Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Systém pro správu informací o studentech

Plzeň, 2012 Roman Polák

Originál zadání

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 11. května 2012

Roman Polák

Abstract

System for managing information about students.

This bachelor thesis deal with creating application for manages information about students under Department of Computer Science and Engineering on Faculty of Applied Sciences, University of West Bohemia. The main objective is treatment of points, semester works, attendances and credits. In the first part are useable technologies, especially for creating web-database application. The second part describes realization with design, cooperation with existing system at university, system implementation and testing.

Obsah

1 Úvod

Cílem práce je vytvořit databázovou aplikaci pro správu informací o studentech, která usnadní práci při evidenci studentů a jejich hodnocení na předmětech. Zároveň bude systém poskytovat výstup se souhrnnými informacemi o výsledcích studentů. Podle typu předmětu bude možné zapisovat u studentů docházku, přiřadit semestrální práci nebo udělit bodové ohodnocení. Všechny tyto informace pak přehledně zobrazí formou webové stránky. Vyučující se po přihlášení dostane k úpravě již zmíněných detailů předmětů a jejich studentů.

V první části práce jsou popsány technologie, které je možné využít při řešení, jako jsou informace o databázích, kde lze uchovávat potřebná data. Dále jsou popsány nejznámější redakční systémy a způsob jejich použití. Využitelné pro tuto práci jsou zejména PHP frameworky usnadňující vytváření webových aplikací.

Druhá část práce se zabývá samotnou realizací aplikace, výběrem technologií, popsaných v této práci, a jejich využitím při realizaci. Nezbytnou součástí je i testování správné funkčnosti aplikace s popisem zjištěných nedostatků a návrhů vylepšení.

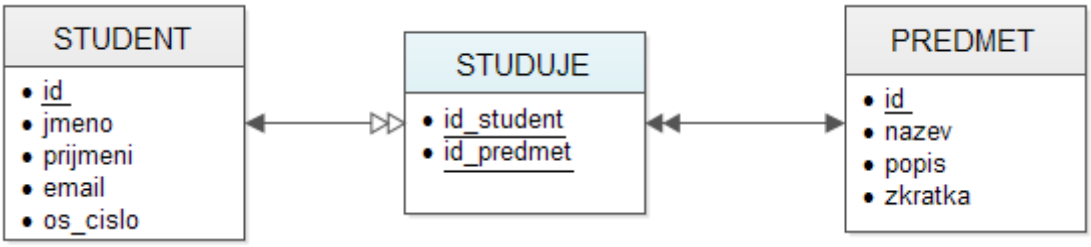
2 Databázové aplikace

Pod databázovou aplikací si lze představit kteroukoliv aplikaci pracující s daty, které je potřeba uchovávat. Na uchování dat slouží databáze, která většinou bývá oddělena od aplikací, které s databází komunikují. Komunikace je ve většině případů typu Klient-Server a jako dotazovací jazyk se používá SQL. Pomocí dotazu je zaslán požadavek na server s databází, následně server provede dotaz a vrací výsledná data zpět.

2.1 Obecná charakteristika

Podle toho jak jsou data ukládána do databáze, tak se dělí na hierarchické, síťové a relační databáze. Tyto názvy vyplývají ze struktury dat, tedy modelu databáze. Nejnovější a nejpoužívanější z nich je relační model. Data se ukládají do tabulek, kde sloupce jsou atributy a jednotlivé řádky uložené záznamy. Charakteristické pro tento model jsou relace mezi tabulkami. Aby taková relace vznikla, je potřeba dvě a více tabulek. Dané tabulky pak obsahují stejné atributy, přes které se tabulky propojí. Například jedinečný identifikátor objektu, jako je třeba osobní číslo studenta či zaměstnance.

2.1.1 Příklad relační databáze



Obrázek 1 – Relační schéma

Ukázka relační databáze znázorňuje vazbu mezi studenty a jejich předměty. V tabulce STUDENT odpovídá jeden záznam jednomu studentovi, stejně tak je tomu u tabulky PŘEDMĚT. Tzv. vazební tabulka STUDUJE slouží k propojení těchto dvou. Uchovává identifikátory studentů a předmětů, pomocí kterých lze později jednoznačně určit předměty, které student studuje nebo naopak zjistit studenty předmětu.

2.2 Databázové systémy

Databázový systém nebo také systém řízení báze dat (SŘBD) tvoří rozhraní mezi databází (uloženými daty) a aplikací. Každý SŘBD se stará o strukturu dat a jejich správu, jako je vytváření, úpravy, výběr dat a jejich ochrana.

Mezi programy SŘBD patří např. Oracle, MS SQL Server, MySQL, PostgreSQL a další. Jako příklad zde uvádím jen některé z nejpoužívanějších.

2.2.1 MySQL

Jedná se o nejoblíbenější databázi s otevřeným zdrojovým kódem, která byla v roce 2011 vyhlášena jako nejlepší řešení pro správu dat [1]. Nejčastěji se používá u jednoduchých webových aplikací. Vyznačuje se zejména rychlostí, spolehlivostí a snadným použitím.

Komunikace probíhá u této relační databáze systémem typu Klient-Server. Výsledná databáze je snadno přenositelná mezi jednotlivými platformami a její použití v menších projektech podporuje fakt, že se jedná o open-source[1] databázi s licencí GPL[2].

2.2.2 PostgreSQL

Stejně jako v případě MySQL databáze jde o software s volnou licencí. Oproti MySQL se ale jedná o licenci BSD, která se od GPL liší lehce licenční politikou volného softwaru. V případě GPL je podmínkou jakoukoliv úpravu softwaru zpětně poskytnout. Naopak vlastní software založený na kódu s licencí BSD[3] je možné distribuovat pod svou licencí.

Z pohledu funkčnosti a využitelnosti databáze jsou v současnosti MySQL i PostgreSQL na srovnatelné úrovni. Při výběru mezi těmito databázemi tedy záleží na volbě programátora, kterou použije.

[1] **Open-source** – otevřený zdrojový kód a legální licence na software
[2] **GPL** – *General Public Licence* (všeobecná veřejná licence)
[3] **BSD** - *Berkeley Software Distribution* – licence pro svobodný software

2.2.3 Oracle Database

Komerční produkt firmy Oracle, který se řadí mezi nejvýkonnější SŘBD. Obsahuje veškeré funkcionality a vlastnosti dnešních SŘBD. Je vhodný především pro využití ve větších firmách a projektech. Důvodem tohoto zařazení je relativně vysoká cena a zbytečně velké požadavky na hardware.

2.3 Srovnání uvedených databází

Shrnutí popsaných systémů řízení báze dat (Tabulka 1 – Srovnání SŘBD). Navíc je zde produkt MS Access, který je dodáván v balíku MS Office. Kombinuje typ relační databáze a grafické uživatelské rozhraní.

Hodnocení použitelnosti je orientačně rozděleno do 3 stupňů na základní, dobré a velmi dobré. Hodnotit má především využití u menších webových aplikací. MS Access je určen spíše pro vývoj a návrh databáze, pro ostrý provoz je od Microsoftu určen MS SQL, proto tedy použitelnost databáze k aplikaci pouze základní. MySQL i PostgreSQL jsou využívány jako databázové servery u mnoha webových aplikací, poskytují většinu potřebných funkcí a to vše pod open-source licencí. PostgreSQL je robustnější a ve srovnání se MySQL zřejmě o stupeň výš [2]. Oracle je se svou databází nejdále, obsahuje všechny nejnovější funkce a technologie. Samotná databáze ale zabírá oproti předcházejícím několikanásobně větší místo na disku a investice do této databáze by se pro menší produkty určitě nevyplatila. Cena produktu je hodnocena kladně nebo záporně. V tomto případě odpovídá licenci softwaru.

	Verze	Použitelnost	Operační systém	Licence	Cena
MySQL	5.1	velmi dobrá	Win, Linux	GPL, K[4]	+
PostgreSQL	9.1.3	velmi dobrá	Win, Linux, Solaris, MacOS	BSD	+
Oracle	11g	dobrá	Win, Linux, Solaris	K	-
MS Access	2010	základní	Win	K	-

Tabulka 1 - Srovnání SŘBD

[4] **K** – Komerční licence

3 Redakční systémy

Redakční systémy, známé také z angličtiny jako CMS[5] jsou aplikace umožňující vytváření a úpravu článků, textů, fotografií a jejich publikaci na webu. Dle funkcí, které systém má a pro jakou oblast je navržen, lze například rozdělit práva uživatelů do skupin nebo využít systém pro správu verzí textů. Správa dat pomocí redakčního systému nevyžaduje znalost programovacího jazyka.

Existují různé typy redakčních systémů podle jejich využití. Nejčastějším typem, s kterým může běžný uživatel přijít do styku, je WCMS[6] (Web CMS) na tvorbu a správu obsahu webových stránek. Při tvorbě webu pomocí redakčního systému si lze vybrat ze šablon, které systém nabízí, dále přidat funkce jako jsou diskuzní fóra, ankety a obsah webu vytvořit v uživatelsky přívětivém prostředí. Dalším typem je systém pro správu dokumentů využitelný v rámci podniku jako interní aplikace.

3.1 Nejpoužívanější redakční systémy

Vzhledem k zaměření této práce na technologie použitelné na webu uvádím pouze příklady redakčních systémů typu WCMS a jejich základní charakteristické vlastnosti.

3.1.1 WordPress, Drupal, Joomla

Pravděpodobně u nás nejpopulárnější [2] systém zdarma je WordPress. Vhodný především pro začínající uživatele díky své jednoduchosti. Postačuje na jednodušší webové prezentace a blogy. Má rozsáhlou českou podporu [2] včetně návodů a aktuální české verze ke stažení.

CMS Drupal je pokročilejší a má větší nároky na uživatelovu znalost vývoje a programování. Na druhou stranu se hodí u větších projektů s rozsáhlejším obsahem, který lze přehledně třídit do kategorií. Dále poskytuje například funkce sociálních sítí, tvorbu e-shopu a jiné. Podporu lze najít na webových stránkách české komunity [3] současně s návody a dalšími informacemi.

[5] **CMS** – Content management system (redakční systém)
[6] **WCMS** – Web content management system (webový redakční systém)

Co se týče uživatelské obtížnosti, tak se Joomla řadí mezi již dva zmíněné systémy. Důvodem je především větší množství funkcí oproti CMS Wordpress a tím i méně přehledné uživatelské prostředí. Stejně jako u předchozích česká komunita s návody dostupná na webu [4].

3.2 Využití

Všechny z uvedených příkladů redakčních systémů jsou pod licencí GPL, tedy tzv. otevřený software. Lze nainstalovat českou verzi a mají také podporu české komunity. Pro uživatele, který si chce vytvořit svůj vlastní blog nebo jednoduše prezentovat svoji firmu na internetu, může využít právě těchto redakčních systémů. Pomocí nich lze vytvořit rychle a jednoduše webové stránky a později upravovat obsah. Slouží tedy především pro tvorbu a správu standardních stránek s využitím univerzálních komponent.

V případě aplikace na správu studentů by bylo nutné doprogramovat veškeré komponenty a ty, které redakční systém nabízí, by zůstaly nevyužity. V tomto směru by nemělo význam instalovat CMS a k němu „přilepit“ svoji vlastní aplikaci.

4 Framework

Jedná se o softwarovou strukturu, která napomáhá při vývoji aplikací. Konkrétně jde o jádro systému, na kterém lze postavit novou aplikaci. Frameworky většinou implementují mnoho typů návrhových vzoru podle potřeby, viz kapitola 4.2 Návrhové vzory. Poskytují knihovnu scriptů a funkcí, které by v jiných případech bylo nutné psát pořád dokola a tím šetří čas při vývoji aplikací.

4.1 Využití

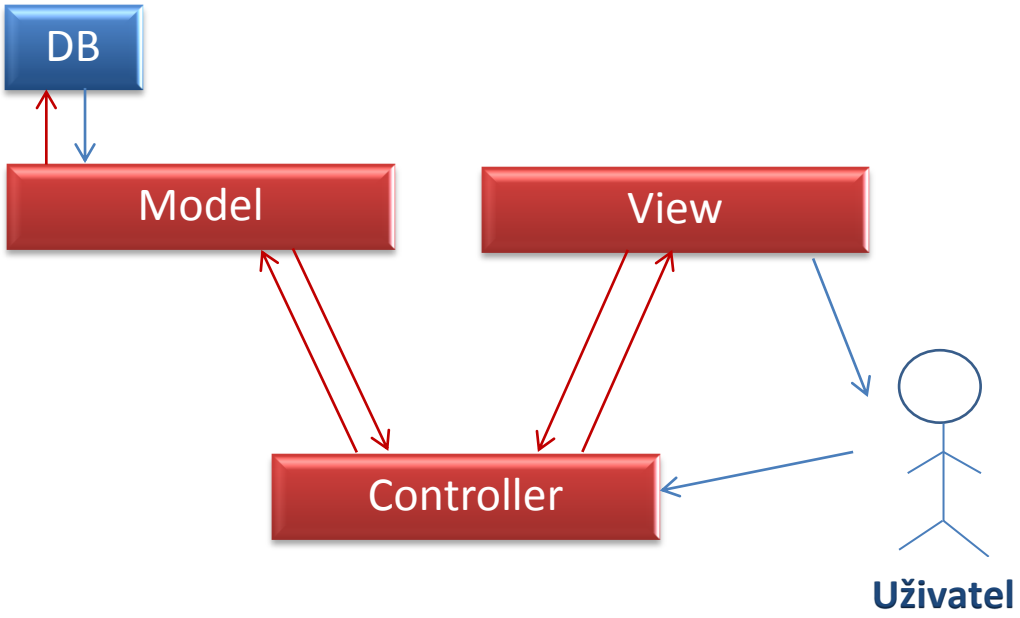
Při použití frameworku jako základu pro vyvíjený software se může programátor spolehnout na fungující jádro systému a zabývat se svou konkrétní aplikací s tím, že ze souboru knihoven frameworku využije to, co právě potřebuje.

4.2 Návrhové vzory

Před programováním aplikace je potřeba udělat analýzu problému a návrh řešení. K tomu mimo jiné slouží návrhové vzory (design pattern). Existují tři skupiny vzorů popisující vytváření objektů, strukturu a jejich chování. Na rozdíl od frameworku nejde o šablonu nebo fungující skript. Návrhové vzory pouze popisují postup jak daný typ problému řešit a definují strukturu navrhovaného systému. Aplikace psaná podle takového vzoru je pak lépe čitelná a snadno upravitelná.

4.2.1 Model-View-Controller

Jedná se o architekturu, která rozděluje aplikaci na tři nezávislé vrstvy: Model, View a Controller. Změny v těchto vrstvách mají pak minimální vliv na ostatní a lze je upravovat samostatně.



Obrázek 2 – Schéma MVC architektury

- **Model** – vrstva pracující s daty aplikace
- **View** – zpracování dat do čitelné podoby uživatel
- **Controller** – zpracování požadavků od uživatele, podle vstupní akce provádí změny v Model a View
- **DB** – databáze nebo jakékoliv úložiště dat pro aplikaci)

Pokud uživatel provede v aplikaci akci jako je stisk tlačítka (např. zobrazení všech studentů předmětu), Controller zpracuje jeho požadavek a požádá Model o data, který je získá z databáze. Výsledek předá zpátky a pomocí View se tyto data zobrazí uživateli.

4.3 Příklady frameworků

Zde jsou uvedeny nejznámější frameworky používané k vývoji webových aplikací.

4.3.1 Zend

Zend Framework lze také najít pouze pod názvem ZF. Jedná se o objektově orientovaný framework psaný v PHP 5 s opensource licencí určený pro webové aplikace. Asi nejdůležitější součástí, které ZF zahrnuje, je implementace vzoru MVC (viz kapitola 4.2.1 Model-View-Controller). Dále obsahuje mnoho modulů zajišťující

autorizaci uživatele, zasílání emailů, tvorbu PDF dokumentů a další. Podporuje všechny databáze zmíněné v kapitole srovnání (2.3 Srovnání uvedených databází).

Celosvětově je velmi rozšířený a používaný. Dokumentace jsou dostupné v angličtině na http://framework.zend.com/docs/overview. Obsahují velké množství příkladů a na stránkách Zendu je možné shlédnout video tutoriály nebo návody v textové podobě.

4.3.2 Nette

Stejně jako v případě ZF je psaný v PHP 5. Poskytován pod licencí opensource GPL a navíc licencí Nette (obdoba BSD licence). Autorem je původně český programátor David Grudl, díky tomu jsou k dispozici dokumentace a podpora v češtině. Jeho vývoj začal déle než ZF, obsahuje tedy prozatím méně modulů oproti ZF, které jsou programátorovi k dispozici.

Dokumentace k tomuto frameworku jsou k dispozici na http://nette.org, kde lze nalézt tutoriály a ostatní podporu.

4.3.3 jQuery

Od dvou předchozích se jQuery podstatně liší. Na Zend i Nette frameworku lze začít psát ihned vlastní webovou aplikaci díky jádru, které obsahují. Proti tomu je jQuery jako doplněk, který můžeme použít k některému z frameworků. Jde o knihovnu javascriptových funkcí využitelných pro zavedení dynamických prvků do webových stránek. Bez větší znalosti javascriptu je možné umístit na stránky například stylovou fotogalerii, kalendář, načítat část obsahu bez nutnosti obnovení celé stránky nebo použít nějaké efektní animace.

Jako u předchozích je šiřitelný pod licencí GPL a navíc kombinací s licencí MIT[7].

4.4 RRSoft Web-engine

Zařazení mezi frameworky možná není až tak přesné. Patří někam mezi frameworky a šablonovací systémy s tím, že využívá architekturu MVC a je psán v direktivě převzaté od Zendu. V současné době je stále ve vývoji, ale v dané fázi

[7] **MIT** – svobondá licence z Massachusets Institute of Technology

poskytuje hlavní jádro s funkčními moduly postačující k tvorbě webových aplikací. Co se týče šíření, patří taktéž mezi opensource software.

Původně jsem zvažoval použít tento web-engine jako základ aplikace pro správu studentů. Při pokusech daný systém rozchodit a použít ho jako nástroj usnadňující tvorbu webových aplikací jsem nakonec tento systém opustil z následujících důvodů.

Předchozí systémy jsou obecně známé, mnohem více rozšířené a častěji používané. Jsou tak dostupné různé návody a dokumentace jak daný systém používat nebo se lze připojit ke komunitě lidí kolem daného softwaru a vyřešit svůj problém. V tomto směru RRSoft Web-engine nemá nic z uvedeného. Na začátku se zdálo využití tohoto enginu jako dobrý nástroj k vytvoření aplikace na správu studentů nebo jakéhokoliv jiného webu. S postupem času jsem ale zjistil, že absence dokumentace je větší problém než jsem předpokládal a proto jsem v další implementaci pod tímto systémem nepokračoval a zvolil vlastní kostru aplikace viz 6. Návrh aplikace.

Tímto nechci říct, že se jedná o špatný nástroj, ale pro jiného člověka než je autor bude podstatně složitější práce se softwarem, který není popsán v žádné dokumentaci oproti nástrojům kompletně zdokumentovaným včetně návodů a příkladů.

5 Webové služby IS/STAG

Na Západočeské univerzitě v Plzni (dále jen ZČU) existuje systém zvaný IS/STAG, který obsahuje mnoho informací (o studentech, předmětech, katedrách, oborech atd.). Přístup k těmto informacím zajišťuje například webové rozhraní portál ZČU. Aby bylo možné vytvářet další aplikace jako rozhraní využívající informací školy, jsou potřeba služby poskytující výstup požadovaných dat v daných formátech.

Zmíněný portál ZČU umožňuje mimo jiné také správu studentů na univerzitní úrovni, jako jsou údaje o studentech a závěrečné hodnocení studenta u zkoušky. Průběžné hodnocení v předmětech, docházku studentů, přidělení semestrálních prací atd. portál neumožňuje a proto si každý vyučující předmětu zajišťuje zpracování těchto informací sám. Právě tyto údaje o studentech zpracovává aplikace popsaná v druhé části práce.

K poskytování informací z IS/STAG slouží také webové služby. Dále se pak využívají na webových stránkách kateder, místních systémech evidence známek a zápočtů nebo jen k exportování dat do tabulky v Excelu.

5.1 Popis poskytovaných služeb

Webové služby mají svoji adresu URL. Podle požadavku uživatele lze sestavit URL adresu tak, aby se na výstupu zobrazili žádané informace (podrobněji v kapitole 5.2 Generování dat). Jsou dva typy webových služeb, standard SOAP a REST [6]. Jedná se o protokoly založené na formě XML a určené pro komunikaci pomocí HTTP. Jednodušším a uživatelsky bližším je standard REST, který se hodí na získání dat pro zmíněné aplikace kateder nebo evidenční systémy studentů.

5.2 Příklad webové služby

Poskytované služby na základě svého názvu a parametrů tvoří adresu URL. Na výstupu je výchozí formát XML[8] nebo možnost výběru z XLS[9], CSV[10] a ICS[11]. V evidenci studentů postačuje výstup v XML. Pro vygenerování příslušných dat je potřeba vytvořit požadavek, tedy URL adresu v následujícím tvaru:

[8] **XML –** značkovací jazyk, standardizovaný formát pro výměnu informací
[9] **XLS** – formát aplikace Microsoft Excel (tabulkový editor)
[10] **CSV** – jednodušší tabulkový formát, který lze otevřít v MS Excel
[11] **ICS** – formát pro data z kalendáře

https://stag-ws.zcu.cz/ws/services/rest/predmety/getPredmetyByKatedra?katedra=KIV

- https://stag-ws.zcu.cz/ws/services
  - základ URL, na této adrese lze nalézt seznam poskytovaných služeb
- rest/predmety/
  - typ standardu a adresa služby
- getPredmetyByKatedra
  - název volané metody (prováděná operace)
- ?katedra=KIV
  - parametry metody, v případě většího počtu parametrů se používá "&" na jejich oddělení

Výstupem tohoto požadavku je XML soubor ve tvaru:

```
<ns1:getPredmetyByKatedraResponse   xmlns:ns1="http://stag-ws.zcu.cz/">
     <predmetyKatedry>
          <predmetKatedry>
               <zkratka>ACG</zkratka>
               <nazev>Počítačová grafika pro pokročilé
               </nazev>
               <katedra>KIV</katedra>
               <rok>2011</rok>
          </predmetKatedry>
          <predmetKatedry>
               <zkratka>ACS</zkratka>
               ……..
               ……..
          </predmetKatedry>
          …….
    </predmetyKatedry>
</ns1:getPredmetyByKatedraResponse>
```

5.3 Zabezpečení

Vzhledem k povinnostem správců při ochraně osobních údajů, které ukládá Úřad pro ochranu osobních údajů, jsou některé metody, poskytující informace o studentech, veřejnosti nepřístupné. K těmto datům se lze dostat až po ověření přihlašovacích údajů proti STAGu. Takto chráněné metody je nutné volat přes zabezpečený protokol HTTPS.

Ostatní služby, které nevyžadují autorizovaného uživatele, je možné volat bez omezení. Zpravidla jde o získání informací, které jsou běžně přístupné, a není důvod omezovat jejich přístup.

6 Návrh aplikace

Před samotným návrhem konkrétní datové struktury neboli databázového modelu a návrhem struktury aplikace, která na tento model navazuje, je potřeba specifikovat požadavky na systém.

6.1 Požadavky na systém

Specifikace požadavku říká, co má daný systém umět a jaké má mít funkce s podrobnějším popisem pro implementaci. Tzv. podnikatelský požadavek je zmíněn hned v úvodu. To znamená, proč zadavatel chce tento systém a k čemu mu bude dobrý oproti stávajícím nástrojům. Nejdůležitější částí požadavků jsou uživatelské požadavky, u kterých jsou přesně specifikovány cíle a úkoly, které budou uživatelé se systémem provádět. Dalším typem jsou systémové požadavky, které definují zejména zdroje a struktury.

1. **Evidence docházky na předmětu**
   Systém umožňuje evidování docházky ve zvoleném předmětu. Možnost změny je buď individuální, nebo hromadná po výběru předmětu.

2. **Evidence semestrálních prací**
   V detailech studenta je možné přidat semestrální práci z předmětu, který studuje a je načtený v této aplikací s požadavky na semestrální práci. Dále u přidané práce měnit její stav zpracování a vkládat doplňující komentáře.

3. **Nastavení požadavků na zápočet u předmětu**
   Při načítání nového předmětu lze vybrat parametry předmětu, jako jsou možnosti bodování, evidování docházky nebo semestrálních prací. Tuto možnost nastavení lze provést i u vytvořeného (načteného) předmětu.

4. **Rozdělení práv přístupu**
   Uživatelé, kteří budou pracovat se systémem, se dělí do dvou skupin. Na vyučující se všemi právy na změny a studenty s právem přístupu na prohlížení vybraných částí (viz obrázek Diagram užití).

5. **Souhrn výsledků pro studenty**
   Zobrazení souhrnné tabulky s výsledky (body a semestrální práce) rozdělené podle předmětů.

6. **Export zobrazovaných tabulek**
   Zobrazované tabulky, jako jsou studenti předmětu, docházka nebo seznam semestrálních prací lze exportovat do souboru *.csv[12] pro další úpravy mimo aplikaci.

7. **Komunikace s IS STAG**
   Systémový požadavek na zdroj použitých dat. Aplikace bude komunikovat s IS STAG a získávat z něj data o studentech a předmětech.

6.2 Výběr technologií

Vzhledem k implementaci systému jako webové aplikace jsou použity technologie spojené s internetem a tvorbou webových stránek. Základním jazykem, který je potřeba pro zobrazení i jednoduchého textu, je HTML[13]. Vše, co se dnes zobrazuje na internetu, je pomocí HTML a i využití jiných technologií nic nemění na tom, že HTML je potřeba vždy. Proto je tedy použití této technologie nezbytné.

Aby stránka nebyla jen prostý text naskládaný pod sebe bez jakékoliv úpravy, využívá se kaskádových stylů CSS[14], které umožňují pokročilejší formátování než samotné HTML. Pomocí tohoto jazyka je možné nastavit vzhled stránky z jednoho místa.

Pomocí HTML a CSS se stránka zobrazí tak jak je napsána (tzv. statická). Pro tvorbu dynamických stránek, které oproti statickým jen nezobrazují napevno nastavené stránky, je potřeba využít programovací jazyk. V případě této aplikace jsem zvolil PHP[15]. Výhodou je, že je přímo určené pro tvorbu webových stránek a podpora u většiny poskytovatelů hostingových služeb je samozřejmostí. Pro spuštění stačí, aby zdrojové soubory byly umístěny na serveru (např. na lokálním PC) a webová aplikace se spustí s prvním http[16] požadavkem.

[12] **CSV** – *Comma-separated* values, obecný tabulkový formát pro výměnu dat
[13] **HTML** – Hyper Text Markup Language – značkovací jazyk pro tvorbu webových stránek
[14] **CSS** – Cascading Style Sheets – jazyk pro formátování prvků v HTML
[15] **PHP** – Hypertext Preprocesor – programovací jazyk pro internetové stránky
[16] **http** – HyperText Transfer Protocol – přenosový protokol pro výměnu HTML dokumentů

S kombinací předchozích technologií je možné vytvořit dynamické webové stránky. Ve většině případů je ale potřeba ukládat nějaká data, která aplikace využívá, spravuje nebo si jen ukládá svůj stav. Pro tyto účely lze použít jednoduché ukládání do souboru. Ovšem následná práce se soubory není moc efektivní, a pokud chceme k datům přistupovat z více míst najednou, je lepší použít systém řízení báze dat (viz kapitola 2.2 Databázové systémy). Z uvedených databází zůstávají na výběr MySQL a PostregeSQL s volnou licencí. Na základě dostupných hodnocení a předchozí zkušeností jsem se rozhodl pro MySQL, které poskytuje veškeré funkce využitelné v této aplikaci. Samozřejmostí je využití strukturovaného dotazovacího jazyka SQL (Structured Query Language) pro definování a manipulaci s daty v databázi.

Ostatní technologie typu JavaScript[17] nebo Flash[18] jsou na zvážení a použitelné spíše pro pozdější vylepšení aplikace.

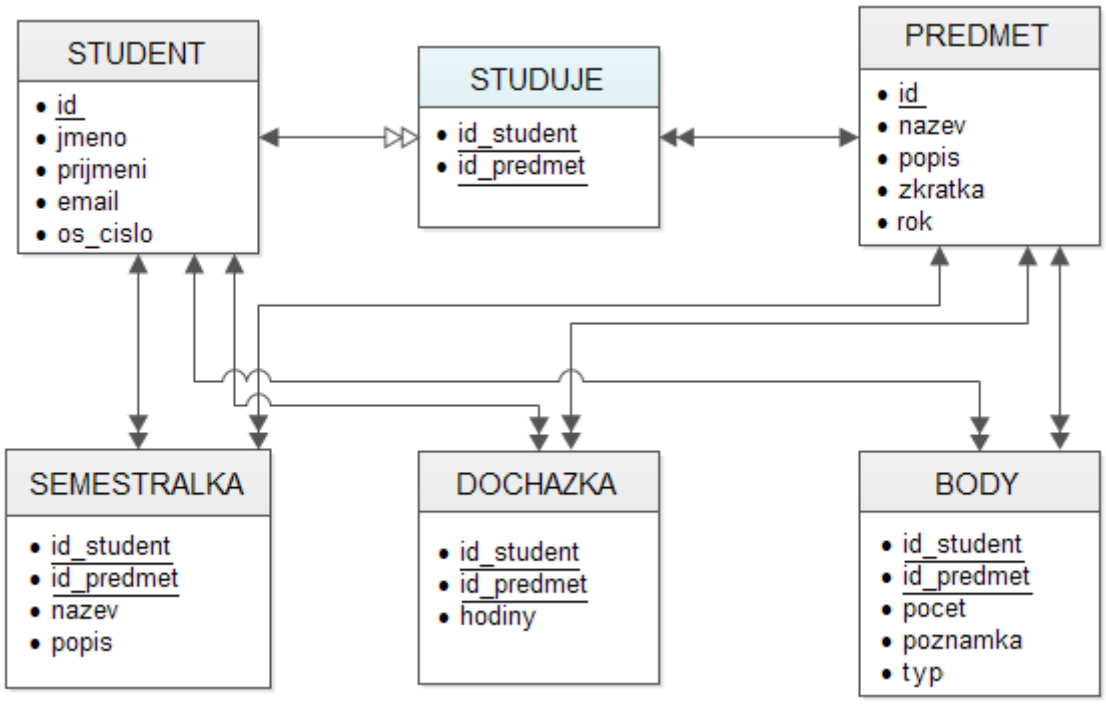
Posledním typem využitelných technologií pro tuto práci je Document Object Model (DOM) a Extensible Markup Language (XML). Využití XML nepřímo přikazuje systémový požadavek č. 7 v kapitole 6.1 Požadavky na systém, protože výměna dat mezi aplikací a IS/STAG probíhá právě pomocí XML souboru. DOM slouží pro práci s webovou stránkou [5], konkrétně přistupuje k souboru XML jako k jednotlivým objektům a umožňuje další práci jako se stromovou strukturou.

6.3 Návrhové modely

Zde jsou uvedeny grafické reprezentace aplikace z pohledu databáze (Datový model), uživatele (Diagram případů užití) a model samotné aplikace (Model aplikace).

[17] **JavaScript** – programovací jazyk pro oživení webových stránek (živé menu, efekty …)
[18] **Flash** – grafický program pro tvorbu interaktivních aplikací

6.3.1 Datový model



Obrázek 3 - Datový model

Popis tabulek a jejich atributů

STUDENT - entita shromažďující osobní informace studenta

- **id** – primární klíč studenta, číslování 1 až N
- **jmeno** – jméno studenta
- **prijmeni** – příjmení studenta
- **email** – e-mailová adresa studenta
- **os_cislo** – osobní číslo pod kterým je student veden na univerzitě

PŘEDMĚT – tabulka s informacemi o předmětu

- **id** – primární klíč předmětu, číslování 1 až N
- **nazev –** celý název předmětu
- **zkratka –** zkrácený název ve tvaru katedra/předmět
- **rok –** rok předmětu určený k možnosti archivace

STUDUJE – vazební tabulka pro STUDENT a PREDMET

- **id_student** – primární cizí klíč z tabulky STUDENT
- **id_predmet –** primární cizí klíč z tabulky PŘEDMĚT

Semestrální práce (SEMESTRALKA) – semestrální práce studentů z různých předmětů

- **id_student** – primární cizí klíč z tabulky STUDENT
- **id_predmet –** primární cizí klíč z tabulky PŘEDMĚT
- **nazev –** název semestrální práce
- **popis –** text popisující semestrální práci, případně její stav zpracování

DOCHÁZKA – tabulka shromažďující informace o docházce studenta v jeho předmětech

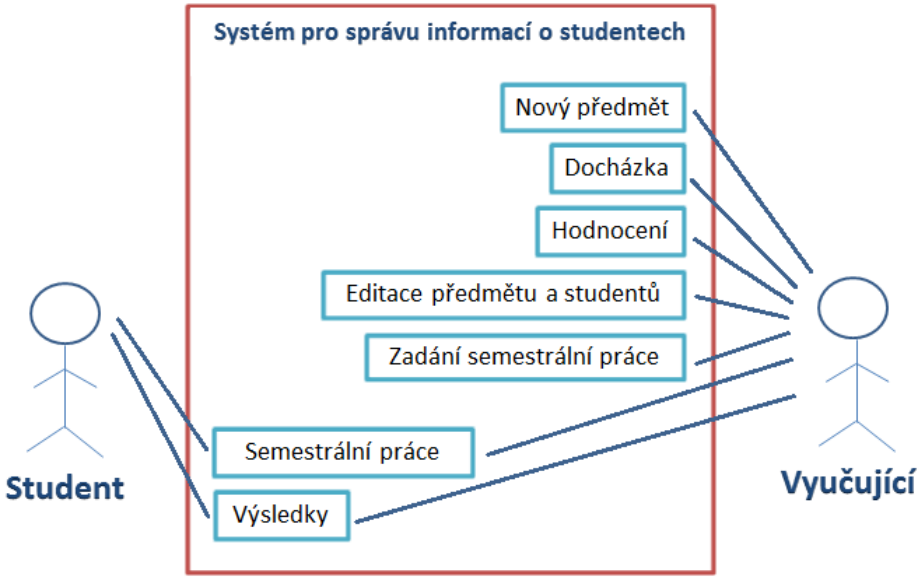
- **id_student** – primární cizí klíč z tabulky STUDENT
- **id_predmet –** primární cizí klíč z tabulky PŘEDMĚT
- **hodiny –** uložené záznamy přítomen/nepřítomen typu Boolean v poli

BODY – bodové hodnocení studenta na cvičeních

- **id_student** – primární cizí klíč z tabulky STUDENT
- **id_predmet –** primární cizí klíč z tabulky PŘEDMĚT
- **pocet –** body získané na cvičení
- **typ –** druh aktivity, za které jsou body uděleny
- **poznamka –** případná poznámka o aktivitě studenta na cvičení

6.3.2 Diagram případů užití

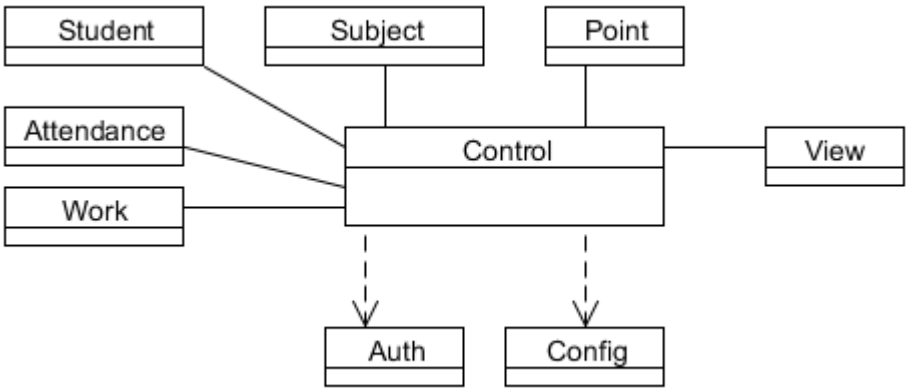
Diagram užití (angl. Use case diagram) graficky znázorňuje pohled uživatelů na systém jako celek s možnostmi (funkcemi) jeho použití.



Obrázek 4 – Diagram případů užití

Vyučující je uživatelská zabezpečená role typu administrátor - má přístup do všech sekcí aplikace. Druhý typ uživatele je student, který má práva běžného uživatele a v tomto případě může pouze prohlížet zpřístupněné seznamy.

6.3.3 Model aplikace



Obrázek 5 – Model aplikace

Tento model (viz Obrázek 4 – Model aplikace) znázorňuje statickou strukturu systému. Podrobněji viz Příloha – Model aplikace.

Třídy Student, Předmět (Subject), Docházka (Attendance), Body (Point), Semestrální práce (Work) jsou datové a spolupracují s databází. To znamená, Že provádějí veškeré úpravy v tabulkách uvedených v datovém modelu (viz 6.3.1 Datový model). Control je hlavní řídící třída, která vyřizuje požadavky od uživatele, z datových tříd získává informace a pomocí View je prezentuje zpět uživateli. View tedy slouží ke generování grafického výstupu, v tomto případě HTML.

Všechny datové třídy obsahují tzv. rozhraní pro komunikaci. Jedná se o kostru třídy, která je dále implementována v dané třídě. Zároveň zpřehledňuje seznam funkcí včetně parametrů.

Pomocná třída Auth je určená k ověření uživatele a umožnění přístupu do zabezpečených sekcí. Veškeré údaje, které je potřeba nastavit před spuštěním aplikace v daném prostředí, obsahuje Config. Jedná se o přístupové údaje k informačnímu systému STAG a dále nastavení pro připojení k lokální databázi.

Dalšími soubory potřebnými v aplikaci jsou connect k připojení na databázi definované v Config a kaskádový styl CSS související s částí View.

7 Realizace

V danou chvíli je systém navržen a je možné se pustit do samotné implementace. Jako vývojové prostředí jsem použil NetBeans IDE 7.0.1 a pro průběžné testy během implementace zejména prohlížeče Google Chrome a Opera.

7.1 Načtení informací

Nejdůležitějším prvkem aplikace jsou data, se kterými se bude dále pracovat. Pro účely této aplikace jsou potřeba údaje o studentech a předmětech, které poskytují webové služby IS/STAG (viz kapitola 5. Webové služby IS/STAG). Každá služba má svoji unikátní adresu, proto je pro získání dat zapotřebí sestavit URL požadavek.

Během práce došlo ke změně v poskytovaných údajích z důvodu ochrany osobních údajů a je tedy nutné rozdělit požadavky do dvou kategorií:

- **Služby s přihlášením** – obsahující jméno, příjmení a osobní rozvrh studenta. Podléhají ochraně osobních údajů a je tedy nutné se při volání těchto služeb přihlásit orion nebo stag loginem a heslem[19].
- **Služby bez přihlášení** – ostatní služby bez těchto osobních údajů. Volání probíhá stejným způsobem jako v ukázce webové služby (kapitola 5.2 Příklad webové služby).

Při ručním prohlížení služeb zadá uživatel přihlašovací údaje do formuláře, který se mu zobrazí pří zavolání požadavku (otevření URL služby). V rámci aplikace je nutné tyto přihlašovací údaje „podstrčit“ již na začátku, tedy v URL samotné. Uvádím ukázku sestavené URL.

Potřebné údaje k sestavení:

- Orion nebo stag přihlašovací údaje (login a heslo) – definované v konfiguračním souboru aplikace „config.php“.
- Zkratka katedry a předmětu – Zadané ve formuláři při načítání předmětu
- Vyučovaný rok – aktuální akademický rok
- Tělo požadavku s názvem služby (ve funkci se nemění oproti předchozím proměnným)

https://orionlogin:orionheslo@stag-ws.zcu.cz/ws/services/rest/student/getStudentiByPredmet?katedra=KIV&zkratka=WEB&rok=2011

[19] **Orion a stag** – systémy s možností ověření identity v rámci ZČU

Odpovědí na takto vytvořený požadavek je XML soubor s informacemi o studentech. Ještě před tímto krokem dojde v aplikaci k ověření, zda zadaný předmět existuje pomocí služby *getPredmetInfo*. Pokud vše proběhne v pořádku, přichází na řadu parsování XML a uložení dat do lokální databáze.

Pro práci s XML souborem jsem zvolil Document Object Model (DOM). Pomocí DOM se načte celý XML soubor do stromové struktury, kde se dá na celou strukturu dívat jako na typy objektů v pořadí Dokument → Uzel → Atribut → Prvek. Celá technologie načítání závisí na zachování názvů jednotlivých tagů[20] jako je „<studentPredmetu>“ nebo „<userName>“. V případě změny těchto identifikátorů v poskytovaných službách by bylo nutné změnit datové třídy „Student“ a „Predmet“, u kterých se provádí načítání z IS/STAG.

Vybraná data (viz 6.3.1 Datový model) z XML se následně ukládají do lokální databáze pomocí příkazů SQL jazyka.

7.2 Změna dat studentů

Změny informací o studentech je možné upravovat zvlášť v detailní kartě studenta, kde je na výběr změna emailu, semestrální práce s komentářem podle předmětu, bodů za aktivity a docházky v předmětech. Stejné tak je možné upravit tyto informace hromadně u vybraného předmětu.

7.2.1 Docházka

Vedení docházky spravuje datová třída „Dochazka“, která komunikuje se stejnojmennou tabulkou v databázi. Identifikace konkrétní docházky probíhá na základě ID[21] předmětu a ID studenta. Samotná docházka je uložena jako pole nabývající hodnot typu Boolean[22], kde true znamená přítomen a false nepřítomen.

7.2.2 Body

Stejně jako docházka patří body studentovi v nějakém předmětu a pomocí ID předmětu i ID studenta se tyto záznamy propojí. Oproti návrhu jsou rozšířeny o atribut

[20] **Tag** – vlastnost prvku uzavřená do hranatých závorek
[21] **ID** – identifikační číslo uvedené v databázi
[22] **Boolean** – datový typ nabývající hodnot true/false

„typ“, aby bylo možné rozlišovat, za co student body získal a nezobrazovat pouze jejich součet.

7.2.3 Semestrální práce

Závislosti platí stejné jako v předchozích případech. Taktéž existuje samostatná datová třída pro semestrální práci. Každá „semestrálka“ má svůj název a komentář. Zadávání nových prací nebo změny v již existujících jsou umožněny v detailní kartě studenta.

7.3 Rozdělení práv přístupu

Existují dvě role uživatelů (viz 6.3 Diagram případů užití) – přihlášený uživatel (vyučující), který má přístup do všech sekcí k prohlížení i úpravám a nepřihlášený uživatel (student) s přístupem pouze do sekcí s výsledky. V těchto seznamech nejsou z důvodu ochrany osobních údajů uvedeny jména studentů, ale zobrazují se pouze jejich univerzitní čísla, podle kterých není možné bez přihlášení zjistit jakékoliv informace o studentovi.

O autorizaci se stará třída „Auth“. Ověřování uživatele (vyučujícího) probíhá na základě nastavených údajů při konfiguraci pomocí souboru „config.php“. Obsah tohoto souboru je detailně popsán v kapitole 6.3.2 Model aplikace a jeho nastavení s příkladem v kapitole 12.1 Uživatelská příručka.

8 Testování

Průběžné testování správné funkčnosti jsem prováděl během vývoje vždy po implementaci části funkcionalit. Šlo zejména o kontrolu správnosti načítaných údajů z IS/STAG, uložení do databáze a kontrolu úprav provedených uživatelem v aplikaci. Veškeré testy byly prováděny manuálně bez využití automatických testů.

Dalším testovaným prvkem je grafické rozhraní, které se může v některých případech zobrazovat odlišně. Z tohoto důvodu jsem použil validační nástroj od společnosti W3C a následně vizuálně zkontroloval zobrazování aplikace v nejpoužívanějších prohlížečích viz níže.

- Google Chrome 11
- Internet Explorer 9
- Mozilla Firefox 12
- Opera 11

Jediným problémem bylo odlišné odsazení prvků, ve kterém se lišil Internet Explorer proti ostatním. Tento problém byl odstraněn a aplikace se tak zobrazuje ve všech testovaných případech stejně.

8.1 Validace

K validaci HTML (resp. XHMTL) výstupu jsem použil již zmíněný validátor od W3C [6]. Po úpravě drobností, jako jsou nesprávně použité atributy nebo chybně umístěné tagy, stránka odpovídá normě XHTML 1.0 Strict a je tedy validní.

W3C nabízí také nástroj k validování kaskádového stylu (CSS). Využil jsem i této možnosti a zkontroloval použitý soubor CSS. Dokument odpovídá CSS level 3. Více o těchto normách na www.w3.org/TR/CSS/#css-levels.

8.2 Zátěžový test

U této aplikace se nepředpokládá, že by k ní zároveň přistupovalo více subjektů a byl kladen důraz na výkon při načítání dat z IS/STAG nebo lokální databáze. Přesto jsem provedl dle zadání testy zaměřené na požadavky směrem k IS/STAG, tedy načítání informací o studentech a předmětech, a jejich uložení do databáze. Při těchto prováděných akcích jsem měřil dobu, za kterou se požadavek vykoná a pro ilustraci zanesl do grafů.

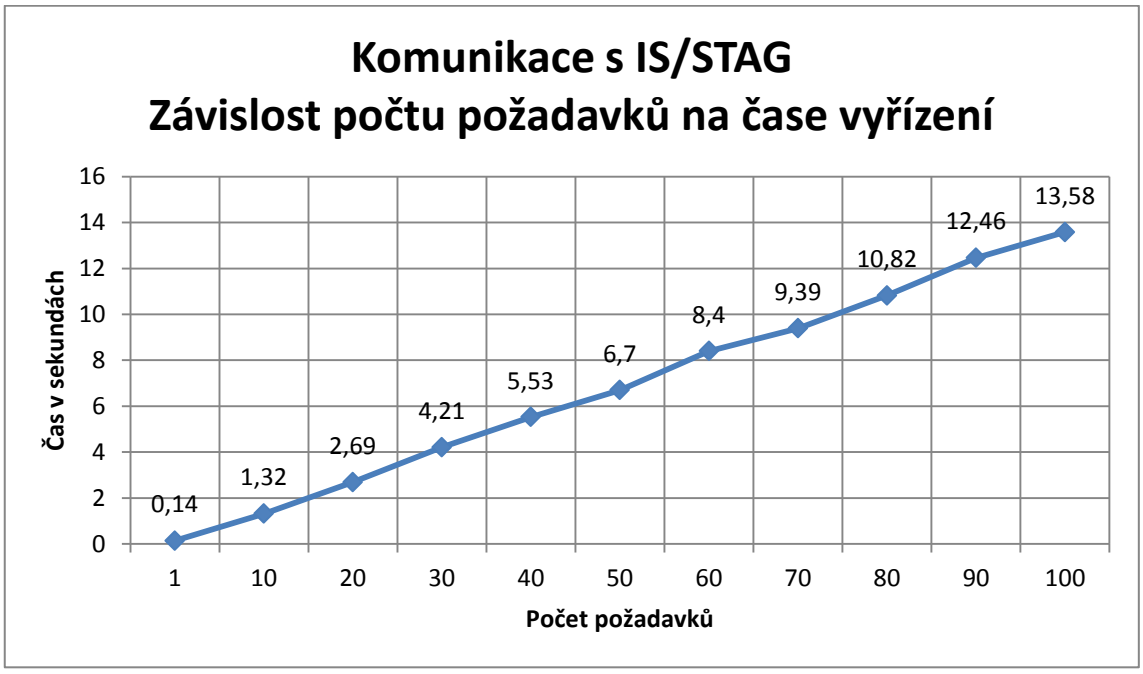
Knihovna PHP poskytuje funkci „microtime()“ [9]. Výstupem této funkce je čas v sekundách, včetně části s mikrosekundami, od data 1. 1. 1970. Uložením dvou bodů, před vykonáním akce a po dokončení, lze měřit dobu, za kterou se akce vykoná.

Ke kontrole naměřených hodnot jsem souběžně spustil vývojářské nástroje aplikace Google Chrome a porovnal oba údaje, aby se dal výsledek označit za správný.

8.2.1 Komunikace s IS/STAG

Pro tento test jsem zvolil jednoduchou funkci na získání informací o předmětu. V měřeném úseku aplikace požádá IS/STAG o data pomocí webové služby a celý výsledek uloží do proměnné ke zpracování. K samotnému zpracování už během měření nedochází.

Naměřené hodnoty se pohybují kolem 140ms. Tato doba je tedy potřeba ke stažení informací z IS/STAG a jejich uložení do proměnné k dalšímu použití. Se vzrůstajícím počtem požadavku roste i jejich doba na vyřízení. Výsledky měření (viz Graf 1 – Komunikace s IS/STAG) ukazují lineární závislost času na počtu požadavků.

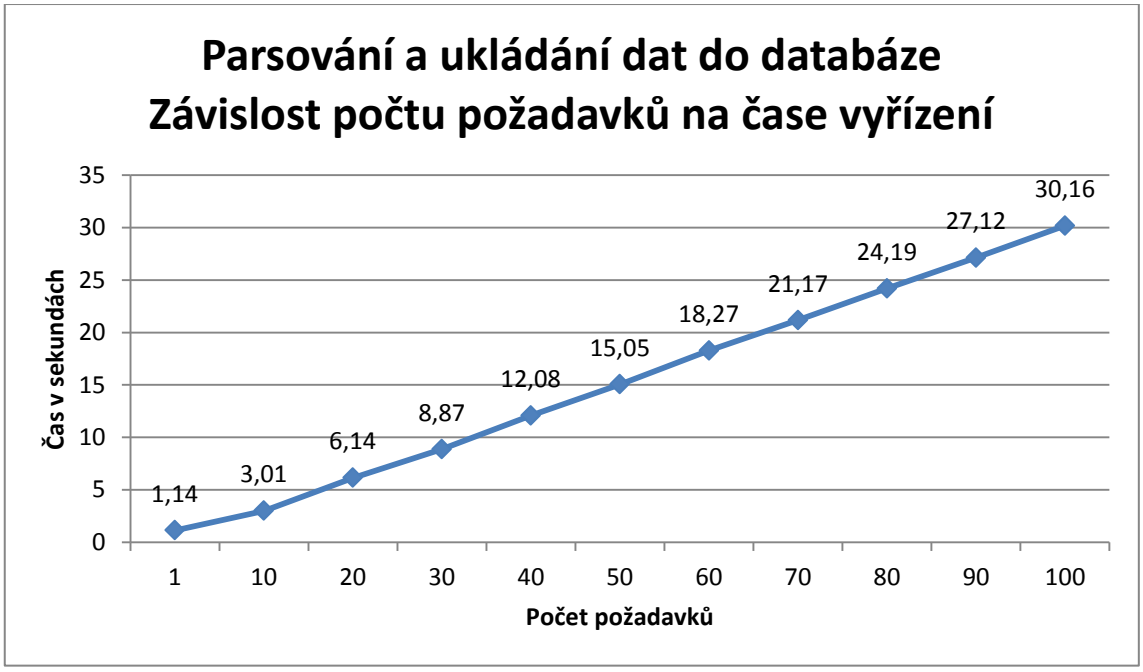


Graf 1 – Komunikace s IS/STAG

8.2.2 Uložení dat do databáze

Tento krok následuje po načtení dat z IS/STAG. Zde jsem zvolil nejnáročnější funkci k ukládání všech studentů předmětu. Měření probíhá po dobu načítání (parsování) dat ze staženého XML dokumentu a vkládání těchto dat do databáze.

Při prvním načtení předmětu se vkládá většina studentů. Při dalších volání této funkce se pouze aktualizují údaje v databázi, pokud je to potřeba. V porovnání prvního načtení a druhého při aktualizaci je rozdíl v načtení téměř 0,8 sekundy. První údaj (viz Graf 2 – Uložení dat do databáze) je vyšší z důvodu ukládání nových záznamů a ostatní měřené hodnoty jsou pouze aktualizacemi databáze.

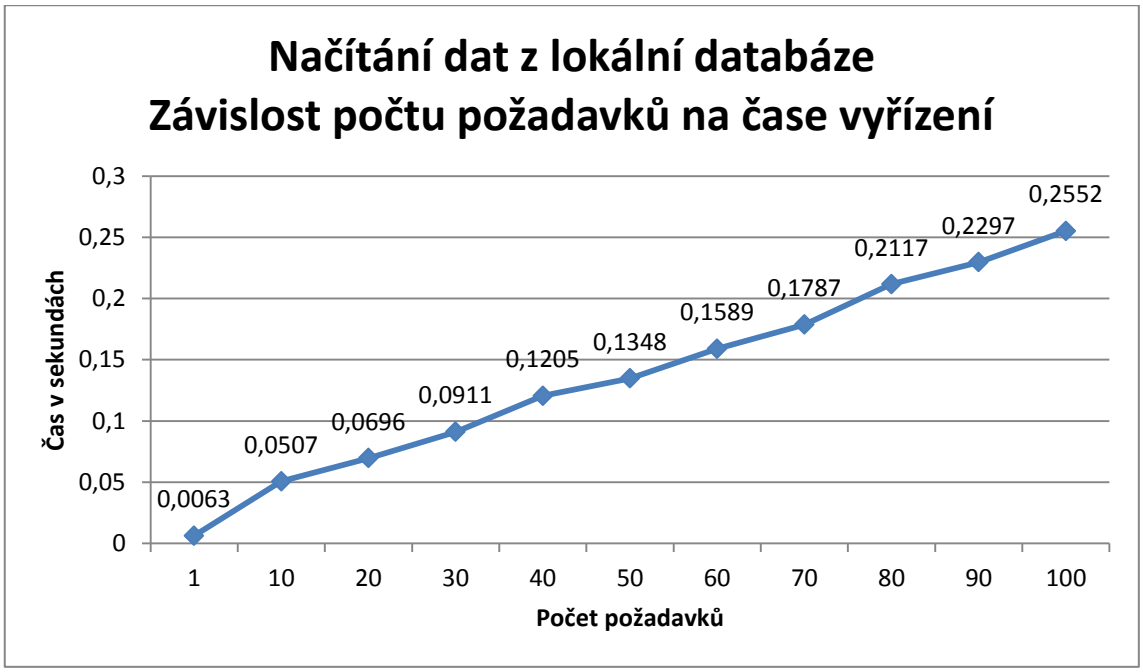


Graf 2 – Parsování a ukládání dat do databáze

8.2.3 Zobrazení načtených dat

Téměř na každé stránce se načítají data z lokální databáze. Proto jsem jako další testovanou část zvolil právě toto načítání. Jako testovací funkci jsem vybral načítání studentů vybraného předmětu. Měřeným úsekem je pouze část načtení požadovaných dat.

Naměřené hodnoty pro jeden požadavek jsou v řádech desítek milisekund, což je pro načítání stránky přijatelné. V grafu (Graf 3 – Načítání dat z lokální databáze) je vidět, že čas roste úměrně s počtem požadavků.



Graf 3 – Načítání dat z lokální databáze

8.3 Shrnutí testů

Byly provedeny testy funkčnosti logické části aplikace a validace s vizuální kontrolou grafického rozhraní. U obou těchto části byly nalezeny drobné chyby a následně opraveny. Testování požadavků je zaměřené na rychlost zpracování, tedy dobu, za kterou je požadavek vyřízen. Tato doba roste s přibývajícím počtem požadavků dle očekávání. Její průběh v závislosti na počtu požadavků je ve všech případech lineární a nejvíce záleží na okolních systémech, za jakou dobu dokáží požadavek zpracovat. Jedná se tedy o systémy poskytující data (IS/STAG) a ukládání/načítání z/do databáze (MySQL server).

9 Závěr

V úvodní fázi bylo potřeba se seznámit s IS/STAG a webovými službami [6], které poskytuje. Pomocí těchto služeb se následně načítají data (informace) o studentech a předmětech. Jedná se tedy o zdroj dat k aplikaci, a proto bylo nezbytné se naučit sestavovat požadavky v akceptovatelném formátu. Vzhledem k obsáhlé dokumentaci včetně příkladů jsem neměl s touto částí problém.

Dále jsem hledal využitelné webové technologie pro tuto aplikaci. V první teoretické části jsou uvedeny směry, kterými se lze vydat při tvorbě podobného systému s porovnáním technologií. Později bylo možné současně s výběrem technologií začít navrhovat systém. První návrh se týkal datové části, tedy datového modelu. Tento model, až na drobné úpravy (přidání) atributů, zůstal v původní navrhované podobě. Co se týče modelu aplikace, tak ten bylo potřeba několikrát upravovat.

Po úspěšném návrhu všech součástí následovala implementace. Zde jsem chtěl využít framework, který ovšem není moc známý a nemá žádnou dokumentaci. Z počátku se zdálo jeho využití s architekturou MVC jako dobrý nápad. Později jsem ale narazil na problémy, které nebylo možné bez dokumentace řešit a musel jsem tak zvolit cestu vlastního kódu bez využití podpůrných frameworků.

Posledním krokem bylo otestování funkčnosti výsledného produktu. Systém pracuje dle požadavků zadavatele a je využitelný jako elektronická evidence studentů na předmětech s okamžitým zveřejněním výsledků studentům.

Přehled zkratek

Použité zkratky jsou uvedeny vždy v poznámce pod čarou na stránce, kde je zkratka použita. Pro přehled je zde seznam všech použitých zkratek se stručným komentářem. Zkratky jsou uvedeny v pořadí, v jakém se vyskytují v textu práce.

Zkratka	Popis zkratky
PHP	Hypertext Preprocesor – programovací jazyk určený zejména pro webové aplikace
SŘBD	Systém řízení báze dat – databáze
GPL	General Public Licence – všeobecná veřejná licence
BSD	Berkeley Software Distribution – licence pro svobodný software
MS	Microsoft
CMS	Content Management Systém – systém pro správu obsahu
WCMS	WebCMS – webový typ systému pro správu obsahu
DB	databáze
ZF	Zend Framework
PDF	Portable Document Format – souborový formát od firmy Adobe
IS/STAG	Informační systém studijní agendy
XML	Extended Markup Language – standardizovaný značkovací jazyk pro výměnu informací
XLS	tabulkový formát aplikace Microsoft Excel
CSV	Comma separated value – jednodušší tabulkový formát
ICS	formát pro data z kalendáře
URL	Uniform resource locator – řetězec ke specifikaci umístění zdrojů
HTTP	HyperText Transfer Protocol – internetový přenosový protokol
HTTPS	HTTP Secure – zabezpečená verze přenosového protokolu
CSS	Cascading Style Sheet – kaskádový styl – jazyk pro formátování prvků v HTML
DOM	Document Object Model – objektový model dokumentu
ID	Identifikace objektu
SQL	Structured Query Language – strukturovaný dotazovací jazyk pro práci s databází
HTML	HyperText Markup Language – značkovací jazyk

Použitá literatura

1. **Oracle.** Oracle|Hardware and Software. *Oracle.* [Online] 2012. [Citace: 2. Květen 2012.] http://www.oracle.com/us/products/mysql/.

2. **Schröder, Carla.** PostgreSQL vs MySQL: Which Is the Best Open Source Database? [Online] 8. červenec 2011. http://olex.openlogic.com/wazi/2011/postgresql-vs-mysql-which-is-the-best-open-source-database/.

3. **Redakční systémy.** *Redakční systémy CMS.* [Online] 22. Červen 2009. [Citace: 10. Květen 2012.] http://www.redakcni-systemy.com.

4. **WordPress - česká podpora.** redakční systém s českou podporou. [Online] http://www.cwordpress.cz/.

5. **Drupal.** O systému Drupal. [Online] [Citace: 11. Květen 2012.] http://www.drupal.cz/o-systemu-drupal.

6. **JoomlaPortal.** Co je Joomla!? *Joomla Portal - česká komunita.* [Online] 25. Únor 2012. [Citace: 11. Květen 2012.] http://www.joomlaportal.cz.

7. **Západočeská univerzita v Plzni.** Webové služby nad IS/STAG. [Online] https://stag-ws.zcu.cz/ws/help.

8. **tvorba-webu.cz.** DOM: Document Object Model. [Online] http://www.tvorba-webu.cz/dom/.

9. **W3C.** Markup Validation Service. [Online] http://validator.w3.org/.

10. **PHP Manual.** [Online] The PHP Documentation Group, 1997-2012. http://docs.php.net/manual/en/index.php.

11. **PostgreSQL.** [Online] PostgreSQL Global Development Group. http://postgres.cz/wiki/PostgreSQL.

12. **Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Timothy Boronczyk.** *PHP6, Apache, MySQL vytváříme webové aplikace.* Brno : Computer Press, 2009. ISBN 978-80-251-2767-4 .

Přílohy

A. Uživatelská dokumentace

Uživatelská příručka stručně popisuje možnosti a ovládání aplikace z pohledu uživatele (vyučujícího). Úvodem je důležité zmínit, že veškeré funkce pro správu studentů a předmětů jsou přístupné pouze po přihlášení. Bez přihlášení je možné prohlížet výsledky studentů a zadané semestrální práce.

V pravém horním rohu je k dispozici přihlášení. Po zadání uživatelského jména a hesla se zpřístupní všechny sekce.

Načtení nového předmětu

V menu úprav je možnost „Nový předmět“. Zadáním zkratky příslušného předmětu (např. „WEB“) a výběrem možností, které chcete u předmětu evidovat, dojde k načtení předmětu a jeho studentů. V kategoriích „Studenti“ a „Předměty“ je pak možné prohlížet informace studentů. U každé položky jsou na výběr možnosti, co lze v daný okamžik provést (viz Správa údajů).

Prohlížení kategorií

Kategorie jsou rozděleny na studenty, předměty, semestrální práce a body. Zároveň jsou přístupem do sekcí, kde je možné upravit údaje. Existuje tedy více možností jak se dostat ke změně údajů studenta či předmětu. Například je možné prohlížet seznam všech studentů a přes detaily studenta se dostat k předmětu nebo naopak výběrem předmětu se dostat až ke změnám detailů studenta.

Úpravy (správa) údajů

Možnosti úprav jsou vždy uvedeny u každé položky v seznamu. Jedná se o zobrazení detailů, úpravy či smazání. Tyto úpravy se týkají jak studentů, tak předmětů. Veškeré údaje týkající se studenta jsou přístupné a upravitelné v jeho detailní kartě (Studenti dle předmětu → Možnosti u studenta → Upravit).

Další možností jsou hromadné změny (Studenti dle předmětu → Hromadná docházka/změna bodů) u všech studentů vybraného předmětu najednou.

Odstranění a archivace záznamů

Smazat je možné předmět i studenta. Tato možnost je dostupná u zobrazení všech studentů předmětu a seznamu předmětů. Ačkoliv je smazání předmětu dostupné, předpokládá se spíše využití archivace předmětu se zachováním všech údajů.

Upozornění: Odstraněním předmětu dojde k jeho úplnému smazání včetně všech jeho souvisejících údajů, jako jsou data studentů vázané k tomuto předmětu (docházka, body, semestrální práce).

B. Model aplikace