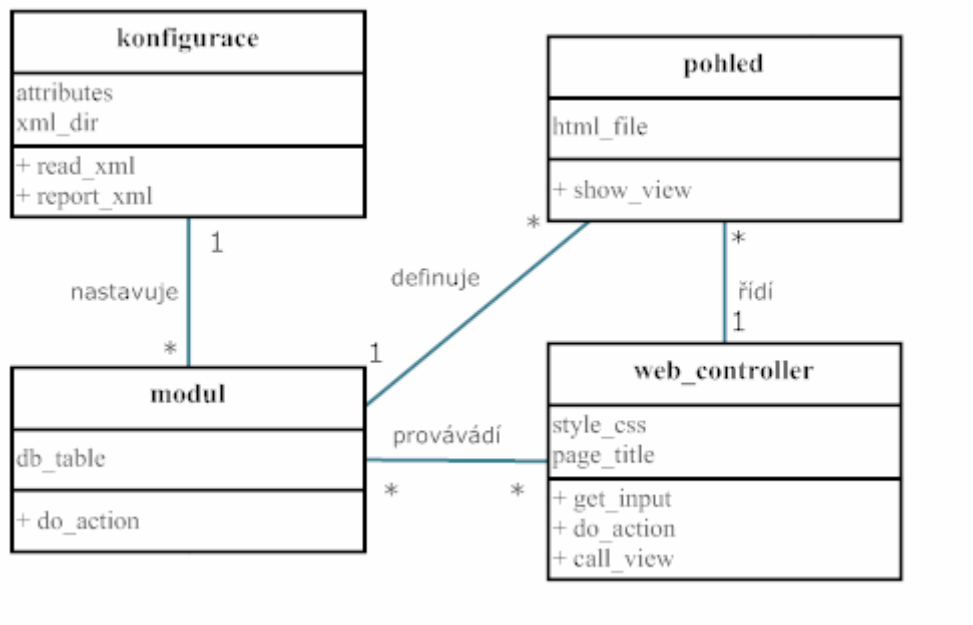
Obrázek 11 zobrazuje konceptuální datový model jádra informačního systému vytvoření na základě požadavků, které jsou na něj kladeny. Entita uživatel představuje uživatele systému. Každý uživatel musí mít pevně přiřazené uživatelské jméno a heslo, které složí k přístupu do systému. Implicitním uživatelem je administrátor. Každému uživateli musí být přiřazena role, naopak roli může mít obecně více uživatelů, jen roli administrátor musí mít přiřazen alespoň jeden uživatel. Samotná entita role obsahuje jen název role. Systém bude umožňovat vytváření nových rolí a k nim přidělování práv. Práva v systému určují, zda uživatel s přiřazenou konkrétní rolí může pracovat s daty modulů k nimž jsou práva vztažena.

Samotná entita modul zastupuje moduly, které budou do informačního systému vloženy administrátorem. Obecně se může jednat o modul s různorodou funkcionalitou, a proto je důležité správné vytvoření vztahu mezi rolí a modulem.



Obrázek 11: Konceptuální ER diagram

Modul je základní stavební jednotkou celé aplikace. Z modulů se skládá výsledná aplikace, v našem případě školní informační systém. Jak již bylo řečeno, jeden fyzický modul, v systému implementovaný jako třída, může využívat několik modulů, při implementaci například děděním této třídy. Proto je dobré, už v této fázi návrhu jádra systému vytvořit obecné schéma modulu, požadavků na jeho funkcionalitu.



Obrázek 12: Diagram tříd jádra aplikace

Na obrázku 12 je návrhový diagram tříd jádra aplikace. Třída web_controller slouží ke zpracování vstupních dat, tedy přebírá činnost Controlleru v architektuře MVC. Dle aktuálního stavu aplikace a přijatých dat komunikuje s třídou modul (Model v MVC), která zastupuje jakýkoliv rozšiřitelný modul aplikace. Modul pracuje s daty datového úložiště podle specifikací od třídy web_controller a třídy konfigurace, která zpracovává konfigurační soubor modulu. Třída pohled (View v MVC) pak data získaná od modulu zobrazí.

Už samotné jádro aplikace poskytuje nadstavbovým modulům základ pro jejich činnost a to třídou modul. Předpokládáme, že činností modulů bude provádění operací s nimi vztaženými daty. Potom jádro aplikace ulehčí práci modulů již předpřipravenými metodami třídy modul, ke kterým bude mít nový modul vložený do informačního systému přístup zděděním této třídy.

Funkcí třídy modul je umožnit práci s daty v datovém úložišti. Veškeré funkce by měli být pouze elementárním návrhem, který se při implementaci dalších návazných modulů může rozšířit.

Mezi tyto funkce zcela jistě patří:

- Přehledné a jednotné zobrazení vše dat z datového úložiště (databázové tabulky).
- Přehledné a jednotné zobrazení konkrétních dat (řádek v databázové tabulce).
- Vložení nového záznamu (přidání záznamu do databáze).
- Editace vložených dat.
- Mazání dat z datového úložiště.

Aby bylo možné provádět operace v datovém úložišti, je zapotřebí získat informace, s kterými daty pracovat. Například nad jakou tabulkou v databázi se budou tyto úlohy provádět, ale také jak

zpracovat data získaná od uživatele, která se mají vložit do databáze. Pro tento účel je zapotřebí navrhnout jak tento modul konfigurovat.

8.7.3 Konfigurace modulu

Je důležité navrhnout co nejjednodušší formát a strukturu konfiguračního souboru. Ideálním v tomto případě je použití XML, kde se pomocí předem stanovených pravidel nadefinují konfigurační data. Volba XML přináší výhodu, kdy jednoduchý modul zvládne nastavit i uživatel, který není programátorem. Konfigurace modulů je ovšem práce administrátora, který musí být dostatečně poučen.

Od konfigurace modulu požadujeme:

- určení tabulky v databázi,
- jednoduchý slovní popis toho, co konkrétní modul provádí,
- možnost definovat text nápovědy, vysvětlující jak s modulem pracovat,
- která vybraná data (sloupce v databázové tabulce) zobrazovat,
- definování všech dat a jejich typů (definování formulářových elementů, jejich návratových hodnot a formát zobrazení).

V příloze A je kompletní návrh DTD (definice typu dokumentu) konfiguračního XML souboru. Existence většiny elementů není povinná, tyto elementy pouze rozšiřují funkčnost modulu. Význam elementů *h1*, *help* a *text* je pouze informativní, při práci s tímto modulem se pak zobrazují na stránce. Element *text* by měl obsahovat textový popis modulu, ale může například obsahovat i nápovědu jak s tímto modulem pracovat. Elementy *select*, *where* a *order_by* již určují jaké informace a jakým způsobem zobrazovat ve výpisu všech dat z tabulky v databázi, jak už jejich název napovídá je z těchto elementů složen SQL dotaz a jeho výsledek je poté zobrazen uživateli.

Element *form* potom ve svých vnořených elementech definuje jak zobrazovat webový formulář pro přidání nebo editaci záznamu v databázi. Jednotlivé části formuláře jsou specifikovány v elementu *elements*. Formulář je často tvořen se závislostí na tabulku databáze, s kterou pracuje, element *col* proto určuje sloupec v této tabulce. Formát v jakém se políčko formuláře zobrazí uživateli určuje element, ten má předdefinovaný obsah, kterými mohou být:

- input – vstupní pole pro editaci textové informace,
- textarea – velké vstupní pole s textovou informací,
- select – výběrové pole, možnost výběru jedné položky z N,
- multipleselect – výběrové pole s možností výběru až N položek z N,
- file – umožní zadat soubot,
- editor – textové pole s možností úpravy rozsáhlého textu pomocí wysiwyg editoru.

Dále element nadpis, pro informaci uživatele jaké políčko edituje. A element type, který určuje hodnotu formulářového pole. Například formulářový element *input* s typem *date*, umožní uživateli zadat pouze platné datum ve správném formátu. Další možnosti a příklady konfiguračních souborů jsou uvedeny v příloze B.

Podle takto vytvořeného DTD, může být příkladem jednoduchý konfigurační soubor:

```
<modul>
<h1>Texty</h1>
<text>Vytváření a editace textů.</text>
<form>
      <elements>
            <col>nadpis</col>
            <element>input</element>
            <nazev>Nadpis</nazev>
            <size>40</size>
            <type>text</type>
            <css>style="background: #CCC;"</css>
            <povinny>1</povinny>
      </elements>
      <elements>
            <col>datum</col>
            <element>input</element>
            <nazev>Vytvořeno</nazev>
            <size>10</size>
            <type>date</type>
      </elements>
      <elements>
            <col>text</col>
            <element>editor</element>
            <nazev>Text</nazev>
            <size>500x600</size>
            <type>profi</type>
      </elements>
</form>
<order_by>order by nadpis</order_by>
<select>nadpis</select>
</modul>
```

8.7.4 Validace vstupních dat

Pro udržení konzistence datového úložiště, z pohledu bezpečnosti a funkčnosti celého systému je zapotřebí validace vstupních dat od uživatelů. Jediným zdrojem vstupních dat, které zadává samotný uživatel je formulář svázaný s určitým modulem. Pokud tento modul využívá konfigurační soubor XML, je zapotřebí už při vytváření konfiguračního souboru rozmýšlet jakých hodnot může element ve formuláři nabývat hodnot. Podle takto vytvořeného XML dokumentu pak kontrolní mechanismus modulu validuje vstupní data.

Jak již bylo uvedeno v příkladu výše, data získaná od uživatele se vylisují pomocí předepsaného typu v elementu *type* konfiguračního souboru. Pokud není uveden typ výstupních dat, automaticky se předpokládá jakýkoliv vstup za správný, je tedy pouze na tvůrci XML souboru na jejich definici.

U každé formulářové položky může být ještě uveden element *povinny*, který určuje povinnost tento údaj vyplnit, aby byl formulář úspěšně zpracován.

8.7.5 Reprezentace modulu v databázi

Konfigurační soubor XML je určitým způsobem svázán s databází a nemusí se vždy jednat pouze o jednu tabulku databáze. Definicí formulářových elementů určuje sloupce v tabulce, dokonce i jejich typ a hodnoty, kterých mohou nabývat.

Díky elementům XML dokumentu, validací hodnot formuláře HTML a metodám modulu je možné určovat vazby mezi databázovými tabulkami.

Vztah typu 1:1 znamená, že právě jednomu záznamu v první tabulce odpovídá právě jeden záznam v tabulce druhé. Jednotlivé záznamy v tabulkách jsou tímto přímo spojeny. Tento typ vztahu je používán velmi málo, protože takto spojené záznamy lze vložit do jedné tabulky v databázi. Proto se tento vztah v aplikace neuvažuje.

Vztah typu 1:N umožňuje, aby jednomu záznamu v první tabulce odpovídalo více záznamů z druhé tabulky. V databázi je tato závislost tvořena pomocí cizího klíče (foreign key).

Příkladem vztahu 1:N může být vztah student – třída, kdy je student zařazen do právě jedné třídy a třída obsahuje více studentů. Realizace v databázi pomocí cizího klíče, znamená přidáním jednoho sloupce do tabulky student, který obsahuje hodnotu primárního klíče záznamu z tabulky tříd.

Tento vztah je také možné řešit přímo v konfiguračním souboru, kde se cizí klíč zapíše jako v příkladě 7. Uživateli se při práci s modulem zobrazí výběrové pole ve formuláři, ve kterém je možné vybrat jeden z více záznamů.

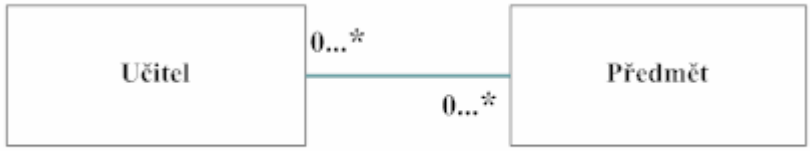
Příklad 7: Zápis cizího klíče v konfiguračním souboru

```
<elements>
      <col>trida_id</col>
      <element>select</element>
      <nazev>Třída</nazev>
      <povinny>1</povinny>
</elements>
```

Vztah typu M:N je méně častým typem vztahu mezi entitami. Protože se tento vztah velmi těžko převádí na schéma relační databáze, je převeden na vztahy typu 1:N pomocí vazební entitní množiny.

Pro vytvoření takové vztahu je nutná dekompozice, tudíž vytvoření tzv. vazební tabulky. Vazební tabulka pak obsahuje n-ární spojení primárních klíčů entit. Princip těchto operací ilustruje příklad 8.

Příklad 8: Vztahu M:N může být mezi učiteli a předměty, kdy učitel učí M předmětů a předmět je vyučován N učiteli.



Obrázek 13: Vztah učitel – předmět

dekompozicí získáme schéma

Obrázek 14: Dekompozice vztahu

Pokud nastane situace, kdy je nutné využít tohoto vztahu, lze jej jednoduše zapsat v konfiguračním souboru a to obdobným způsobem jako v příkladu 9. Vazební tabulka je určena elementem *saveto.* Uživateli modulu, ve kterém se použije toto řešení, je zobrazeno výběrové pole o možnosti výběru více záznamů. Výběrové pole je v jazyce HTML řešeno tagem select s atributem multiple.

Příklad 9: Řešení vztahu M:N

```
<elements>
      <col>predmety_id</col>
      <element>multipleselect</element>
      <nazev>Učí</nazev>
      <size>10</size>
      <saveto>predmety_ucitele</saveto>
      <db>0</db>
</elements>
```

Významy XML elementů jsou již popsány výše. Důležitým elementem je saveto, který definuje název vazební tabulky.

8.7.6 Zpracování výstupu

Při zpracování výstupu je hlavním cílem, jak už architektura MVC naznačuje odděleni logické a prezentační vrstvy. Třída modul zpracuje data, zastává tedy roli Modelu v MVC a pomocí třídy pohled data uživateli zobrazí. Třída pohled využívá předpřipravených šablon pro zobrazení dat uživateli. Šablony jsou soubory jazyka PHP tří druhů, do kterých se pouze doplní proměnné.

1. Šablona zobrazující obsah tabulek v databázi, samozřejmě co zobrazit lze také nadefinovat v konfiguračním XML souboru.
2. Šablona, která zobrazuje pouze detail jednoho záznamu v databázi.
3. Šablona, která podle zpracovaných dat zobrazí uživateli formulář pro vkládání nových záznamů nebo editaci záznamu.

Pokud ani jedna ze šablon nepokrývá data, která jsou v rámci modulu zobrazována, musí být pro tento modul vytvořena zvláštní šablona.

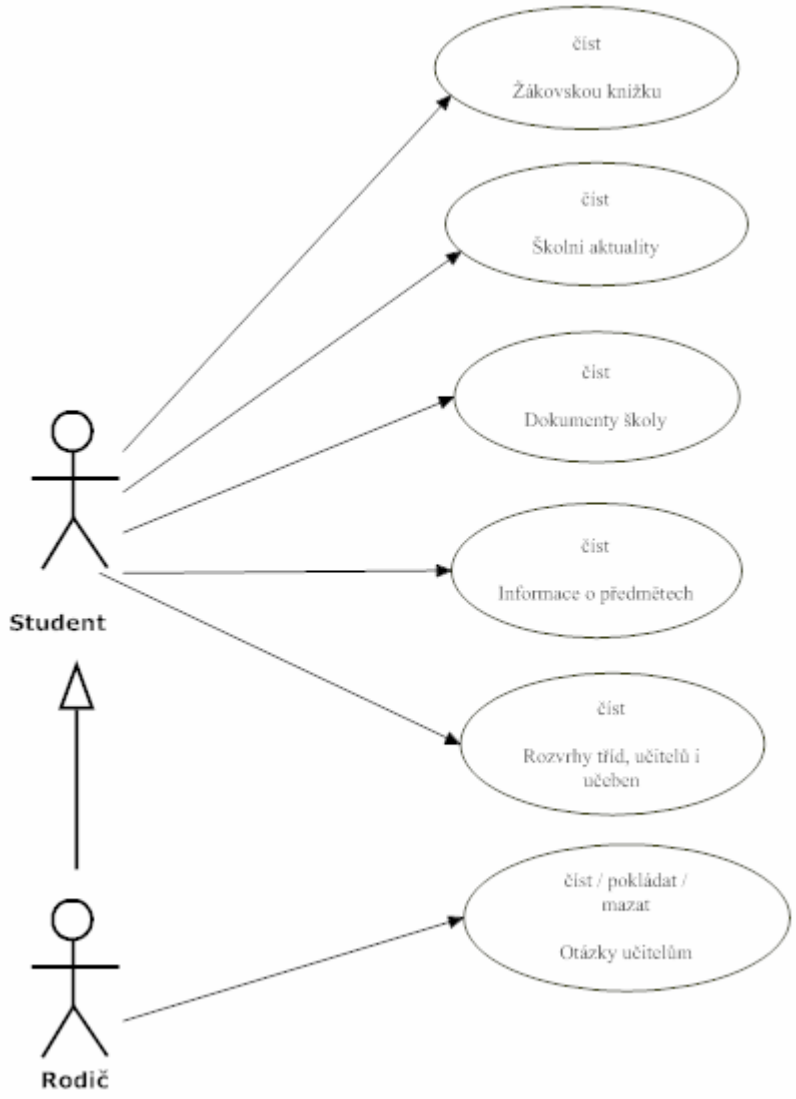
8.8 Návrh školního informačního systému

Proces návrhu se nyní zaměří na samotný školní informační systém, v této kapitole se již nevěnuji obecnému řešení informačních systémů, ale právě navrhovanému školnímu.

8.8.1 Případy užití

Na obrázku 15 je zobrazen model užití pro role student a rodič. Role studenta má v systému nejmenší oprávnění, student nemá u žádnému modulu právo vkládání nových záznamů, ani jejich editaci. Student přistupuje k datům o jeho klasifikaci formou žákovské knížky, vytvořeným rozvrhům, informacím o aktuálním dění ve škole a informacím o předmětu.

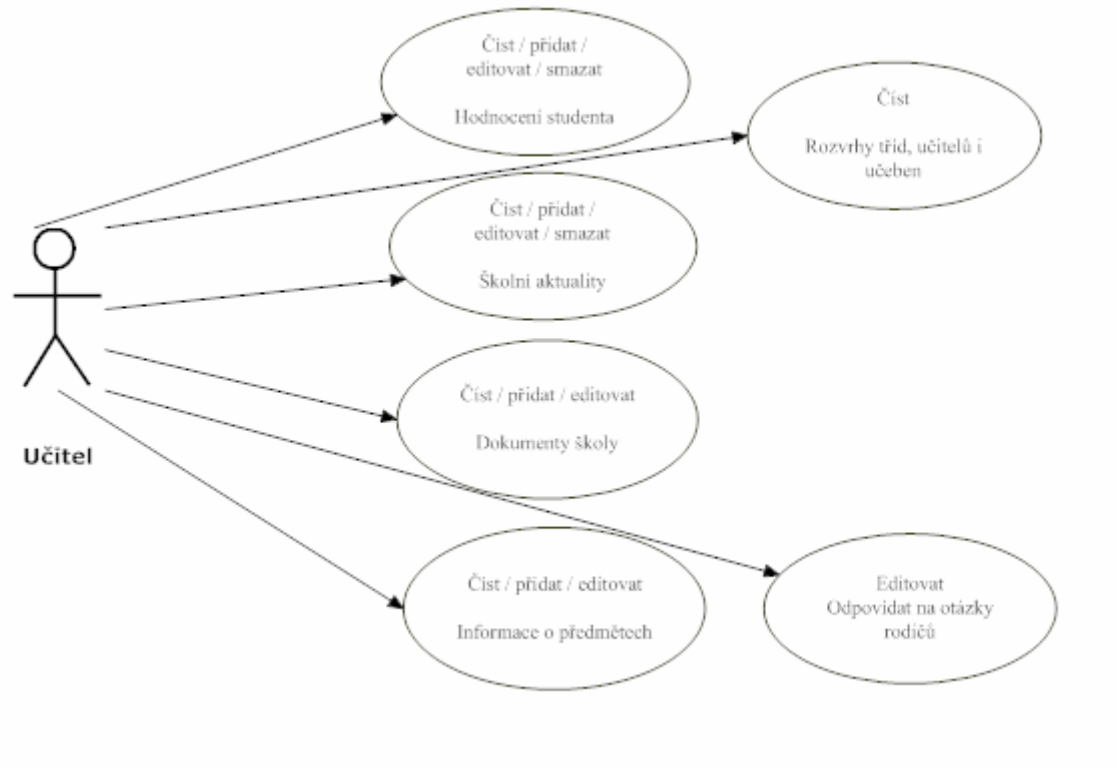
Dalším aktérem na obrázku 15 rodič, který má stejná oprávnění jako student, ale navíc mu systém umožňuje komunikaci s učiteli.



Obrázek 15: Model užití role student a rodič

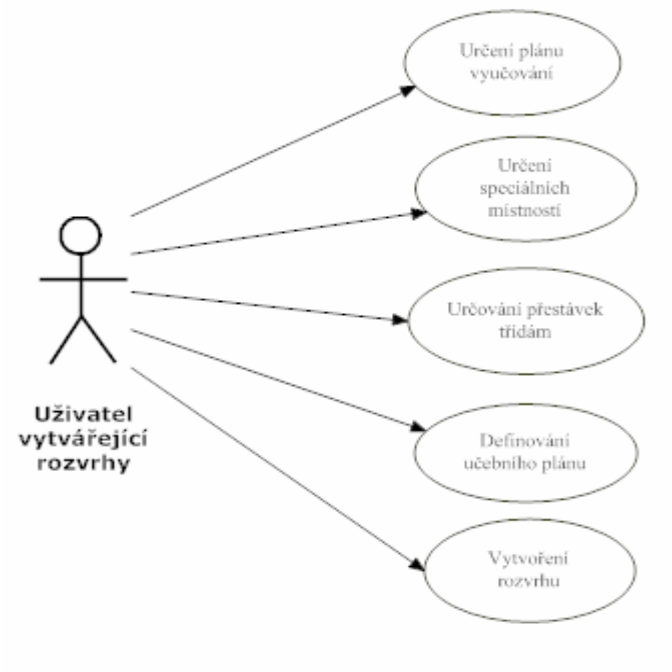
Dalším modelem užití je na obrázku 16 model pro roli učitel. Učitel už nemá natolik omezená práva, jako například student. Učitel se již podílí na zaznamenávání informací do školního informačního systému. Učitel může vytvářet školní aktuality, nebo doplňovat školní publikace. Jeho

hlavním úkolem je ovšem spravovat a aktualizovat informace o vyučovaných předmětech a samozřejmě klasifikace studentů. Učitel také komunikuje s rodiči studentů, není mu ale přiděleno právo tuto komunikace začít, nebo jí ukončit smazáním záznamu.



Obrázek 16: Model užití role učitel

Model užití na obrázku 17 zobrazuje možnosti uživatele, který bude v informačním systému vytvářet rozvrhy. Pouze tento uživatel může definovat omezující podmínky pro vytvoření rozvrhu. Určením učebního plánu definuje vyučování předmětů v jednotlivých ročnících. Dále určuje, které učebny mají speciální význam, tedy výuka jakých předmětů v nich může probíhat. Nastavuje průběh vyučování, určením hodinových přestávek na oběd.



Obrázek 17: Model užití pro uživatele, který vytváří rozvrhy

8.8.2 Struktura modulů

Po navržení jádra aplikace je zapotřebí navrhnout strukturu modulů, které budou tvořit školní informační systém. V návrhu je výhodné využít samotné jádro aplikace, které pokrývá požadavky na většinu modulů. Vytvoření struktury a závislosti modulů je nejdůležitějším krokem. Moduly budou konfigurovány XML soubory, dle navrženého DTD dokumentu.

Na obrázku 18 je návrh architektury modulů. Všechny tyto moduly dědí obecný modul navržený pro jádro aplikace.

Například komponenta třídy. Její třída *tridy* zdědí vlastnosti jádra aplikace, pouze pro detailní nastavení zobrazovaných dat a konfiguraci webového formuláře využívá konfigurační soubor *tridy.xml*. Provázáním databázového schématu a konfiguračních souborů je dáno, že třída *tridy* bude pracovat s daty v databázové tabulce *tridy*. Při zobrazení přehledu dat modulu v informačním systému se dle řádků v konfiguračním souboru

```
<order_by>order by nazev</order_by>
<select>nazev, ucitel_id</select>
```

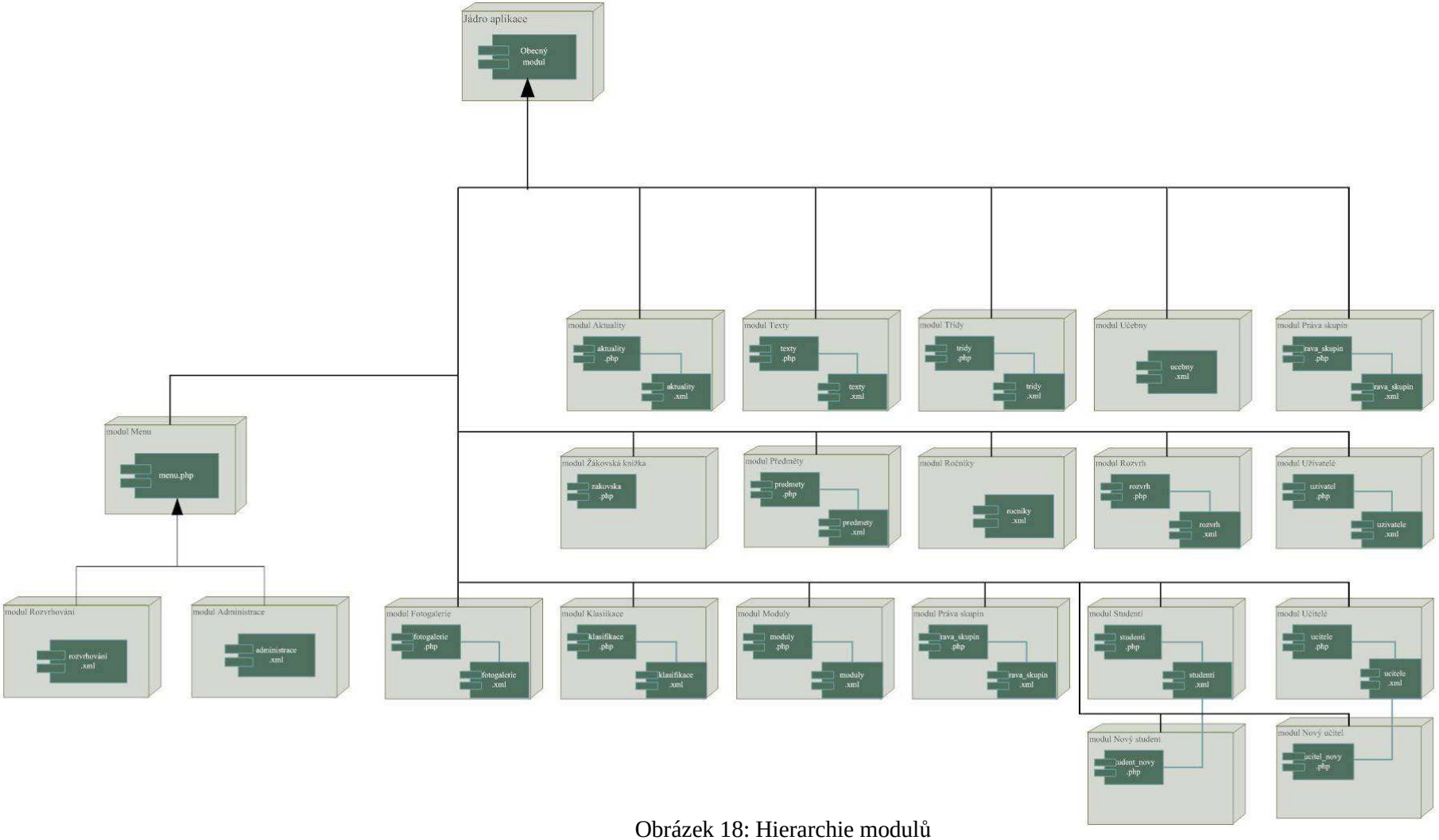
zobrazí všechny řádky tabulky seřazené podle sloupce název. Zobrazená tabulka bude obsahovat jen sloupce název a učitel. Ostatní data budou automaticky zobrazena v detailu jednoho řádku tabulky.
Pro vložení nového záznamu se zobrazí formulář, načtený také z konfiguračního souboru. Použití třídy *tridy* je v tomto modulu nutné, protože sloupec *ucitel_id* je cizím klíčem do tabulky učitelů.

Třída *tridy* zaručí načtení používaných dat z tabulky učitelů. Pokud bychom nemuseli tento případ řešit, stačilo by pouhé použití konfiguračního souboru a jádro aplikace by tyto operace provedlo samo (tento problém je řešitelný na úrovni jádra aplikace). Bez použití třídy, ale pouhého konfiguračního souboru je například řešen modul učeben.

Na stejném principu jako je ukázána práce modulu třídy, mohou pracovat i ostatní moduly. Znovu je zde kladen důraz na správné sestavení konfiguračního souboru a je vidět, že pokud je XML dokument správně nastaven odlehčuje tím práci třídě, která s ním pracuje.

Neobvyklým řešením je modul menu. Tento modul vytváří pouhou složku v systémovém menu, která může obsahovat další moduly, ale nepracuje s žádnými daty v databázi. Proto je modul menu řešen pouze třídou, bez konfiguračního souboru. Dává totiž modulům, které ho dědí další možnosti. A to především tu možnost, že komponenty administrace a rozvrhování nepotřebují vlastní třídy.

Dalším modulem, bez využití konfiguračního souboru bude modul žákovské knížky. Přístup k tomuto modulu bude mít pouze uživatel s rolí student, nebo rodič. Modul nebude žádným způsobem měnit data v databázi, ale bude data číst a zobrazovat. Přihlášenému studentovi zobrazí přehled jeho studijních výsledků, samozřejmě přihlášený uživatel (rodič) může přistupovat k údajům studenta, kterého je zákonným zástupcem.



Obrázek 18: Hierarchie modulů

8.8.3 Rozvrhovací algoritmus

Pro vytvoření školního týdenního rozvrhu je potřeba vytvořit vztah třída – učitel – předmět – učebna – hodina tak, aby byly splněny omezující podmínky.

Podmínky plynoucí už ze samého rozvrhu jsou, aby jedna třída měla v jednu vyučovací hodinu měla maximálně jeden předmět. Obdobné pravidlo platí pro učitele, kdy učitel v jednu vyučovací hodinu může učit pouze jeden předmět v dané třídě, a také pro učebny, kde se může konat vyučování pouze jedné třídy. Obecně může být jeden předmět vyučován vícekrát v jednu vyučovací hodinu, pokud jsou splněny předchozí podmínky, tedy vyučuje ho vždy jiný učitel v rozdílných třídách.

Učební plán školy přesně určuje vztahy mezi třídami a předměty, známe kolik hodin týdně má třída určitý předmět. Pokud máme konečnou množinu tříd t_1, t_2, … , t_i, které mají zapsané předměty p_1, p_2, … p_j v určitém týdenním rozsahu, pak je učební plán dán trojicí

$$(t_r, p_s, h),\ r \in 1..i \wedge s \in 1..j \qquad (8)$$

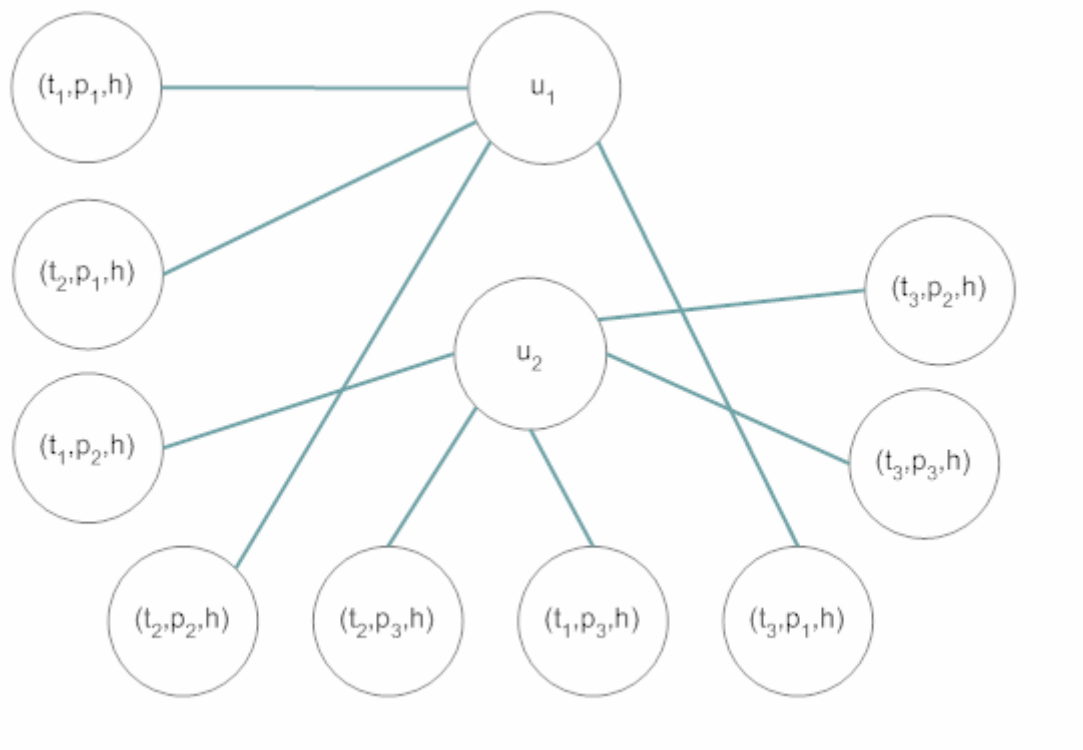
která vyjadřuje, že třída t_r má vyučování předmětu p_s a to h hodin týdně.

Nyní už jen zbývá doplnění o vztah mezi trojicí (t_r, p_s, h) a učitelem. Který předmět učí jaký učitel je také známé z informací dodané školou, ovšem jeden předmět p_s může učit až n učitelů. Proto jsem zvolil metodu úměrného rozdělení vyučování mezi učitele daného předmět. Například pokud je předmět p_s v učebním plánu pro dvě třídy a tento předmět mohou učit dva učitelé, pak každý z nich bude učit jednu třídu. Tato metoda je do určité míry variabilní, kdy například chceme, aby obě třídy mělo stejného učitele, tak do systému nebudeme zadávat druhého učitele, který tento předmět může učit.

Tento problém rozvrhování může být převeden na problém barvení grafu. Nejdříve je nutné problém převést na graf. Jako uzel grafu budeme znázorňovat učitele u_1 … u_l a trojici (t_r, p_s, h) a hrana v grafu jako vztah mezi trojicí a učitelem.

Obrázek 19: Návrh vytvoření grafu

Obrázek 19 zobrazuje situaci, kdy třída t_r má předmět p_s h hodin týdně a jako vyučujícího na tento předmět má učitele u_v. Velmi jednoduchý příklad takto vytvořeného grafu pak může vypadat jako na obrázku 20.



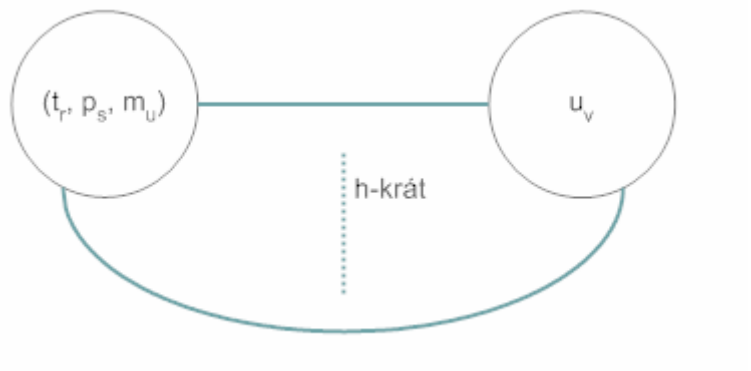
Obrázek 20: Příklad grafu

Další podstatnou informací, kterou lze jednoduše získat je, v které učebně bude vyučování pro trojici (t_r, p_s, h) probíhat. Pokud je předmětu p_s školou definována tzv. speciální místnost, ve které výuka tohoto předmětů musí z nějakého důvodu probíhat, může rovnou tuto učebnu přidat do vztahu s touto trojicí. Pokud pro předmět p_s není žádná speciální místnost určena vybereme jednu z místností m_1 až m_k, takovou, aby kapacita míst v učebně byla co nejmenší, ale zároveň aby platilo pravidlo počet studentů třídy t_r > kapacita učebny. Těmito operacemi získáme čtveřici (t_r, p_s, h, m_u), která říká třída t_r má předmět p_s v místnosti m_u h hodin za týden.

Přidáním místností do vztahů rozvrhu se promítne v grafu jako na obrázku 21, kdy jeden uzel je dán čtveřicí (t_r, p_s, h, m_u).

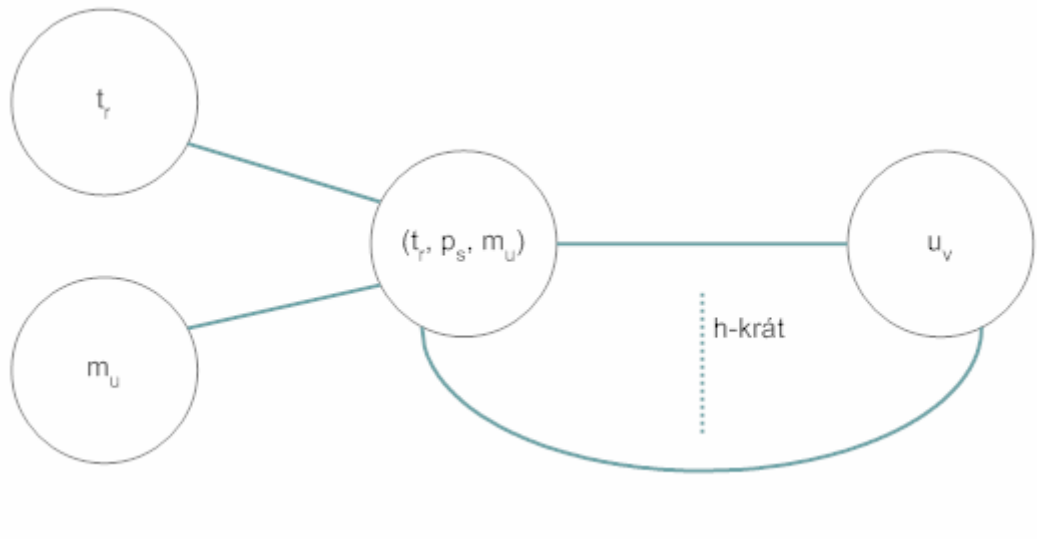
Obrázek 21: Návrh grafu

Pokud bychom chtěli takto vytvořený graf již obarvit, stále nedostaneme požadovaný týdenní rozvrh. Získaná barva hrany nám určuje, kterou hodinu v týdenním rozvrhu je předmět p_s v třídě t_r a místnosti m_u vyučován učitelem u_v. Ale potom potřebujeme těchto barev získat h, protože čtveřice (t_r, p_s, h, m_u) je v týdenním rozvrhu obsažena právě h-krát. Převést tuto situaci do grafu, znamená vytvoření multigrafu, kde vztah (hrana) mezi čtveřicí (t_r, p_s, h, m_u) a u_v je dán h hranami, přesněji řečeno trojicí (t_r, p_s, m_u) a u_v, protože hodnotu h je dána počtem hran a už ji nepotřebujeme. Takto upravený graf je na obrázku 22.



Obrázek 22: Vytvoření multigrafu

Aby byl graf kompletní, musíme ještě zavést uzly reprezentující třídu a místnost. Jelikož budeme barvit hrany grafu, tak by se mohlo stát, že v jednu hodinu (jedna barva) bude probíhat vyučování dvou tříd v jedné místnosti. Takovýto graf, sloužící jako šablona na vytvoření kompletního grafu je uveden na obrázku 23.



Obrázek 23: Graf reprezentující rozvrh

Obrázek 23 zobrazuje závislost mezi třídou t_r, předmětem p_s, který má v učebním plánu, místností mu, ve které bude vyučování probíhat a učitelem u_v, který bude učit h hodin týdně. Tímto způsobem vytvoříme graf pro celou školu, tedy všechny předměty, třídy, učitele a místnosti.

Nyní je potřeba tento graf obarvit, barvit budeme hrany grafu. Barva bude určovat číslo hodiny, kdy bude probíhat vyučování. Na barvení grafu využijeme metodu sekvenčního barvení, která dle [16] dává velmi slušné výsledky.

Algoritmus 8: Navrhované barvení grafu

1. Proměnná k se nastaví na 0.
2. Vybere se doposud neobarvená hrana mezi uzly trojice (t_r, p_s, m_u) a u_v.
3. Nalezení nejmenší přirozené číslo p takové, že není barvou žádné z hran spojující uzel (t_r, p_s, m_u) s jakýmkoliv jiným.
4. Hraně ((t_r, p_s, m_u), u_v) se přiřadí barva p.
5. Hraně ((t_r, p_s, m_u), m_u) se také přiřadí barva p.
6. Hraně ((t_r, p_s, m_u), t_r) se také přiřadí barva p.
7. Pokud p > k, pak se nastaví k = p.
8. Pokud nejsou obarveny všechny hrany tak se pokračuje krokem 2

Po průchodu tohoto algoritmu jsme zjistili, že graf je obarvitelný k barvami a pro každou n-tici učitel, třída, předmět, místnost známe barvu.

Nyní potřebujeme přepočítat barvu na číslo hodiny v rozvrhu. Nejdříve je zapotřebí jednoznačně označit každou školní hodinu. Řekněme, že týdenní rozvrh školy je dán 50 hodinami, 10 hodin denně 5 dní v týdnu (od pondělí do pátku). A čísla hodin jsou dána po dnech (po řádcích v rozvrhu), pondělní 1. hodina bude hodina číslo 1, druhá číslo 2 atd., úterní 1. hodina bude hodina číslo 11 atd. až páteční 10. hodina bude hodina číslo 50.

Pokud bychom i tímto způsobem přiřazovali barvy k číslům hodin, je velice pravděpodobné, že rozvrh by byl dán přibližně třemi dny v týdnu, každou vyučovací hodinu. Proto jsem zvolil opačný postup, kdy barvy označují číslo hodiny po sloupcích, například barva číslo 5 označuje páteční 1. hodinu a barva číslo 6 pondělní 2. hodinu.

8.8.3.1 Další omezující podmínky

Vzhledem k rozsáhlé oblasti školního rozvrhování je potřeba nastínit řešení dalších omezujících podmínek. Nastavení obdobných podmínek může být školu od školy jiné, každá škola má své zavedené postupy a zásady, na kterých si zakládá.

Přestávky

Jak už bylo řečeno v analýze školního informačního systému, škola je povinna dle zákona [14] definovat při vytváření rozvrhů povinou přestávku na oběd. Čas této přestávky určen ročníkem.

S touto situací musí rozvrhovací algoritmus počítat. Nejprve je nutné, aby uživatel systému, nejlépe ten, který vytváří rozvrhy, nastavil po které hodině má daný ročník povinnou přestávku.

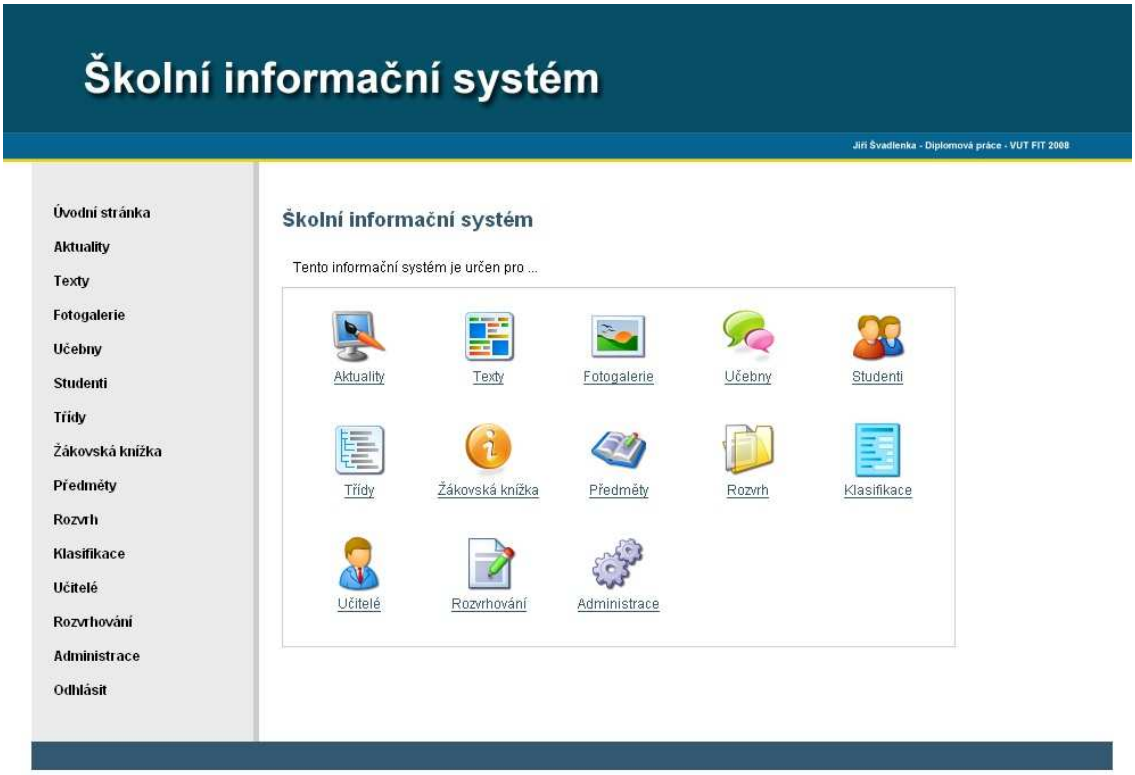
Řešením v samotném algoritmu je vytvoření fiktivního uzlu a jeho spojení hranou s uzlem (t_r, p_s, m_u), kde t_r je třída, p_s předmět a mu místnost. Tato hrana bude obarvena hodnotou čísla hodiny převedeného na barvu, ještě před samotným barvením. Algoritmus pak tuto barvu vynechá a proto nebude mít třída tu danou hodinu žádný předmět. Číslo hodiny převáděné na barvu je samozřejmě dáno ročníkem dané třídy a je zapotřebí tuto operaci provést pro každý den v týdnu.

Požadavky učitelů

Další omezující podmínky pro vytváření rozvrhu mohou být kladeny přímo od učitelů. Proto bude mít školní informační systém modul, ve kterém uživatel, který bude vytvářet rozvrhy (v našem případě uživatel s rolí rozvrhář) může zadávat pro každého učitele hodiny v týdnu, které nemůže vyučovat.
Řešení může být obdobné jako v případě přestávek. Je tedy vytvořen fiktivní uzel a s ním hranou spojen uzel udávající učitele a hrana je předem obarvena na hodnotu barvy udávající, kdy učitel nemůže učit.

8.8.4 Grafický návrh aplikace

Tato kapitola obsahuje pouze obrázek 24, který je zpracovaným grafickým návrhem školního informačního systému.



Obrázek 24: Grafický návrh

9 Implementace

Tato kapitola se bude věnovat implementací námi navrženým informačním systémem. Rozebereme všeobecné principy vhodné při implementaci aplikací a popíšeme konkrétní řešení jednotlivých částí systému.

Jak už bylo napsáno v kapitole návrhu architektury, rozhodl jsem se informační systém pro školy implementovat v jazyce PHP, konkrétně objektově orientovaném PHP 5. Výhodou tohoto řešení může být bezproblémová instalace aplikace, v případě praktického nasazení v praxi. Kombinace jazyka PHP a relační databáze MySQL je již velmi ověřena, prakticky nedochází k nějakému neočekávanému chování vlivem chyb v jazyce PHP.

Protože k implementaci nepoužívám žádný IDE (Integrated Development Environment) nástroj, je vhodné psát zdrojové kódy čitelně a dodržovat určitá pravidla, o což jsem se při vývoji systému snažil.

9.1 Adresářová struktura

Pro modulově založenou aplikaci je také potřeba volby optimální struktury uložených tříd v jazyce PHP a jejich řazení do adresářů. Zvolená adresářová struktura:

- /class/ - adresář obsahující třídy jádra aplikace,
- /css/ - adresář s nadefinovanými CSS soubory, které určují vzhled informačního systému,
- /data/ - tato složka obsahuje data uložená uživateli, většinou jde o obrázky,
- /icon/ - knihovna ikon, které jsou přiřazeny jednotlivým modulům,
- /images/ - adresář se statickými obrázky, které určují vzhled webové aplikace,
- /log/ - adresář obsahující všechny losovací soubory,
- /modules/ - složka se všemi moduly informačního systému,
- /pages/ - většinou statické webové stránky, nebo šablony určující zobrazení stránek pro modul,
- /scripts/ - externí skripty, které se využívají v systému,
- /sites/ - adresář s konfiguračními skripty pro jednotlivá nasazení systému.

Důležitými soubory v systému jsou:

- /index.php – výchozí stránka webu, která načte třídu web a vytvoří její instanci,
- /class/web_controller.php – třída napsaná v jazyce PHP, která slouží jako Controller v architektuře MVC. Po vytvoření její instance, zpracuje uživatelský vstup, načte třídu případného modulu a té předá zpracovaný vstup,

- /class/modul.php – třída označena jako tzv. obecný modul. Tuto třídu dle návrhu školního informačního systému využívá velké procentu všech modulů. Třída pracuje s daty v datovém úložišti,
- /class/mysql.php – je třída poskytující rozhraní k databázi MySQL.

9.2 Konfigurace systému

Při nasazení systému na serveru, nebo při změně systémových údajů, které se týkají celého informačního systému je potřeba aplikaci nakonfigurovat. Ke konfiguraci se využívá nastavení globálních konstant jazyka PHP.

Nastavení stačí provést pouze v jednom souboru const.php. Ale jeho umístění může být dáno názvem domény, na které je aplikace spuštěna. Jeho umístění by poté mělo být v adresáři /sites/název_domény/, pokud zde není soubor nalezen, využije se tzv. defaultní soubor v adresáři /class/. Tento proces provádí třída *web*. Výhodou tohoto nastavení může být provoz dvou odlišných informačních systémů, které využívají stejné jádro a názvy jejich domén jsou směrovány na kořenový adresář jednoho systému.

V souboru const.php se potom nastaví informace nutné pro provoz informačního systému, jako je kódování dat, nadpis stránky atd. Ukázkový soubor /class/const.php je velmi dobře okomentován pro případ jeho editace.

9.3 Implementace jádra

Implementace jádra programu spočívá v naprogramování tříd web_controller a modul, při implementace těchto tříd jsem se držel navrhované funkčnosti jádra. Třída web, která pracuje s uživatelským vstupem a reaguje na události vzniklé v aplikaci, se také stará o tzv. layout webové aplikace. I když je její hlavní funkcí je plnit roli Controlleru, tak také dává informace skriptu skeleton.php, který se stránku zobrazuje. Především se jedná o nalezení souboru style.css pro zobrazení kaskádových stylů stránky. Soubor style.css má stejná pravidla uložení jako již zmiňovaný const.php.

Poté co třída web reaguje na přijaté události (přečtených z parametrů GET a POST), vytvoří instanci modulu, který se aktuálně používá (určuje ho proměnná mod v parametru GET) a této instanci předá zpracovaná vstupní data. Instancí modulu je vytvoření instance třídy, kterou modul používá. Třída je dána svým fyzickým uložením souboru, který může být:

- /sites/název_domény/modules/název_třídy/název_třídy.php – v případě, že modul používá vlastní třídu a ta je navíc ve vztahu v jakém informačním systém je použita (dána doménou),

- /modules/název_třídy/název_třídy.php / - kdy modul používá vlastní třídu, ale ta je použita globálně pro všechny stejné moduly v rámci jednoho systému,
- /class/modul.php – kdy to není ani jedna z předchozích možností a modul použije předdefinovanou třídu.

Pokud není použit žádný modul, například po přihlášení do informačního systému, zobrazí třída soubor /pages/home.php.

Pokud je vytvořená instance třídy modul.php, tak se provádí funkce kladené při návrhu na tzv. obecný modul. Podmínkou je, aby byl tento modul konfigurován XML souborem a tím vytvářel jeden samostatný modul. XML soubor má naprosto stejná pravidla pro jeho uložení jako má třída modulu. Po nalezení konfiguračního souboru je načten, podle definovaného DTD, třídou /class/xml.php a třídou modul zpracován. Dále provádí činnost dle návrhu jádra aplikace, výstupní data zobrazují uživateli skripty plnící roli view (dle MVC).

Implementace metod pro zobrazení webového formuláře vyžadovala využití již hotového wysiwyg editoru a jeho nasazení na informační systém. Po výběru z wysiwyg editorů, jsem nakonec zvolil open-source projekt FCK-editor [25], šířený pod licencí GPL [26]. Po řadě drobných úprav kódu editoru jsem jej úspěšně nasadil na systém. Dalším externím skriptem je doplněk Date Picker [27] napsaný v Javascriptu, který slouží k výběru data do formulářového vstupu typu datum, přímo z kalendáře.

9.4 Implementace modulů

Implementace modulů školního informačního systému je pouhým vytvořením jednoduché třídy a konfiguračních souboru pro každý z navržených modulů, všechny moduly využívají třídu jádra modul.php. Některé rovnou, kdy je vytvořen jen konfigurační soubor, jiné alespoň děděním této třídy a stavění modulu na jejím základu.

Implementace modulů podle navrženého schématu na obrázku 18 nenesla žádné vetší problémy, naopak sloužila jaké testování jádra aplikace.

9.5 Rozvrhování

Modul tvorby rozvrhů je tvořen třídou, která pracuje přesně podle navrženého algoritmu. Uživateli, který bude rozvrhy vytvářet je díky dalším modulům usnadněno definování omezujících podmínek.

Modul hodiny umožní definování učebního plánu. Každé třídě se může přesně nadefinovat, kolik hodin výuky týdně má jaké předměty. Další modul určený k definici omezujících pravidel na vytváření rozvrhů je modul přestávky, ve kterém uživatel určí vztah mezi ročníkem a zákonem

definovanou přestávkou. A posledním takovým modulem je modul pro určení požadavků učitele na vyučování v daných hodinách.

Aby měl uživatel přehled nad zadanými daty, pomocí kterých bude systém generovat rozvrhy, je vytvořen modul přehled, kde jsou všechna zadaná data zobrazena v tabulkách.

Samotná třída pro generování rozvrhu nejprve načte všechny informace. Z těchto informací podle metody popsané v 8. kapitole vytvoří graf, ten obarví podle algoritmu 8 ve stejné kapitole. Ze zjištěných barev hran vytvoří rozvrh a ten zobrazí uživateli. Pokud je uživatel s rozvrhem spokojen, uloží se tento rozvrh do databáze, do tabulky rozvrh.

9.6 Nároky na konfiguraci serveru

Pro nasazení informačního systému je potřeba webový server s PHP a přístup k MySQL databázi. Protože je programovací jazyk PHP nezávislý na platformě, je možnost provozu webového serveru na různých operačních systémech.

Nejčastěji se vyskytující variantou je provoz serveru na operačním systému GNU Linux, kde je jako webový server použit Apache (dnes Apache 2), dále samotné PHP a ideálně ještě MySQL server.

Nastavení PHP direktiv pro plně funkční běh aplikace je magic_quotes_gpc = Off, magic_quotes_sybase = Off a magic_quotes_runtime = Off.

Osobně jsem informační systém vyvíjel s použitím standardního nastaveni verze PHP 5.2.6 s upravením direktiv.

10 Nasazení školního informačního systému

Nasazení školního informačního systému znamená uvedení celé aplikace do provozu. Obdobné systémy nasazené v tzv. ostrém provozu obsahují celou řadu soukromých informací. Pro nasazení systému je tedy není možné použít. Proto jsem nejdříve systém naplnil testovacími daty, ale vytvoření fiktivního provozu školy je velice složitý proces, obzvlášť v definování vztahu učitel – předmět. Všechny implementované moduly pracovaly správně, jen řešení problému generování rozvrhů nemělo potřebnou vypovídající hodnotu o složitosti testovaných dat.

Proto jsem se dohodl s panem Janečkem na poskytnutí reálných údajů jejich základní školy. Tím jsem dostal učební plány všech tříd, seznam učitelů včetně informací jaké učí předměty, seznam vyučovaných předmětů a také jimi používanou aplikací vygenerované rozvrhy. Všechny tyto informace jsem převedl do mnou navrženého a implementovaného informačního systému. Systém bez jakýchkoliv problému s daty pracuje dle očekávání a vygenerování rozvrhů tříd dává také vcelku dobré výsledky. Takto vygenerované rozvrhy jsou přiloženy v příloze D. Pro porovnání se skutečnými rozvrhy, jsou tyto také obsaženy v příloze E.

Informačním systémem vygenerované rozvrhy jsem ukázal panu Janečkovi, který byl z výsledku velmi příjemně překvapen.

10.1 Návrh na vylepšení

Spravování školní agendy je velice široký pojem a každá ze škol si může pod tímto pojmem představit různé činnosti. Z tohoto důvodu, je aplikace postavena na jednoduché možnosti rozšiřitelnosti. Jako vylepšení stávajícího řešení navrhuji

- Rozšíření jádra systému o další funkce, především modul pro práci s daty. Tím se zjednoduší tvorba nových modulů, které mohou být ideálně postavené pouze na konfiguračním souboru.
- S předcházejícím návrhem souvisí i možnost rozšíření XML konfiguračního souboru.
- Doplnění rozvrhovacího algoritmu o další omezující podmínky, které mohou v reálných situacích nastat.
- Sestavení nových modulů například pro řešení třídní knihy, suplování, výukových kroužků, atd.

11 Závěr

Při seznámení se správou informací ve školách, jsem zjistil, jak je tato problematika rozsáhlá. Velkým přínosem pro tuto práci byly informace poskytnuté samotnými školami, obzvláště pak osobní setkání s panem Janečkem. Školy jsou nuceny spravovat velké množství informací, typy a formy těchto dat jsou shrnuty v prvních kapitolách této práce.

Dále jsem se zaměřil na techniky pro automatickou tvorbu rozvrhů. V úvodu do této problematiky bylo nejdříve zapotřebí definovat základní pojmy problému rozvrhování a metody jeho řešení. Zaměřil jsem se na metody založené na barvení grafu.

Po analýze požadavků, kladených školou na provoz školního informačního systému, jsem navrhl možnost řešení modulově založenou aplikací. Návrh informačního systému se skládal z návrhu jádra aplikace a na něm postavených modulech. Cílem práce byl také návrh algoritmu, pomocí kterého lze řešit problém školního rozvrhování. Navržený algoritmus se skládá z převedení omezujících podmínek na graf a modifikované sekvenční metody barvení grafu, která převedený graf obarví a tím se získají požadované rozvrhy.

Samotná implementace navrženého systému je plně funkční a splňuje běžné požadavky pro provoz školního informačního systému. Přínos této práce vidím především v návrhu takto založeného informačního systému a v návrhu řešení problému školního rozvrhování.

Literatura

[1] *Projekt Bakaláři* [online]. 1999 , listopad 2007 [cit. 2007-11-20]. Dostupný z WWW: <http://www.bakalari.cz/>.

[2] M2000, spol. s r.o.. *EŠkola* [online]. 2000 [cit. 2007-11-20]. Dostupný z WWW: <http://www.eskola.cz/>.

[3] Computer Media s.r.o.. *IŠkola* [online]. 2005 [cit. 2007-11-20]. Dostupný z WWW: <http://www.iskola.cz/>.

[4] MP-Soft, a.s.. *Systém agend pro školy* [online]. 2001 [cit. 2007-11-20]. Dostupný z WWW: <http://www.mp-soft.cz/>.

[5] SCHAERF, Andrea. A Survey of Automated Timetabling. *Artificial Intelligence Review* [online]. 1999 [cit. 2007-11-15], s. 87-127. ISSN 1573-7462.

[6] SANTOS, Haroldo G., OCHI, Luiz S., SOUZA, Marcone J. F. A Tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *Journal of Experimental Algorithmics.* [online]. 2005, no. 10 [cit. 2007-11-24], s. 16-44. ISSN 1084-6654.

[7] SMITH, Graham. On Maintance of the Opportunity List for Class-Teacher Timetable Problems. *Management Science* [online]. 1975 [cit. 2007-12-15], s. 203-208.

[8] MARCZYK, Adam. *Talk.origins : Genetic Algorithms and Evolutionary Computation* [online]. 2004 [cit. 2007-12-20]. Dostupný z WWW: <http://www.talkorigins.org/faqs/genalg/>.

[9] CARRASCO, Marco P., PATO, Margarida V. *PATAT '00: Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling*. London : [s.n.], 2001. A multiobjective genetic algorithm for the class/teacher timetabling problem, s. 3-17.

[10] DE WERRA, Dominique. Construction of school timetables by flow methods. *INFOR – Canadian Journal of Operations Research and Information Processing* [online]. 1971 [cit. 2007-12-27], s. 12-22.

[11] ABRAMSON, David. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science* [online]. 1991 [cit. 2007-12-27], s. 98-113.

[12] SCHAERF, Andrea. Tabu search techniques for examination timetabling. *Lecture Notes in Computer Science* [online]. 2001 [cit. 2007-12-28], s. 104-117.

[13] PATAT. *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling* [online]. 2007 [cit. 2007-12-28]. Dostupný z WWW: <http://www.asap.cs.nott.ac.uk/patat/patat06/Proceedings.pdf>.

[14] Vyhláška ze dne 18. ledna 2005 o základním vzdělávání a některých náležitostech plnění povinné školní docházky, vztahující se k zákon č. 561/2004 Sb. *o předškolním, základním, středním, vyšším odborném a jiném vzdělávání (školský zákon)*, 2004. ISBN 80-7208-069-5.

[15] ŠEDA, Miloš.: *Teorie grafů*. VUT FSI, Brno, 2003.

[16] KOLÁŘ, Josef. *Teoretická informatika*. [s.l.] : [s.n.], 2004. 205 s. ISBN 80-900853-8-5.

[17] *XML Specification* [online]. 2007 [cit. 2008-04-20]. Dostupný z WWW: <http://www.w3.org/XML/Core/#Publications>.

[18] TICHÝ, Jan. Architektura aplikace. *Funkční požadavky [online]*. 2004-2008 [cit. 2008-04-27]. Dostupný z WWW: < http://www.jantichy.cz/diplomka/pozadavky/architektura >.

[19] BURBECK, Steve. How to use Model-View-Controller. *Applications Programming in Smalltalk-80* [online]. 1992 [cit. 2008-04-25]. Dostupný z WWW: <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>.

[20] Sun Microsystems. *J2EE Architecture Approaches*. 2002. Dostupný z WWW: <http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/app-arch/app-arch2.html>.

[21] TICHÝ, Jan. Autentizace a správa uživatelů. *Funkční požadavky [online]*. 2004-2008 [cit. 2008-04-27]. Dostupný z WWW: <http://www.jantichy.cz/diplomka/pozadavky/autentizace >.

[22] The Open Web Application Security Project. *A Guide to Building Secure Web Applications*. 2002. Dostupný z WWW: <http://www.owasp.org/documentation/guide>.

[23] TOPLIST. Globální statistiky [online]. 2008 [cit. 2008-05-07]. Dostupný z WWW: < http://www.toplist.cz/global.html>

[24] KOSEK, Jiří. *XML pro každého : podrobný průvodce.* 1. vyd. Praha : Grada Publishing, 2000. 163 s. Obsahuje bibliografii, bibliografické odkazy a rejstřík. ISBN 80-7169-860-1.

[25] *FCKeditor* [online]. 2007 [cit. 2008-03-02]. Dostupný z WWW: < http://www.fckeditor.net/>.

[26] *GNU Lesser General Public License* [online]. 2008 [cit. 2008-03-05]. Dostupný z WWW: < http://www.gnu.org/licenses/lgpl.html>.

[27] *Date picker* [online]. 2006 [cit. 2008-03-05]. Dostupný z WWW: < http://www.frequency-decoder.com/2006/10/02/unobtrusive-date-picker-widgit-update>.

[28] CHMELAŘ, Petr. *Zjišťování izomorfizmu mezi grafy*. [s.l.] : [s.n.], 2006. 16 s.

Seznam příloh

A. Návrh referenčního DTD konfigurace

B. Možnosti konfiguračních souborů

C. Rozvrhy vytvořené navrhovaným systémem

D. Rozvrhy poskytnuté školou

E. Obsah CD

A. Návrh referenčního DTD konfigurace

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE modul [
<!ELEMENT col ( #PCDATA ) >
<!ELEMENT control ( #PCDATA ) >
<!ELEMENT control_line ( #PCDATA ) >
<!ELEMENT control_links ( #PCDATA ) >
<!ELEMENT css ( #PCDATA ) >
<!ELEMENT destination_base ( #PCDATA ) >
<!ELEMENT destination_web ( #PCDATA ) >
<!ELEMENT element ( #PCDATA ) >
<!ELEMENT elements ( col | css | destination_base | destination_web
| element | help | js | nazev | options | popis | povinny | size |
type | value | db )* >
<!ELEMENT form ( elements+ ) >
<!ELEMENT h1 ( #PCDATA ) >
<!ELEMENT help ( #PCDATA ) >
<!ELEMENT info ( #PCDATA ) >
<!ELEMENT js ( #PCDATA ) >
<!ELEMENT modul ( h1 | text | info | form | order_by | select |
control | control_line | control_links )* >
<!ELEMENT nazev ( #PCDATA ) >
<!ELEMENT options ( #PCDATA ) >
<!ELEMENT order_by ( #PCDATA ) >
<!ELEMENT popis ( #PCDATA ) >
<!ELEMENT povinny ( #PCDATA ) >
<!ELEMENT select ( #PCDATA ) >
<!ELEMENT size ( #PCDATA ) >
<!ELEMENT text ( #PCDATA ) >
<!ELEMENT type ( #PCDATA ) >
<!ELEMENT value ( #PCDATA ) >
]>
```

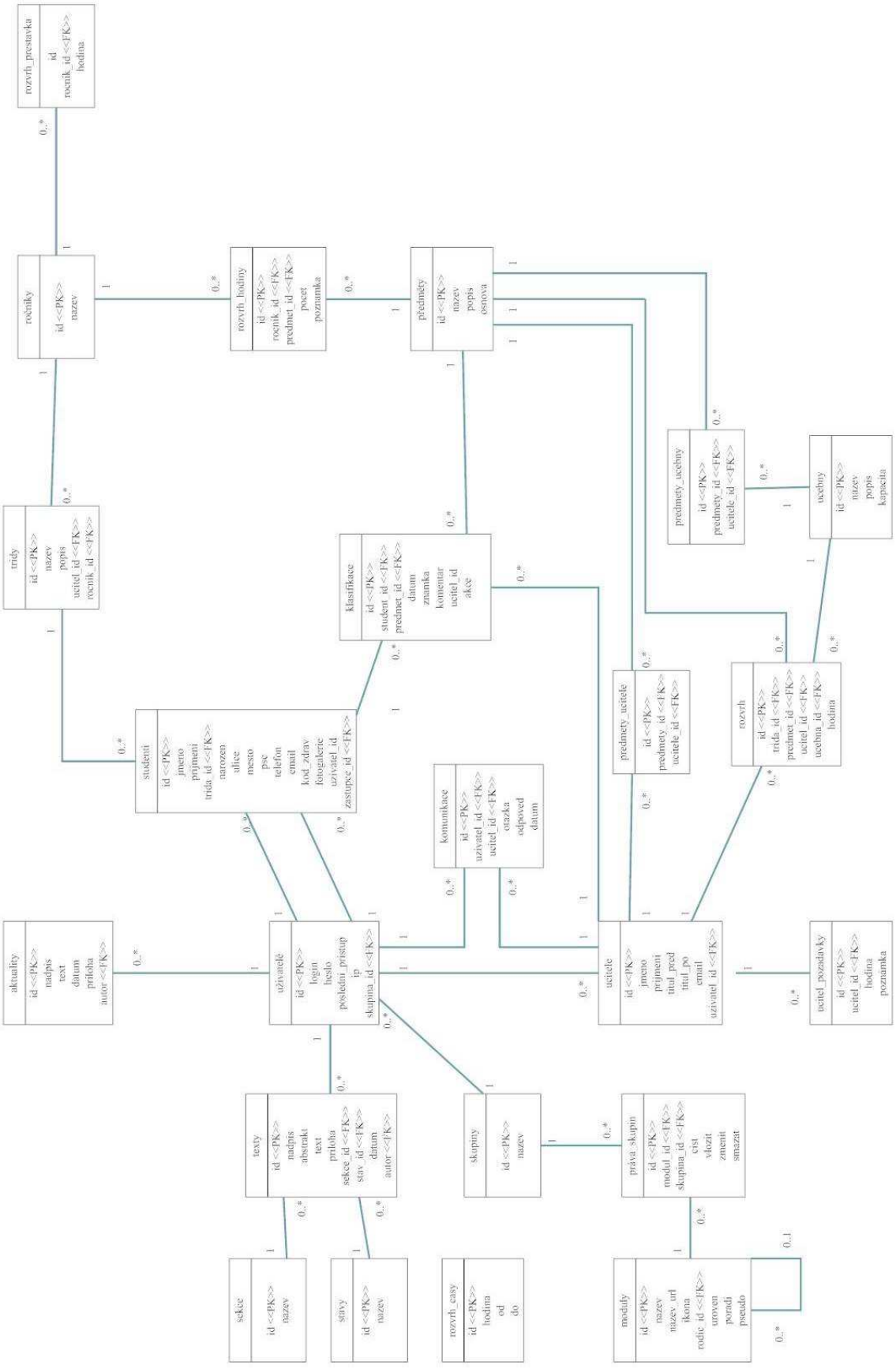
B. Možnosti konfiguračních souborů

Příklad konfiguračního souboru modulu učitelé:

```
<modul>
<h1>Učitelé</h1>
<text>Seznam učitelů.</text>
<form>
  <elements>
        <col>titul_pred</col>
        <element>input</element>
        <nazev>Titul před jménem</nazev>
        <size>5</size>
        <type></type>
  </elements>
  <elements>
        <col>jmeno</col>
        <element>input</element>
        <nazev>Jméno</nazev>
        <size>20</size>
        <type>text</type>
        <css>style="background: #CCC;"</css>
        <povinny>1</povinny>
  </elements>
  <elements>
        <col>prijmeni</col>
        <element>input</element>
        <nazev>Příjmení</nazev>
        <size>20</size>
        <type>text</type>
        <css>style="background: #CCC;"</css>
        <povinny>1</povinny>
  </elements>
        <elements>
        <col>titul_po</col>
        <element>input</element>
```

```
        <nazev>Titul za jménem</nazev>
        <size>5</size>
        <type></type>
    </elements>
    <elements>
        <col>email</col>
        <element>input</element>
        <nazev>E-mail</nazev>
        <size>25</size>
        <type>email</type>
    </elements>
    <elements>
        <col>predmety_id</col>
        <element>multipleselect</element>
        <nazev>Učí</nazev>
        <size>10</size>
        <saveto>predmety_ucitele</saveto>
        <db>0</db>
    </elements>
</form>
<order_by>order by prijmeni, jmeno</order_by>
<select>prijmeni, jmeno</select>
<control>Akce</control>
<control_line>data</control_line>
</modul>
```

C. E-R diagram


rozvrh_prestavka
id
rocnik_id <<FK>>
hodina
ročníky
id <<PK>>
nazev
rozvrh_hodiny
id <<PK>>
rocnik_id <<FK>>
predmet_id <<FK>>
pocet
poznamka
předměty
id <<PK>>
nazev
popis
osnova
třídy
id <<PK>>
nazev
popis
ucitel_id <<FK>>
rocnik_id <<FK>>
klasifikace
id <<PK>>
student_id <<FK>>
predmet_id <<FK>>
datum
znamka
komentar
ucitel_id
akce
predmety_ucebny
id <<PK>>
predmety_id <<FK>>
ucitele_id <<FK>>
ucebny
id <<PK>>
nazev
popis
kapacita
studenti
id <<PK>>
jmeno
prijmeni
trida_id <<FK>>
narozen
ulice
mesto
psc
telefon
email
kod_zdrav
fotogalerie
uzivatel_id
zastupce_id <<FK>>
predmety_ucitele
id <<PK>>
predmety_id <<FK>>
ucitele_id <<FK>>
rozvrh
id <<PK>>
trida_id <<FK>>
predmet_id <<FK>>
ucitel_id <<FK>>
ucebna_id <<FK>>
hodina
komunikace
id <<PK>>
uzivatel_id <<FK>>
ucitel_id <<FK>>
otazka
odpoved
datum
aktuality
id <<PK>>
nadpis
text
datum
priloha
autor <<FK>>
uživatelé
id <<PK>>
login
heslo
posledni_pristup
ip
skupina_id <<FK>>
ucitele
id <<PK>>
jmeno
prijmeni
titul_pred
titul_po
email
uzivatel_id <<FK>>
ucitel_pozadavky
id <<PK>>
ucitel_id <<FK>>
hodina
poznamka
texty
id <<PK>>
nadpis
abstrakt
text
priloha
sekce_id <<FK>>
stav_id <<FK>>
datum
autor <<FK>>
skupiny
id <<PK>>
nazev
práva_skupin
id <<PK>>
modul_id <<FK>>
skupina_id <<FK>>
cist
vlozit
zmenit
smazat
sekce
id <<PK>>
nazev
stavy
id <<PK>>
nazev
rozvrh_casy
id <<PK>>
hodina
od
do
moduly
id <<PK>>
nazev
nazev_url
ikona
rodic_id <<FK>>
uroven
poradi
pseudo

D. Rozvrhy vytvořené navrhovaným systémem

5. A

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Český jazyk 5 *Černá Ivana* **X 10**	**Český jazyk 5** *Černá Ivana* **X 10**	**Matematika 5** *Černá Ivana* **X 10**	**Vlastivěda** *Černá Ivana* **X 10**	**Anglický jazyk 5** *Černá Ivana* **X 10**		**Tělesná výchova** *Vykydal Martin* **Z 103**			
Český jazyk 5 *Černá Ivana* **X 10**	**Český jazyk 5** *Černá Ivana* **X 10**	**Matematika 5** *Černá Ivana* **X 10**	**Přírodověda** *Šestáková Lenka* **X 10**	**Výtvarná výchova** *Pecinová Ivana* **X 10**					
Tělesná výchova *Vykydal Martin* **Z 103**	**Český jazyk 5** *Černá Ivana* **X 10**	**Matematika 5** *Černá Ivana* **X 10**	**Vlastivěda** *Černá Ivana* **X 10**	**Anglický jazyk 5** *Černá Ivana* **X 10**					
Výtvarná výchova *Pecinová Ivana* **X 10**	**Český jazyk 5** *Černá Ivana* **X 10**	**Matematika 5** *Černá Ivana* **X 10**	**Hudební výchova 5** *Černá Ivana* **X 10**	**Anglický jazyk 5** *Černá Ivana* **X 10**					
Český jazyk 5 *Černá Ivana* **X 10**	**Přírodověda** *Šestáková Lenka* **X 10**	**Matematika 5** *Černá Ivana* **X 10**	**Anglický jazyk 5** *Černá Ivana* **X 10**	**Informační technologie 6** *Völklová Ludmila* **X 10**					

6. A

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Matematika 6 *Völklová Ludmila* **A 108**	**Tělesná výchova** *Vykydal Martin* **Z 103**	**Matematika 6** *Völklová Ludmila* **A 108**	**Tělesná výchova** *Vykydal Martin* **Z 103**	**Člověk a svět práce** *Krajhanzlová Zdena* **A 108**		**Český jazyk 5** *Černá Ivana* **A 108**			
Zeměpis 6 *Dytrt Rudolf* **A 108**	**Matematika 6** *Völklová Ludmila* **A 108**	**Zeměpis 6** *Dytrt Rudolf* **A 108**	**Informační technologie 6** *Völklová Ludmila* **A 108**	**Výchova k občanství** *Ostatková Irena* **A 108**		**Český jazyk 5** *Černá Ivana* **A 108**			
Anglický jazyk 6 *Lorenčíková Jana* **A 108**	**Výtvarná výchova** *Pecinová Ivana* **A 108**	**Tělesná výchova** *Vykydal Martin* **Z 103**	**Dějepis 6** *Ostatková Irena* **A 108**	**Anglický jazyk 6** *Lorenčíková Jana* **A 108**		**Český jazyk 5** *Černá Ivana* **A 108**			
Fyzika 6 *Švadlenková Štěpánka* **A 108**	**Výtvarná výchova** *Pecinová Ivana* **A 108**	**Matematika 6** *Völklová Ludmila* **A 108**	**Dějepis 6** *Ostatková Irena* **A 108**	**Anglický jazyk 6** *Lorenčíková Jana* **A 108**		**Přírodopis 6** *Šestáková Lenka* **A 108**			
Fyzika 6 *Švadlenková Štěpánka* **A 108**	**Český jazyk 5** *Černá Ivana* **A 108**	**Matematika 6** *Völklová Ludmila* **A 108**	**Hudební výchova 6** *Vikrorínová Ludmila* **A 108**	**Český jazyk 5** *Černá Ivana* **A 108**					

6. B

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Matematika 6 *Švadlenková Štěpánka* **A 201**	**Hudební výchova 6** *Vikrorínová Ludmila* **A 201**	**Výtvarná výchova** *Pecinová Ivana* **A 201**	**Přírodopis 6** *Šestáková Lenka* **A 201**	**Fyzika 6** *Pecinová Ivana* **A 201**		**Anglický jazyk 6** *Dytrtová Martina* **A 201**			
Tělesná výchova *Krajhanzlová Zdena* **Z 103**	**Výtvarná výchova** *Pecinová Ivana* **A 201**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Český jazyk 5** *Kratochvílová Šárka* **A 201**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**		**Výchova k občanství** *Ostatková Irena* **A 201**			
Matematika 6 *Švadlenková Štěpánka* **A 201**	**Český jazyk 5** *Kratochvílová Šárka* **A 201**	**Český jazyk 5** *Kratochvílová Šárka* **A 201**	**Matematika 6** *Švadlenková Štěpánka* **A 201**	**Dějepis 6** *Ostatková Irena* **A 201**		**Matematika 6** *Švadlenková Štěpánka* **A 201**			
Člověk a svět práce *Janeček Vladimír* **A 201**	**Český jazyk 5** *Kratochvílová Šárka* **A 201**	**Český jazyk 5** *Kratochvílová Šárka* **A 201**	**Informační technologie 6** *Völklová Ludmila* **A 201**	**Anglický jazyk 6** *Dytrtová Martina* **A 201**		**Zeměpis 6** *Dytrt Rudolf* **A 201**			
Zeměpis 6 *Dytrt Rudolf* **A 201**	**Matematika 6** *Švadlenková Štěpánka* **A 201**	**Fyzika 6** *Pecinová Ivana* **A 201**	**Dějepis 6** *Ostatková Irena* **A 201**	**Anglický jazyk 6** *Dytrtová Martina* **A 201**					

7. A

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Dějepis 7 *Kratochvílová Šárka* **A 204**	**Matematika 7** *Lorenčíková Jana* **A 204**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Matematika 7** *Lorenčíková Jana* **A 204**	**Matematika 7** *Lorenčíková Jana* **A 204**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**				
Fyzika 7 *Švadlenková Štěpánka* **A 204**	**Český jazyk 7** *Vikrorínová Ludmila* **A 204**	**Výchova ke zdraví** *Janeček Vladimír* **A 204**	**Český jazyk 7** *Vikrorínová Ludmila* **A 204**	**Člověk a svět práce** *Janeček Vladimír* **A 204**	**Volitelný předmět** *Lorenčíková Jana* **A 204**		**Volitelný předmět** *Lorenčíková Jana* **A 204**		
Výtvarná výchova *Pecinová Ivana* **A 204**	**Přírodopis 7** *Šestáková Lenka* **A 204**	**Matematika 7** *Lorenčíková Jana* **A 204**	**Zeměpis 7** *Vykydal Martin* **A 204**	**Výtvarná výchova** *Pecinová Ivana* **A 204**	**Zeměpis 7** *Vykydal Martin* **A 204**				
Anglický jazyk 7 *Dytrtová Martina* **A 204**	**Český jazyk 7** *Vikrorínová Ludmila* **A 204**	**Český jazyk 7** *Vikrorínová Ludmila* **A 204**	**Fyzika 7** *Švadlenková Štěpánka* **A 204**	**Přírodopis 7** *Šestáková Lenka* **A 204**	**Anglický jazyk 7** *Dytrtová Martina* **A 204**				
Český jazyk 7 *Vikrorínová Ludmila* **A 204**	**Matematika 7** *Lorenčíková Jana* **A 204**	**Dějepis 7** *Kratochvílová Šárka* **A 204**	**Anglický jazyk 7** *Dytrtová Martina* **A 204**	**Hudební výchova 7** *Vikrorínová Ludmila* **A 204**					

7. B

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Český jazyk 7 *Ostatková Irena* **A 203**	**Matematika 7** *Völklová Ludmila* **A 203**	**Fyzika 7** *Švadlenková Štěpánka* **A 203**	**Matematika 7** *Völklová Ludmila* **A 203**	**Hudební výchova 7** *Vikrorínová Ludmila* **A 203**	**Přírodopis 7** *Dytrt Rudolf* **A 203**				
Český jazyk 7 *Ostatková Irena* **A 203**	**Dějepis 7** *Kratochvílová Šárka* **A 203**	**Anglický jazyk 7** *Dytrtová Martina* **A 203**	**Výtvarná výchova** *Pecinová Ivana* **A 203**	**Volitelný předmět** *Lorenčíková Jana* **A 203**	**Anglický jazyk 7** *Dytrtová Martina* **A 203**				
Český jazyk 7 *Ostatková Irena* **A 203**	**Anglický jazyk 7** *Dytrtová Martina* **A 203**	**Matematika 7** *Völklová Ludmila* **A 203**	**Člověk a svět práce** *Janeček Vladimír* **A 203**	**Dějepis 7** *Kratochvílová Šárka* **A 203**	**Matematika 7** *Völklová Ludmila* **A 203**				
Přírodopis 7 *Dytrt Rudolf* **A 203**	**Výchova ke zdraví** *Janeček Vladimír* **A 203**	**Český jazyk 7** *Ostatková Irena* **A 203**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Fyzika 7** *Švadlenková Štěpánka* **A 203**	**Volitelný předmět** *Lorenčíková Jana* **A 203**				
Zeměpis 7 *Vykydal Martin* **A 203**	**Zeměpis 7** *Vykydal Martin* **A 203**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Výtvarná výchova** *Pecinová Ivana* **A 203**	**Český jazyk 7** *Ostatková Irena* **A 203**	**Matematika 7** *Völklová Ludmila* **A 203**				

8. A

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Anglický jazyk 8 *Dytrtová Martina* **A 301**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Anglický jazyk 8** *Dytrtová Martina* **A 301**	**Český jazyk 8** *Vikrorínová Ludmila* **A 301**	**Chemie 8** *Šestáková Lenka* **A 301**	**Výchova k občanství** *Ostatková Irena* **A 301**		**Matematika 8** *Švadlenková Štěpánka* **A 301**		
Český jazyk 8 *Vikrorínová Ludmila* **A 301**	**Přírodopis 8** *Dytrt Rudolf* **A 301**	**Český jazyk 8** *Vikrorínová Ludmila* **A 301**	**Matematika 8** *Švadlenková Štěpánka* **A 301**	**Anglický jazyk 8** *Dytrtová Martina* **A 301**	**Člověk a svět práce** *Janeček Vladimír* **A 301**		**Hudební výchova 8** *Vikrorínová Ludmila* **A 301**		
Český jazyk 8 *Vikrorínová Ludmila* **A 301**	**Matematika 8** *Švadlenková Štěpánka* **A 301**	**Fyzika 8** *Pecinová Ivana* **A 301**	**Dějepis 8** *Kratochvílová Šárka* **A 301**	**Chemie 8** *Šestáková Lenka* **A 301**	**Volitelný předmět** *Lorenčíková Jana* **A 301**				
Tělesná výchova *Krajhanzlová Zdena* **Z 103**	**Zeměpis 8** *Vykydal Martin* **A 301**	**Volitelný předmět** *Lorenčíková Jana* **A 301**	**Volitelný předmět** *Lorenčíková Jana* **A 301**	**Fyzika 8** *Pecinová Ivana* **A 301**					
Dějepis 8 *Kratochvílová Šárka* **A 301**	**Výtvarná výchova** *Pecinová Ivana* **A 301**	**Přírodopis 8** *Dytrt Rudolf* **A 301**	**Matematika 8** *Švadlenková Štěpánka* **A 301**	**Zeměpis 8** *Vykydal Martin* **A 301**	**Český jazyk 8** *Vikrorínová Ludmila* **A 301**				

8. B

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Chemie 8 *Šestáková Lenka* **A 300**	**Anglický jazyk 8** *Dytrtová Martina* **A 300**	**Dějepis 8** *Kratochvílová Šárka* **A 300**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Matematika 8** *Völklová Ludmila* **A 300**	**Český jazyk 8** *Vikrorínová Ludmila* **A 300**		**Matematika 8** *Völklová Ludmila* **A 300**		
Matematika 8 *Völklová Ludmila* **A 300**	**Anglický jazyk 8** *Dytrtová Martina* **A 300**	**Výchova k občanství** *Ostatková Irena* **A 300**	**Anglický jazyk 8** *Dytrtová Martina* **A 300**	**Matematika 8** *Völklová Ludmila* **A 300**	**Fyzika 8** *Pecinová Ivana* **A 300**				
Chemie 8 *Šestáková Lenka* **A 300**	**Zeměpis 8** *Vykydal Martin* **A 300**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**	**Český jazyk 8** *Vikrorínová Ludmila* **A 300**	**Zeměpis 8** *Vykydal Martin* **A 300**			**Hudební výchova 8** *Vikrorínová Ludmila* **A 300**		
Dějepis 8 *Kratochvílová Šárka* **A 300**	**Přírodopis 8** *Dytrt Rudolf* **A 300**	**Výtvarná výchova** *Pecinová Ivana* **A 300**	**Fyzika 8** *Pecinová Ivana* **A 300**	**Přírodopis 8** *Dytrt Rudolf* **A 300**	**Český jazyk 8** *Vikrorínová Ludmila* **A 300**		**Volitelný předmět** *Lorenčíková Jana* **A 300**		
Volitelný předmět *Lorenčíková Jana* **A 300**	**Český jazyk 8** *Vikrorínová Ludmila* **A 300**	**Český jazyk 8** *Vikrorínová Ludmila* **A 300**	**Volitelný předmět** *Lorenčíková Jana* **A 300**	**Člověk a svět práce** *Janeček Vladimír* **A 300**					

9. A

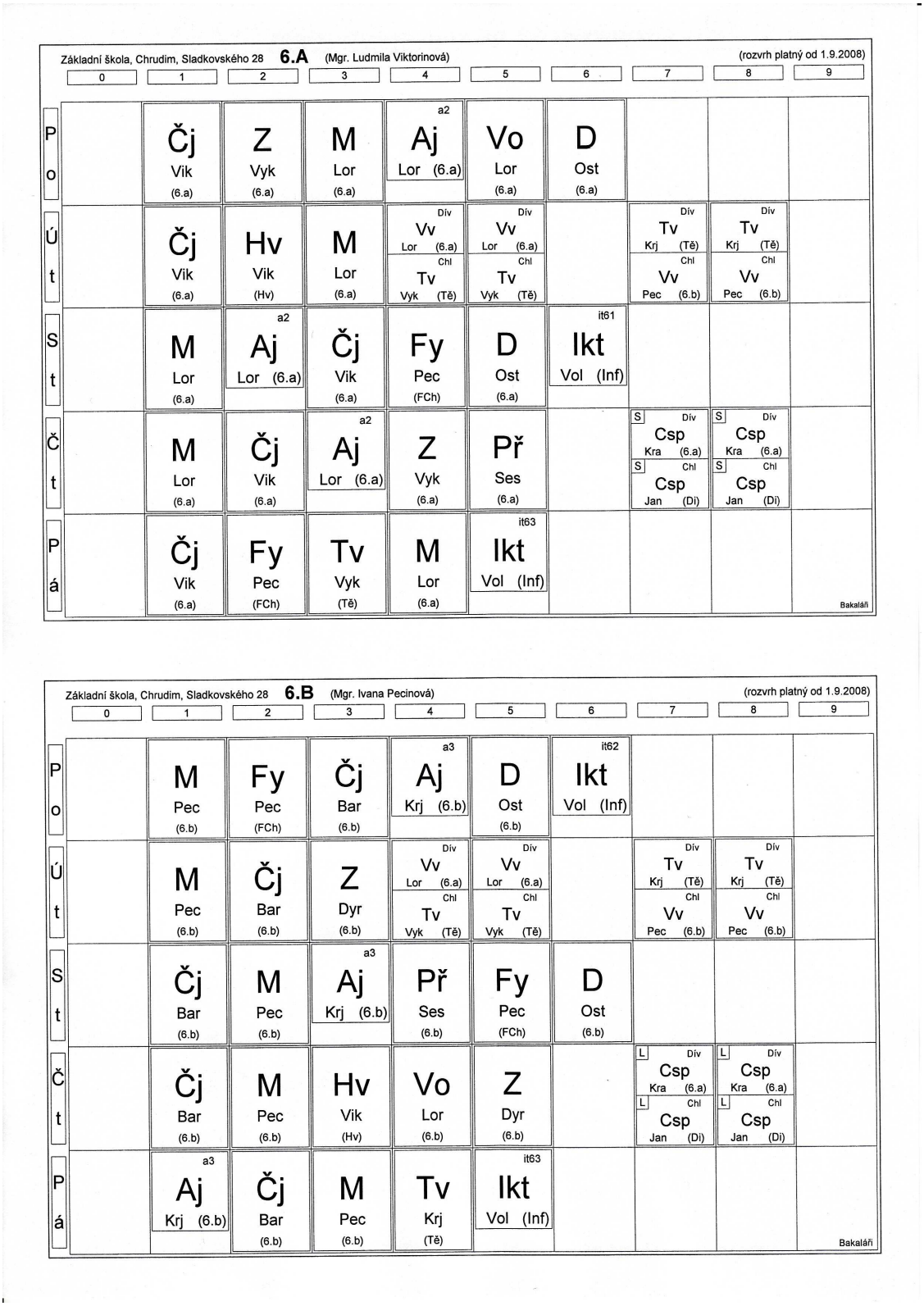
8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Tělesná výchova *Vykydal Martin* **Z 103**	**Chemie 9** *Šestáková Lenka* **A 205**	**Volitelný předmět** *Lorenčíková Jana* **A 205**	**Matematika 9** *Švadlenková Štěpánka* **A 205**	**Anglický jazyk 9** *Dytrtová Martina* **A 205**	**Fyzika 9** *Pecinová Ivana* **A 205**		**Volitelný předmět** *Lorenčíková Jana* **A 205**		
Výtvarná výchova *Pecinová Ivana* **A 205**	**Člověk a svět práce** *Janeček Vladimír* **A 205**	**Fyzika 9** *Pecinová Ivana* **A 205**	**Výchova ke zdraví** *Černá Ivana* **A 205**	**Matematika 9** *Švadlenková Štěpánka* **A 205**	**Český jazyk 9** *Vikrorínová Ludmila* **A 205**				
Zeměpis 9 *Dytrt Rudolf* **A 205**	**Český jazyk 9** *Vikrorínová Ludmila* **A 205**	**Anglický jazyk 9** *Dytrtová Martina* **A 205**	**Volitelný předmět** *Lorenčíková Jana* **A 205**	**Matematika 9** *Švadlenková Štěpánka* **A 205**	**Český jazyk 9** *Vikrorínová Ludmila* **A 205**				
Český jazyk 9 *Vikrorínová Ludmila* **A 205**	**Chemie 9** *Šestáková Lenka* **A 205**	**Přírodopis 9** *Šestáková Lenka* **A 205**	**Český jazyk 9** *Vikrorínová Ludmila* **A 205**	**Dějepis 9** *Ostatková Irena* **A 205**	**Matematika 9** *Švadlenková Štěpánka* **A 205**		**Hudební výchova 9** *Vikrorínová Ludmila* **A 205**		
Anglický jazyk 9 *Dytrtová Martina* **A 205**	**Dějepis 9** *Ostatková Irena* **A 205**	**Volitelný předmět** *Lorenčíková Jana* **A 205**	**Tělesná výchova** *Vykydal Martin* **Z 103**	**Matematika 9** *Švadlenková Štěpánka* **A 205**	**Volitelný předmět** *Lorenčíková Jana* **A 205**				

9. B

8:00 - 8:45	8:55 - 9:40	10:00 - 10:45	10:55 - 11:40	11:50 - 12:35	12:45 - 13:30	13:40 - 14:25	14:35 - 15:20	15:30 - 16:15	16:25 - 17:10
Český jazyk 9 *Vikrorínová Ludmila* **B 122**	**Fyzika 9** *Pecinová Ivana* **B 122**	**Český jazyk 9** *Vikrorínová Ludmila* **B 122**	**Fyzika 9** *Pecinová Ivana* **B 122**	**Zeměpis 9** *Dytrt Rudolf* **B 122**	**Volitelný předmět** *Lorenčíková Jana* **B 122**		**Hudební výchova 9** *Vikrorínová Ludmila* **B 122**		
Dějepis 9 *Kratochvílová Šárka* **B 122**	**Volitelný předmět** *Lorenčíková Jana* **B 122**	**Chemie 9** *Šestáková Lenka* **B 122**	**Volitelný předmět** *Lorenčíková Jana* **B 122**	**Český jazyk 9** *Vikrorínová Ludmila* **B 122**	**Výchova ke zdraví** *Černá Ivana* **B 122**		**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**		
Matematika 9 *Völklová Ludmila* **B 122**	**Volitelný předmět** *Lorenčíková Jana* **B 122**	**Člověk a svět práce** *Janeček Vladimír* **B 122**	**Matematika 9** *Völklová Ludmila* **B 122**	**Český jazyk 9** *Vikrorínová Ludmila* **B 122**	**Chemie 9** *Šestáková Lenka* **B 122**				
Matematika 9 *Völklová Ludmila* **B 122**	**Anglický jazyk 9** *Dytrtová Martina* **B 122**	**Anglický jazyk 9** *Dytrtová Martina* **B 122**	**Dějepis 9** *Kratochvílová Šárka* **B 122**	**Český jazyk 9** *Vikrorínová Ludmila* **B 122**	**Tělesná výchova** *Krajhanzlová Zdena* **Z 103**				
Výtvarná výchova *Pecinová Ivana* **B 122**	**Matematika 9** *Völklová Ludmila* **B 122**	**Anglický jazyk 9** *Dytrtová Martina* **B 122**	**Matematika 9** *Völklová Ludmila* **B 122**	**Volitelný předmět** *Lorenčíková Jana* **B 122**	**Přírodopis 9** *Šestáková Lenka* **B 122**				

E. Rozvrhy poskytnuté školou

Pro možnost porovnání vygenerovaných rozvrhů, zde uvádíme příklad rozvrhů poskytnutých školou.

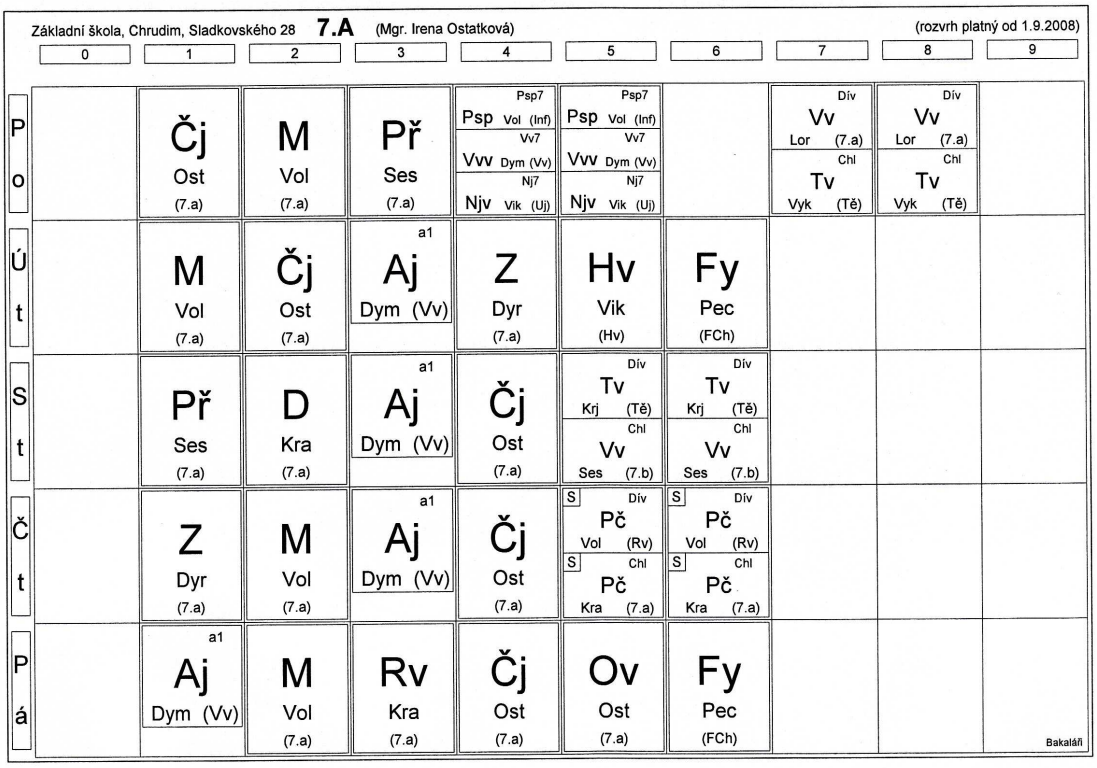


Základní škola, Chrudim, Sladkovského 28 **6.A** (Mgr. Ludmila Viktorinová) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		Čj Vik (6.a)	Z Vyk (6.a)	M Lor (6.a)	a2 Aj Lor (6.a)	Vo Lor (6.a)	D Ost (6.a)			
Út		Čj Vik (6.a)	Hv Vik (Hv)	M Lor (6.a)	Div Vv Lor (6.a); Chl Tv Vyk (Tě)	Div Vv Lor (6.a); Chl Tv Vyk (Tě)		Div Tv Krj (Tě); Chl Vv Pec (6.b)	Div Tv Krj (Tě); Chl Vv Pec (6.b)	
St		M Lor (6.a)	a2 Aj Lor (6.a)	Čj Vik (6.a)	Fy Pec (FCh)	D Ost (6.a)	it61 Ikt Vol (Inf)			
Čt		M Lor (6.a)	Čj Vik (6.a)	a2 Aj Lor (6.a)	Z Vyk (6.a)	Př Ses (6.a)		S Div Csp Kra (6.a); S Chl Csp Jan (Di)	S Div Csp Kra (6.a); S Chl Csp Jan (Di)	
Pá		Čj Vik (6.a)	Fy Pec (FCh)	Tv Vyk (Tě)	M Lor (6.a)	it63 Ikt Vol (Inf)				Bakaláři

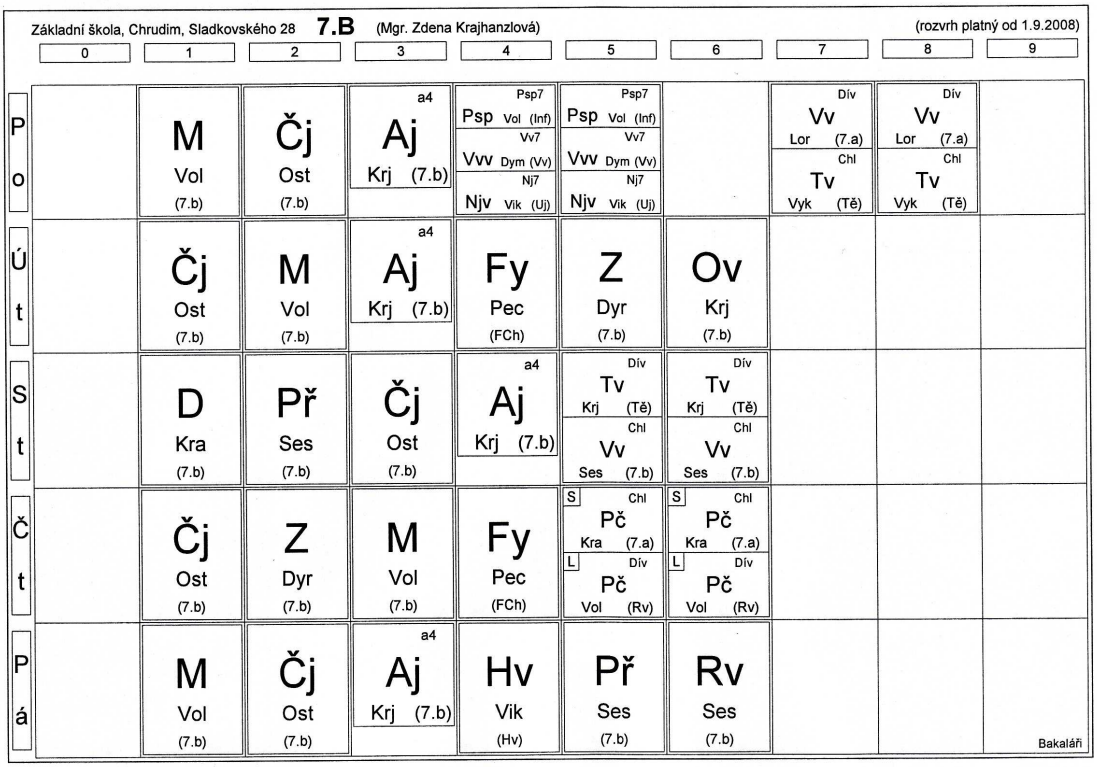
Základní škola, Chrudim, Sladkovského 28 **6.B** (Mgr. Ivana Pecinová) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		M Pec (6.b)	Fy Pec (FCh)	Čj Bar (6.b)	a3 Aj Krj (6.b)	D Ost (6.b)	it62 Ikt Vol (Inf)			
Út		M Pec (6.b)	Čj Bar (6.b)	Z Dyr (6.b)	Div Vv Lor (6.a); Chl Tv Vyk (Tě)	Div Vv Lor (6.a); Chl Tv Vyk (Tě)		Div Tv Krj (Tě); Chl Vv Pec (6.b)	Div Tv Krj (Tě); Chl Vv Pec (6.b)	
St		Čj Bar (6.b)	M Pec (6.b)	a3 Aj Krj (6.b)	Př Ses (6.b)	Fy Pec (FCh)	D Ost (6.b)			
Čt		Čj Bar (6.b)	M Pec (6.b)	Hv Vik (Hv)	Vo Lor (6.b)	Z Dyr (6.b)		L Div Csp Kra (6.a); L Chl Csp Jan (Di)	L Div Csp Kra (6.a); L Chl Csp Jan (Di)	
Pá		a3 Aj Krj (6.b)	Čj Bar (6.b)	M Pec (6.b)	Tv Krj (Tě)	it63 Ikt Vol (Inf)				Bakaláři



Základní škola, Chrudim, Sladkovského 28 **7.A** (Mgr. Irena Ostatková) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		Čj Ost (7.a)	M Vol (7.a)	Př Ses (7.a)	Psp7 Psp Vol (Inf) / Vv7 Vvv Dym (Vv) / Nj7 Njv Vik (Uj)	Psp7 Psp Vol (Inf) / Vv7 Vvv Dym (Vv) / Nj7 Njv Vik (Uj)		Dív Vv Lor (7.a) / Chl Tv Vyk (Tě)	Dív Vv Lor (7.a) / Chl Tv Vyk (Tě)	
Út		M Vol (7.a)	Čj Ost (7.a)	a1 Aj Dym (Vv)	Z Dyr (7.a)	Hv Vik (Hv)	Fy Pec (FCh)			
St		Př Ses (7.a)	D Kra (7.a)	a1 Aj Dym (Vv)	Čj Ost (7.a)	Dív Tv Krj (Tě) / Chl Vv Ses (7.b)	Dív Tv Krj (Tě) / Chl Vv Ses (7.b)			
Čt		Z Dyr (7.a)	M Vol (7.a)	a1 Aj Dym (Vv)	Čj Ost (7.a)	S Dív Pč Vol (Rv) / S Chl Pč Kra (7.a)	S Dív Pč Vol (Rv) / S Chl Pč Kra (7.a)			
Pá		a1 Aj Dym (Vv)	M Vol (7.a)	Rv Kra (7.a)	Čj Ost (7.a)	Ov Ost (7.a)	Fy Pec (FCh)			Bakaláři



Základní škola, Chrudim, Sladkovského 28 **7.B** (Mgr. Zdena Krajhanzlová) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		M Vol (7.b)	Čj Ost (7.b)	a4 Aj Krj (7.b)	Psp7 Psp Vol (Inf) / Vv7 Vvv Dym (Vv) / Nj7 Njv Vik (Uj)	Psp7 Psp Vol (Inf) / Vv7 Vvv Dym (Vv) / Nj7 Njv Vik (Uj)		Dív Vv Lor (7.a) / Chl Tv Vyk (Tě)	Dív Vv Lor (7.a) / Chl Tv Vyk (Tě)	
Út		Čj Ost (7.b)	M Vol (7.b)	a4 Aj Krj (7.b)	Fy Pec (FCh)	Z Dyr (7.b)	Ov Krj (7.b)			
St		D Kra (7.b)	Př Ses (7.b)	Čj Ost (7.b)	a4 Aj Krj (7.b)	Dív Tv Krj (Tě) / Chl Vv Ses (7.b)	Dív Tv Krj (Tě) / Chl Vv Ses (7.b)			
Čt		Čj Ost (7.b)	Z Dyr (7.b)	M Vol (7.b)	Fy Pec (FCh)	S Chl Pč Kra (7.a) / L Dív Pč Vol (Rv)	S Chl Pč Kra (7.a) / L Dív Pč Vol (Rv)			
Pá		M Vol (7.b)	Čj Ost (7.b)	a4 Aj Krj (7.b)	Hv Vik (Hv)	Př Ses (7.b)	Rv Ses (7.b)			Bakaláři

Základní škola, Chrudim, Sladkovského 28 **8.A** (Mgr. Štěpánka Švadlenková) (rozvrh platný od 1.9.2008)

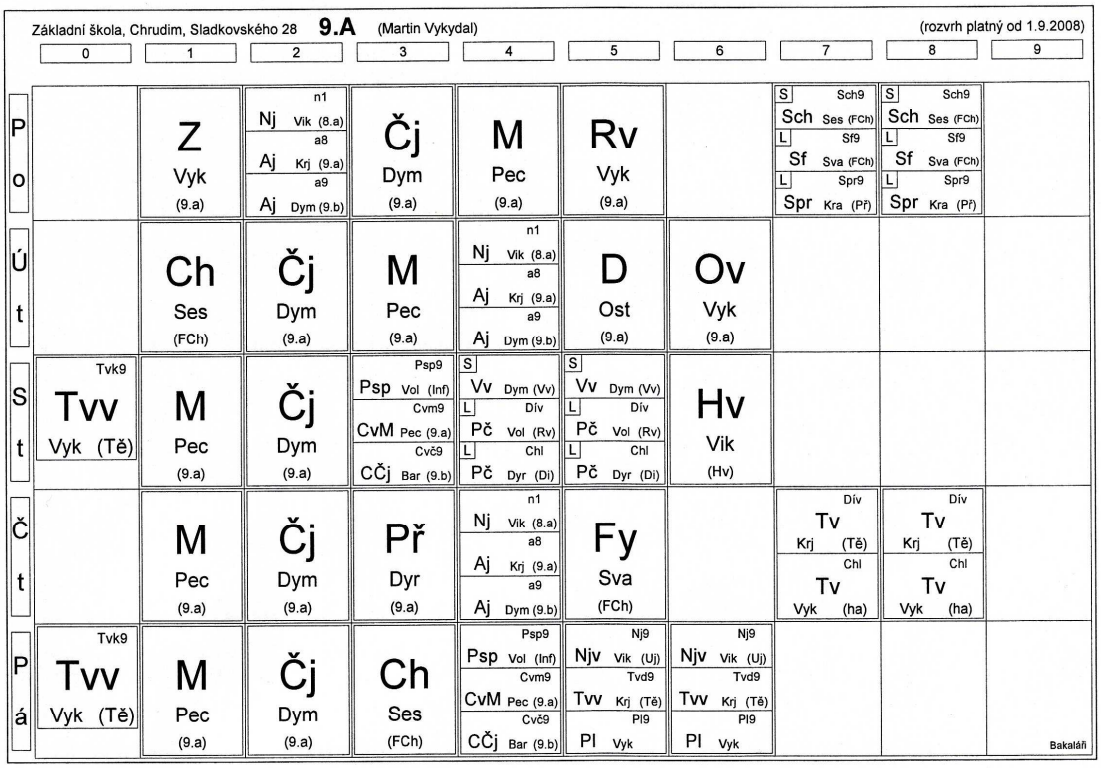
	0	1	2	3	4	5	6	7	8	9
Po		M Sva (8.a)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj)	Čj Ost (8.a)	Ch Ses (FCh)	Ov Bar (8.a)		Psp8 Psp Vol (Inf) / Pz8 PrZ Dyr (8.a) / Nj8 Njv Vik (8.b)	Psp8 Psp Vol (Inf) / Pz8 PrZ Dyr (8.a) / Nj8 Njv Vik (8.b)	
Út		M Sva (8.a)	Př Kra (8.a)	Čj Ost (8.a)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj)	S Vv Dym (Vv) / L Dív Pč Vol (Rv) / L Chl Pč Jan (Di)	S Vv Dym (Vv) / L Dív Pč Vol (Rv) / L Chl Pč Jan (Di)			
St		Čj Ost (8.a)	Z Vyk (8.a)	Fy Sva (FCh)	D Kra (8.a)	Hv Vik (Hv)	Rv Kra (8.a)			
Čt		Ch Ses (FCh)	Čj Ost (8.a)	M Sva (8.a)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)			
Pá		Čj Ost (8.a)	Z Vyk (8.a)	M Sva (8.a)	D Kra (8.a)	Fy Sva (FCh)	Př Kra (8.a)			

Bakaláři

Základní škola, Chrudim, Sladkovského 28 **8.B** (Mgr. Šárka Kratochvílová) (rozvrh platný od 1.9.2008)

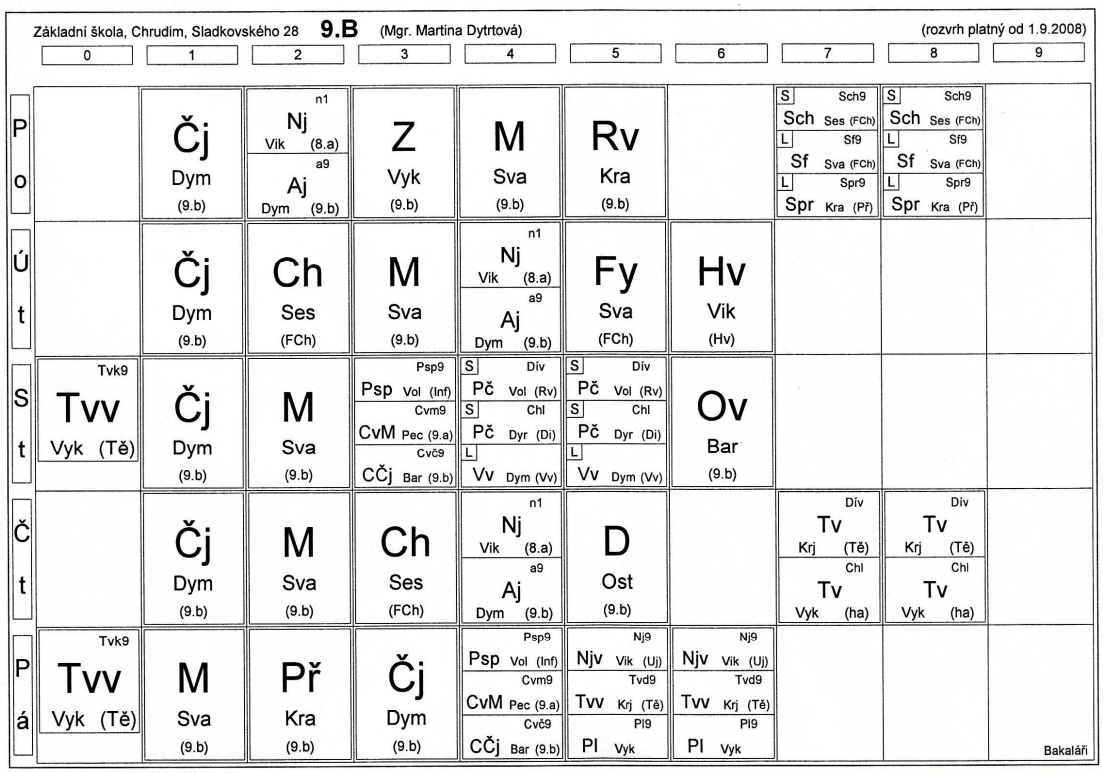
	0	1	2	3	4	5	6	7	8	9
Po		D Kra (8.b)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj) / a6 Aj Sva (8.b)	M Sva (8.b)	Čj Bar (8.b)	Ch Ses (FCh)		Psp8 Psp Vol (Inf) / Pz8 PrZ Dyr (8.a) / Nj8 Njv Vik (8.b)	Psp8 Psp Vol (Inf) / Pz8 PrZ Dyr (8.a) / Nj8 Njv Vik (8.b)	
Út		Př Kra (8.b)	M Sva (8.b)	Čj Bar (8.b)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj) / a6 Aj Sva (8.b)	S Dív Pč Vol (Rv) / S Chl Pč Jan (Di) / L Vv Dym (Vv)	S Dív Pč Vol (Rv) / S Chl Pč Jan (Di) / L Vv Dym (Vv)			
St		Z Vyk (8.b)	Čj Bar (8.b)	D Kra (8.b)	Hv Vik (Hv)	Rv Kra (8.b)	Fy Sva (FCh)			
Čt		M Sva (8.b)	Ch Ses (FCh)	Čj Bar (8.b)	n1 Nj Vik (8.a) / a5 Aj Bar (Uj) / a6 Aj Sva (8.b)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)			
Pá		Z Vyk (8.b)	M Sva (8.b)	Čj Bar (8.b)	Fy Sva (FCh)	Př Kra (8.b)	Ov Bar (8.b)			

Bakaláři



Základní škola, Chrudim, Sladkovského 28 **9.A** (Martin Vykydal) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		Z Vyk (9.a)	n1 Nj Vik (8.a) / a8 Aj Krj (9.a) / a9 Aj Dym (9.b)	Čj Dym (9.a)	M Pec (9.a)	Rv Vyk (9.a)		S Sch9 Sch Ses (FCh) / L Sf9 Sf Sva (FCh) / L Spr9 Spr Kra (Př)	S Sch9 Sch Ses (FCh) / L Sf9 Sf Sva (FCh) / L Spr9 Spr Kra (Př)	
Út		Ch Ses (FCh)	Čj Dym (9.a)	M Pec (9.a)	n1 Nj Vik (8.a) / a8 Aj Krj (9.a) / a9 Aj Dym (9.b)	D Ost (9.a)	Ov Vyk (9.a)			
St	Tvk9 Tvv Vyk (Tě)	M Pec (9.a)	Čj Dym (9.a)	Psp9 Psp Vol (Inf) / Cvm9 CvM Pec (9.a) / Cvč9 CČj Bar (9.b)	S Vv Dym (Vv) / L Dív Pč Vol (Rv) / L Chl Pč Dyr (Di)	S Vv Dym (Vv) / L Dív Pč Vol (Rv) / L Chl Pč Dyr (Di)	Hv Vik (Hv)			
Čt		M Pec (9.a)	Čj Dym (9.a)	Př Dyr (9.a)	n1 Nj Vik (8.a) / a8 Aj Krj (9.a) / a9 Aj Dym (9.b)	Fy Sva (FCh)		Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	
Pá	Tvk9 Tvv Vyk (Tě)	M Pec (9.a)	Čj Dym (9.a)	Ch Ses (FCh)	Psp9 Psp Vol (Inf) / Cvm9 CvM Pec (9.a) / Cvč9 CČj Bar (9.b)	Nj9 Njv Vik (Uj) / Tvd9 Tvv Krj (Tě) / Pl9 Pl Vyk	Nj9 Njv Vik (Uj) / Tvd9 Tvv Krj (Tě) / Pl9 Pl Vyk			Bakaláři



Základní škola, Chrudim, Sladkovského 28 **9.B** (Mgr. Martina Dytrtová) (rozvrh platný od 1.9.2008)

	0	1	2	3	4	5	6	7	8	9
Po		Čj Dym (9.b)	n1 Nj Vik (8.a) / a9 Aj Dym (9.b)	Z Vyk (9.b)	M Sva (9.b)	Rv Kra (9.b)		S Sch9 Sch Ses (FCh) / L Sf9 Sf Sva (FCh) / L Spr9 Spr Kra (Př)	S Sch9 Sch Ses (FCh) / L Sf9 Sf Sva (FCh) / L Spr9 Spr Kra (Př)	
Út		Čj Dym (9.b)	Ch Ses (FCh)	M Sva (9.b)	n1 Nj Vik (8.a) / a9 Aj Dym (9.b)	Fy Sva (FCh)	Hv Vik (Hv)			
St	Tvk9 Tvv Vyk (Tě)	Čj Dym (9.b)	M Sva (9.b)	Psp9 Psp Vol (Inf) / Cvm9 CvM Pec (9.a) / Cvč9 CČj Bar (9.b)	S Dív Pč Vol (Rv) / S Chl Pč Dyr (Di) / L Vv Dym (Vv)	S Dív Pč Vol (Rv) / S Chl Pč Dyr (Di) / L Vv Dym (Vv)	Ov Bar (9.b)			
Čt		Čj Dym (9.b)	M Sva (9.b)	Ch Ses (FCh)	n1 Nj Vik (8.a) / a9 Aj Dym (9.b)	D Ost (9.b)		Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	Dív Tv Krj (Tě) / Chl Tv Vyk (ha)	
Pá	Tvk9 Tvv Vyk (Tě)	M Sva (9.b)	Př Kra (9.b)	Čj Dym (9.b)	Psp9 Psp Vol (Inf) / Cvm9 CvM Pec (9.a) / Cvč9 CČj Bar (9.b)	Nj9 Njv Vik (Uj) / Tvd9 Tvv Krj (Tě) / Pl9 Pl Vyk	Nj9 Njv Vik (Uj) / Tvd9 Tvv Krj (Tě) / Pl9 Pl Vyk			Bakaláři

F. Obsah CD

Na přiloženém CD jsou k dispozici zdrojové kódy implementovaného systému a tato dokumentace ve formátu PDF.