VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
CENTRUM VÝPOČETNÍCH A INFORMAČNÍCH SLUŽEB

FACULTY OF INFORMATION TECHNOLOGY
COMPUTER AND INFORMATION SERVICES CENTRE

GRAFICKÝ PROHLÍŽEČ MAPY STUDIJNÍCH PROGRAMŮ A STUDIÍ NA VUT

STUDY PROGRAMME MAP GRAPHICAL VIEWER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — MARTIN BRETTSCHNEIDER
AUTHOR

VEDOUCÍ PRÁCE — ING. JAROMÍR MARUŠINEC, PH.D., MBA
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Centrum výpočetních a informačních služeb Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Brettschneider Martin**
Obor: Výpočetní technika a informatika
Téma: **Grafický prohlížeč mapy studijních programů a studií na VUT**
Kategorie: Databáze

Pokyny:

1. Prostudujte analýzu a ER diagramy datového schématu ST01 celoškolského studijního systému.
2. Navrhněte grafický prohlížeč mapy návaznosti studijních programů, oborů a zaměření a grafický prohlížeč studií studenta. Každý program, obor nebo zaměření bude reprezentován blokem. Spojující čáry budou představovat možnosti postupu s uvedením podmínek. Časová osa zleva doprava bude označovat semestry.
3. Realizujte grafický prohlížeč v prostředí webového portálu VUT.
4. Funkcionalitu ověřte vytvořením grafické podoby mapy studijních programů Fakulty strojního inženýrství.
5. Navrhněte rozvoj aplikace zejména o možnost editace mapy na webovém portálu pro studenty s uvedením současného stavu a dalších možností studia.
6. Zhodnoťte přínos Vaší práce.

Literatura:

- Podle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Marušinec Jaromír, Ing., Ph.D.**, CVIS VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 22. května 2007

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

L.S.

prof. Ing. Tomáš Hruška, CSc.
děkan FIT

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací dvou webových aplikací (modulů) pro portál VUT. První z nich je mapa studií studenta, která v přehledné grafické formě zobrazuje všechna dosavadní studia jednotlivých studentů. Druhá aplikace je mapa studijních programů, která v grafické podobě zobrazuje návaznosti jednotlivých programů a oborů na zvolené fakultě VUT.

Klíčová slova

Informační systém, Webová aplikace, UML, HTML, PHP, SQL, Databáze, JpGraph, Graphviz

Abstract

This thesis deals with design and implementation of two web applications for BUT portal. The first is called the map of studies. It shows for student all his studies in well-arranged graphical form. The second application is the map study programme graphical viewer. It shows in graphical form the relationships between study programs and departments for each BUT faculty.

Keywords

Information system, Web application, UML, HTML, PHP, SQL, Database, JpGraph, Graphviz

Grafický prohlížeč mapy studijních programů a studií na VUT

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaromíra Marušince, Ph.D., MBA.
Další informace mi poskytli Ing. Rudolf Musil, Ing. Michal Jurosz a Ing. Marek Strakoš.
Všem jmenovaným tímto děkuji.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Brettschneider
19.4.07

Obsah

Úvod

Internet je fenoménem současnosti. Asi nejpoužívanější službou Internetu je WWW (World Wide Web). Slouží mimo jiné k prohlížení tzv. www stránek, využívajících grafiku, zvuky a často i animace a video. Na těchto www stránkách se dá nalézt množství informací. Ale co to vlastně je informace? Budu citovat z České terminologické databáze knihovnictví a informační vědy.

V nejobecnějším slova smyslu se informací chápe údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Informace snižuje nebo odstraňuje neurčitost systému (např. příjemce informace); množství informace je dáno rozdílem mezi stavem neurčitosti systému (entropie), kterou měl systém před přijetím informace a stavem neurčitosti, která se přijetím informace odstranila. V tomto smyslu může být informace považována jak za vlastnost organizované hmoty vyjadřující její hloubkovou strukturu (varietu), tak za produkt poznání fixovaný ve znakové podobě v informačních nosičích. V informační vědě a knihovnictví se informací rozumí především sdělení, komunikovatelný poznatek, který má význam pro příjemce nebo údaj usnadňující volbu mezi alternativními rozhodovacími možnostmi. Významné pro informační vědu je také pojetí informace jako psychofyziologického jevu a procesu, tedy jako součásti lidského vědomí (např. N. Wiener definuje informaci jako "obsah toho, co se vymění s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním"). V exaktní vědě se např. za informaci považuje sdělení, které vyhovuje přísným kriteriím logiky či příslušné vědy. V ekonomické vědě se informací rozumí sdělení, jehož výsledkem může být zisk nebo užitek. V oblasti výpočetní techniky se za informaci považuje kvantitativní vyjádření obsahu zprávy. Za jednotku informace se ve výpočetní technice považuje rozhodnutí mezi dvěma alternativami (0, 1) a vyjadřuje se jednotkou nazvanou bit. (JONÁK)

Pod pojmem informační systém lze chápat zdroj informací. Webový portál VUT, jehož rozšíření o grafický prohlížeč mapy studijních programů a studií na VUT, jež je předmětem mé práce, je příkladem takového informačního systému.

V první kapitole mé diplomové práce se budu zabývat problémy návrhu a implementace informačního systému. Tedy od modelování systému, po jazyky a prostředky pro tvorbu webového informačního systému, jako je HTML, PHP či SQL.

V kapitole druhé se zabývám návrhem prvního vytvořeného modulu, tj. grafického prohlížeče mapy studií.

Ve třetí kapitole je popsán princip implementace tohoto modulu, včetně problémů, které při implementaci vznikly a jejich řešení. V této kapitole je také popsán grafický nástroj JpGraph, který jsem s výhodou využil při implementaci.

Kapitola čtvrtá je zaměřena na návrh druhého modulu, kterým je mapa studijních programů na fakultách VUT.

V předposlední kapitole, je popsán postup implementace mapy studijních programů a dále problémy s ní spojené. Jedna podkapitola je věnována také grafickému nástroji Graphviz a možnostem rozvoje aplikace.

Závěrečná kapitola je celkovým shrnutím práce.

1 Informační systém

1.1 Úvodem

Vzhledem k tomu, že zadání mé diplomové práce se týká rozšíření stávajícího webového informačního systému VUT, tak v této kapitole popíši hlavní metody a programovací jazyky, které se při implementaci informačního systému používají.

Informačním systémem (zkráceně IS) rozumíme počítačový firemní systém pro sběr, zpracovávání a prezentací informací a dat. Informacemi míníme sdělení, které odstraňuje nejistotu nebo nevědomost, daty míníme jakékoli zaznamenané poznatky či fakta. Informaci lze také chápat jako data s nějakým přidaným významem (data + význam).

1.1.1 Vlastnosti informačního systému

Pod pojmem „vlastnosti informačního systému“ si můžeme představit faktor jeho účinnosti, jelikož jejich prostřednictvím působí systém na své okolí. Pokud systém dodává informace pozdě, je těžkopádný (pomalý), ovládání je složité (sám systém tvoří bariéru styku s uživatelem), je obtížně modifikovatelný, nefunguje spolehlivě, příliš nebere v úvahu ochranu informace před jejím zneužitím, pak je v podstatě nepoužitelným produktem. Vlastnosti IS nejsou navzájem nezávislé, proto byly užitečné snahy o jejich kategorizaci, o vymezení vzájemných souvislostí a jednoznačného popisu.

Vlastnosti informačního systému roztřídila například paní Kotteová do šesti skupin. Je to schopnost uspokojit informační potřeby uživatele, tzn. dodat informace, které uživatel žádá a schopnost pružně se přizpůsobit změnám v informační potřebě uživatele.

Dalšími kritérii jsou aktuálnost poskytovaných informací, tj. poskytovaná informace má vyjádřit nejaktuálnější stav v řízení řízeného objektu.

Včasnost poskytovaných informací má zajistit přístupnost informace vždy v čase potřebném pro řízení.

Dále pak orientace informace pro potřebu řízení, tj. relevantnost poskytovaných informací, kvantování informace v souladu s její potřebou v řízení, obsahovou orientaci informace pro algoritmy řízení, dialogový-interakční způsob přebírání informace uživatelem IS a jejich uplatnění v řízení.

Spolehlivostí IS se rozumí celková funkční spolehlivost (lidí, technických prostředků, programového vybavení, ...), spolehlivost spočívající v rychlosti a jednoznačnosti poskytované informace, v bezpečnosti informace před poškozením a zničením.

A nakonec také uživatelská výhodnost, kdy IS pomáhá uživateli naplnit jeho strategické cíle, je tedy ekonomicky výhodný v prioritních směrech aktivit uživatele.

Existuje více způsobů, jak roztřídit vlastnosti systému. Třídit lze také některé význačné části systému, jako je např. programové vybavení. To rozdělili pánové Ross, Goodenouch a Irvine do čtyřech kategorií: modifikovatelnost, efektivnost, spolehlivost a srozumitelnost.

1.1.2 Cíle informačního systému

Cíle informačního systému se objevují v zadavatelem vytvořeném zadání, později také v popisu stávajícího systému pro zpracování informace a jsou určeny především tvůrcům informačních systémů. V popisu cílů jde o verbální integraci samotného poslání IS a zadavatel je formuluje z pozic organizace, která chce zavedením IS docílit ekonomicko-finanční progres.

Význam jednotlivých vlastností informačních systémů není stejný. To je logické, neboť i významy cílů systému, z nichž jsou vlastnosti odvozeny, nejsou stejné. Stačí si uvědomit jaký má význam spolehlivost v systému typu „IS lékařské záchranné služby“. Spolehlivost má tedy klíčové postavení mezi všemi vlastnostmi. Její zabezpečení je kardinální otázkou při tvorbě IS, což je těsně spojeno i s tím, jakými postupy spolehlivost prověřit.

1.2 Návrh IS

1.2.1 Požadavky a jejich specifikace

Správný návrh je důležitou částí při tvorbě informačního systému. Základem při návrhu systému je definice požadavků, tj. zadání v přirozeném jazyce, příp. diagramy, udávající požadované služby systému a jeho omezení. Návrh je zpravidla vytvořen na základě informace od zákazníka.

Z definice požadavků se dále vytvoří strukturovaný dokument specifikace požadavků. Tento detailně popisuje služby a omezení systému. Může mimo jiné sloužit pro uzavření kontraktu se zákazníkem.

Dokument požadavků na programové dílo je oficiální dokument o tom, co se od vývojářů požaduje. Je kombinací definice a specifikace požadavků. Dokument by měl být dobře strukturovaný s minimem vzájemných odkazů. Hlavní částí dokumentu jsou modely systému, ukazující vztahy prvků systému a vztahy systému a okolí, definice funkčních požadavků, popisující poskytované služby a definice nefunkčních požadavků, což jsou omezení kladená na produkt a proces vývoje (výkonnost, odezva, standardy, …). Dalšími částmi dokumentu mohou být rozvoj systému, validační kritéria (třídy testů pro ověření implementace) a další.

1.2.2 Modely systému

Pod pojmem model si můžeme představit abstrakci něčeho (systému, jeho části) pro účely pochopení, experimentování apod. před jeho vlastním vytvořením. Použití modelu může být např. náhrada fyzického zařízení pro experimentování (simulace), komunikace se zákazníkem, vizualizace, redukce složitosti (abstrakce) či dokumentace.

Cílem modelování je tedy i vizualizace (jaký systém je nebo jaký má být), specifikace struktury a chování systému, vytvoření „šablony“ pro konstrukci systému a také dokumentace provedených rozhodnutí. Pro modelování se nejčastěji používají objektově orientované techniky modelování, které představují reprezentaci určitého pohledu na systém.

Typů modelů je celá řada. Mezi hlavní patří model objektové struktury, který zpravidla zahrnuje třídy domény problému a třídy pro implementaci, atributy a operace tříd, statické vztahy mezi třídami a objekty. Model objektové struktury je základním modelem, který je používán ve fázi analýzy i návrhu.

Důvodem pro použití dalších modelů je zjišťování a pochopení požadavků na systém, včetně dynamického chování, dokumentace v podobě srozumitelné zákazníkovi, dostatečně podrobná informace pro návrháře a programátora a nakonec také pro zvládnutí vývoje rozsáhlých systémů (rozdělení na menší jednotky).

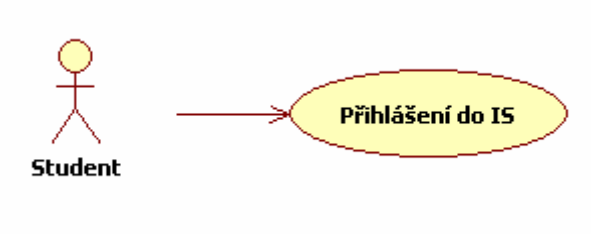
Mezi další modely systému patří modely chování systému, což jsou modely požadované funkčnosti, použití systému, toků aktivit a stavů systému a přechodu mezi stavy. Dále jsou to modely interakce objektů, ve kterých se modelují interakce systému s okolím, ale i interakce vnitřních objektů. Nakonec také existují modely stavů objektů a přechodů mezi nimi a modely rozčlenění systému na podsystémy (pro složitější systémy).

1.2.3 Jazyk UML

Mezi nejpoužívanější metody objektově orientovaného modelování patří jazyk UML. Unified Modeling Language je jednotný grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů.

V UML se nejčastěji modeluje diagram použití a diagram tříd, samozřejmě to není jediné, co by se dalo v jazyku UML modelovat, ale je to základní a nejdůležitější.

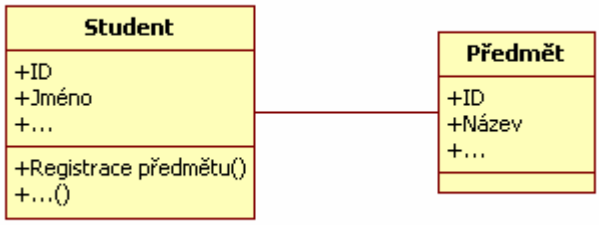
Diagram použití (use case diagram) je diagram, který ukazuje množinu případů použití, aktérů a jejich významy. Mezi prvky tohoto diagramu patří aktér, což je role objektu vně systému, která s ní přímo komunikuje a případ použití.



Obrázek 1.1: Příklad diagramu použití

V diagramu použití je nutné rozlišit aktéra (role uživatele) a uživatele (objekt komunikující se systémem). Jeden uživatel totiž může vystupovat ve více rolích. Seznam aktérů pak může být východiskem pro určení případů použití., tj. posloupnosti kroků popisujících interakci mezi uživatelem a systémem.

Diagram tříd popisuje typy objektů a statické vztahy mezi nimi. Při popisu tříd (objektů) se používají atributy, což je vlastnost, jež třída nese a operace, tj. procesy, které umí třída provádět. Vztahy mezi třídami lze reprezentovat několika způsoby. Statické vztahy mezi instancemi tříd se reprezentují jako asociace. Třídy s řadou společných vlastností a několika odlišnými (zobecňující třída) se reprezentují pomocí generalizace atd..



Obrázek 1.2: Příklad diagramu tříd

Diagram tříd je základem většiny objektově orientovaných metod programování. Pokročilými technikami modelování diagramů tříd se v této práci zabývat nebudu.

1.3 Databáze

Pod pojmem databáze rozumíme určitou uspořádanou množinu informací (dat) uložených na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento software se v české odborné literatuře označuje jako SŘBD (Systém Řízení Báze Dat). Běžně se označením databáze (v závislosti na kontextu) myslí jak uložená data, tak i software (SŘBD). Mezi hlavní funkce SŘBD patří definice dat, vytváření slovníku dat, manipulace s daty, zajištění bezpečnosti a integrity dat, zotavení po chybách, souběžný

přístup (transakční zpracování) a samozřejmě zajištění co nejvyšší výkonnosti. Databáze tedy slouží v informačním systému jako skladiště informací.

Podle způsobu ukládaní dat a vazeb mezi nimi můžeme databáze rozdělit do několika skupin. Jsou to databáze síťové, hierarchické, relační, objektové a v neposlední řadě objektově relační.

1.3.1 Relační databáze

Relační databáze je databázový systém, který je založen na relačním modelu dat a relační algebře. Data jsou strukturována do normalizovaných tabulek (relací), kde hodnoty v tabulce musí být z hlediska významu atomické (skalární). Nad relacemi jsou definovány přípustné operace, jako je výběr tabulky, vložení, změna, zrušení řádku tabulky apod.. Jazykem používaným pro ovládání databáze je v dnešní době obvykle dotazovací jazyk SQL.

1.3.2 Objektové databáze

V současné době se vývoj ubírá směrem k objektovým databázím, jejichž hlavní výhoda spočívá v možnosti uchovávání širokého spektra dat – od znakových, přes obrazová a zvuková data až po video.

Zatím je možné pozorovat dva trendy v implementaci objektů v databázových systémech. Jednak vznikají nové databázové systémy založené na objektových technologiích. Jejich výhodou je logická čistota týkající se konzistentního objektového přístupu ke všem objektům vyskytujícím se v databázi. V druhém případě se tvůrci relačních databázových systémů snaží implementovat objekty a objektový přístup do svých, již existujících produktů. V tomto případě nemůže být zaručen striktně objektový přístup a k některým datům se stále používá přístup relační. Nespornou výhodou je však zpětná kompatibilita těchto systémů s velkým množstvím dříve napsaných programů a tak jejich vyspělost díky dlouhé době jejich používání.

1.4 Implementace IS

Existuje mnoho způsobů, jak implementovat informační systém. Na webovém portálu VUT, kde budu svoji práci realizovat se používá asi nejčastější viděná kombinace programovacích jazyků. Jsou to HTML, CSS, PHP a dotazovací jazyk SQL pro komunikaci s databázovým serverem.

1.4.1 HTML

1.4.1.1 Úvod

HyperText Markup Language (zkráceně HTML) je značkovací jazyk pro tvorbu dokumentů, který definuje vzhled textu (velikost nadpisů, použité fonty písma, okraje, …). Jazyk HTML byl vyvinut, a neustále se vyvíjí, za účelem publikování dokumentů na World Wide Webu.

Již z názvu je patrné, že původně byl tento jazyk určen pro zobrazování textové dokumentace. Ovšem během krátké doby byla tato funkce nedostačující a jazyk musel být rozšířen např. o různé multimediální prvky, jako jsou grafika, animace, či hudba. Tímto krokem se několikanásobně znásobila jeho hodnota coby média pro přenos informace. V dnešní době již nikoho alespoň minimálně znalého Internetu nepřekvapí, že si může prohlížet fotky z družic, či se podívat na videoklip své oblíbené hudební skupiny. Zvládnutí jazyka HTML je nutné i pro užívání složitějších skriptovacích jazyků, jako je např. PHP.

HTML je interpretovaným jazykem, což znamená, že zdrojový kód je přímo předložen prohlížeči, není tedy překládán do strojového kódu. To znamená, že žádné vnější elementy, jako jsou třeba obrázky či animace, se nenachází uvnitř HTML dokumentu, ale je na ně pouze odkazováno. Jazyk je to zároveň velmi pružný, pokud totiž prohlížeč narazí na chybu, či na příkaz který nezná, prostě ho ignoruje a přesune se na příkaz další.

Jazyk HTML je vlastně základním pilířem při tvorbě webového informačního systému, protože informace, která uživatel hledá, se mu musí zobrazit v přehledné formě, tak aby pro něj byla dobře srozumitelná.

1.4.1.2 Tagy

Jak jsem již zmínil, elementem HTML dokumentu může být téměř cokoli, od prostého textu, po obrázky, či video. Základními elementy HTML dokumentů jsou tzv. „tagy“. V našem mateřském jazyce se asi nejčastěji používá pojem „značka“. Značky reprezentují prvky, ze kterých se stránka skládá. Uvnitř těchto elementů mohou být další elementy, v nich vnořené další elementy atd.. Značky se dají rozdělit na dvě základní skupiny, mezi párové a nepárové. Párové značky jsou dvojicí značek, jako např. <html> </html>, kde první je značka otevírací a druhá je uzavírací. Oproti tomu existují značky nepárové, tyto nemají značku uzavírací, jako například nový řádek
.

Poslední verzí HTML je 4.01. Dále se HTML vyvíjí pod názvem XHTML (The Extensible HyperText Markup Language). XHTML v dosavadních verzích používá prakticky stejné značky a atributy jako HTML. Asi hlavním rozdílem mezi HTML a XHTML je, že XHTML definuje přísná, ale zároveň jednoduchá pravidla při psaní dokumentů. To neznamená, že HTML žádná pravidla

nemá, ale díky tomu, že nejsou tak přísná, může vzniknout hodně problémů při prezentaci dokumentu.

1.4.2 CSS

Každý text má svůj obsah a formu. Formou dokumentu (webové stránky) rozumíme třeba barvu a velikost písma, pozadí, zarovnání apod., prostě všechno, co nepatří do obsahu. Tomu se říká formátování.

Při vývoji HTML vznikaly různé způsoby, jak formátovat text. Dnes existují dva způsoby, jak v HTML třeba obarvit nebo ztučnit text. Starší způsob pro to používá přímo HTML tagy. Ovšem tímto způsobem nelze provést vše a HTML kód se také stává méně přehledným a hůře modifikovatelný.

Druhým způsobem je používání CSS (Cascading Style Sheets), které jsou v literatuře překládány jako kaskádové styly. Tento způsob je sice trochu komplikovanější, ale vnáší do kódu stránky přehlednost, jelikož kaskádové styly lze implementovat do externích souborů. Tím lze také docílit toho, že může být definováno více stylů na stránku a uživatel si může vybrat, který vzhled stránky se mu líbí nejvíc.

1.4.3 JavaScript

JavaScript je objektově orientovaným skriptovacím jazykem, který se používá především jako interpretovaný programovací jazyk pro WWW stránky. Javascript se vkládá přímo do HTML kódu stránky. Obvykle se využívá pro ovládání různých interaktivních prvků grafického uživatelského rozhraní (GUI), jako jsou tlačítka, či textová pole, případně jsou jím tvořeny animace a efekty obrázků či textu.

Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe. JavaScript byl původně obchodní název implementace společnosti Netscape, kde byl vyvíjen nejprve pod názvem Mocha, později LiveScript, ohlášen byl společně se společností Sun Microsystems v prosinci 1995 jako doplněk k jazykům HTML a Java. Pro verzi firmy Microsoft je použit název JScript.

Program v JavaScriptu se spouští obvykle na straně klienta, tzn. že není spouštěn na serveru ještě před stažením stránky, jako je to u jiných interpretovaných jazyků (PHP, ASP). Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

JavaScript je možné použít i na straně serveru. První implementací JavaScriptu na straně serveru byl LiveWire firmy Netscape vypuštěný roku 1996, dnes existuje několik možností včetně opensource implementace Rhinola založená na Rhino, gcj a Apache.

1.4.4 PHP

1.4.4.1 Úvod

Hypertext Preprocesor (zkráceně PHP) je rozšířený univerzální skriptovací jazyk, který je obzvláště vhodný pro vývoj dynamických webových aplikací a lze jej jednoduše zapouzdřit do HTML. PHP je interpretovaný, ne kompilovaný jazyk. To znamená, že jeho kód není přeložen do binárního a neběží v prohlížeči návštěvníka, ale je interpretován přímo na serveru. Tímto se podobá Perlu, ale PHP byl vytvořen specificky tak, aby šlo vkládat skript do HTML stránek jednoduchým způsobem. Toto je obrovská výhoda jazyka PHP, protože můžeme v podstatě libovolně přecházet s HTML kódu do skriptu jazyka PHP a naopak. PHP kód je uzavřen mezi zvláštní otevírací a uzavírací značky (<?php, ?>), které nám umožňují vstoupit a opustit "PHP mód".

Jako výhody tohoto jazyka můžeme označit, že je relativně jednoduchý na pochopení, má syntaxi velmi podobnou jazyku C a je tedy velkému množství vývojářů dosti blízký. PHP podporuje velkou řadu souvisejících technologií, formátů a standardů. Obrovskou výhodou je také to, že je multiplatformní a lze jej provozovat s většinou webových serverů a na většině dnes existujících operačních systémů.

Jak jsem se již zmínil, PHP je vykonáván na straně serveru, to má mimo jiné tu výhodu, že klient obdrží pouze výstup ze skriptu, bez možnosti zjistit, jaký může být použit zdrojový kód skriptu, proto je mnohem obtížnější takový kód napadnout nějakou nekalou praktikou.

S PHP nejsme omezeni pouze na výstup HTML. Schopnosti PHP zahrnují výstup obrázků, souborů PDF a dokonce Flash animací (použitím libswf a Ming) generovaných za běhu. Výstupem může být rovněž libovolný text jako třeba XHTML nebo jakýkoli jiný XML soubor. PHP umí tyto soubory nejen vypisovat, ale také automaticky generovat a ukládat je do souborového systému, čímž vznikne cache dynamického obsahu na straně serveru.

Jedna z nejsilnějších a nejvýznamnějších vlastností PHP je jeho podpora pro širokou škálu databází. Vytvoření webové stránky spolupracující s databází je neuvěřitelně jednoduché. Podporovaných databází je velké množství, pro tento projekt je zvláště důležitá podpora databáze Oracle.

1.4.4.2 Princip PHP

V dobách, kdy internet začínal, byly webové stránky statické. Prostě se stránka prohlížeči poslala přesně tak, jak je napsaná a tak se také zobrazila. To po čase začalo být samozřejmě nedostačující, bylo zapotřebí zobrazovat také obsah dynamický a proto vznikla řada technologií, které měly stránky rozpohybovat. Tyto technologie by se dali rozdělit na „klientské“ a „serverové“.

„Klientské“ technologie fungují tak, že se spolu s HTML stránkou pošlou prohlížeči i nějaký kus programového kódu, který je ve vhodnou chvíli na „cílovém“ počítači spuštěn. Vhodnou chvílí se myslí například stisknutí tlačítka, najetí myší nad odkaz apod.. Toto řešení může mít tu nevýhodu, že o spuštění klientského kódu se stará prohlížeč, takže on musí mít podporu pro programovací jazyk, v němž je programový kód napsán. Příkladem takového programovacího jazyka může být například JavaScript.

Oproti tomu "serverové" technologie, jímž je například PHP, jsou založeny na jiném principu. Když prohlížeč požaduje webovou stránku ze serveru, server ji nejdříve sestaví a pak ji teprve odešle. Servery mohou (a také to často dělají) sestavovat pokaždé jinou stránku v závislosti na tom, co přesně prohlížeč požaduje. Typický PHP skript se skládá z kusů normálního HTML kódu, mezi nějž jsou vkládány kusy PHP kódu. Pak sestavování stránky probíhá asi následujícím způsobem: kusy HTML kódu tak, jak jsou, části PHP programového kódu proveď, výsledek zkombinuj a odešli prohlížeči. Tato filozofie fungování je nesmírně mocná, neboť server může provést jednu nebo několik operací a výsledek poslat do prohlížeče jako obyčejnou HTML stránku.

1.4.5 SQL

SQL (Simple Query Language nebo Structured Query Language) je standardní dotazovací jazyk používaný pro práci s databázemi. To znamená, že jej můžeme využít na různých platformách a s různými databázemi. Na internetu může být takovou databází MySQL, Oracle server a další. SQL se skládá z mnoha příkazů (dotazů), které se velmi podobají klasickému jazyku - angličtině.

Příkazy jazyka SQL jsou členěny do několika skupin. Základní skupinu příkazů tvoří příkazy pro manipulaci s daty. Manipulací rozumíme zejména vyhledávání dat, ale také přidávání nových řádků do tabulek, aktualizací hodnot v řádcích nebo vymazání existujících řádků. Další skupiny příkazů umožňují vytváření a rušení tabulek, indexů a jiných databázových objektů a nastavování parametrů databázového systému.

Jazyk SQL byl navržen jako tzv. neprocedurální jazyk. V příkazech se popisuje, „co“ chceme získat, a ne „jak“ (postup, proceduru) to chceme získat. Liší se tak od většiny programovacích jazyků, které jsou procedurální a ve kterých vždy popisujeme přesný postup toho, co se má provést.

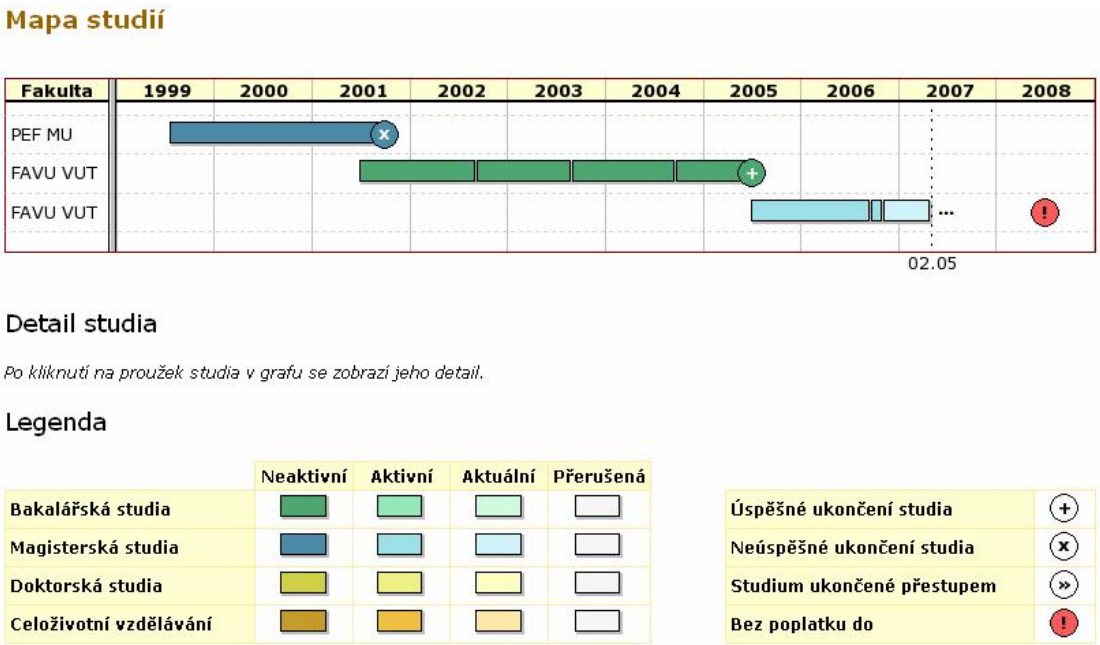
Na webu se velmi často používají databáze, i zde se ke komunikaci používá SQL. Pokud používáme PHP, tak se dají SQL dotazy vkládat přímo do PHP kódu.

2 Grafický prohlížeč studií studenta

2.1 Úvod

Na webovém portálu VUT jsem implementoval grafický prohlížeč studií studenta (mapa studií) a grafický prohlížeč návazností studijních programů, oborů a zaměření (mapa studijních programů). V této kapitole se zaměřím na prvního z nich.

Tento prohlížeč je určen pro všechny studenty VUT. Jeho cílem je, aby v přehledné grafické formě studentovi zobrazil všechna jeho dosavadní studia na Vysoké škole (školách). Vzhledem k tomu, že databáze celoškolského studijního systému VUT obsahuje data i z matriky, tak tento prohlížeč nebude omezen pouze na studia na Vysokém učení technickém, ale bude zobrazovat studia studenta i na jiných vysokých školách, i když ne tak podrobně, jako studia na VUT (vzhledem k omezeným informacím o těchto studiích v databázi). Tento grafický způsob reprezentace je užitečný zejména v agendě sociálních a ubytovacích stipendií, jejichž podmínky jsou úzce svázány s kombinací předešlých a aktivních studií. Taktéž lze jednoznačně vidět, kdy dojde k financování studia ze studentovy kapsy.



Obrázek 2.1: Studia na VUT i na jiných školách

2.2 Návrh prohlížeče

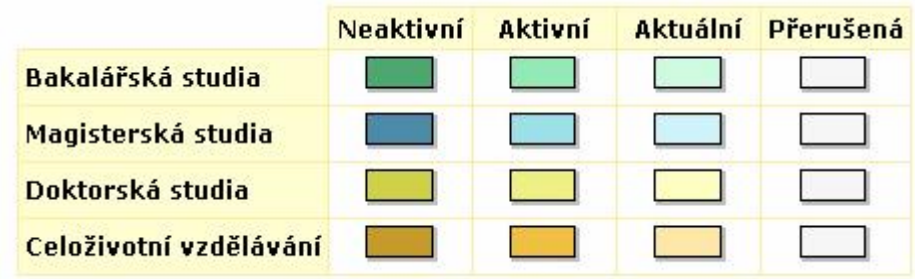
Jak jsem již zmínil, grafický prohlížeč studií studenta bude zobrazovat veškerá studia na Vysokých školách daného studenta.

Od svého počátku prošel návrh prohlížeče mnoha změnami. Důležité bylo udělat návrh prohlížeče tak, aby byl maximálně přehledný. Jedním z prvních návrhů, který jsem vytvořil bylo rozlišovat barevně studia na VUT od studií na jiných školách a typy studií rozlišovat různými typy šrafů. Nakonec se ukázalo, že šrafování v grafu nepůsobí příliš efektním dojmem a celkově spíše aplikaci znepřehledňuje. Barevné rozlišení studií na VUT od ostatních se nakonec ukázalo jako zbytečné a tak vznikl další návrh který popíši dále.

Jak je z předchozího obrázku 2.1 patrné, jednotlivá studia jsou rozlišena barevně, případně symbolem na konci proužku studia. Pro celkové zpřehlednění zobrazovaných studií se barevně rozlišuje typ studia. Různé odstíny barev dále rozlišují studia na neaktivní, aktivní, aktuální (právě probíhající) a přerušená. Symbolem na konci je ukázán typ ukončení studia. Proužky grafu jsou u studií na VUT navíc rozděleny podle jednotlivých změn ve studiu tak, aby se z mapy studií dalo vyčíst co nejvíce dostupných informací o daném studiu. U studií na jiných školách toto rozdělení proužků podle jednotlivých změn není možné, protože jsou v databázi jen omezené informace o těchto studiích.

2.2.1 Typy studií

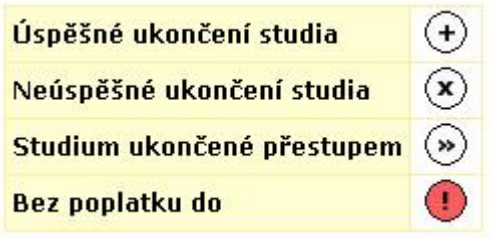
Jednotlivé typy studií jsou rozlišeny pomocí různých barev proužků. Toto je základní rozlišení studií a je společné pro studia na VUT i na jiných školách. Použité barvy ukazuje následující tabulka.



Obrázek 2.2: Typy studií

2.2.2 Ukončená studia

Ukončená studia mají nejtmavší odstín barvy proužku a jsou označena symbolem vpravo. Tento symbol značí typ ukončení studia. Jednotlivé symboly uvedu v následující tabulce.

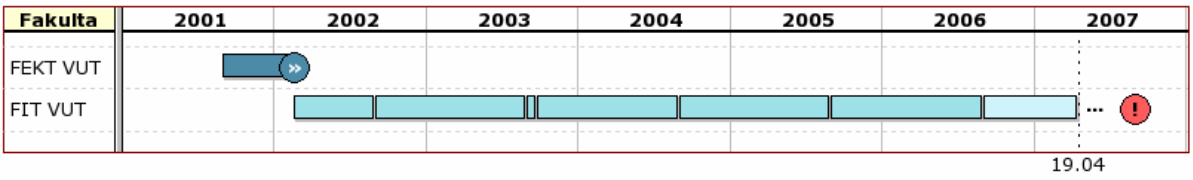


Obrázek 2.3: Typy ukončení studia

V tabulce si lze všimnout také symbolu „bez poplatku do“. Tento symbol neoznačuje typ ukončení studia, ale označuje v aktivním, či přerušeném studiu datum, do kdy si studium student nemusí hradit „z vlastní kapsy“. U ukončených studií tato informace již postrádá smysl, proto se u nich nezobrazuje.

2.2.3 Aktivní a přerušená studia

Aktivní či přerušená studia lze vidět pouze u studií na VUT. Aktivní studia, nejsou označena symbolem vpravo, protože ještě nejsou ukončena. Tato studia jsou ovšem rozlišena barevně. Oproti ukončeným studiím mají světlejší odstín barvy proužku. Dále je u těchto studií vyznačeno aktuální datum a symbol pro značku „bez poplatku do“. Navíc u aktivních studií je vyznačeno i aktuální studium, což je studium, které právě probíhá, tj. od posledního zápisu po dnešek (které má opět o něco světlejší barevný odstín).

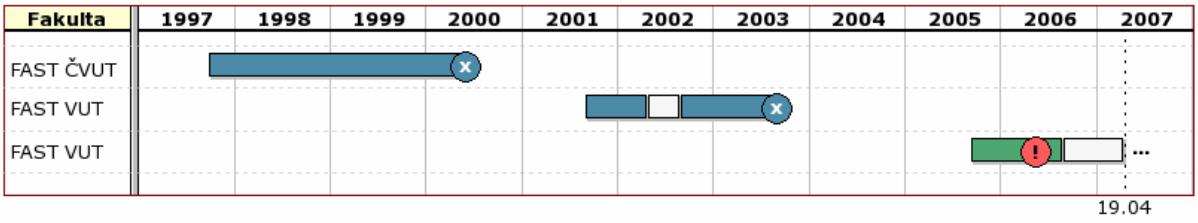


Obrázek 2.4: Aktivní studium

Ve studiích na VUT se mohou objevit také přerušená studia a ty mohou být dvou typů. Přerušení studia může být poslední změnou ve studiu. Pokud tomu tak je, přerušené studium se zobrazuje obdobně jako studium aktivní, protože k takovému studiu se dá ještě vrátit, i když momentálně není označeno jako aktivní. To znamená, že se od poslední změny (přerušení studia)

vykreslí proužek až po aktuální datum a vykreslí se symbol (…) který naznačuje možnost pokračování ve studiu.

Druhou možností je, že přerušené studium není poslední změnou. To znamená, že někde ve studiu se objeví přerušení studia a za ním následuje další změna studia, která značí návrat po přerušení. Toto přerušení je opět označeno bílým proužkem (od začátku přerušení, po jeho konec). V následujícím obrázku jsou vidět obě možnosti, tj. přerušení uprostřed studia i na konci.

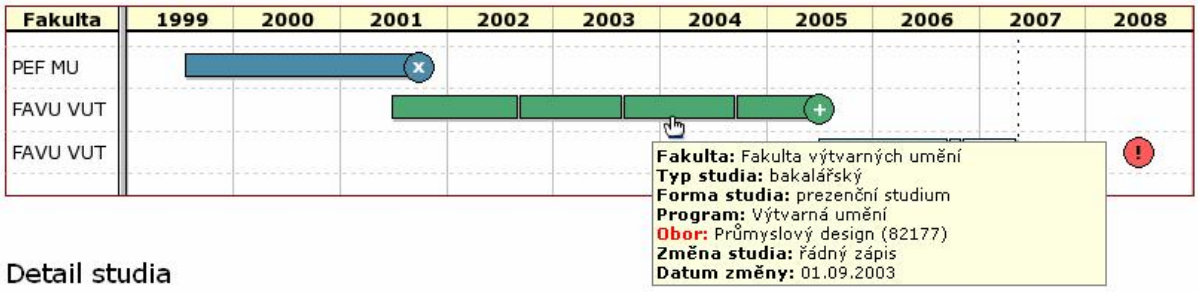


Obrázek 2.5: Přerušené studium uprostřed a na konci

2.2.4 Nápověda a detail studia

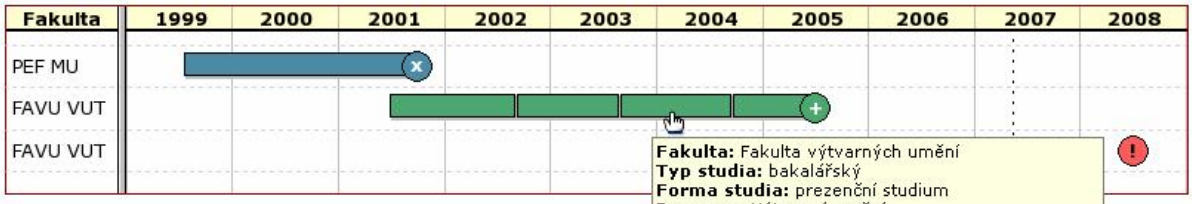
Pokud kurzorem „najedeme“ nad jakýkoliv proužek, či symbol v grafu, zobrazí se přehledná nápověda, ve které jsou všechny potřebné informace o dotyčném studiu, respektive o části studia (změně ve studiu).

Pokud se podíváme na následující obrázek 2.6, vidíme, že je zvýrazněna položka „obor“. Je to z toho důvodu, že v předchozím proužku (tj. části studia) byl obor jiný a tímto se dává jasně vědět o tom, že se obor od předchozí části studia změnil. Vzhledem k tomu, že při studiu mohou nastávat různé změny ve formě studia, programu, či oboru jsou tyto informace zvýrazněny, pokud dojde k jejich změně od předchozí části studia.



Obrázek 2.6: Nápověda

Po kliknutí na jakýkoliv proužek, či symbol v grafu se zobrazí detail studia, ve kterém jsou přehledně vypsány všechny změny ve studiu tak, jak jsou naznačeny v grafu.



Detail studia

Fakulta	Fakulta výtvarných umění (FAVU)
Typ studia	bakalářský

datum	popis změny	roč.	zam.	obor	program	forma
27.06.2001	řádný zápis	1	APD	Průmyslový design	Výtvarná umění (4-letý)	P
09.09.2002	řádný zápis	2	APD	Průmyslový design	Výtvarná umění (4-letý)	P
01.09.2003	řádný zápis	3	APD	Průmyslový design (82177)	Výtvarná umění (4-letý)	P
23.09.2004	zápis se změnou oboru, programu, formy	4	ATD	Průmyslový design	Výtvarná umění (4-letý)	P
01.07.2005	ukončení studia včetně SZZ	4	ATD	Průmyslový design	Výtvarná umění (4-letý)	P

Obrázek 2.7: Detail studia

V detailu studia se tedy zobrazují informace jako fakulta, na které studium probíhá (probíhalo), typ studia, datum a popis změny ve studiu, ročník, zaměření, obor, program (obvyklá délka studia programu) a forma studia.

3 Implementace mapy studií

3.1 Vytvoření nového modulu v systému StudIS

Jak jsem již v předchozí kapitole zmínil, mapa studií i mapa studijních programů je součástí webového portálu VUT. Oba tyto prohlížeče jsem vytvořil, jako moduly do studentské části tohoto portálu (StudIS). V následujících několika řádcích popíši postup, který je nutno dodržovat při vytváření nového modulu v systému StudIS.

- V adresáři „studis/app" vytvoříme nové skripty
  „mapa_studii.php", další pokud jsou potřeba
- Upravíme skript „studis/student.phtml", do „switch ($script_name)" přidáme

```
case 'mapa_studii':
      $GLOBALS['Page_Title'] = get_lang('title_mapa_studii');
      $modul_name = 'mapa_studii';
      break;
```

- Upravíme skript „_base/site_lib/func_tr_data.php" (doplníme název stránky modulu, název odkazu na modul, a nápovědu pro odkaz)

```
$tr_mem['title_mapa_studii'] = array(
      'Mapa studií',
      'The map of studies'
);
```

  obdobně `$tr_mem['mapa_studii']` a `$tr_mem['mapa_studii_hint']`
- Upravíme skript „studis/libs/navigation_conf.php", do „switch ($modul_name)" přidáme

```
case 'mapa_studii':
      $left_menu[ $modul_name ] = array(
            'TITLE' => get_lang('mapa_studii_hint'),
            'ANCHOR'=> get_lang('mapa_studii'),
      );
      break;
```

- Přidání modulu do menu docílíme opět úpravou „studis/libs/navigation_conf.php"

```
pridej_do_menu( 'mapa_studii' );
```

- Pokud je modul vázán na parametrizaci, doplníme ještě v souboru „studis/libs/modul_conf.php" funkci

```
c_parametr_is_id2modul_name( $id )
```

Tímto postupem jsme přidali modul do studentské části portálu VUT a v levém menu se zobrazí odkaz na tento modul.

Samotný princip programování modulu je pak takový, že ve vytvořeném skriptu „mapa_studii.php“ pomocí SQL dotazů vybereme všechna potřebná data z databáze a ty si uložíme do vhodně pojmenované proměnné (případně pole).

Následné zpracování a zobrazení těchto vybraných dat pak provádíme v jiném skriptu v adresáři „studis/templates“. Pojmenujeme ho např. „tpl_mapa_studii.php“ a z původního skriptu ho pouštíme pomocí klíčového slova require_once.

Důvod rozdělení modulu mezi dva skripty je jasný, celkové zpřehlednění modulu, tj. separace dotazů na SQL server od samotného zobrazování sesbíraných dat.

3.2 Vícejazyčné rozhraní systému StudIS

Při implementaci modulu pro portál VUT se musí také myslet na česky nemluvící studenty. Na portálu se tedy v každém modulu dá zvolit jazyk anglický, místo standardního českého.

V praxi to funguje tak, že všechny statické (tj. neměnící se) texty jsou uloženy v souboru „_base/site_lib/func_tr_data.php“ v poli „$tr_mem”. Způsob ukládání jsme si ukázali v předchozí kapitole, při vytváření nového modulu (resp. jeho názvu). Ve skriptu se texty potom zobrazují pomocí funkce „get_lang(‘polozka z $tr_mem’)”. Tato funkce podle zvoleného jazyka vrátí buď český nebo anglický text.

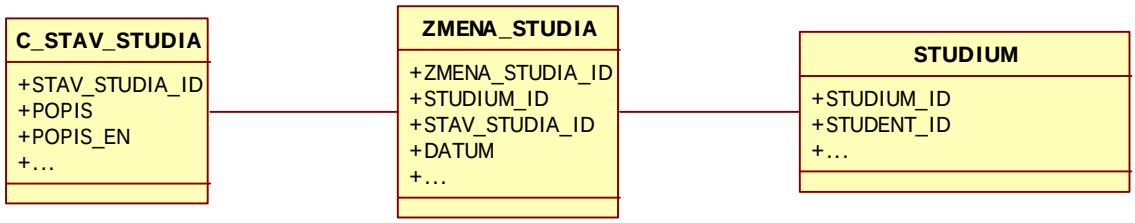
Vzhledem k tomu, že v databázi jsou u každé tabulky nejen české, ale i anglické překlady, tak zobrazování dynamických dat je o něco jednodušší. V SQL dotazu se vyberou jak české, tak anglické překlady a podle zvoleného jazyka se pouze rozhodne, který z těchto dvou textů se má vybrat k zobrazení na stránce.

3.3 Postup při implementaci

3.3.1 Výběr dat z databáze

Prvním krokem při implementaci mapy studií bylo vybrat potřebná data z databáze. Nejprve tedy pro přihlášeného studenta (uživatele) vyberu všechna jeho studia na VUT. U těchto studií vybírám informace, které jsou pro dané studium stejné a nemění se postupem času. Jsou to informace, které když se změní, tak se zakládá pro studenta nové studium. Jsou to například začátek studia, název a zkratka fakulty, do kdy je studium bez poplatku a typ studia. Vybírají se pouze tyto informace, protože všechny další se mohou v průběhu studia změnit a nové studium se kvůli jejich změně nezakládá (např. program, či obor).

Pro takto vybraná studia vyberu všechny změny, které ve studiu proběhly. Mezi tyto změny patří například řádný zápis, různé typy ukončení studia, změny programu a oboru apod.. Základní vazba mezi studiem, jeho změnami a popisem těchto změn je vidět na následujícím obrázku.



Obrázek 3.1: Vazba mezi studiem a jeho změnami

Ve změnách ve studiu se tedy vybírají z databáze informace, jako jsou stav studia (zápis, přerušení studia, přestup, ukončení apod.), datum změny, program, obor, stupeň, forma a ročník v době změny studia.

Tímto mám připravená všechna potřebná data pro vykreslení studií na VUT. Nyní je na čase vybrat studia na jiných školách. Tato studia se vybírají trochu odlišným způsobem. Tato studia jsou uložena v tabulce „studia_na_jine_skole“. Jsou v ní uložená pouze ukončená studia (ať již úspěšně, či naopak) a nelze k nim zjistit veškeré změny, které toto studium provázely. Jediné tedy, co o studiu na jiné škole můžeme zjistit je datum začátku a konce studia, na jaké vysoké škole (fakultě) probíhalo, v jakém studijním programu, jakého typu bylo, jakou formou probíhalo a některé další.

Studia na VUT i na jiných školách si ukládám do stejného pole, pouze si každou položkou v tomto poli označím, zda se jedná o studium na VUT, či ne. To, že si oba typy studií ukládám do stejného pole má prostý důvod a to, že pro zpřehlednění vykreslené mapy studií potřebuji studia

seřadit podle data začátku studia a to není možné dělat na úrovni SQL dotazu, protože na vybírání studií na VUT a jiných studií používám dva různé dotazy.

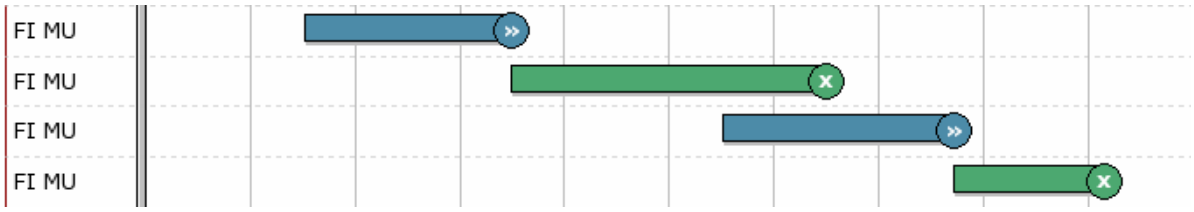
3.3.2 Vykreslení mapy studií

Pro vykreslení mapy studií používám grafickou knihovnu JpGraph, kterou popíši kapitole 3.4. V principu vykreslení funguje tak, že si vytvořím graf a postupně procházím všechna studia studenta a vykresluji proužky (gantt bars), signalizující průběh studia do grafu.

3.3.2.1 Studia na jiných školách

Vykreslení studií na VUT probíhá odlišným způsobem, než studií na jiných školách. Jediné, co je u zobrazování těchto studií společné, je nastavení barvy proužku, podle typu studia (bakalářský, magisterský, celoživotní). Studia na jiných školách se vykreslují jednodušším způsobem, vzhledem k tomu, že se nezobrazují změny, které tato studia provázela, proto nejprve objasním zobrazování těchto studií.

U zobrazení studií na jiných školách jde vlastně jen o vykreslení proužku od data začátku studia do data jeho ukončení. Na konec tohoto proužku ještě umístím značku (milestone), který ukazuje typ ukončení tohoto studia. Pak již stačí nastavit javascriptovou nápovědu, která se zobrazí po najetí kurzorem nad proužek, či symbol ukončení studia a studium na jiné škole je zobrazeno.



Obrázek 3.2: Zobrazení studií na jiných školách

3.3.2.2 Studia na VUT

Jak jsem již zmínil, vykreslení studií na VUT je o něco komplikovanější. U těchto studií totiž zobrazuji veškeré změny, které takové studium potkalo. Navíc oproti studiím na jiných školách mohou být studia na VUT stále aktivní (neukončená), či přerušená. To znamená, že pro každé studium na VUT postupně procházím změny v tomto studiu a vykresluji na jeden řádek v grafu proužky vždy od poslední změny, po změnu aktuální. Pokud je poslední změnou nějaký typ ukončení studia, vykreslím také značku, která toto ukončení označuje.

Obrázek 3.3: Zobrazení studií na VUT

Pokud je studium stále aktivní (tj. ještě neukončené) nebo je poslední změnou ve studiu přerušení studia, je to nutné v grafu také naznačit. Toto zobrazuji tak, že od poslední změny ve studiu vykreslím ještě proužek po aktuální datum (tj. datum zobrazení grafu, které i v grafu zobrazím). Dále vykreslím symbol (…), který ukazuje, že studium stále probíhá, resp. může probíhat u přerušených studií.

Obrázek 3.4: Zobrazení aktivního studia

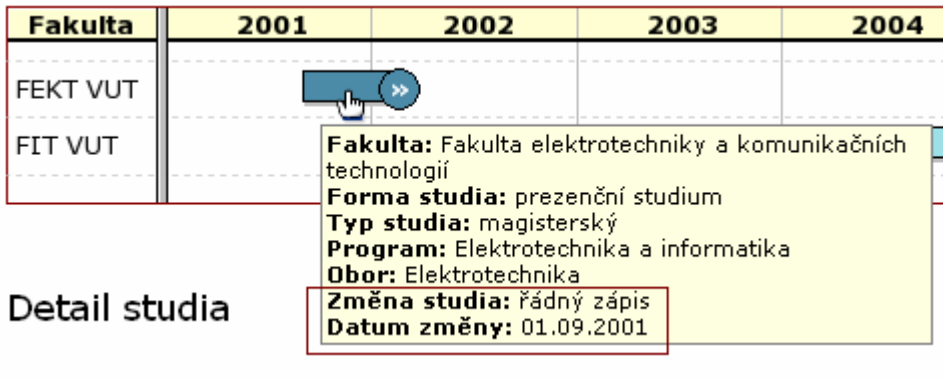
Obrázek 3.5: Zobrazení přerušeného studia

Při zobrazování neukončených studií také zobrazuji značku pro datum, do kdy je studium nezpoplatněno. Tato informace je uložena v tabulce „studium“, díky tomu toto datum nemusím složitě počítat. U ukončených studií již tato informace není podstatná, proto ji u nich nezobrazuji.

Obrázek 3.6: Zobrazení značky "bez poplatku do"

3.3.2.3 Nápověda pro objekty v grafu

Pro všechny objekty (proužky a značky) v grafu zobrazuji podrobnou nápovědu, co který objekt představuje. V předchozí kapitole jsem uvedl, že proužek v grafu představuje buď celé studium (v případě, studia na jiné škole) nebo část studia (v případě studií na VUT). Pokud tedy najedeme kurzorem nad proužek grafu, zobrazí se nápověda k tomuto studiu (resp. části studia). Nápověda se zobrazí vždy k počáteční změně ve studiu (resp. části studia). Pro značky v grafu zobrazuji obdobnou nápovědu, která přesně značí, co daná značka znamená.



Obrázek 3.7: Zobrazení nápovědy pro proužek

Tuto nápovědu zobrazuji pomocí skriptu „DHTML Tooltip Skript“ napsaného v jazyce JavaScript. Jeho použití je jednoduché. Pro jakýkoliv html odkaz, lze vytvořit událost onmouseover a onmouseout. To znamená, že pokud najedeme kurzorem nad odkaz, vyvolá se událost onmouseover a zobrazí se nápověda. Pokud kurzorem přejedeme z odkazu pryč, vyvolá se událost onmouseout a nápověda se skryje. Uvedu jednoduchý příklad použití.

```
<a href="http://dynamicdrive.com"
     onMouseover="showtip('<b>Nápověda</b>')"
     onMouseout="hidetip()"> Dynamic Drive
</a>
```

Při přidávání této nápovědy do grafu jsem se setkal s jedním problémem. V JpGraphu můžeme jakémukoliv objektu přiřadit URL odkaz, na který tento objekt ukazuje. Problém ovšem je, že v JpGraphu lze definovat k objektu pouze cílovou adresu URL a popis této adresy. Pokud jsem tedy chtěl přidat ještě například událost „onmouseover“, musel jsem trochu improvizovat.

Takto například vypadá použití funkce pro vytvoření odkazu k proužku v grafu.

```
$bar->SetCSIMTarget('priklad.com', 'priklad');
```

Pokud chci přidat událost přejetí myší, musí kód vypadat následujícím způsobem.

```
$bar->SetCSIMTarget('priklad.com', 'priklad" onmouseover="funkce()');
```

Je to sice trošku nepěkné, ale jiným způsobem bych nedocílil zobrazování nápovědy. Na fóru JpGraphu jsem dal podnět tvůrcům této knihovny k rozšíření metod používaných pro vytváření URL odkazů o nadstandardní parametry, takže snad budou použity v některé z nových verzí knihovny JpGraph.

3.3.2.4 Detail studia

Detail studia se zobrazuje pod grafem jako klasická HTML tabulka. Obsahuje informace, které lze vyčíst přímo z grafu pomocí nápovědy, jejíž vytvoření jsem popsal v předchozí kapitole. Tento detail je ovšem přehlednější v tom, že můžeme vidět veškeré změny ve studiu vypsané pod s sebou.

Detail studia

Fakulta	Fakulta informačních technologií (FIT)
Typ studia	magisterský
Bez poplatku do	31.08.2007

datum	popis změny	roč.	zam.	obor	program	forma
18.02.2002	řádný zápis	1	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
01.09.2002	řádný zápis	2	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
01.09.2003	řádný zápis	1	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
25.09.2003	změna zařazení - programu, formy, oboru, ročníku	3	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
01.09.2004	řádný zápis	2	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
01.09.2005	řádný zápis	3	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P
01.09.2006	řádný zápis	4	--	Výpočetní technika a informatika	Elektrotechnika a informatika (5-letý)	P

Obrázek 3.8: Zobrazení detailu studia

Princip zobrazování detailů studia spočívá v tom, že po vykreslení mapy studií, modul znovu projde všechna studia a ke každému se vytvoří jeho detail. Tyto detaily se vloží mezi párové tagy . Tyto tagy se nastaví tak, aby se nezobrazovaly (style="display:none"). Pokud v grafu kliknu na některé studium, zavolám JavaScriptovou funkci, která tag , odpovídající danému studiu, nastaví tak, aby se zobrazil (style="display:block"). Poslední zobrazený, se naopak skryje, takže zůstane zobrazený pouze detail právě „odkliknutého" studia.

JavaScriptová funkce pro zobrazování, resp. skrývání detailu studia se volá podobně jako předchozí legenda po události „onclick". Při implementaci s tím byl obdobný problém, který je popsán na konci předchozí kapitoly.

3.3.2.5 Legenda

Legenda se zobrazuje jako klasická HTML tabulka. Formátování této tabulky, stejně jako tabulky s detaily studia, je pomocí CSS stylů používaných na celém portálu VUT tak, aby mapa studií zapadla svým vzhledem mezi ostatní moduly na portálu.

Legenda

	Neaktivní	Aktivní	Aktuální	Přerušená
Bakalářská studia				
Magisterská studia				
Doktorská studia				
Celoživotní vzdělávání				

Úspěšné ukončení studia	⊕
Neúspěšné ukončení studia	⊗
Studium ukončené přestupem	»
Bez poplatku do	!

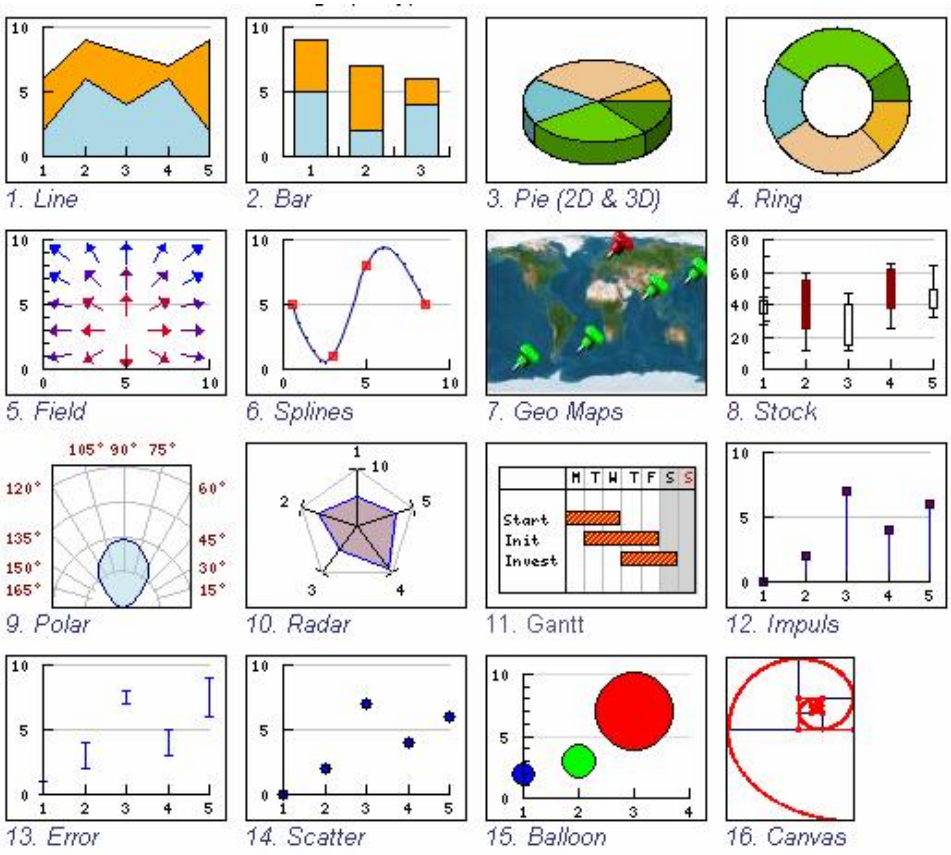
Obrázek 3.10: Zobrazení legendy

3.4 JpGraph

JpGraph je objektově-orientovaná knihovna pro PHP vytvořená pro generování grafů. Tato knihovna je kompletně napsaná v PHP a je připravena pro použití v jakémkoli PHP skriptu. Tuto knihovnu jsem s výhodou využil při implementaci mapy studií.

JpGraph může být použit k vytvoření obrovského množství typů grafů. Grafy mohou být generované on-line, ale lze je i uložit do souborů. JpGraph umožňuje vykreslení jak jednoduchých grafů za pomoci jednoduchého zdrojového kódu, tak grafů složitých. V knihovně má většina parametrů, které se používají pro formátování grafu standardní hodnoty, což snižuje délku učení, jak knihovnu používat. Nadstandardní hodnoty tedy můžeme používat, jen když je potřebujeme.

Na následujícím obrázku jsou ukázány některé grafy, které lze pomocí JpGraph generovat.



Obrázek 3.11: Ukázky grafů (JpGraph)

Pro implementaci mapy studií jsem použil graf Ganttův. Ten je ideální pro takový typ zobrazování grafu, který jsem potřeboval u mapy studií. V následujících kapitolách popíši základní principy použití knihovny JpGraph a postupy pro generování Ganttových grafů. Nehodlám opisovat celý manuál (proč také), ale pouze uvést některé základní postupy a popsat důležité metody základních tříd JpGraphu.

3.4.1 Instalace JpGraph

Pro zajištění chodu knihovny je nutné zajistit správný přístup k souborům knihovny (bude popsáno dále) a zajistit, že instalace PHP na serveru podporuje alespoň jeden grafický formát, např. že podporuje „image“ rozšíření.

Nejjednodušší způsob jak zjistit jestli je toto rozšíření dostupné, je pomocí standardní funkce „phpinfo()“ nebo se ujistíme, že je v instalaci PHP dostupná funkce „imagecreate()“. To znamená, že instalace musí mít funkční GD-knihovnu předtím, než JpGraph může být použit. Pro správný chod knihovny je také nutné mít PHP ve verzi minimálně 4.3.8. Pro dostupnost pokročilých vlastností JpGraphu je nutná GD2-knihovna. V nejnovější verzi JpGraphu (1.21b) je již dokonce nutná knihovna GD2, GD byla označena jako zastaralá, tudíž se již nepoužívá.

Celá knihovna JpGraph může být nakopírována v podstatě kamkoli na server tak, aby se k ní dalo přistupovat pomocí klíčového slova „include“ z PHP skriptu. Základním souborem knihovny je „jpgraph.php“, který je potřebný vždy. Vykreslovacím rozšířením, které jsem použil pro mapu studií je „jpgraph_gantt.php“. Zpřístupnění knihovny s Ganttovými grafy v PHP skriptu může tedy vypadat následujícím způsobem:

```
include($GLOBALS['lib_gdir'] . 'jpgraph-1.20.5/jpgraph.php');
include($GLOBALS['lib_gdir'] . 'jpgraph-1.20.5/jpgraph_gantt.php');
```

3.4.2 Dynamické generování obrázků

3.4.2.1 Úvod

Základním pravidlem pro generování obrázků v PHP je, že obrázek musí být specifikován ve zvláštním souboru, který je volán v HTML tagu „<img>”. Například následující část HTML kódu vloží na stránku obrázek, který generuje skript „fig1.php“.

```
<img src="fig1.php">
```

Knihovna automaticky generuje všechny potřebné hlavičky, které se posílají zpátky prohlížeči, a ten díky tomu správně rozezná, že přijímaná data jsou obrázek ve formátu PNG, GIF, či JPEG.

Jak jsem již zmínil, k přístupu ke knihovně JpGraph je potřeba minimálně dvou souborů, které se musí zahrnout do skriptu. Je to základní soubor knihovny a jeden z jeho rozšíření. Takže soubor „fig1.php“ z předchozího příkladu by mohl vypadat asi následujícím způsobem.

```
<?
include ('jpgraph.php');
include ('jpgraph_line.php');
...
 // Kód knihovny jpgraph
...
?>
```

3.4.2.2 Základní principy

Obvyklý způsob pro vytvoření grafů je vytvoření skriptu, který generuje obrázek. Ten „obalit“ jiným skriptem, který obsahuje jeden, či více „<img>” tagů, které umístí grafy na správné místo v HTML stránce.

Samozřejmě je také možné zavolat PHP skript přímo prohlížečem, čímž docílíme prostého zobrazení vygenerovaného obrázku v prohlížeči.

Nyní je na čase ukázat základní strukturu obrázkového skriptu.

```
// ... Vložení potřebných souborů
$graph  = new Graph($width, $height, ...);
// ... kód k vytvoření detailů grafu
$graph->Stroke();
```

JpGraph je kompletně objektově orientovaný, takže všechna volání budou akcí na specifickou instanci třídy. Jednou ze základních tříd je třída Graph(), která reprezentuje celý graf. Všimněme si parametrů metody, které udávají velikost (šířku a výšku) výsledného obrázku. Po vytvoření objektu Graph() se do kódu vkládají veškeré detaily grafu.

Poslední metodou volanou v obrázkovém skriptu je Graph::Stroke(). Tato metoda pošle výsledný obrázek zpátky prohlížeči. Pokud v grafu potřebujeme využívat obrazové mapy, bude poslední metoda volaná ve skriptu Graph::StrokeCSIM().

Parametrem metody Graph:Stroke($file) lze také výsledný obrázek vygenerovat do souboru, namísto toho, aby se poslal zpátky prohlížeči.

```
$graph-> Stroke("/usr/home/peter/images/result2002.png");
```

3.4.2.3 Výběr formátu obrázku

Standardně JpGraph automaticky vybírá formát obrázků v pořadí PNG, JPEG, GIF. Přesný formát obrázku záleží na dostupnosti na systému serveru. Existují dvě cesty, jak lze ovlivnit výběr formátu.

Změnit standardní nastavení formátu pomocí klíčového slova DEFINE nebo pomocí metody SetImgFormat().

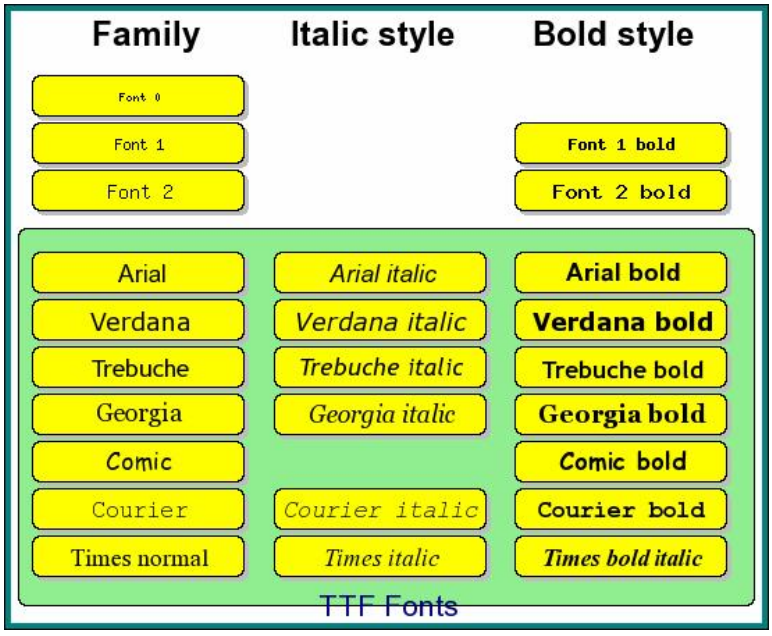
```
DEFINE ("DEFAULT_GFORMAT", "auto")
$graph->img->SetImgFormat("jpeg")
```

3.4.2.4 Práce s fonty

JpGraph podporuje dva typy fontů. Jsou to vestavěné bit-mapové fonty a také true typové fonty (TTF). Pro grafy se doporučuje používat vestavěné fonty. Je to z jednoho prostého důvodu, že jsou pro většinu lidí snadněji čitelné a také jsou rychleji renderovány. TTF se doporučuje používat jen tam, kde je to nezbytné (např. podpora češtiny v nadpisech).

Pro zprovoznění true typových fontů je potřeba JpGraphu sdělit, kde se nacházejí. Je třeba nastavit cestu v souboru „jpg-config.inc“. Cesta musí být uvedena v absolutním tvaru.

```
DEFINE("TTF_DIR", "/usr/local/fonts/ttf/");
```



Obrázek 3.12: JpGraph - Ukázka některých fontů

A takto může vypadat příklad správného použití změny fontů.

```
$graph->title->SetFont(FF_FONT2);
$graph->title->SetFont(FF_FONT2, FS_BOLD);
$graph->title->SetFont(FF_ARIAL);
$graph->title->SetFont(FF_ARIAL, FS_BOLD,24);
```

3.4.2.5 Specifikace barev

Barva může být změněna téměř u každého objektu v grafu (barvy čar, textu, výplně apod.). Barvy mohou být definovány třemi různými způsoby.

Prvním způsobem je využití jednoho asi ze 400 předdefinovaných názvů barvy. Jméno barvy lze také upravit pomocí čísla, které zesvětluje, či ztmavuje původní barvu. Číslo menší než 1 barvu ztmavuje, a naopak.

```
SetColor("khaki");
SetColor("khaki:0.5");   // Tmavší verze „khaki“
SetColor("yellow:1.2");  // Tmavší verze „yellow“
```

Druhým způsobem je specifikace pomocí trojice barev RGB.

```
SetColor(array(65, 100,176));
```

Posledním způsobem je specifikace pomocí hexa kódu barvy.

```
SetColor ("#A16BFF");
```

Obdobně jako lze barvy zesvětlovat, či ztmavovat, lze jim také nastavovat průhlednost. Průhlednost některého objektu lze vytvořit následujícím způsobem.

```
SetColor("red@0.4");  // 40% průhlednost
SetColor("red@0.9");  // 90% průhlednost
```

3.4.3 Ganttovi grafy (Gantt charts)

Ganttův graf se skládá ze čtyř různých oblastí. V levé části je sloupec popisů aktivit. V horní části je prostor pro hlavičky (mohou být až čtyři). Další částí je samozřejmě místo, pro samotné vykreslení Ganttových proužků (bars) a značek. Poslední oblastí je okraj grafu, kde se zobrazují např. nadpisy.

Ganttovův graf je tvořen z následujících objektů: Ganttův proužek (značí délku aktivity), milníky (značka v určitém datu) a vertikální čára, která může být použita např. k vyznačení různých fází projektu.

Všechny tyto objekty mohou být všemožně formátovány. Je možné specifikovat barvu, velikost, popisy, styly, výplně apod.

3.4.4 Jednoduchý Ganttův graf

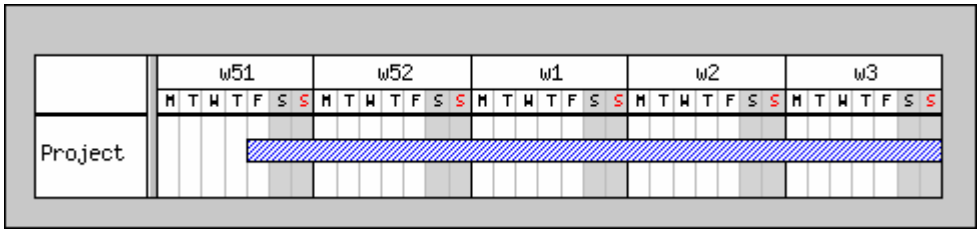
Tento jednoduchý příklad ukazuje použití knihovny JpGraph. V grafu budeme zobrazovat jednu aktivitu, pod jménem „Project“, která trvala od 2001-12-21 do 2002-01-20.

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

// Nový graf s automatickou velikostí
$graph  = new GanttGraph (0,0, "auto");

//  Nová aktivita na řádku 0
$activity  = new GanttBar (0,"Project", "2001-12-21", "2002-01-20");
$graph->Add( $activity);

// Zobraz graf
$graph->Stroke();
?>
```



Obrázek 3.13: Jednoduchý příklad Ganttova grafu

Z příkladu vidíme, že všechny použité objekty mají svoje standardní hodnoty (jako je barva, styl písma, apod.) a proto je není nutné nastavovat.

3.4.5 Složitější Ganttův graf (princip vytvoření mapy studií)

V této kapitole popíši většinu metod JpGraphu, které bylo nutno použít při vytváření grafu mapy studií. Budu zde postupně ukazovat skript, který takový graf vytvoří. Některé metody budou popsány přímo ve zdrojovém kódu, o jiných se rozepíši více. Ukáži zjednodušené vykreslení jednoho studia.

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");
// Nový graf o velikosti 800px na šířku
```

```
$graph = new GanttGraph('800', 0, 'auto');

// V hlavičce budu zobrazovat pouze roky
$graph->ShowHeaders(GANTT_HYEAR);
// Nadpis pro jména aktivit
$graph->scale->actinfo->SetColTitles(array('Fakulta'));
```

Nyní vytvořím aktivitu (proužek) pro část studia.

```
$bar = new GanttBar(0, 'FSI VUT', '2004-09-01', '2005-09-01');
// nastavím typ a barvu výplně pro toto studium
$bar->SetPattern(GANTT_SOLID, "#c59b2a", 100);
```

Pro proužek nastavím URL odkaz (v reálu je mnohem složitější, viz kapitola 3.3.2.3)

```
$bar->SetCSIMTarget('#detaily', 'Detaily');
// přidáme proužek do grafu
$graph->add($bar);
```

Dále vytvořím další proužek, který odpovídá další části studia. Vytvořím ho na stejném řádku jako proužek předchozí.

```
$bar = new GanttBar(0, 'FSI VUT', '2005-09-01', '2006-09-01');
$bar->SetPattern(GANTT_SOLID, "#c59b2a", 100);
$bar->SetCSIMTarget('#detaily', 'Detaily');
$graph->add($bar);
```

Nyní vytvořím značku označující nějaký typ ukončení studia. Značka pro označení data, do kdy je studium bez poplatku se vytváří obdobným způsobem.

```
$ms = new MileStone(0, '', '2006-09-01');
// tvar značky
$ms->mark->SetType(MARK_FILLEDCIRCLE);
// vybarvení značky
$ms->mark->SetFillColor("#c59b2a");
// barva písma ve značce
$ms->mark->title->SetColor("white");
// barva obrysu značky
$ms->mark->SetColor("black");
// text ve značce (tento označuje neúspěšné ukončení)
$ms->mark->title->Set("x");
// obdobně jako u proužků vytvoří URL odkaz značky
$ms->SetCSIMTarget('#detaily', 'Detaily');
// přidání značky do grafu
```

```
$graph->Add($ms);
```

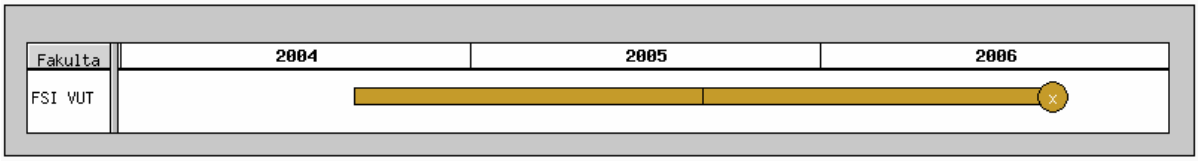
Kdybych takovýto graf vykreslil, nastaví se rozsah grafu automaticky, tj. začínal by prvním použitým datem (2005-09-01) a končil posledním (2006-09-01). Pro zpřehlednění grafu je ovšem lepší zobrazovat celé roky.

```
$graph->SetDateRange('2004-01-01', '2006-12-31');
```

Nyní graf uložíme do souboru a vytvoříme z něj klikací mapu (vytvoří se mapa, ve které lze klikat na objekty, které mají nastavenou URL adresu).

```
$graph->Stroke('graf.png');
$graph->StrokeCSIM('graf.png');
?>
```

Na následujícím obrázku je vidět výstup ze skriptu. Je vidět, že se už dosti podobá mapě studií, i když v ní je použito ještě více různých formátování grafu.



Obrázek 3.14: Složitější příklad Ganttova grafu

4 Grafický prohlížeč mapy studijních programů

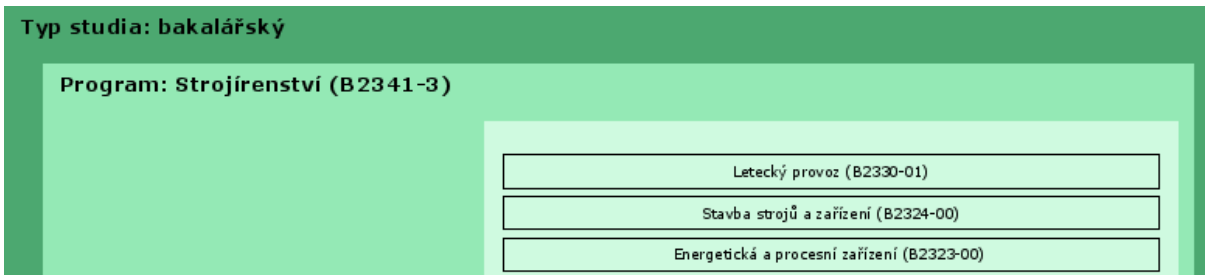
4.1 Úvod

Tento prohlížeč je jako předchozí prohlížeč studií, určen pro všechny studenty VUT. Cílem prohlížeče je v přehledné formě ukázat studentovi mapu návaznosti jednotlivých studijních programů, oborů a zaměření pro zvolenou fakultu VUT. Grafické zobrazení návaznosti program, či oborů má význam převážně pro fakulty s komplikovaným systémem studia, jako je například Fakulta strojního inženýrství.

4.2 Návrh prohlížeče

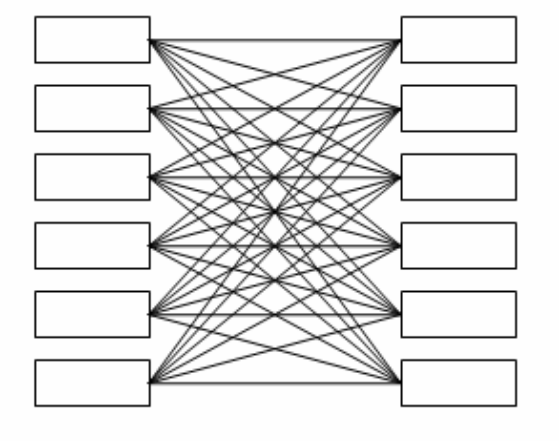
Grafický prohlížeč mapy studijních programů je určen pro webový portál VUT, takže k němu bude mít přístup každý student Vysokého učení technického v Brně. Názornou formou ukazuje možnosti studia studenta na zvolené fakultě VUT. Vzhledem k tomu, že staré magisterské programy již pomalu a jistě dobíhají, zobrazuje pouze programy a k nim náležící obory bakalářské a magisterské navazující.

Každý program, či obor je zařazen do odpovídajícího bloku programů, či oborů, které spolu souvisejí. To znamená, že programy jsou zařazovány do dvou bloků (bakalářský a magisterský navazující) a samy tvoří další bloky. Do těchto bloků jsou zařazovány obory, podle toho, do kterého programu náleží.



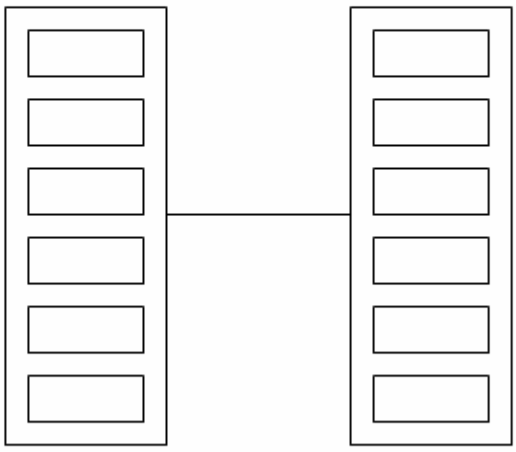
Obrázek 4.1: Bloky programů a oborů

Obory pak tvoří další bloky se společnými rysy. Tyto bloky oborů jsou vytvořeny z důvodu zpřehlednění celé mapy studijních programů. Obory jsou totiž čarami spojovány s jinými obory. Tyto čáry představují možnosti přestupu mezi jednotlivými obory. Představme si, že bychom ke každému oboru kreslili čáru k oboru, na který se dá z tohoto oboru přestoupit. U několika málo oborů to zas takový problém není, ale s větším počtem oborů to už začíná problém být. Vezměme kupříkladu situaci, kdy máme šest bakalářských oborů a z jakéhokoliv takového oboru lze přestoupit na jiných šest oborů magisterských. Tímto postupem získáme 36 čar (6x6), což je velice nepřehledné (viz obrázek 4.2).



Obrázek 4.2: Skupiny oborů 1

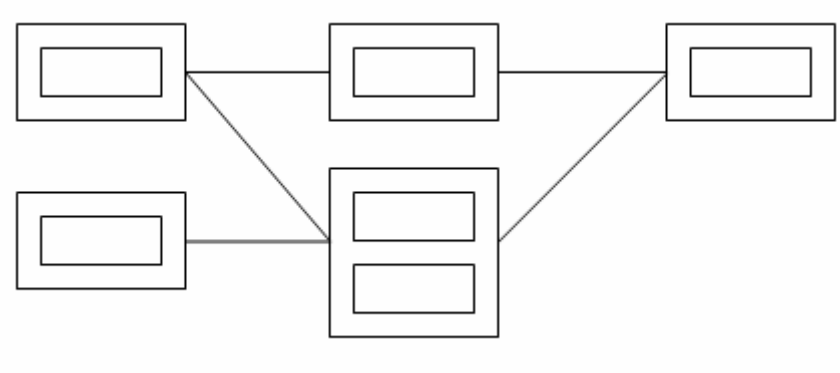
Z tohoto důvodu je lepší obory, které mají stejné obory předcházející i následující (z hlediska přechodu mezi jednotlivými obory) zařadit do stejné skupiny (bloku), čímž bychom například v předchozím případě mohli místo 36 čar vykreslit čáru jednu (toto ukazuje obrázek 4.3). Tato čára pak signalizuje možnost přestupu mezi jakýmkoliv oborem z první skupiny oborů do jakéhokoliv oboru ze skupiny druhé.



Obrázek 4.3: Skupiny oborů 2

Schválně jsem napsal, že se skupiny oborů vytvářejí nejen podle oborů, na které lze z těchto oborů přestoupit, ale i podle oborů předcházejících. Je to z prostého důvodu. Na některých fakultách totiž existují tzv. všeobecné obory, či obory společné například pro první dva roky. Potom jsou přestupy mezi obory možné nejen v rámci přechodu mezi bakalářským a magisterským studiem, ale i v rámci jednoho typu studia (např. bakalářského). Takových všeobecných oborů v rámci jedné fakulty může být samozřejmě víc (a také je, například na Fakultě strojního inženýrství) a ne vždy je možné z těchto oborů přestoupit na všechny obory následující (ne-všeobecné). Proto je nutné vytvářet skupiny oborů i podle toho, ze kterých oborů se na ně dá přestoupit.

Všimněme si následujícího obrázku. První dvě skupiny představují dva všeobecné obory. Nemají žádné předchozí, ale následující obory mají různé, proto je každý obor ve zvláštní skupině. Uprostřed obrázku jsou tři další obory, rozdělené do dvou skupin. Pokud bychom obory zařazovali do skupin jen podle následujících oborů, vytvořili by tyto obory pouze jednu skupiny (protože následující obor mají společný), což by bylo chybné.



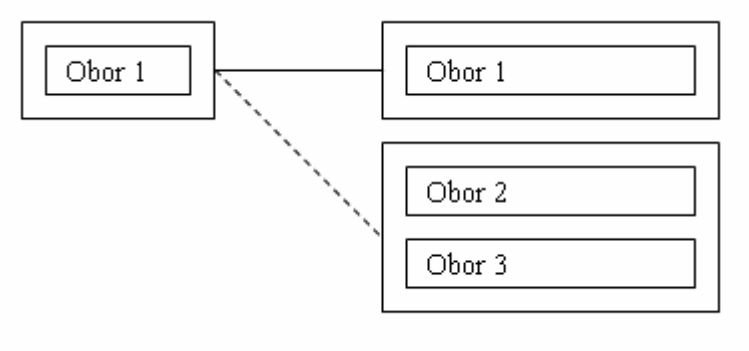
Obrázek 4.4: Skupiny oborů 3

Jak jsem již zmínil jednotlivé návaznosti mezi obory (programy) jsou tvořeny čarami, respektive šipkami. U těchto šipek je vhodné zobrazit i podmínku, kterou je nutné pro přestup splnit. Některé obory totiž vyžadují například talentovou zkoušku apod..

U vykreslování oborů je také nutné zajistit, aby uživatel z grafu poznal, jak dlouho se daný obor studuje. Toto je vhodné zajistit tím, že budeme vykreslovat horizontální mřížku, která bude představovat jednotlivé ročníky studia nebo zajistíme, aby šířky jednotlivých obdélníků, do kterých obory vykreslujeme měli takové šířky, aby z nich bylo na první pohled patrné, jak dlouho se obor studuje.

Na Fakultě strojního inženýrství jsem se setkal také s tím, že existují obory (třeba tříleté), ze kterých se po prvním ročníku dá přestoupit na jiné specializované obory. Takové obory je pak nutno rozdělit na dvě části, abychom tuto možnost mohli v grafu naznačit. To znamená, že tento obor rozdělíme na jeho první část (kde je pouze první ročník) a druhou část, kde jsou zbývající dva.

Přestup mezi takto rozděleným oborem je vhodné nějakým způsobem označit jako povinný a přestup na jiné obory po prvním ročníku tohoto oboru jako nepovinné (viz následující obrázek).

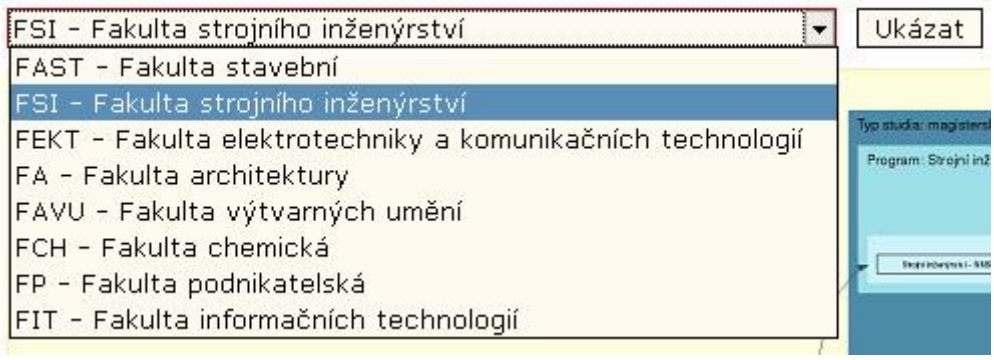


Obrázek 4.5: Skupiny oborů 4

4.3 Celkový pohled na modul

Celkový náhled na modul (pro Fakultu strojního inženýrství) jsem s ohledem na jeho velikost umístil do příloh (Příloha 2: Celkový pohled na mapu studijních programů).

Jak jsem se již zmínil, mapa studijních programů zobrazuje návaznosti programů a oborů na všech fakultách VUT. Fakulta, pro kterou se má mapa zobrazit, lze vybrat v záhlaví modulu. Momentálně je ovšem plně funkční pouze pro Fakultu strojního inženýrství, kde jsou návaznosti jednotlivých oborů asi nejkomplikovanější. Důvody, proč mapa není plně funkční na ostatních fakultách uvedu v kapitole 4.4.3.



Obrázek 4.6: Výběr fakulty

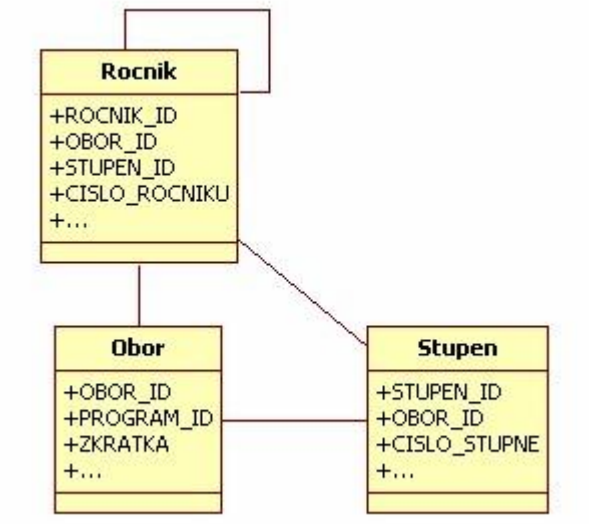
5 Implementace mapy studijních programů

5.1 Výběr dat z databáze

Jako první je nutno vybrat všechny fakulty na VUT (tabulka mv_soucasti_vut). Tyto fakulty jsou pak zobrazeny v hlavičce grafu a je možno se mezi nimi přepínat (zobrazovat pro ně jednotlivé mapy studijních programů). Standardně je vybrána Fakulta strojního inženýrství (pro ni je modul odladěn).

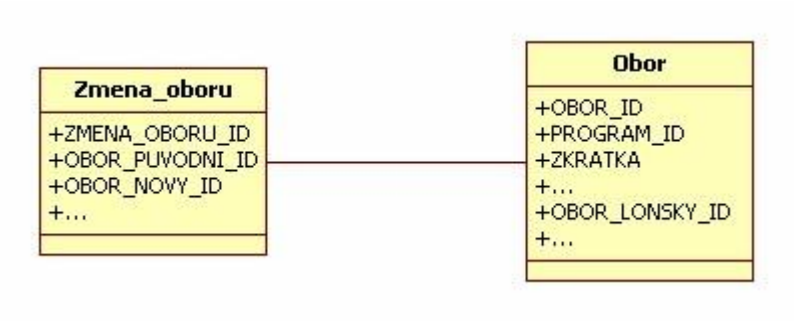
Podle vybrané fakulty se pak vyberou z databáze všechny obory na této fakultě. Jsou vybírány jen ty obory, které patří do programů platných pro aktuální rok. Dále jsou vybírány pouze bakalářské a navazující magisterské obory, resp. programy (mapa by mohla být doplněna i o obory doktorské). O oborech z databáze zjišťujeme mnoho informací, jednou z nejdůležitějších (když pomineme název, zkratku apod.) je délka studia tohoto oboru.

U oboru je v databázi uvedena jeho délka studia, ovšem tato není přesně ta, kterou potřebujeme ke správnému vykreslení mapy. Tato délka totiž ukazuje skutečnou délku studia oboru (tzn. i s možnými předchozími přestupy). Můžeme si to představit tak, že máme všeobecný obor (jednoletý) a za ním následuje obor navazující (dvouletý). U tohoto následujícího oboru bude však v databázi uložena délka studia 3 roky (1+2). Délku studia proto zjišťujeme tak, že spočítáme, kolik ročníků k danému oboru patří, viz obrázek 4.7.



Obrázek 4.7: Vazba mezi oborem, ročníkem a stupněm studia

Když jsou vybrány všechny obory na příslušné fakultě, můžeme začít zjišťovat návaznosti mezi nimi. K tomuto slouží tabulka „st01.zmena_oboru“, jejíž svázanost s tabulkou „st01.obor“ je vidět na obrázku 4.8.



Obrázek 4.8: Vazba mezi oborem a změnou oboru

Jak jsem zmínil v kapitole 4.2 je nutné zjišťovat ke každému oboru obory následující, ale i předcházející. Pokud chceme k určitému oboru zjistit obory, které za ním následují, podíváme se do tabulky „st01.obor“ na sloupec „obor_lonsky_id“. Tento identifikátor odpovídá sloupci „obor_puvodni_id“ z tabulky „st01.zmena_oboru“, takže můžeme zjistit všechny obory, na které se dá z tohoto oboru přestoupit („obor_novy_id).

Obdobným způsobem lze zjistit, které obory danému oboru předcházejí. Vybereme všechny řádky tabulky „st01.zmena_oboru“, kde sloupec „obor_novy_id“ odpovídá identifikátoru oboru, pro který předcházející obory hledáme. Tímto si ze sloupce „obor_puvodni_id“ zjistíme „obor_lonsky_id“ z tabulky „st01.obor“ a tím získáme předchozí obory k danému oboru.

Když máme vybrané všechny obory a k nim všechny předchozí a následující, tak začneme vytvářet skupiny oborů, o kterých jsem mluvil v kapitole 4.2. To znamená, že obory zařazujeme do skupin právě podle toho, jaké mají předchozí a následující obory.

5.2 Vykreslení mapy studijních programů

Pro vykreslení mapy studijních programů, používám grafický nástroj, pro generování grafů pod názvem Graphviz. Tento grafický nástroj se mi zdál velice vhodný pro vykreslení mapy návaznosti programů, protože velice dobře generuje stromové grafy. Bohužel má své mouchy a mapa studijních programů je sice funkční, ale přece jen její grafické znázornění není přesně takové, jak jsem si ho představoval. Uvažoval jsem, že vytvořím vlastní framework (nástroj), pro generování jednoduchých stromových grafů v jazyce PHP tak, aby přesně vyhovoval mým požadavkům, které jsem měl na mapu studijních programů. Snažil jsem se tedy alespoň co nejlépe odladit zobrazování právě pomocí grafického nástroje Graphviz.

Vzhledem k tomu, že jsem pro Graphviz nenašel žádné pořádné rozšíření pro PHP, generuji grafy tak, že si v PHP skriptu vytvořím zdrojový soubor Graphvizu, ve kterém je celý graf popsán. Z něho posléze vygeneruji graf do obrázku (PNG). Generování probíhá pomocí rozšíření „Image_GraphViz“, které umí volat přímo unixové příkazy, pomocí kterých lze s nástrojem Graphviz také pracovat.

Samotné vykreslení mapy studijních programů pak probíhá tak, že se postupně prochází veškeré vytvořené skupiny oborů. Ty se zařazují do skupin (clusterů) podle typu studia a programu. V každé skupině oborů se následně projdou všechny obory, které do ní patří a vykreslí se. Když jsou všechny obory vykresleny, musí se ještě mezi s sebou pospojovat čarami (představujícími možnosti postupu mezi nimi).

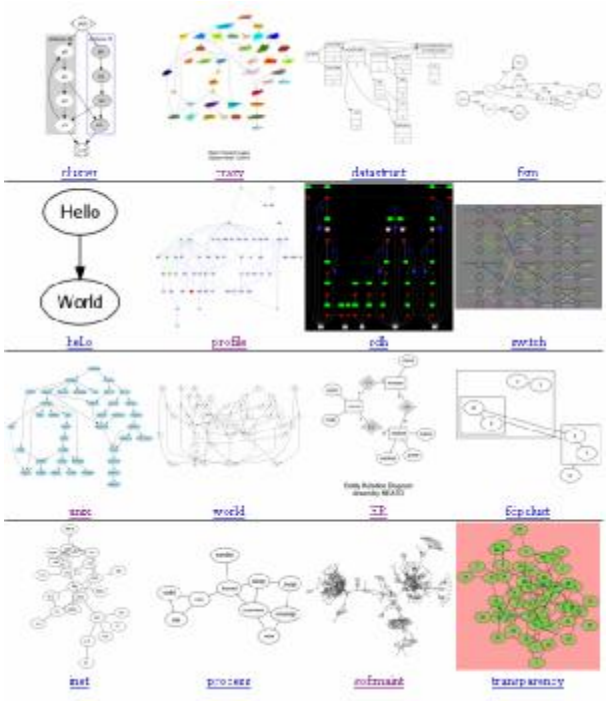
5.3 Problémy při implementaci

Jak jsem již v kapitole 4.3 zmínil, mapa studijních programů je plně funkční pouze pro Fakultu strojního inženýrství. Je to z toho důvodu, že v databázi celoškolského systému (ST01) nejsou uvedeny veškeré vazby mezi obory. Chybí totiž vazby mezi bakalářskými a magisterskými obory. Je to z toho důvodu, že pokud student dokončí bakalářské studium a pak pokračuje ve studiu v magisterském navazujícím, tak se mu pro něj zakládá v databázi nové studium. Z tohoto důvodu není nutné mít v databázi informaci o tom, že je z nějakého bakalářského oboru možno přestoupit na některý magisterský. Pro Fakultu strojního inženýrství jsem v testovací databázi tyto vazby mezi bakalářskými a magisterskými obory doplnil, abych mohl prezentovat funkčnost modulu alespoň na jedné fakultě. Zároveň vazby mezi obory jsou na této fakultě asi nejsložitější, takže by po doplnění potřebncých informací i u jiných fakult, neměl být problém s funkčností celého modulu.

Dalším problémem, který jsem uvedl v předchozí kapitole 4.4.2 je samotný nástroj Graphviz, který jsem použil pro vykreslování mapy studijních programů. Jak jsem uvedl, grafické znázornění pomocí tohoto nástroje není přesně takové, jak jsem si ho představoval, ale jiný vhodný nástroj pro generování grafů jsem nenašel. Také v tomto nástroji nakonec nebylo možné udělat JavaScriptové nápovědy podobně, jako u mapy studií, což je velká škoda.

5.4 Graphviz

Graphviz je nástroj pro generování grafů, podobně jako JpGraph, o kterém jsem psal v předchozí kapitole. Graphviz je ale oproti předešlému vhodný pro grafy, které mají stromovou strukturu. Graphviz je program pro vizualizaci grafů. Obsahuje několik základních typů grafů, jak můžete vidět na následujícím obrázku.



Obrázek 5.1: Ukázky grafů (Graphviz)

Graphviz získává popis grafu z jednoduchých textových souborů. Z nich pak vytváří grafy do několika užitečných formátů, jako jsou obrázky a SVG pro webové stránky, Postscript pro vkládání do PDF dokumentů nebo je dokáže zobrazit v interaktivním prohlížeči grafů.

Je několik možností, jak generovat grafy za pomoci Graphviz. Za pomoci „dot“ lze vytvářet hierarchické nebo vrstvené orientované grafy. Vykreslovací algoritmus kreslí hrany stejným směrem (z vrchu dolů, z leva doprava), dále se snaží předejít křížení hran a redukovat jejich délku. Dále je to „neato“ a „fdp“, které se využívají pro vykreslování neorientovaných grafů. Nakonec ještě „twopi“ a „circo“ pro generování grafů kruhových, či cilkických.

5.4.1 Kreslení grafů pomocí „dot“

Dot kreslí hierarchické orientované grafy. Lze spouštět jako program příkazové řádky, vizualizační služba pro web nebo s kompatibilním grafickým rozhraním. Mezi jeho vlastnosti patří využívání propracovaného kreslícího algoritmu pro umisťování uzlů a hran (křivek), popisů hran, tvarů typu „záznam“ pro kreslení datových struktur, a mnoho dalších.

Dot čte atributy grafu z textových souborů a zapisuje vykreslené grafy do různých formátů jako je GIF, PNG, SVG nebo Postskript. Textové soubory, ve kterých jsou popsané grafy musí být ve formátu jazyka DOT. Tento jazyk popisuje tři základní objekty: graf, uzel a hrana. Hlavní (nejsvrchnější) graf může být orientovaný (directed) nebo neorientovaný (graph). Uvnitř hlavního grafu mohou být definovány podgrafy (subgraph), které mohou obsahovat další skupinu uzlů a hran.

Následující tabulka ukazuje abstraktní gramatiku, která popisuje jazyk DOT. Terminální symboly jsou označeny tučným písmem a nonterminální symboly kurzívou. V jednoduchých uvozovkách jsou uvedeny znaky. Závorky označují spojování, kde je potřeba. Hranaté závorky odznačují volitelné položky. Vertikální čára | odděluje alternativy.

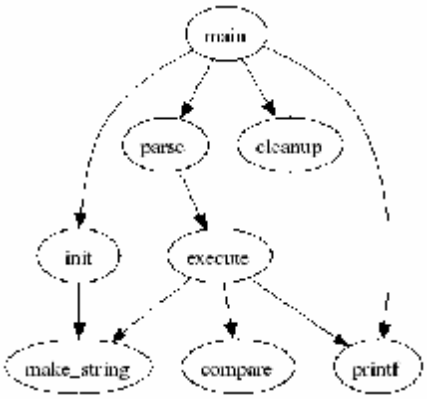
graph : [**strict**] (**graph** | **digraph**) [*ID*] '**{**' *stmt_list* '**}**'
stmt_list : [*stmt* ['**;**'] [*stmt_list*]]
stmt : *node_stmt*
| *edge_stmt*
| *attr_stmt*
| *ID* '=' *ID*
| *subgraph*
attr_stmt : (**graph** | **node** | **edge**) *attr_list*
attr_list : '**[**' [*a_list*] '**]**' [*attr_list*]
a_list : *ID* ['=' *ID*] ['**,**'] [*a_list*]
edge_stmt : (*node_id* | *subgraph*) *edgeRHS* [*attr_list*]
edgeRHS : *edgeop* (*node_id* | *subgraph*) [*edgeRHS*]
node_stmt : *node_id* [*attr_list*]
node_id : *ID* [*port*]
port : '**:**' *ID* ['**:**' *compass_pt*]
| '**:**' *compass_pt*
subgraph : [**subgraph** [*ID*]] '**{**' *stmt_list* '**}**'
| **subgraph** *ID*
compass_pt : (**n** | **ne** | **e** | **se** | **s** | **sw** | **w** | **nw**)

Následující zdrojový kód je ukázkou jazyka DOT. Na prvním řádku je graf pojmenován a je zvolen jeho typ. V následujících řádcích se vytváří uzly a hrany. Uzel je vytvořen, jakmile se poprvé jeho jméno objeví ve zdrojovém souboru. Hrany jsou vytvářeny, pokud jsou dva uzly spojeny dohromady hranovým operátorem „->".

```
digraph G {
      main -> parse -> execute;
      main -> init;
      main -> cleanup;
      execute -> make_string;
      execute -> printf;
      init -> make_string;
      main -> printf;
      execute -> compare;
}
```

Pokud takový zdrojový kód uložíme například do souboru „graph.dot", můžeme ho pak vykreslit za pomoci následujícího příkazu. Pomocí volby –Tps se vybírá jako výstup PostScript. Tento výstup můžeme vidět na obrázku 5.2.

$ dot –Tps graph.dot –o graph.ps



Obrázek 5.2: Kreslení jednoduchého grafu

Existuje mnoho možností, jak formátovat objekty v Graphvizu. Celý graf, případně uzel, či hranu lze popisovat velkým množstvím různých atributů. Základní použití těchto atributů popíši v následující kapitolce.

5.4.1.1 Atributy grafu – uzly

Uzly jsou v grafu standardně vykreslovány s atributy shape=elipse, width=0.75, height=0.5 a označeny názvem uzlu. První atribut shape označuje tvar uzlu. Další možné tvary jsou box, circle, record, plaintext a další. Pokud se při vykreslování zjistí, že velikost uzlu je větší, než je nutné pro vykreslení jeho popisu jsou parametry width (šířka) a height (výška) upraveny tak, aby uzly nebyly zbytečně velké. Funguje to i obráceně, tj. pokud je velikost uzlu nastavena tak, že by se do něj nevešel jeho popis, je automaticky zvětšen.

Toto chování lze samozřejmě zakázat použitím atributu fixedsize=true. Pro popis uzlu se používá atribut label, pokud nechceme, aby byl popis uzlu stejný, jako jeho jméno. Následuje jednoduchý příklad, na kterém si ukážeme použití některých atributů uzlů v grafu.

```
main[shape=box,label=”Hlavní”,weight=10];
```

U uzlů lze také nastavovat jejich styl. Ten se používá ke změně vzhledu uzlu. V současnosti je dostupných 8 stylů jako je filled, invisible, diagonals, rounded. dashed, dotted, solid a bold. Hodnota atributu style může být nejen jedním typem, ale může být seznamem více typů oddělených čárkou.

```
node1[style="dotted, solid"];
```

5.5 Možnosti rozvoje aplikace

5.5.1 Vlastní nástroj pro generování grafů

Jako první by bylo vhodné navrhnout a implementovat vlastní nástroj pro generování mapy studijních programů. Graphviz se nezdá být příliš vhodný, pro další rozvoj aplikace.

V následujícím odstavci se budu snažit popsat návrh takového nástroje. Celá pracovní plocha, kam by se vykreslovaly programy, obory a zaměření by byla rozdělena mřížkou, do které by se jednotlivé objekty umisťovaly. Umisťování uzlů do této mřížky by mohlo vypadat obdobně, jako je to nyní při použití nástroje Graphviz. U vkládaných objektů je nutné zajistit, aby byla možnost tyto objekty popsat i nějakými javascriptovými funkcemi tak, aby se k nim dala jednoduše vytvořit nápověda, protože všechny potřebné informace se do samotného grafu kvůli jeho velikost prostě nevejdou.

Proti Graphvizu je nutné vylepšit vykreslování šipek, představujících možnosti přestupu mezi programy, či obory. Rozhodně je lepší čáry vykreslovat jako lomené čáry, což Graphviz nedokáže. Pokud se ovšem zavede mřížka, o které jsem psal v předešlém odstavci, tak by se vykreslování čar dalo jednoduše vyřešit. Graphviz se snaží jednotlivé čáry vykreslovat tak, aby se nepřekrývaly. Tato vlastnost ovšem při vykreslování mapy studijních programů není vhodná, takže by bylo lepší vykreslovat čáry, které vycházejí z jednoho uzlu co nejdéle jako jednu čáru. Tímto se rozhodně zpřehlední a celkově vylepší vzhled celé aplikace.

Dále je samozřejmou nutností v návrhu nezapomenout na různé měnění barev a stylů jednotlivých objektů apod..

5.5.2 Editace mapy

Celá aplikace mapy studijních programů by se dala rozšířit o možnosti editace. Studentovi by se v mapě programů po otevření zvýraznil jeho dosavadní postup ve studiu. To znamená, že programy a obory, které zatím studoval by byly nějakým vhodným způsobem v mapě zvýrazněny. Možnosti dalšího postupu ve studiu by se také zvýraznily. Takto by student měl jasný přehled o tom, jaké další možnosti studia má.

Ovšem s editací bychom mohli zajít ještě dál. Student by si mohl například vybrat jeden z oborů, který je mu nabízen jako další možnost studia a tím by se mu ukázala další možnost studia i po tomto oboru. Pokud by byl student pro tento obor rozhodnut, mohl by si ho označit k zápisu a mohla by se i automaticky odeslat přihláška k tomuto oboru.

Mapa studijních programů by rovněž mohla být upravena pro zaměstnance školy. Vzhledem k tomu, že jsem si sám vyzkoušel, jak komplikované je zadávat návaznosti jednotlivých oborů do databáze ručně, mohla by se mapa studijních programů rozšířit o přidávání vazeb mezi obory. Tyto editace by mohli provádět zaměstnanci jednotlivých fakult. Uživatel by prostě „natáhnul“ čáry mezi jednotlivými obory tak, jak je možné mezi nimi přestupovat a tyto informace by se uložili do databáze. Samozřejmě by mohly vznikat problémy se špatně vytvořenými vazbami, proto by bylo potřeba ošetřit i mazání takto špatně vytvořených vazeb z databáze.

6 Závěr

Cílem této diplomové práce bylo navrhnout a implementovat dva moduly pro webový portál Centra výpočetních a informačních služeb Vysokého učení technického v Brně. Jsou to moduly grafický prohlížeč studií studenta a grafický prohlížeč mapy studijních programů na VUT. Jak se dá z názvu těchto modulů poznat, je jejich cílem poskytovat co nejvíce informací uživateli (studentovi) v grafické podobě tak, aby byly pro něj přehledné a jasné.

Před samotným započetím práce na těchto modulech bylo potřeba prostudovat analýzu a ER diagramy datového schématu ST01 celoškolského studijního systému a pravidel studování na fakultách VUT. Samozřejmě nebylo nutné prostudovat schéma systému ST01 celé, ale pouze jeho části týkající se studií. Tato část ER diagramu (i když ne celá) je přiložena v kapitole přílohy, takže si každý, kdo tuto práci bude číst může udělat alespoň částečnou představu o schématu systému ST01.

V první kapitole této práce se zabývám pojmem informační systém (IS), protože webový portál VUT je příkladem tohoto pojmu. Informačním systémem rozumíme počítačový firemní systém pro sběr, zpracování a prezentaci informací a dat. V této kapitole se tedy zabývám vlastnostmi a cíli informačního systému. Dále jsou rozebrány postupy, používané při návrhu IS, jako je např. modelování systému v jazyce UML. Nakonec jsou také v této kapitole prozkoumány programovací jazyky, pomocí kterých je webový portál VUT implementován, tudíž jejich znalost byla nutná i pro implementaci modulů, která jsem vytvářel.

Ve druhé kapitole se zabývám návrhem prvního modulu, tj. grafického prohlížeče studií studenta. Tento modul je navržen jako náhrada za stávající modul Studium, který uživateli ukazuje jeho dosavadní studia. Modul Studium zobrazuje pouze studia na VUT a ty jsou zobrazovány do tabulek. Oproti tomu modul Mapa studií, který je v této kapitole popsán, zobrazuje studia nejen na fakultách VUT, ale i na jiných školách. Další výhodou je, že studia nejsou v tomto modulu zobrazovány pouze do tabulek, ale jsou prezentována grafickou podobou. Tento způsob prezentace má nespornou výhodu např. při udělování různých typů stipendií, protože podmínky těchto stipendií bývají úzce svázány s kombinací předešlých a aktivních studií a tato návaznost je z grafu Mapy studií jasně viditelná a přehledná. Díky tomuto modulu může uživatel také jasně vidět, kdy dojde k financování aktivního studia „z vlastní kapsy“. Využití modulu je tedy široké a nic nebrání jeho reálnému nasazení na ostrou verzi portálu VUT.

Kapitola třetí je zaměřena na implementaci modulu Mapa studií, od postupů, které je nutno dodržet při vytvoření nového modulu na portálu VUT, přes vlastní vykreslení mapy, až po popis grafického nástroje JpGraph, který byl při implementaci použit pro generování grafů.

V kapitole čtvrté se zabývám návrhem druhého implementovaného modulu, tj. grafického prohlížeče mapy studijních programů na VUT. Tento modul je zvláště významný pro ty fakulty VUT,

které mají velké množství různých oborů, mezi kterými je možno přestupovat. Grafická prezentace těchto možností přestupu mezi obory pak uživateli dá jasnou představu o tom, co může na dané fakultě studovat a co pro to musí splnit. V této kapitole jsou popsány teoretické základy principu zobrazení mapy studijních programů, které byly využity při implementaci.

Kapitola pátá se pak zabývá samotnou implementací mapy studijních programů na VUT. Je v ní také popsán nástroj Graphviz, který byl pro generování mapy při implementaci použit. Součástí této kapitoly jsou také návrhy na možné vylepšení celého modulu o nové vlastnosti.

Závěrem bych chtěl říci, že modul Mapa studií momentálně čeká na schválení ze strany softwarových integrátorů jednotlivých fakult, kteří modul testují. V nejbližší době by mohlo být testování dokončeno a modul by se měl objevit na ostré verzi portálu VUT. Toto považuji za velký úspěch a jsem rád, že moje diplomová práce bude v budoucnu sloužit studentům k získávání informací o jejich studiích.

Vzhledem k tomu, že databázový systém VUT momentálně není připraven na nasazení druhého modulu, tj. Mapy studijních programů, jeho nasazení do všedního provozu není v tuto chvíli možné. Je to škoda, ale snad někdo v budoucnu na moji práci naváže a odladí tento modul tak, že se databázové schéma upraví, aby se modul mohl používat v praxi.

Literatura

[1] Gutmans, A., Bakken, S., Rethans, D. *Mistrovství v PHP*. CP Books, 1, 2005.

[2] Zendulka, J., *Databázové systémy (materiály k přednáškám)*

[3] Mišovič, M., *Informační systémy - projektování (materiály k přednáškám)*

[4] http://www.php.net, PHP, Hypertext Preprocesor

[5] http://en.wikipedia.org, Wikipedia, The free encyclopedia,

[6] http://www.aditus.nu/jpgraph/, JpGraph, PHP Graph Creating Library

[7] http://www.graphviz.org/, Graphviz, Graph Visualization Software

Užitečné odkazy

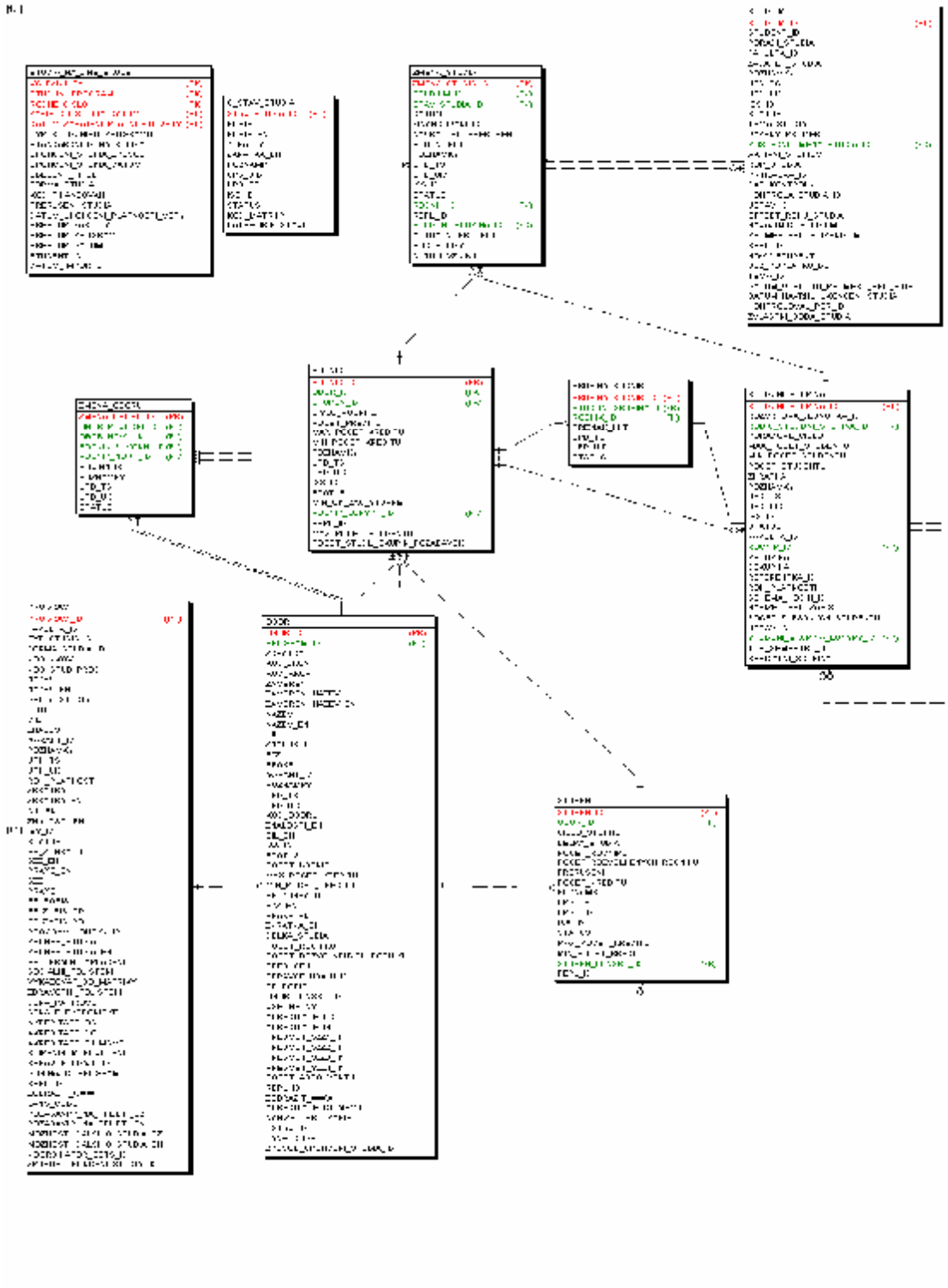
[1] https://shreckdog.ro.vutbr.cz/studis/student.phtml?script_name=mapa_studii, Mapa studií (volně přístupné pro studenty VUT).

[2] https://3wtest.ro.vutbr.cz/brett/studis/student.phtml?script_name=mapa_programu, Mapa studijních programů (není volně přístupné, musí se povolit IP adresa uživatele).

Seznam obrázků

Přílohy

Příloha 1: Část diagramu tříd ST01 celoškolského systému (studijní programy)



Příloha 2: Celkový pohled na mapu studijních programů