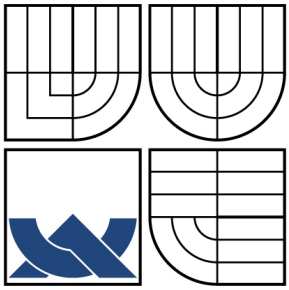
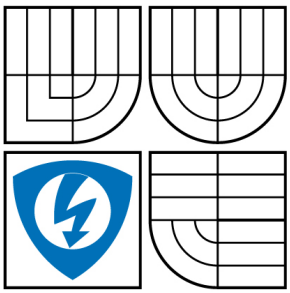


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

DATABÁZOVÝ SYSTÉM PRO VÝROBU DESEK PLOŠNÝCH SPOJŮ

DATABASE SYSTEM FOR PRINTED CIRCUIT BOARD PRODUCTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

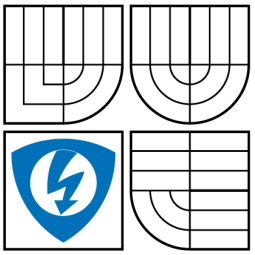
AUTOR PRÁCE — Bc. JIŘÍ PROCHÁZKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK ŠEBELA

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Jiří Procházka	***ID:***	89772
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Databázový systém pro výrobu desek plošných spojů

POKYNY PRO VYPRACOVÁNÍ:

Zjistěte parametry, které jsou potřebné pro výrobu desek plošných spojů (rozměry desky, popis, platba, obrázek). Prostudujte nástroje pro provoz webové aplikace (Apache, MySQL, PHP, PhpMyAdmin). Prostudujte zásady navrhování tabulek v databázích a case nástroje pro design databází. Prostudujte binární ukládání dat do databáze MySQL. Prostudujte bezpečnostní opatření databazí (SQL Injection, Non-String). Navrhněte strukturu databáze pro systém výroby plošných spojů.

Vytvořte uživatelsky přívětivý webový systém pro správu výroby desek plošných spojů. Systém bude využívat různá oprávnění: studenti mají práva pouze zobrazovat "stav zakázky", vyučující mají práva zadávat (rušit) zakázky a rozhodovat o placení, dílna má oprávnění zadávat vyhotovení zakázky.

Systém realizujte pomocí pluginů (snadné rozšíření dalších funkcí). Optimalizujte systém proti různým možným útokům (SQL Injection, Non-String). Optimalizujte stránky pro standard W3C.

DOPORUČENÁ LITERATURA:

[1] http://www.apache.org
[2] http://www.cz.php.net
[3] http://www.mysql.org
[4] http://www.phpmyadmin.net

Termín zadání:	9.2.2009	***Termín odevzdání:***	29.5.2009

Vedoucí práce: Ing. Radek Šebela

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

Abstrakt:

Rozbor problematiky návrhu webové databáze a její struktury pro potřeby provozu dílny na výrobu plošných spojů. Prostudování použití aplikaci určených pro vytváření, správu a zabezpečení dat databázového systému. Prostředky pro provoz webového serveru. Zásady navrhování tabulek v databázích. Konkrétní návrh struktury databáze pro systém výroby desek plošných spojů.

Abstract:

Analysis of problems during designing web database and their structure for requirement of manufacturer of PCBs. Study of using applications intended for creation, administration and protection of database system. Systems for operation of web server. Principles of projection tables in databases. Design of database's structure for production system of PCBs.

Klíčová slova:

webová databáze, provoz webového serveru, zásady návrhu tabulek

Keywords:

web database, operation of web server, principles of projection tables

PROCHÁZKA, J. *Databázový systém pro výrobu desek plošných spojů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 58 s. Vedoucí diplomové práce Ing. Radek Šebela.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Databázový systém pro výrobu desek plošných spojů jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 29. května 2009 .....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Radku Šebelovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 29. května 2009 .....
podpis autora

OBSAH

1 Úvod 8
2 Http server Apache 9
3 Vlastní databázový systém 10
3.1 Zásady návrhu tabulek v relačních databázích 10
3.1.1 Typy relací 11
3.1.2 Normalizace 12
4 Jazyk SQL 14
5 MySQL 15
5.1 Typy tabulek v MySQL 15
5.1.1 Typ ISAM 15
5.1.2 Typ MyISAM 15
5.1.3 Typ Merge 15
5.1.4 Typ InnoDB 16
5.1.5 Typ BDB 16
5.1.6 Typ Memory 16
5.1.7 Typ Federated 16
5.1.8 Typ CSV 16
5.2 Ukládání binárních dat do databáze MySQL 16
6 PHP 17
6.1 Zabezpečení databáze 18
6.1.1 Na straně databáze 18
6.1.2 Na straně webové aplikace 18
7 PhpMyAdmin 20
8 Case nástroje pro design databází 21
8.1 Case Studio 22
9 Rozbor následného vývoje systému 23
9.1 Typy uživatelů 23
9.1.1 Student 23
9.1.2 Zaměstnanec 24
9.1.3 Dílna 24
9.2 Stav objednávky 24
9.3 Návrh databáze 25
10 Návrh obslužných skriptů PHP 28
10.1 Návrh základních souborů 28
10.2 Návrh použité struktury 28
10.3 Grafická šablona 30

10.4 Přihlašovací formulář ... 32
10.5 Řešení přihlašování ... 32
10.6 Řešení odhlašování ... 34
10.7 Zadávací formulář ... 34
10.7.1 Zpracování odeslaných dat ... 36
10.8 Výpis objednávek ... 37
10.9 Stránka nastavení ... 39
10.10 Vytvoření tabulkové struktury databáze ... 40
11 Bezproblémový chod databázové aplikace ... 41
12 Závěr ... 44
13 Seznam použité literatury ... 45
14 Seznam obrázků a tabulek ... 46
15 Seznam zkratek ... 47
16 Seznam příloh ... 48

1 Úvod

Databázové systémy a aplikace s nimi spojené jsou dnes již základem většiny dynamických www stránek. Dávají administrátorům velkou škálu možností, jak přehledně pracovat s velkým množstvím dat a jak z nich vyhledávat potřebné informace nebo generovat stránky co nejvíce se přizpůsobujícím uživatelovým potřebám. Vše, co je dnes spojeno s počítači, internetem a informačními technologiemi vůbec zažívá veliký rozvoj. Webové aplikace dnes usnadňují život svým uživatelům, šetří jejich čas, a především jsou dostupné z kteréhokoliv místa na světě, které je připojené k internetu

Tato diplomová práce si klade za cíl prozkoumat možnosti v oblasti webových databázových aplikací, skriptovacích jazyků i zabezpečení takovýchto systémů. Dle teoretických znalostí bude navržena struktura databázového serveru spolupracující s webovou aplikací. Jako výsledek by měl být realizovaný obslužný program s funkční databází, systém by měl být realizován pomocí pluginů, a zabezpečen proti různým druhům útoků.

2 Http server Apache

První verze Apache byl napsán Robem McCoolem, tehdy nesl server jméno NCSA HTTPd. Vývoj samotného Apache započal během roku 1994. Jako základ je použita poslední verze NCSA httpd serveru, a to verze 1.3. K ní se přidali všechny publikované opravy chyb a rozšíření a v dubnu 1995 byla vydána první verze Apache Web Serveru (0.6.2). (samotný název byl odvozen právě od množství použitých záplat – záplatovaný – patched)

Apache si okamžitě získal velikou oblibu a vývojáři se rozhodli, že pro další vývoj serveru potřebuje Apache přepsat od základu. Během dalších měsíců byla navržena zbrusu nová architektura serveru, který měl označení Shambhala.

Ve verzi 2 byla značná část zdrojových kódů Apache verze jedna přepsána s velkým zaměřením na modernizaci a vývoj jeho vrstvy přenosnosti – Apache portability runtime. Jádro Apache 2 má několik významných odlišností od jádra Apache 1, jako například UNIX threading, lepší podporu pro ne-Unixové platformy jako je MS Windows.

Server jak takový je řešen jako modulární (jádro + řada modulů zajišťujících různé akce), lze jej prakticky libovolně rozšiřovat. Na Internetu jsou dostupné ke stažení moduly pro různé druhy zabezpečené komunikace, pro různé druhy souborů, pro administraci, výpisy stavu serveru přes www atd.

Server Apache se stal velice populární zejména pro své volné šíření, nezávislosti na platformě a snadné konfigurovatelnosti. Například oblíbený vyhledávač Google má svůj engine založený na upravené verzi Apache.

Hlavní rysy serveru:

- HTTP 1.1 web server
- Rozsáhlé možnosti konfigurovatelnosti při zachování jednoduchosti
- rozšiřitelnost (moduly třetích stran)
- kompletní zdrojový kód k dispozici
- běží na mnoha platformách, Unix, MS
- restrikce dle adresářů, domén, hesel, individuální omezení uživatelských stránek jednotlivými uživateli - oddělení obsahu serveru od ostatních částí systému
- zavádění tzv. alias - přezdívek adresářů
- virtuální servery
- Integrovaná proxy

Nové funkce od verze 2.0.

- nové API **o**bsahuje např. zlepšení podpory modulů - jejich řazení při natahování se již provádí zcela automaticky, byla přidána nová volání bez nutnosti patchovat jádro Apache
- podpora IPv6
- podpora Unicode na Windows NT platformě - podpora UTF-8 pro jména souborů

3 Vlastní databázový systém

Pojem databáze je nejčastěji spojován pouze se souborem jakýchsi dat, která máme někde fyzicky uložené. Pravý význam slova databáze může být poněkud obsáhlejší, označuje totiž jak soubor dat uložených na nějakém záznamovém médiu tzv. datovou základnu, tak i celý systém sloužící k obsluze a přístupu k datům tzv. systému řízení báze dat neboli SŘBD tento název vznikl překladem anglického názvu DBMS - database management systém. Spojením obou částí získáme DBS neboli databázový systém.

SŘBD - systém řízení báze dat lze chápat jako souhrn procedur a datových soustav, které zajišťují nezávislost databázové aplikace na detailech vytváření, výběru, uchování, modifikaci a zabezpečení ochrany databází na fyzických paměťových strukturách počítače [1]. Příkladem takového systému je třeba Oracle, MS SQL Server nebo MySQL

Datová základna je vlastně soubor všech informací a dat v databázovém systému uložených. Ukládá se nejčastěji formou záznamů neboli entit, což je souhrn údajů (atributů) o určité části popisovaného objektu. Každý atribut je uložen v samostatné položce (poli, field), které je charakteristické především svým názvem a datovým typem.

Datový typ je popis dané položky, kterým se předem sdělujeme systému, jakého typu budou zpracovávaná data v této položce, jejich rozsah, kolik místa v paměti maximálně zaberou nebo i to jaké operace budou pro tuto položku přípustné. Typické datové typy používané v databázích se dají rozdělit na:

- Textový typ - ten se obvykle dále dělí podle povolené délky na
  - krátký textový řetězec, například pro uložení jména, příjmení, názvu atd.
  - dlouhý textový řetězec, např. pro delší popis, či jiný druh textu.
- Číselné typy - pro uložení číselných údajů, většinou se dále dělí na typy pro celá čísla, reálná čísla a typy podle hodnoty jejich rozsahů.
- Logický typ - pro uložení hodnot typu ano/ne On/off apod..
- Datumový typ - slouží pro uložení data případně času

3.1 Zásady návrhu tabulek v relačních databázích

Tabulka je základním prvkem celé databáze. Relace odpovídá celé tabulce a prvku relace odpovídá jeden konkrétní řádek. Jeden řádek bývá nazýván databázovým záznamem. Soubor tabulek (relací) pak tvoří celou databázi (relační schéma).

Pro práci s relačními databázemi je velmi užitečné (ne-li přímo nutné) mít alespoň jednu položku (sloupec) zvolenu jako primární klíč, hodnota tohoto sloupce nám bude jednoznačně identifikovat záznam v tabulce.

Primární klíč má tu vlastnost, že jeho hodnota je jedinečná, tj. pro žádné dva řádky v tabulce nemůže, a také nesmí, nastat situace, že by hodnota primárního klíče

byla pro dva záznamy totožná. Databázové systémy většinou umožňují definovat jako primární klíč n-tici položek, např. dvojici nebo trojici položek. V takovém případě se mohou některé položky v klíčích opakovat, ale nesmí být shodné všechny položky dvou primárních klíčů najednou. Primární

Typicky se pro tyto účely vloží do tabulky navíc jeden sloupec, jenž bude typu integer (celé číslo) a vhodně zajistíme, aby u každého dalšího záznamu bylo toto číslo o jedničku vyšší. Vytvoří se tak jakési počítadlo záznamů. Drtivá většina databází má takovýto mechanismus již implementovaný přímo v sobě. Obvykle nese označení *auto-increment* a dá se nastavit jako další vlastnost daného sloupce.

Jako primární klíč lze však zvolit jakýkoliv sloupec tabulky, o kterém je známo, že bude u každého záznamu obsahovat rozdílné hodnoty. Takovým příkladem, může být např. databáze, jež obsahuje seznam studentů a zároveň jejich rodné číslo. To by měl mít každý jedinečné a proto, se sloupec obsahující rodná čísla dá s výhodou použít jako primární klíč dané tabulky.

3.1.1 Typy relací

V rámci jedné databáze většinou se nachází více tabulek, které spolu nějakým způsobem souvisí. Mezi jednotlivými tabulkami v databázi může existovat několik možností, jak jsou tabulky na sobě závislé, nebo-li jaká mezi nimi funguje relace

3.1.1.1 Relace 1:1

Nejzákladnější typ relace je 1:1. Každý řádek jedné tabulky je svázaný (může být svázaný – relace mohou být povinné či volitelné, tj. v jednotlivých tabulkách odpovídající řádky musí či nemusí existovat) právě s jedním řádkem druhé tabulky. Pro zajištění této relace je nutno vytvořit v obou tabulkách unikátní klíče.

3.1.1.2 Relace 1:N

Druhý, velmi běžný typ relace je relace 1:N. Každý řádek primární tabulky je (může být) svázaný s jedním či více řádky druhé tabulky. Relace 1:N se vytvoří tak, že se do souvisejícího objektu vloží primární klíč primárního objektu. Primární klíč primárního objektu ale nemusí být v souvisejícím objektu jedinečný. Související objekt bude tedy obsahovat svůj primární klíč, který bude jedinečný, a cizí klíč, který bude moci obsahovat i duplicitní hodnoty

Jako příklad lze uvést databázi pro výrobu desek plošných spojů, kdy v jedné tabulce jsou záznamy o zadávajících (jméno, příjmení, atd.) a v druhé jsou záznamy o zakázkách (číslo zakázky, Popis projektu…), jeden uživatel může mít zadáno několik typů desek najednou, viz Tabulka 3.1.

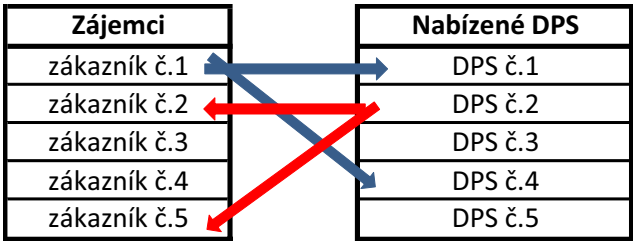
číslo zadávajícího	jméno	příjmení
1	Jan	Novák
2	Pavel	Lánský
3	Richard	Porůček

číslo zadávajícího	číslo desky	Popis projektu
1	1	Regulovatelný zdroj
2	2	NF zesilovač
1	3	V-metr
3	4	Stabilizovaný zdroj

Tabulka 3.1 Relace 1:N

3.1.1.3 Relace N:M

Více řádků primární tabulky je svázáno s více řádky sekundární tabulky. Tento vztah je z hlediska relačních databází celkem poměrně komplikovaný a nejčastěji se řeší dělením na více vztahů typu 1:N.
Jako příklad lze uvést případ, kdy by subjekt vyrábějící desky plošných spojů (dále jen DPS) nabízel již předem vyrobené typy DPS, například pro různé elektronické stavebnice. Systém by mohl obsahovat opět dvě tabulky. První tabulka by obsahovala seznam zájemců o tyto DPS, druhá pak seznam nabízených DPS viz. Tabulka 3.2. Tentokrát však nejen, že jeden zákazník může mít rezervováno více typů DPS ale i více zákazníků může žádat o výrobu stejné DPS.



Tabulka 3.2 Relace N:M

3.1.2 Normalizace

Normalizace databáze je v podstatě upravení struktury tabulek podle určitého souhrnu pravidel, které nám pomáhají vytvářet správné objekty (tabulky), které obsahují správná pole. Normalizace také usnadňuje pochopení vzájemných vazeb (relací) mezi objekty a v neposlední řadě zefektivňují práci v relační databázi.
Je definováno pět stupňů normalizace, běžně se však používají maximálně tři stupně.

3.1.2.1 Nultá normalizovaná forma (0NF)

Tabulka je v této formě právě tehdy, když alespoň jedno pole (sloupec) záznamu obsahuje více než jednu informaci (údaj). Příklad nulté normalizované formy je v Tabulka 3.3

číslo	Zákazník
1	Jan Novák, Brno
2	Pavel Lánský, Blansko
3	Potůček Richard, Praha

Tabulka 3.3 Nultá normalizovaná forma

Máme sice uložena všechna potřebná data, ale nelze například vypsat seznam zákazníku podle abecedy, jelikož někde může být první jméno, jinde příjmení, zkrátka

do tohoto typu tabulky prakticky není možný efektivní přístup a výhody databázového zpracování prakticky ztrácejí svůj výhody.

3.1.2.2 První normalizovaná forma (1NF)

Tabulku můžeme zařadit do první normalizované formy tehdy, když všechny atributy jsou tzv. atomické, což znamená, že je již dále nelze dělit. To znamená, že sloupec v tabulce, která je v 1NF, neobsahuje, a nesmí obsahovat více druhů údajů viz Tabulka 3.4.

číslo	jméno	příjmení	město
1	Jan	Novák	Brno
2	Pavel	Lánský	Blansko
3	Richard	Porůček	Praha

Tabulka 3.4 První normalizovaná forma

3.1.2.3 Druhá normalizovaná forma (2NF)

Tabulka splňuje zařazení do druhé normalizované formy tehdy, pokud splňuje podmínky pro první normalizovanou formu a zároveň neobsahující sloupce s duplicitními položkami, které mezi sebou vytvářejí částečné závislosti. Následující Tabulka 3.5 by do 2NF patřit nemohla, protože sloupec *název pracoviště* je přímo závislý se sloupcem *číslo pracoviště*

číslo	jméno	příjmení	město	číslo pracoviště	název pracoviště
1	Jan	Novák	Brno	2	sklad
2	Pavel	Lánský	Blansko	1	výdej
3	Richard	Porůček	Praha	2	sklad

Tabulka 3.5 Špatně navržená 2. normalizovaná forma

K opravě takovéto tabulky je třeba rozdělit ji na dvě dílčí části a vytvořit mezi nimi relaci 1:N jak ukazuje Tabulka 3.5

číslo	jméno	příjmení	město	číslo pracoviště
1	Jan	Novák	Brno	2
2	Pavel	Lánský	Blansko	1
3	Richard	Porůček	Praha	2

Tabulka 3.6 Správně navržená tabulka v 2. NF

číslo pracoviště	název pracoviště
2	sklad
1	výdej

3.1.2.4 Třetí normalizovaná forma (3NF)

Tabulka splňuje třetí normalizovanou formu tehdy, pokud splňuje 2NF a zároveň důsledně dodržuje pravidlo odstranění a oddělení veškerých dat, která nejsou v

přímém vztahu s daným objektem, nebo také splňuje-li 2NF a všechny neklíčové atributy jsou navzájem nezávislé.

3.1.2.5 Čtvrtá normalizovaná forma (4NF)

Tabulka je ve čtvrté normalizované formě tehdy, pokud je ve třetí normální formě a popisuje jen jeden fakt, či souvislost.

3.1.2.6 Pátá normalizovaná forma (5NF)

Tabulka je v páté normalizované formě, pokud je ve 4NF a není možno přidat do tabulky nový sloupec (atribut) či sloupce aniž by se rozpadla na několik dílčích tabulek.

4 Jazyk SQL

Jeho historie se začíná odvíjet od začátku sedmdesátých let, kdy se objevují první relační databáze, které pohlížejí na data jako na tabulky. Vznikl tak požadavek na vytvoření programovacího jazyka pro tyto aplikace a to takového, který by nebyl příliš složitý, aby ho zvládla používat co nejširší, i méně technicky zdatná veřejnost, a proto se měl co nejvíce podobal klasické angličtině.

Tento požadavek začala v roce 1974 řešit známá firma IBM, která tak dala vzniku jazyku, jenž měl název SEQUEL (Structured English Query Language). Postupně se k používání a vývoji tohoto jazyka přidaly i další firmy, Oracle vydává v roce 1979 svoji databázovou platformu Oracle Database, IBM v roce 1981 systém databázový systém SQL/DS a o dva roky později DB2, další systémy jsou např. Informix, SyBase, Progres atd. Všechny tyto databázové platformy jsou přímo založené na jazyku SEQUEL. Tímto rozšířeným použitím se z jazyka SEQUEL stává nepsaný standart pro vývoj a správu databází a vžívá se zkrácený název SQL (Structured Query Language = strukturovaný dotazovací jazyk). Po zhruba deseti letech vývoje přinesl výkonově použitelný systém, srovnatelný se síťovými a hierarchickými databázemi.

Jak jazyk procházel vývojem v různých firmách, bylo na zapotřebí jazyk SQL sjednotit, to se daří až roku 1986, kdy byl přijat jeho první oficiální standart, označován jako SQL86. I po té jeho vývoj pokračoval, v roce 1992 po odstranění některých nedostatků spatřil svět další vylepšenou verzi SQL92 zkráceně SQL-2. Po té přišly další standardy SQL1999 jinak nazýván SQL-3, definující použití triggerů, regulárních výrazů, rekurzivních dotazů, neskalárních typů a některých objektově orientovaných vlastností, dále SQL 2003 a nakonec SQL 2006, kde už jsou standardizovány i práce databáze se soubory XML.

Jazyk v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje na manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací).

V podstatě se skládá ze čtyř hlavních částí. První část je DDL (Data Definition Language) je to jazyk, který je určený pro vytváření databázových schémat a katalogů. Druhou částí je SDL (Storage Definition Language), který upravuje způsob, jakým se budou tabulky v databázích vytvářet. Další částí je VDL (View Definition Language), který slouží k v popisu vytváření náhledů. Poslední čtvrtá část je DML

(Data Manipulation Language), právě s touto částí se koncoví uživatelé a tvůrci databázových aplikací setkají nejvíce, jelikož tato část zajišťuje samotnou manipulaci s daty, od vkládání, přes přesouvání až po mazání záznamů.

5 MySQL

je relační databázový systém, vytvořený švédskou firmou MySQL AB a je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Databáze MySQL je prakticky nezávislá na použité platformě, lze ji instalovat jak na v současnosti nejrozšířenější operační systém MS Windows, tak i na Linux a další.

MySQL bylo od počátku optimalizováno především na rychlost, a to i přes některá zjednodušení, které s pro zvýšení rychlosti byly nutné jako např. jednoduché způsoby zálohování. MySQL donedávna nepodporovalo pohledy, uložené procedury ani triggery, což jsou automaticky spouštěné příkazy, které provedou nějaké, předem naprogramované operace v případě vzniku určité události. Především vysoká oblíbenost tohoto databázového systému mezi webovými designéry zapříčinila, že v posledních verzích tyto funkce přibyly a přidaly tak uživatelům větší komfort.

MySQL má vysoký podíl mezi v současné době používanými databázemi, právě díky své multi-platformnosti, vysokého výkonu a možnosti volného šíření.

Komunikace s tímto typem databáze probíhá pomocí jazyka SQL, který má jako u ostatních databází založených na jazyku SQL některé rozšíření a úpravy.

5.1 Typy tabulek v MySQL

MySQL umožňuje ukládat data do několika různých typů tabulek, každá z nich má své specifické výhody.

5.1.1 Typ ISAM

První typ tabulek, který se kdy v MySQL používal. ISAM (*indexed sequential access method*) metoda indexového sekvenčního přístupu. V nové verzi MySQL5 již ani není podporován, indexy ukládal nekomprimované, a proto byly příliš velké.

5.1.2 Typ MyISAM

Tento typ tabulky se používá nejčastěji a je i defaultním typem. Oproti předchozímu typu již pracuje s komprimovanými indexy, systém s nimi pracuje o něco déle.

5.1.3 Typ Merge

Někdy je třeba mít jednu tabulku rozdělenou na více částí, např. z důvodů omezené velikost souboru při velké tabulce apod. poté by se nám ale hodilo pracovat s tabulkou jako by to byla jedna a právě pro tento případ je tu tabulka typu Merge. Je to virtuální tabulka nad několika a nejméně dvěma tabulkami MyISAM se stejnou strukturou, tj stejný počet atributů i jejich názvů a typů a stejné indexy.

5.1.4 Typ InnoDB

Tabulky tohoto typu při zpracování dat používají techniku zamykání dat na úrovni řádků. Jsou vhodné všude tam, kde se data mnohem více editují či vkládají, než vybírají. Tento typ tabulky se navenek tváří jako jediný soubor, ve skutečnosti se ale skládá z více částí. Tabulky tohoto typu rovněž podporují tzv. transakce.

5.1.5 Typ BDB

Další typ tabulky podporující transakce je BDB (Berkeley database) Vývoj tohoto typu tabulky je ještě v plném proudu a proto se její nasazení doporučuje jen zkušeným uživatelům. Opět je vhodná tam, kde je mnohem více příkazů na vkládání a editaci dat, než jejich výběr.

5.1.6 Typ Memory

Méně často používaný typ tabulky, jejich obsah je celý načtený v paměti počítače, to jakou mají strukturu, je samozřejmě uloženo na disku, po vypnutí serveru je jejich obsah ztracen, na druhou stranu je možnost takovou tabulku vytvořit jako kopii již existující tabulky a jelikož je poté celá uložená v paměti počítače, práce s ní je mnohem rychlejší.

5.1.7 Typ Federated

Novinky mezi tabulkami, podporována až od verze MySQL 5.0.3. Data jsou uložena na jiném serveru, v tabulce tohoto typu je pouze uložena struktura a informace pro správné navázání spojení se serverem, na němž jsou uložena data.

5.1.8 Typ CSV

Také méně často používaný typ tabulky, hlavní odlišnost od ostatních typů je v tom, že svá data ukládá do textového souboru a odděluje je čárkami. Defaultně tento typ tabulky není povolen a celkově jej lze použít až od verze MySQL 4.1.4

5.2 Ukládání binárních dat do databáze MySQL

Jak již bylo řečeno, do databáze ukládáme data různých typů, od řetězců, přes čísla až po výčtové typy. Do databáze MySQL ale i do jiných lze však umisťovat i přímo binární data, jako jsou obrázky, či hudební soubory apod. Je však třeba říct, že většinou se programátor snaží tomuto řešení, pokud to není nezbytně nutné, vyhnout, protože např. práce se sérií obrázků uložených v databázi je pomalá a zabere víc místa. Obejít tento problém se dá například tak, že uložením takovýchto souborů do nějakého adresáře a do databáze se uloží pouze odkaz na tyto soubory. Přesto se ukládat binární soubory do databáze může být v některých případech výhodnější např. kvůli bezpečnosti.

Databáze MySQL pro takové případy přímo umožňuje vložení sloupce, jehož atribut zvolíme jako BLOB (Binary Large OBject). Tento typ se dále člení na čtyři druhy rozlišené maximální velikostí uložených dat, tak jak ukazuje Tabulka 5.1

název	max. bajtů
TINYBLOB	255
BLOB	65535
MEDIUMBLOB	16777215
LONGBLOB	4294967295

Tabulka 5.1 Možné atributy pro uložení binárních souborů

6 PHP

Jazyk PHP vznikl před několika lety jako jednoduchý soukromý projekt Rasmuse Lerdorfa pro vylepšení webových stránek, především pro sledování návštěvnosti. Později celý systém přeprogramoval v jazyce C a uvolnil pro volné užívání jako Personal Home Page – PHP. Velkého rozšíření se dočkal především díky své druhé verzi, která umožňovala uživatelům velice snadnou komunikaci s formuláři. V roce 1998 pak přichází další, v mnohém výrazně vylepšená, verze PHP3. Byla nejen výrazně rychlejší, ale obsahovala podporu mnoha databázových systémů, objektů a cookies. Další verze PHP4 doznalo výrazné změny především v použití zcela nového jádra Zend, přibylo pár nových vlastností a funkcí a opět došlo ke zvýšení výkonu. Nyní je opět po mnoha vylepšeních PHP dostupné ve verzi pět.

PHP je velmi komplexní programovací jazyk pro dynamické webové aplikace. PHP se neustále rozvíjí a umožňuje svým uživatelům spolupracovat s mnoha dalšími technologiemi, ať už zmiňované databáze nebo XML a XSLT. Umožňuje upravovat a vytvářet grafické soubory či dokumenty ve formátu PDF, stručně řečeno umožňuje téměř vše co je na webových stránkách potřeba.

PHP je ve své podstatě hypertextový procesor, který na serveru interpretuje stránky HTML s vlastními příkazy před jejich odesláním ke klientovi (obvykle webový prohlížeč) to znamená, že PHP umožňuje vkládat vlastní skripty přímo do hypertextových stránek podobně jako třeba Java Script. Rozdíl je však v tom, že Java Script je interpretován přímo klienta, kdežto PHP na straně serveru. Zpracování kódů přímo na serveru má oproti zpracování u klienta několik výhod, především:

- Snadněji spolupracuje s ostatními aplikacemi běžícími na serveru, jelikož není třeba neustále přenášet data po internetu
- Malá náročnost na hardware klienta, na straně klienta se totiž jen zobrazí vygenerovaná stránka, veškeré výpočty a vykonávání skriptů se provádí na straně serveru
- Menší objem dat, která je potřeba přenášet přes síť, právě proto, že vše již je vytvořeno na serveru a klientovy tak přijde jen stránka HTML
- Výrazně vyšší ochrana zdrojových textů programu, jelikož nejsou po síti vůbec přenášeny, kdežto při zpracování u klienta, jsou v rámci HTML stránky poslány.

Samozřejmě existují i nevýhody:

- Větší zátěž serveru
- Nemožnost spolupráce s objekty na již vygenerované stránce jako je například pohyb či klepnutí myši.

PHP umožňuje vložit skript přímo do zdrojového text stránky HTML, aby server poznal, kterou část má beze změny poslat klientovi a kterou má zpracovat jako PHP skript, musí se tato část programu v zdrojovém textu označit, např.

```
<Body >
… HTML kód …
<?php  … PHP script…  ?>
… HTML kód …
</body>
```

6.1 Zabezpečení databáze

Jak už bylo řečeno, PHP je ve svých vyšších verzích přímo předurčeno k spolupráci s databázemi. Umožňuje totiž snadné vkládání údajů. Pomocí formulářů v HTML stránce se odešlou data obsluhujícímu php skriptu a ten pomocí jednoduchého příkazu odešle data přímo příslušné databázi, či data jinak vhodně zpracuje. Právě zde vzniká velké bezpečnostní riziko, které by se nemělo přehlížet, jde o napadení tzv. podvržení SQL dotazu známějšího spíše pod anglickým označením *SQL injection*.

K SQL Injection může být náchylná jakákoli stránka, která si mění data s databází viz [9]. Základním vstupním prvkem, kudy se útočník tímto způsobem snaží napadnout aplikaci, jsou různé přihlašovací formuláře. Útočník se v tomto případě snaží místo požadovaných vstupních dat podstrčit vhodně sestavené SQL příkazy, které se pak odešlou, tak jako by to byla vstupní data. To jaké škody se podaří útočníkovi způsobit, závisí na mnoha aspektech. Relativně méně závažné škody vzniknou, např. pokud se podaří vykonat příkaz SELECT * FROM, který „pouze“ načte obsah databázové tabulky. Záleží pak na struktuře napsaného kódu, jestli se pak tato vybraná data podaří i zobrazit. Závažnější problém nastává, pokud se útočníkovi podaří vykonat příkaz DELETE, jenž maže data, ještě hůř příkaz DRPOP TABLE, který smaže všechna data i s celou tabulkou, nebo nastane-li nejhorší varianta – vykonání příkazu DROP DATABASE, jenž smaže celou databázi včetně veškerých dat.

6.1.1 Na straně databáze

V databázi můžeme útoku zabránit (nebo ho přinejmenším extrémně ztížit) vhodným nastavením práv uživatele, se kterými bude k databázi aplikace přistupovat. Velmi zřídka je nutné, aby měl přímo uživatel možnost z aplikační vrstvy mazat tabulky či dokonce databáze. Je-li to možné, pak se na straně databázového serveru vytvoří dva typy účtů. Jeden bude mít k dispozici administrátor a s neomezenými právy. Pomocí něj vytvoří celou strukturu databáze a následně ji bude spravovat. Naproti tomu druhý typ účtu bude sloužit pro běžný provoz systému a některé příkazy v něm budou blokovány.

6.1.2 Na straně webové aplikace

Nejlepší obrana proti SQL injection je ošetření veškerých uživatelských vstupů. Prakticky každý skriptovací program s podporou databáze má nějakou vestavěnou funkci pro escapování znaků před jejich použitím v dotazu. Escapováním se rozumí vložení speciálního znaku (většinou zpětného lomítka či uvozovek) před potencionálně nebezpečné znaky, které mají v databázi svůj specifický význam.

6.1.2.1 Funkce addslashes()

Tato funkce vrátí řetězec se zpětnými lomítky před uvozovkami, apostrofem a zpětným lomítkem. Takto escapovaná data jsou již vhodná k vložení do databáze. Je třeba však neopomenout na všechny proměnné, které mohou vstupovat do databáze. Z dokumentace vyplívá, že se zpětná lomítka použijí pouze pro přenos dat a do databáze jako takové se neuloží. Nevýhodou této funkce je nevhodnost použití pro ošetření binárních dat.

6.1.2.2 Funkce mysql_escape_string

Funkce mysql_escape_string je přímo určená pro escapovaní znaků v SQL dotazech vkládaných do databáze MySQL. Oproti funkci addslashes() ošetřuje všechny znaky, které by mohly mít bezpečnostní potíže. Toto opatření zabrání tomu, aby neupravený vstup mohl způsobit nějakou škodu, nebo vedlejší efekt. Umožňuje zabezpečit i binární data.

Tato funkce se používá přímo uvnitř SQL příkazů pro MySQL a smí se použít všude, kde jsme uvnitř apostrofů v SQL příkazu. Samotná MySQL rozumí takto upraveným znakům se zpětným lomítkem, které funkce mysql_escape_string vytvoří.

6.1.2.3 Funkce mysql_real_escape_string

Další možností, jak přeměnit nebezpečné znaky na bezpečné, je použití funkce mysql_real_escape_string. Je obdobná jako předchozí funkce s tím rozdílem, že mysql_real_escape_string bere v úvahu i nastavení znakové sady aktuální spojení. Proto má také tato funkce dva parametry, stejně jako mysql_escape_string je prvním parametrem právě ošetřovaný řetězec a jako druhý parametr je identifikátor spojení k databázi. Tato funkce je však relativně nová, a pokud se používá PHP starší verze, než je 4.3.0, nebude tato funkce správně pracovat.

6.1.2.4 Direktiva magic_quotes_gpc

Jedna se o direktivu, která se nastavuje v konfiguračním souboru PHP serveru. Při jejím zapnutí (magic_quotes_gpc=on), se automaticky escapuji všechny proměnné, které získáváme ze superglobálních polích Get Post a Cookie. Takto zapsaná data můžeme rovnou vložit do databáze. Nicméně tato funkce je od PHP verze 5 označena za zastaralou a od verze 6 bude odstraněna úplně.

6.1.2.5 Proměnné non-string

Jelikož se proměnné typu integer neukládají s uvozovkami či s lomítky musí být jejich zabezpečení provedeno jinak, a to důsledným testováním vstupních hodnot a typů proměnných. Nejen, že tak zabráníme případnému útočníkovi napadnout databázový systém, ale zabráníme i běžnému uživateli aby neúmyslným zadáním špatných hodnot nenarušil chod aplikace, a tím neohrozil nejen svá data.

Existuje několik možností jak testovat vstupní data na svoji hodnotu či typ. Jazyk PHP umožňuje takové testování provádět pomocí funkcí pro práci s proměnnými. Použitím např. funkcí *is_numeric($cislo), is_int ($cislo)* otestujeme, zda je proměnná *$číslo* skutečně číslo, respektive celé číslo. Další možnost jak kontrolovat správnost vstupních dat, je funkce *empty($cislo).* Ta pro změnu zjišťuje, zda je v proměnné uložená nějaká hodnota.

7 PhpMyAdmin

phpMyAdmin je webová aplikace napsaná v jazyce PHP obsahující i CSS a JavaScript. Tato aplikace umožňuje efektivní správu databází MySQL a jejich tabulkami. Umožňuje také administrátorům spravovat možné uživatele databáze a jejich přístupová práva.

Její vznik se datuje do roku 1998, kdy její první verzi napsal Tobias Ratschiller a vydal ji v říjnu téhož roku, jako verzi 1.0.1 Již v této rané verzi poskytovala aplikace sice omezené, ale užitečné funkce jako vyváření databází i tabulek, jejich editaci, zadávání a mazání dat.

Dalším mezníkem ve vývoji této aplikace bylo zapojení do vývoje Marcem DeLislem, ten se zapříčinil o transformaci kódu do podoby, ve které bylo možné jednoduše přidávat různé druhy jazykových souborů, což mělo za následek velké rozšíření aplikace. V dalších verzích se pracovalo na vylepšení navigačních schopností, dalších jazykových verzích a především se řešili různé záplaty a nové nápady přicházející do diskusního fóra projektu od stále přibývajících uživatelů. Verze 2.1.0. byla pak poslední verze vydaná právě Tobiasem Ratschillerem, z důvodů nedostatku času.

Projekt už byl ale natolik rozšířen, že se o různé další verze začali starat sami uživatelé, kteří vyvíjeli různé záplaty, avšak chybělo nějaké společné centrum, které by koordinovalo práce na vylepšeni aplikace. Toho se zhostil v na počátku roku 2001 Olivier Müller a projekt phpMyAdmin zaregistroval. Nové zázemí a sjednocenost přinesly další vývoj projektu, do dnešního dne bylo vydáno nespočet dalších verzí, které se především snaží držet krok s vývojem samotného MySQL a webu jako takového. Jedny z posledních verzí jsou dokonce dostupné v 52 různých jazycích. Do dnešního dne byla jako poslední stabilní verze vydána verze **2.11.6**

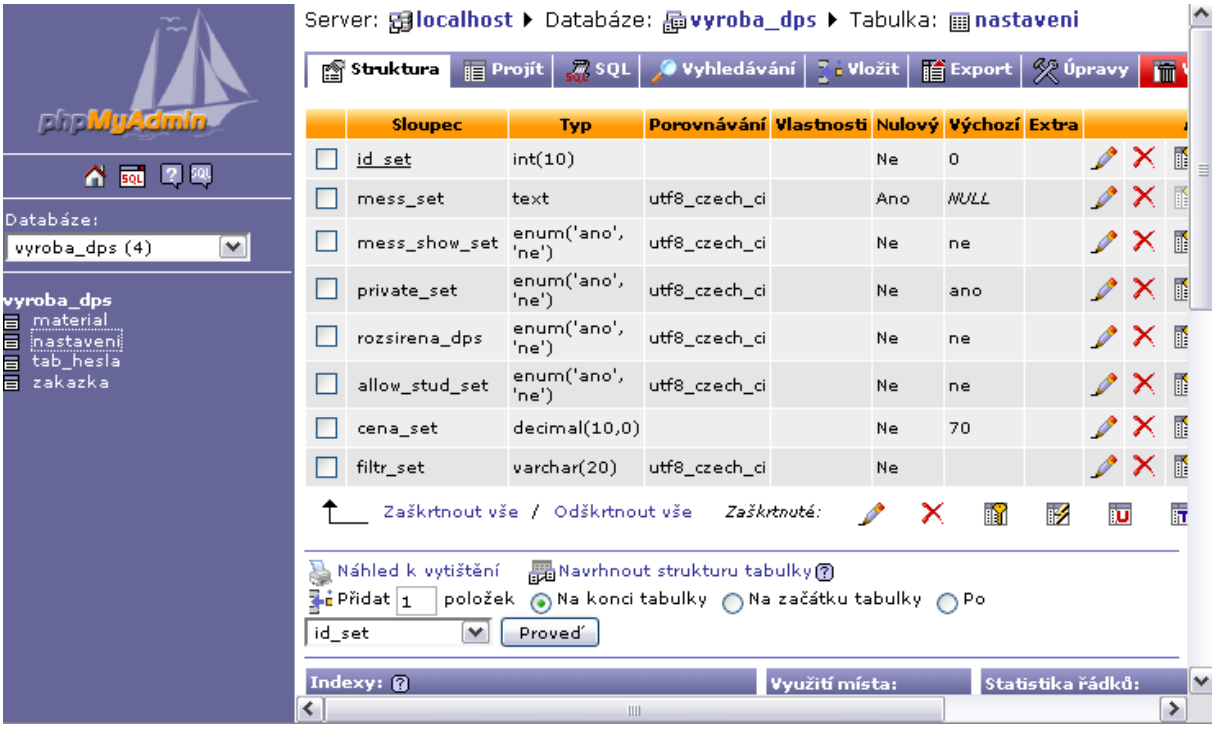
phpMyAdmin umožňuj při práci s MySQL především:

- Vytvoření a odstranění databáze
- Vytvoření tabulky, její kopírování, přejmenování a odstranění
- Speciální funkce prováděné nad tabulkou (změna typu, optimalizace)
- Údržba struktury tabulky včetně indexů
- Vkládání, upravování a editaci dat
- Nahrávání binárních dat
- Vyhledávání dat v tabulce i v celé databázi
- Zobrazení dat

Pro administrátory jsou pak funkce jako:

- Správa uživatelů a jejich oprávnění
- Kontrola databázových oprávnění
- Stav serveru
- Export databází a to i více najednou

phpMyAdmin spravuje databázi pomocí grafického zobrazení tabulek. Pracovní plocha je rozdělena na levý a pravý panel (Obrázek 7.1). Kde levý panel obvykle slouží pro určitý druh navigace mezi databázemi nebo tabulkami databáze, zatím co pravý panel bychom mohli nazvat jako pracovní plocha, kde se spravují data, tabulky, jejich struktury a objevují se zde výsledky našich operací. Celkově je správa databáze pomocí phpMyAdmin velmi intuitivní.



Obrázek 7.1 Okno aplikace phpMyAdmin

8 Case nástroje pro design databází

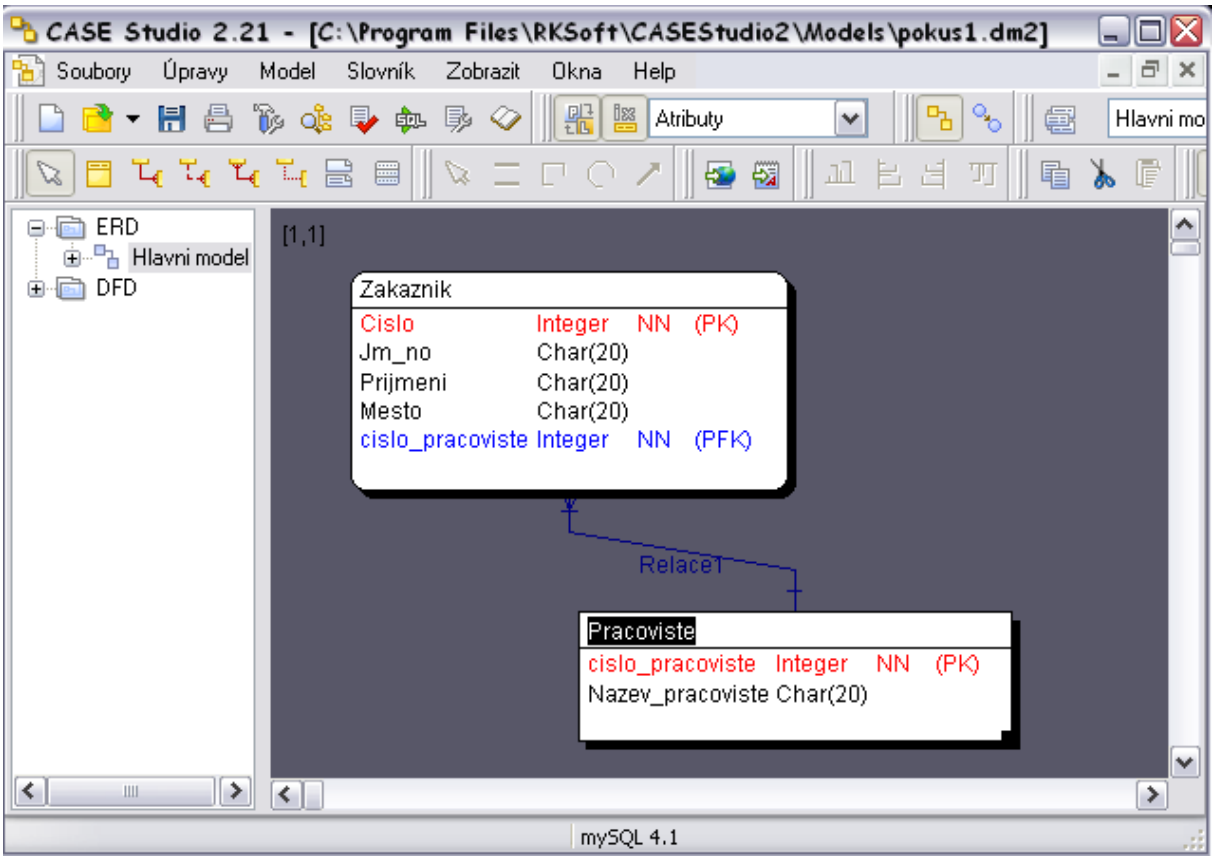
Pro návrh databázových projektů s menším rozsahem není problém pro jejich tvorbu s výhodou použít samotné SQL příkazy či některý z dostupných administrativních správců databází např. phpMyAdmin, DatAdmin apod. Není až tak obtížné, udržet přehled o tom, jaké jsou relace mezi tabulkami v projektu, kde jsou takové tabulky dvě nebo tři. Navrhujeme-li však složitý systém s tabulkami v řádu desítek a ještě k tomu v páté normalizované formě, je téměř jisté, že se v takovém projektu brzy stratí i zkušený programátor. V těchto případech se může s výhodou využít některý s dostupných CASE nástrojů např. Power Designer, Case Studio, apod. Ty umožňují mnohem rychleji a přehledněji všechny běžné operace prováděné při návrhu databáze. Tvorbu tabulek v rámci databází, možnost velmi jednoduchým způsobem vkládání relací a standardem bývá i tzv. reverse engineering, neboli možnost načtení entit (tabulek), jejich atributů, relaci, indexů apod. z již existující databáze a následná možnost práce s nimi jako s databází od počátku vyvíjenou daným Case nástrojem. To vše doplněno grafickým přehledem.

Většina Case nástrojů umožňuje nejen přehlednou tvorbu databází, ta je většinou jen součástí komplexního balíku funkcí, ale především efektivní návrh

celých informačních systémů. Z toho je patrné, že většina takovýchto nástrojů je velmi drahá a vyplatí se opravdu jen pro rozsáhlejší projekty.

8.1 Case Studio

Jeden z představitelů case nástrojů, jeho předností je především česká lokalizace, má implementovanou podporu velkého množství typů databází, jako je MySQL, MS SQL, Oracle, PostgreSQL, IBM DB2, Infomix a mnohé další. Podporuje již zmiňovaný reverse engineering a dalších funkce, které výrazně ulehčí jak vytváření databázových entit, tak i jejich následnou správu.



Obrázek 8.1 Okno aplikace Case Studio

Nespornou výhodou tohoto case nástroje je i fakt, že jej lze používat zdarma s omezením maximálně šesti tabulek.

Case nástroje lze doporučit všude tam, kde velký počet relací znepřehledňuje představu o vazbách v databázi i tam, kde se chceme vyhnout vytváření jednotlivých tabulek prostřednictvím jazyka SQL.

9 Rozbor následného vývoje systému

Má být vytvořen funkční databázový systém využívající ke své obsluze webovou aplikaci. Systém má splňovat co možná nejvyšší bezpečnostní požadavky a hlavně zjednodušit a zefektivnit zpracování objednávek desek plošných spojů oproti nynější verzi.

Dle zadání měl být celý systém realizován pomoci databáze MySQL. Tato volba se i s ohledem na předešlý teoretický rozbor zdá být jako nejvýhodnější. Jako vhodný prostředek pro správu databáze byla zvolena aplikace phpMyAdmin. Představované case nástroje mají sice více možností nastavení a funkcí, nicméně pro tento projekt budou bohatě stačit funkce nabízené právě aplikací phpMyAdmin. Navíc je tato aplikace nainstalovaná i na řadě serverech, což se o case nástroji říci nedá. Při následném testování a přenášení různých verzí programu mezi lokálním počítačem a webhostingovým serverem se pohybuje v stále stejném prostředí, což se jeví jako výhodnější.

Nyní je namístě shrnout základní požadavky na systém.

- Rozlišení zakázek, ty můžou být jak pro školní účely (bakalářské nebo diplomové práce) tak pro soukromé účely.
- Je žádoucí, aby jak soukromou tak školní zakázku musel někdo potvrdit
  - Školní zakázku schvaluje studentovi vedoucí jeho práce
  - Soukromou zakázku schvaluje dílna nebo jiný zaměstnanec UREL
- Soukromá zakázka je zpoplatněna

9.1 Typy uživatelů

Z výše uvedeného vyplívá, že systém obsluhují tři typy uživatelů – studenti, zaměstnanci a dílna. Právě s ohledem na toto bude vytvořen systém, který bude mít tři typy účtů.

9.1.1 Student

Základním typem účtu bude „student“. Bude mít nejmenší rozsah práv a je určen právě studentům, kteří chtějí do systému vkládat svoje zakázky výroby desek plošných spojů. Po jejich vložení do databáze mohou sledovat jejich stav a při splnění určitých podmínek (bude vysvětleno později) je mohou editovat či případně zrušit. Student má možnost sledovat a editovat jen své zakázky, z tohoto důvodu bude pro přihlášení do systému ještě zadávat svoje VUT ID (unikátní číslo přidělené každému studentovi VUT), při zobrazování stavu zakázek mu budou zobrazeny jen ty zakázky, které odpovídají jeho přihlašovacím údajům.

Byla řešena otázka, zda by si mohl student vkládat své soukromé zakázky do systému naprosto samostatně. Při prvotním pohledu na problematiku se zdá být logické, aby zakázky, které si student platí, mohl přímo vkládat do databáze. Nicméně po konzultacích přímo s dílnou byla tato možnost zavržena. Dílna projevila obavu z přílišného nárůstu soukromých zakázek, v případě, kdyby byla možnost objednání desky takto jednoduchá. Přání dílny bylo respektováno a systém bude navržen tak, aby i soukromé desky bylo nutné zapsat do systému za asistence pracovníka dílny případně s pomocí jiného zaměstnance UREL .

Původně bylo zadávání desek možné dvěma způsoby. Buď chce zadávající do systému vložit svoji soukromou zakázku a ta je okamžitě označena za schválenou (přípustnou pro přijetí do výroby) a bude vyrobena za poplatek, nebo vloží zakázku související např. s jeho diplomovou prací, tato mu bude vyrobena zdarma, ale k její výrobě se přistoupí až po potvrzení zakázky (autorizaci) vedoucím jeho diplomové (či jiné práce). To se ale vzhledem k připomínkám dílny změnilo. Student již nemá právo cokoliv zapisovat. Veškerá vkládání objednávek do databáze budou nyní za studenta v případě soukromé zakázky vykonávat pracovníci dílny, nebo v případě školního projektu, provede zápis vedoucí práce, či jiný zaměstnanec Ústavu radioelektroniky.

9.1.2 Zaměstnanec

Vedoucí práce, bude využívat druhého účtu s názvem „zaměstnanec“. Tento účet bude mít stejné možnosti vkládání dat (zakázek) jako „student“, navíc však může zobrazovat všechny zakázky, autorizovat zakázky jednotlivých studentů, měnit jejich parametry, nebo je i rušit. Hlavní smyslem tohoto účtu je právě autorizace zakázek.

Poté, co jsou zakázky autorizovány, bude již dílna moci přistoupit k výrobě dané desky. Dílna je nejen posledním článkem v řetězci výroby plošného spoje, ale zároveň figuruje i jako řídící segment, který nastavuje parametry systému podle svých aktuálních potřeb.

9.1.3 Dílna

Třetím typem účtu proto bude účet s názvem „dílna“. Má všechna práva jako předchozí dva účty, může doplňovat zbylé parametry týkající se výroby každé zakázky a rozhoduje o konečném stavu každé zakázky, může upravovat i mazat objednávky v jakémkoliv stadiu výroby

9.2 Stav objednávky

Pro efektivní komunikaci mezi jednotlivými uživateli podílejícími se na výrobě desky plošných spojů, bylo navrženo čtyřstavové oznamování stavu každé zakázky.

- První stav má název „*zadáno*“ - základní stav - bude určovat zakázku, která je pouze vložena do systému a zatím se nemá přistupovat k její výrobě. Zakázka čeká na potvrzení vyhotovení vedoucím nebo zaměstnancem.
- Druhý stav „*schváleno*“ – pro zakázky potvrzené učitelem/vedoucím práce, či pro soukromé zakázky, jež budou zpoplatněny. Takto označené zakázky mohou být vyrobeny.
- Třetí stav „*přijato*“ – stav, který nastavuje dílna, oznamuje, že zakázka byla přijata do výroby. Zároveň tím nastaví, že parametry dané zakázky již nepůjde změnit.
- Čtvrtý stav „*hotovo*“ – opět nastavovaný dílnou – určen pro oznámení, že je požadovaná deska již vyrobena.

9.3 Návrh databáze

Databáze i s obslužnými skripty budou nejprve vytvořeny na lokálním serveru (tedy na serveru pracujícím na stolním počítači) a po konečném doladění se kompletní projekt přesune na některý webový server, kde se systém otestuje v ostrém provozu.

Jako první byla vytvořena databáze s názvem *vyroba_dps*. K vytvoření databáze se použije SQL příkaz.

```
CREATE DATABASE vyroba_dps
```

Následuje návrh struktury tabulek v databázi. Struktura může vycházet z v současnosti používaného systému v podobě tabulky v Excelu (Obrázek 9.1 a Obrázek 9.2). Je to jedna tabulka, opticky je rozdělená do dvou částí, rozdělených dle uživatele, který ji má vyplnit. První část vyplňuje na žádost studenta vedoucí jeho práce, který má oprávnění zadávat zakázky.

Vyplní zadávající								
Datum	**Student**		**Popis konstrukce**	**Rozměr DPS**		**Platit**	**Zadal**	**Poznámka**
	příjmení	**jméno**		**a [mm]**	**b [mm]**	**ano ne**		

Obrázek 9.1 První část původní záhlaví původní tabulky

Druhou část vyplní pracovník dílny, který zakázku vyřizuje.

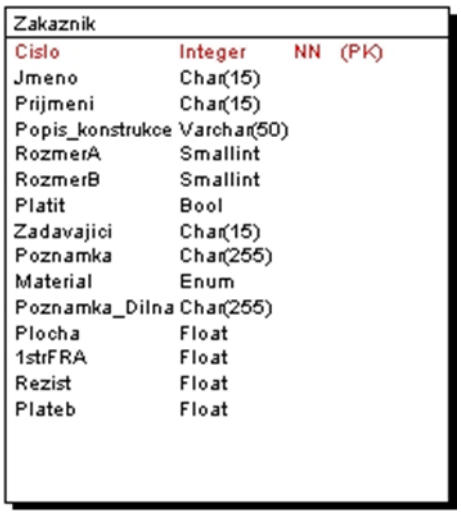
Vyplní dílna							
Platba	**Materiál**	**Poznámka**	Plocha	**Suma**			
	1str/FR4/1,5			**1strFR4**	**2strFR4**	**rezistu**	**plateb**
[Kč]	**2str/FR4/1,5 ostatní**		[dm²]	**[dm²]**	**[dm²]**	**[dm²]**	**[Kč]**

Obrázek 9.2 Druhá část záhlaví původní tabulky

Nově vytvořená tabulka by se mohla jmenovat „zakazka“ a byla by uložena v databázi „vyroba_dps“. Podobu tabulky ukazuje Obrázek 9.3. Tabulka i databáze byla vytvořena s pomocí programu phpMyAdmin. Pokud by tabulka měla být vytvořena ručně, musel by se na serveru php vykonat skrip jenž by začínal následujícím kódem:

```
1. <?php mysql_connect(SERVER, UZIVATEL, HESLO);
2. CREATE DATABASE vyroba_dps;
3. mysql_select_db("vyroba_dps");
4. mysql_query("CREATE TABLE zakazka
5. (     id int(10) NOT NULL auto_increment,
6. datum date NOT NULL,
```

```
7. vutid varchar(8) NOT NULL,
8. .
9. PRIMARY KEY  (id))"); ?>
```



Obrázek 9.3 Tabulka navržená dle staré struktury

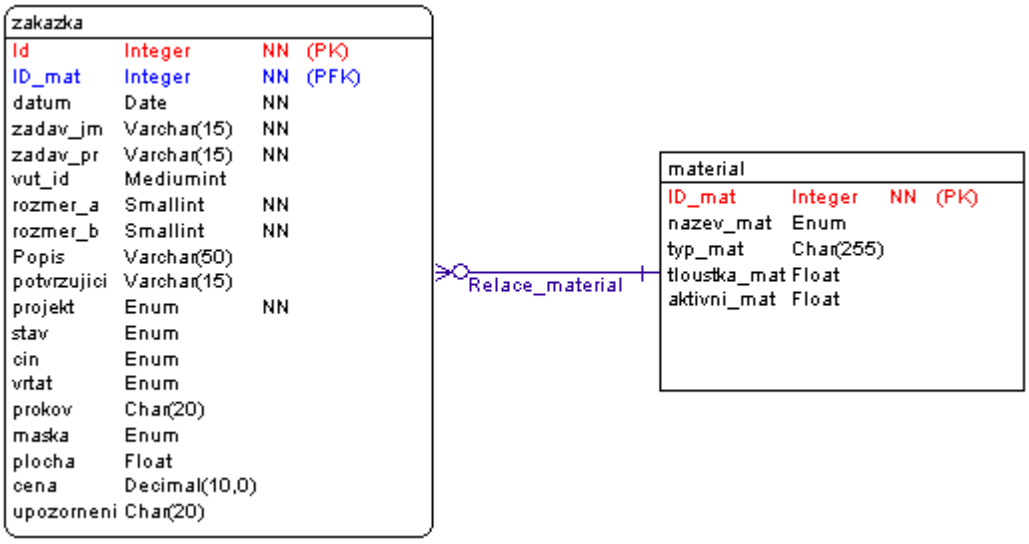
Nejprve musí proběhnout připojení k databázovému serveru pomocí funkce „`mysql_connect()`" se správnými parametry nastavení požadovaného serveru, jména uživatele a hesla. Následuje příkaz „`CREATE DATABASE vyroba_dps`" čímž se vytvoří databáze jako taková, následně se databáze označí jako pracovní – příkaz `mysql_select_db("vyrobadps")` a nakonec se pomocí funkce `mysql_query(SQL)` s vhodně zvoleným dotazem SQL vytvoří všechny sloupce, které jsou potřeba pro navrhovaný systém.

Po konzultaci s pracovníky dílny však bylo rozhodnuto, že vytváření nového systému správy se využije i k lepšímu rozvržení tabulky. Dle požadavků dílny byl např. úplně zrušen sloupec „Rezist". Pracovníci dílny rovněž projevili zájem rozšíření možnosti ukládání typu materiálů. Pokud by do tabulky byly vloženy další sloupce, v kterých by se zadával název, typ a případně další parametry materiálu, zbytečně by tak v tabulce vznikala duplicitní data. V tomto případě bude výhodné vytvořit další tabulku s názvem „material" a v ní uchovávat potřebné údaje o materiálech. Do tabulky „material" se vloží navíc sloupec *id_mat* jenž bude jedinečným identifikátorem každého materiálu, tzv. primární klíč. Bude mít podobu čísla. Toto číslo se pak vloží do tabulky „zakazka" a bude tak odkazovat právě na jeden záznam v tabulce „material". Vytvoří se tak vlastně relace 1:N, kdy jeden záznam zakázky může mít zvolen právě jeden typ materiálu a jeden materiál může mít zvoleno více zakázek. Novou verzi tabulky teď ukazuje Obrázek 9.4.

Jednotlivé sloupce tabulek jsou pojmenovány tak, aby co nejvíce charakterizovaly ukládanou hodnotu. Použité typy jednotlivých sloupců jsou voleny tak, aby byly co nejvíce přiblížené potřebám daného atributu.

V tabulce „zakazka" přibyl k původním sloupcům „jméno" a „příjmení", další sloupec „vut_id", což se ukáže jako výhodnější pro budoucí funkci načítání a filtrování dat z databáze. Číselným sloupcům byl zvolen takový typ, jehož rozsah by měl plně pokrýt předpokládané vstupní hodnoty.

Nový sloupec ID bude navíc zvolen jako primární klíč, tudíž s jeho pomocí budeme každý řádek jednoznačně určovat. Abychom toho mohli dosáhnout, zvolíme mu ještě parametr „autoincrement“ jenž zajistí za prvé automatické zvyšování ID u každé zakázky a zadruhé pohlídá, aby se každé ID opakovalo pouze jednou.



Obrázek 9.4 Nově navržená struktura tabulek

Další nový sloupec se jmenuje „stav“, je výčtového typu a jeho úkolem je zobrazit stav vložené objednávky. Čtveřice stavů, které mohou nastat, byly popsány dříve. Dále přibyl sloupec „datum“, jenž bude využit pro sledování poslední změny objednávky.

Modernizací vybavení dílny, došlo i k rozšíření možností výroby desek plošných spojů a je možnost nechat vyhotovit desku v tzv. rozšířené variantě. Proto bylo nutné do tabulky přidat ještě další čtyři sloupce:

- Cín
- Vrtat
- Prokov
- Maska

Všechny jsou výčtového typu (enum) se dvěma možnostmi hodnot buď „ano“ nebo „ne“. Výroba DPS v rozšířené variantě je časově náročná a proto se umožňuje jen u školních projektů.

V tabulce „materiál“ je pět sloupců, již zmiňovaný sloupec *id_mat* sloužící jako primární klíč, sloupec *nazev_mat* pro uložení názvu materiálu, sloupec *typ_mat* výčtového typu, sloužící pro rozlišení jednostranného nebo dvoustranného materiálu, sloupec *tloustka_mat,* pro uložení tloušťky materiálu a nakonec sloupec *aktivni_mat,* sloužící k rozlišení toho, zdali je materiál momentálně k dispozici pro výrobu.

Struktura databáze je tedy navržena, nyní je třeba navrhnout obslužné skripty, které s vytvořenou databází budou spolupracovat.

10 Návrh obslužných skriptů PHP

10.1 Návrh základních souborů

Navrhovaná webová aplikace musí umožňovat práci pro tři typy účtů, musí umožňovat zadání objednávky, její zobrazení a případnou úpravu již zadaných parametrů, to vše s určitými rozdíly pro každý typ účtu. V neposlední řadě musí mít aplikace, jak již bylo mnohokrát zmíněno, ošetřena, pokud možno všechna bezpečnostní rizika.

Rozborem předchozích požadavků (a nejen jich) bylo zjištěno, že pro funkci webové aplikace bude potřeba tří základních stránek, každá se svojí specifickou funkcí.

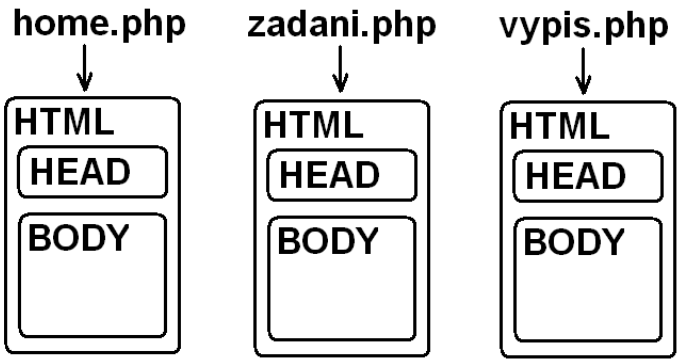
První stránka se bude jmenovat home.php, bude sloužit pro přihlášení uživatelů do systému. Bude obsahovat jednoduchý přihlašovací formulář, jehož vyplněním a odesláním zavoláme obslužný skript, jenž zkontroluje přihlašovací údaje.

Druhá stránka se jménem zadani.php bude sloužit pro vkládání nových objednávek do databáze. Bude obsahovat formulář obsahující políčka pro zadání všech parametrů potřebných pro výrobu objednávané DPS. Tento formulář bude zároveň využit pro zobrazení parametrů již zadané aplikace, za účelem možných úprav.

Třetí stránka bude sloužit k přehlednému zobrazení (tabulkový výpis) již zadaných objednávek. Tato stránka také bude zajišťovat filtraci zobrazovaných objednávek upravovaných dat.

10.2 Návrh použité struktury

Pro dosažení předešlých cílů by bylo z krátkodobého hlediska nejjednodušší vytvořit pomocí jazyka PHP webový systém skládající se ze tří samostatných PHP stránek – souborů, které by dynamicky generovaly tři HTML stránky, jak ukazuje Obrázek 10.1. Význam jednotlivých sekcí lze vyhledat v [3] nebo online v [7].

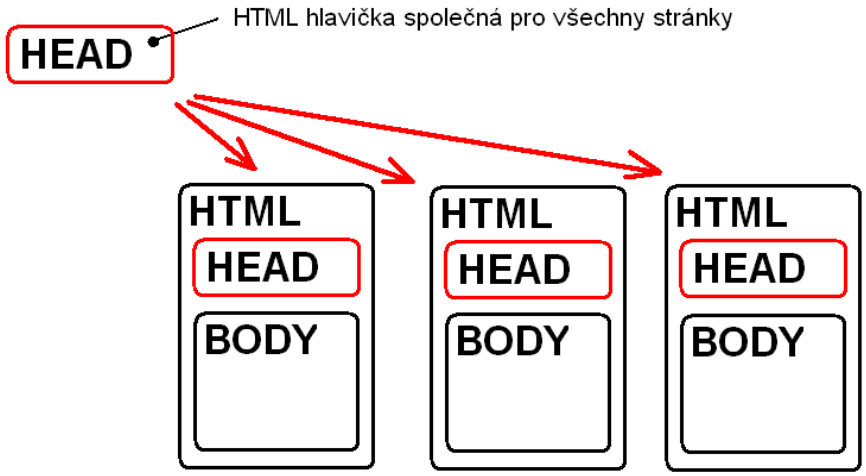


Obrázek 10.1 Struktura systému navržená klasickým způsobem

Tato varianta je sice logicky dobře pochopitelná, ale co se týče údržby takového systému, je velmi nepraktická. Pokud má být dosaženo společného

vzhledu, je třeba v takto navrženém systému nastavit všechny parametry v každé stránce zvlášť. Nastane-li pak nějaká změna např. v grafickém vzhledu, je třeba tyto změny provést v každé stránce zvlášť, což může vést k chybám a proto je výhodnější se tomuto řešení vyhnout.

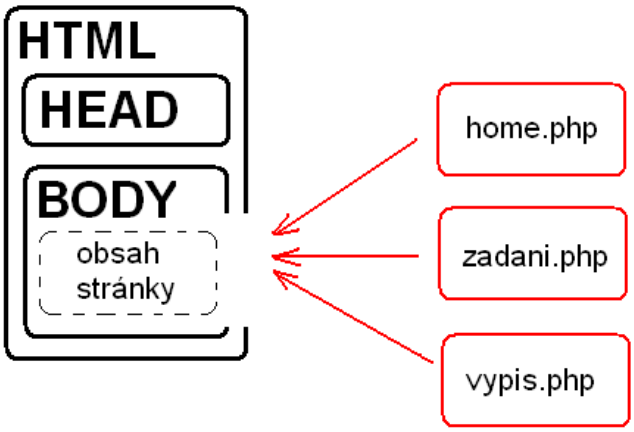
Lepší řešení bude, když si společné části jednotlivých HTML stránek uložíme do jednoho souboru a výsledné HTML stránky se budou vytvářet vložením tohoto společného souboru do jednotlivých stránek, viz Obrázek 10.2.



Obrázek 10.2 Vkládání společné hlavičky.

Toto řešení již dává webu jisté sjednocení. Pro budoucí rozšíření takového webu to však stále znamená, že při přidání nové stránky si bude muset programátor pamatovat, či bude muset prozkoumat, které všechny soubory jsou pro daný web společné a tím pádem, které bude muset do nové stránky vložit, aby zachoval stejný vzhled.

Tento nedostatek řeší následující varianta pomocí tzv. pluginů. V podstatě se jedná o stejný princip s tím rozdílem, že společná bude celá struktura stránky (tzv. šablona) a vkládat se bude jen jednotlivý specifický obsah (plugin) každé stránky viz Obrázek 10.3.

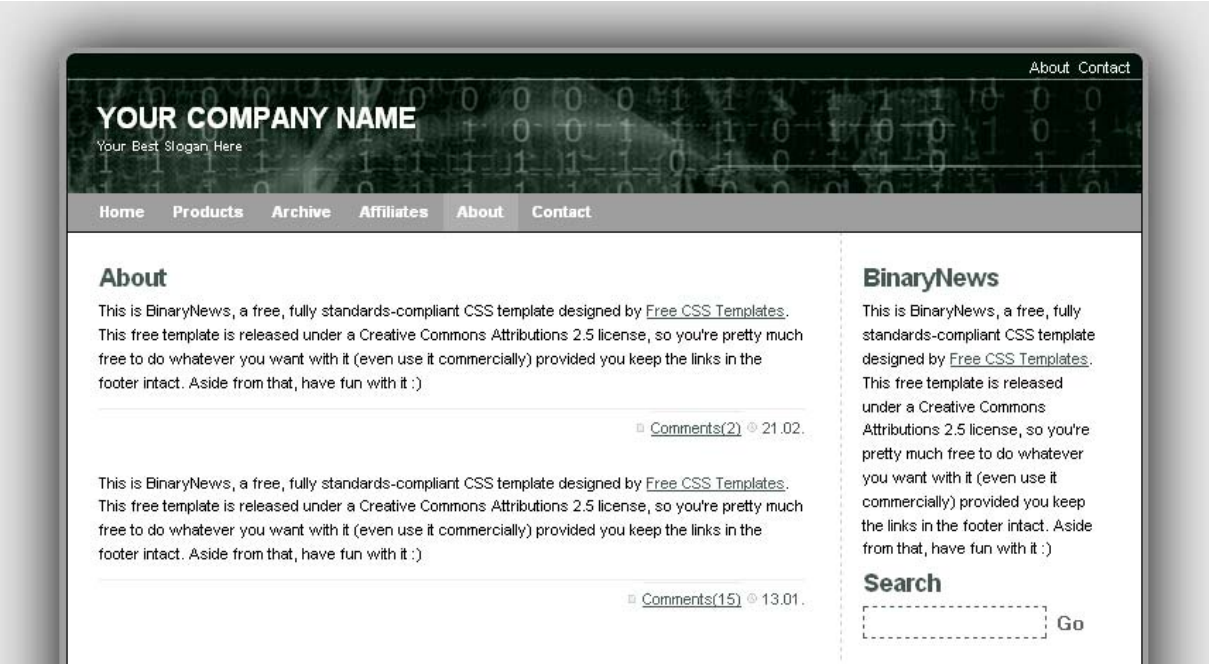


Obrázek 10.3 Systém využívající HTML šablony.

Jednotlivé soubory (home.php, zadani.php, vypis.php) se už nestarají o celkový vzhled, ale zajišťují jen zobrazení svého konkrétního obsahu. Takovýto systém je nejen jednoduše upravovatelný, co se týče jednotného vzhledu stránek, ale zároveň umožňuje snadné rozšíření pomocí nového obsahu, který se nechá do společné šablony vložit.

10.3 Grafická šablona

Ne každý tvůrce webových aplikací musí být i dobrý grafický designér. Webový grafik, který vhodnou kombinací obrázků a jejich rozmístěním vždy velmi pozvedne úroveň stránek, musí mít jistou dávku uměleckého citu. Pokud tvůrce nedisponuje takovýmto nadáním, nabízí se v takovémto případě řešení přímo v podobě počítačové sítě internet. Na internetu se dá nalézt a stáhnout spousty šablon, které jsou předpřipraveny pro vložení pluginů. Jako u každého softwaru, je i u šablon možnost stažení zdarma nebo za poplatek. V případě stažení zdarma bývá od tvůrce požadavek, aby většinou ve spodní části stránek, zůstal odkaz na jeho domovské stránky. Šablona z internetu bude s výhodou použita i v tomto projektu. Po delším prohledávání byla zvolena šablona, kterou ukazuje Obrázek 10.4.



Obrázek 10.4 Šablona "binarynews"

Šablona je tvořena obrázky, vhodně rozmístěnými pomocí kaskádních stylů uložených v soubor s příponou CSS, v tomto případě soubor *style.css*. Podrobnější informace o možnostech a použití kaskádních stylů nabízí např. [7].

Pro použití šablony je však třeba udělat řadu změn. Především je to změna její šířky. Při výpisu objednávek by se při množství vypisovaných parametrů do původní šablony parametry vedle sebe těžko vměstnaly. Proto se ve vhodném programu obrázky použité na pozadí šablony zvětší a nové rozměry se zapíší i na všech potřebných místech v souboru s kaskádními styly. Pro potřeby různých upozornění a oznámení je nutno přidat do souboru další styly, které sjednotí vzhledy používaných tabulek, odkazů a nadpisů.

Bylo potřebné provést změny v nastaveném kódování jazyka a odstranit části, jež pro používání webového serveru nebudou nutné, jako například přídavný sloupec v pravé části stránky nadepsaný nadpisem „Binarry news“ viz Obrázek 10.4.

Výpis kódu souboru *index.*html zbavený všech ukázkových dat je následující, ve výpisu kódu je vyznačeno, do které části se budou vkládat obsahy jednotlivých pluginů.

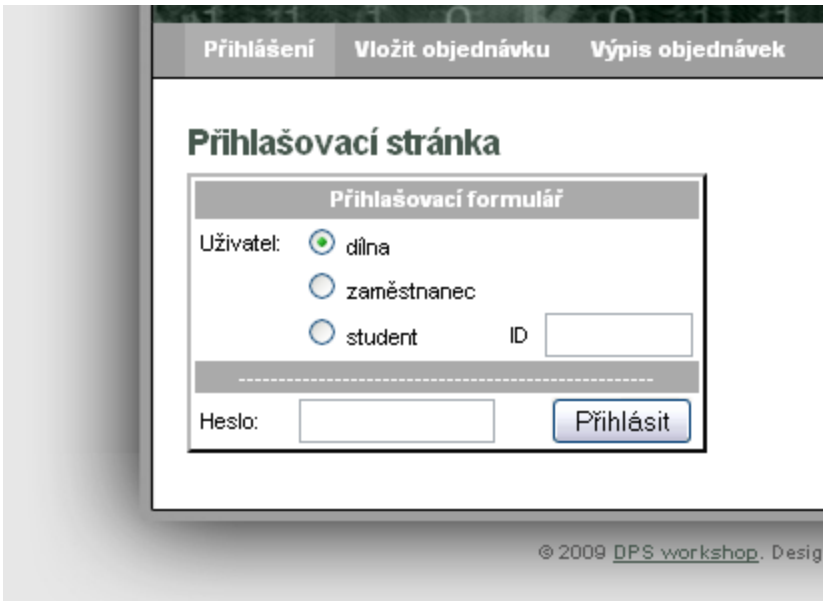
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <meta http-equiv="Content-Language" content="cz">
       <link rel="stylesheet" href="images/style.css" type="text/css" />
       <title></title>
</head>
<body>
  <div class="content">
        <div class="header">
    <div class="title">YOUR COMPANY NAME</div>
             <div class="slogan">Your Best Slogan Here</div>
         </div>
         <div id="nav">
               <ul>
                 <li><a href="">Přihlásit</a></li>
                 <li><a href="">Vložit objednávku</a></li>
                 <li><a href="">Výpis objednávek</a></li>
                 <li><a href="">Help</a></li>
                 <li><a href="#">Kontakt</a></li>
               </ul>
         </div>
         <div class="main_content">
          <div class="sd_left">
           <div class="text_padding">

<!-- Do tohoto místa se budou vkládat jednotlivé pluginy -->

          </div>
         </div>
          <div class="footer">
           <div class="padding">
               &copy; 2007 <a href="#"><strong>SiteName</strong></a>.
         <!-- nasledujici odkazy jsou nutne vlozit pro free pouziti této sablony -->
   Designed by <a href="http://www.free-css-templates.com/">Free CSS Templates</a>,
    Thanks to <a href="http://www.injuryexperts.com/">Personal Injury Attorney</a>
          </div>
         </div>
        </div>
       </div>
</body>
</html>
```

Poslední změnou je umístění obsahu souboru *index.html*, v kterém je zapsána společná struktura stránek do nového souboru *index.php*. To v budoucnu umožní vkládat do souboru ještě před samotným začátkem HTML tagů různé příkazy, kontrolu přihlášení a především vytvořit logiku řízení vkládaného obsahu.

Nyní máme vytvořenou strukturu databáze i šablonu pro zobrazení na webu, zbývá vytvořit příslušné skripty, které budou řídit celou aplikaci i obsah jednotlivých pluginů.

10.4 Přihlašovací formulář

Přihlašovací formulář vytvoříme v souboru home.php, ten bude v případě kliknutí v menu na odkaz „Přihlášení“ načten do místa naznačeném v předchozí kapitole. Jeho vzhled ukazuje Obrázek 10.5.



Obrázek 10.5 Přihlašovací formulář

Dle předešlého rozboru bylo vytvořeno členění přihlašovacího formuláře na jednotlivé části. Zatím co uživatel účtu „dílna“ a „zaměstnanec“ jen zvolí typ účtu a vyplní správné heslo, „student“ ještě musí zadat svoje VUT ID. Pod tímto identifikačním číslem pak budou uloženy jeho zakázky. Po stisknutí tlačítka „Přihlásit“ se data z formuláře odešlou metodou POST ke zpracování.

10.5 Řešení přihlašování

Pro ověření přihlašovacích údajů je vhodné vytvořit funkci, která toto bude realizovat. Odeslaná data z formuláře budou dostupná v superglobální proměnné `$_POST["název_proměnné"]`. Srovnáním hesla pro zvolený účet se kontroluje, zda je heslo správné.

Přístupová hesla musí mít PHP server k dispozici, tudíž musí být někde uložená. Jednou z možností je uložit hesla přímo ve skriptu. Při správně nastaveném serveru Apache, takový způsob nepředstavuje bezpečnostní riziko. Bude-li však třeba v budoucnu zadaná hesla změnit, bude to muset provést administrátor přímo ve skriptu. Naproti tomu, pokud by byla hesla uložena v databázi, bylo by velmi jednoduché naprogramovat obslužný skript, kterým by mohla být hesla z databáze načítána a také měněna.

Do struktury vytvořené databáze proto bude přidána tabulka s názvem *tab_hesla.* Bude mít čtyři sloupce. Jednoznačným identifikátorem bude sloupec *id_pass*. Zbylé slupce budou sloužit pro uložení hesel jednotlivých účtů. V tabulce tak bude vložen jen jediný záznam, viz Obrázek 10.6.

Obrázek 10.6 Tabulka hesel zobrazená v programu phpMyAdmin

Pro přehlednost je v následujícím výpisu kódu ukázáno, jak vypadá SQL dotaz, jimž může být tabulka vytvořena.

```
CREATE TABLE IF NOT EXISTS tab_hesla1
          ( `id_pass` int(10) unsigned NOT NULL default '0',
            `dilna` char(50) collate utf8_czech_ci NOT NULL default '',
            `zamestnanec` char(50) collate utf8_czech_ci NOT NULL default '',
            `student` char(50) collate utf8_czech_ci NOT NULL default '',
            PRIMARY KEY  (`id_pass`)
          )  ENGINE=MyISAM    DEFAULT    CHARSET=utf8    COLLATE=utf8_czech_ci
PACK_KEYS=0 ";
```

Nyní tedy existuje tabulka obsahující hesla, později bude uvedeno, kde a kdo bude mít práva hesla měnit.

Aby celý systém přihlašování měl vůbec smysl, je nutné, aby při návštěvě jakékoliv stránky, měl PHP server informace o tom, zdali uživatel úspěšně prošel autentizací a mohl tak stránky, jenž by měly být dostupné pouze pro přihlášené, chránit proti přístupu. Proto by bylo vhodné, aby se po úspěšném přihlášení nastavil nějaký ukazatel, který bude říkat PHP serveru, zda je uživatel přihlášen či nikoliv. Samotný protokol HTTP je ve svém principu bezestavový a proto se musí tato bezestavovost nějakým způsobem nahradit. Jazyk PHP nabízí dva mechanizmy jak toho docílit.

Prvním z nich jsou tzv. *cookies* jsou to textové soubory malé velikosti, které může webový server vytvořit a zaslat uživateli, kde se uloží přímo v jeho počítači. Takovýto soubor může obsahovat například údaje o autentizaci. Pokud uživatel znovu přistupuje na stejný webový server, odešle soubor cookies zpět webovému serveru a ten si tak může zkontrolovat, jaká data u uživatele uložil např., zda je již uživatel přihlášen. Soubory cookies jsou mezi serverem a uživatelem přenášeny v rámci hlavičky http v nezakódované podobě.

Druhou možností je použití takzvaných *session proměnných*. Session proměnnou opět vytvoří server, přenos dat u těchto proměnných je sice založený na používání *cookies,* avšak není na nich závislý, jelikož umí používat k přenosu dat i například URL. Oproti funkci samotných *cookies* si však server s uživatelem nevyměňuje přímo data, ale jedinečná dlouhá čísla, která danou proměnnou zastupují. Odesláním dat serveru tak nemůžeme nastavit samotnou proměnnou, ale pouze ovlivnit obslužný PHP skript, který ji vytvoří, nastaví či zruší. Samotný obsah session proměnné tak zůstává stále na serveru.

Vezmeme-li v úvahu, že v každém webovém prohlížeči existuje možnost přijímání *cookies* zrušit, že při jejich používání dochází k výměně samotných dat v nezašifrované podobě, pak je nanejvýš logické zvolit jako spolehlivější metodu pro uchování informace o přihlášení *session proměnnou*.

Po úspěšném přihlášení tedy bude nastavena session proměnná „prihlaseni“ (`$_SESSION["prihlaseni"]`), zároveň bude využita k uchování informace o úrovních povolených práv (odpovídajícím danému účtu). Session proměnná „přihlášení“ bude obsahovat číslo od jedné do tří, přičemž jednička odpovídá účtu „student“, dvojka účtu „učitel“ a trojka účtu „dílna“. Pokud je přihlášen uživatel s účtem „student“, nastaví se navíc další session proměnná „vut_id“, která, jak název napovídá, bude obsahovat VUT ID. Díky tomu bude jednoduše k dispozici při zpracování jakéhokoliv skriptu.

Funkce, která bude ověřovat přihlášení uživatele, bude vložena do souboru *index.php* ještě před výpisem HTML tagů stránky, aby mohlo být umožněno po úspěšném přihlášení ještě ovlivnit následný obsah stránky. Stejně tak i mechanismus kontroly nastavené session proměnné. Pokud proměnná nebude mít nastavenou jednu ze tří úrovní práv, pak bude přístup na stránku zamítnut. Pro větší komfort je vhodné následně přesměrovat uživatele na přihlašovací stránku spolu s adresou stránky, na kterou chtěl vstoupit, aby ho po úspěšném přihlášení mohl systém opět přesměrovat zpátky.

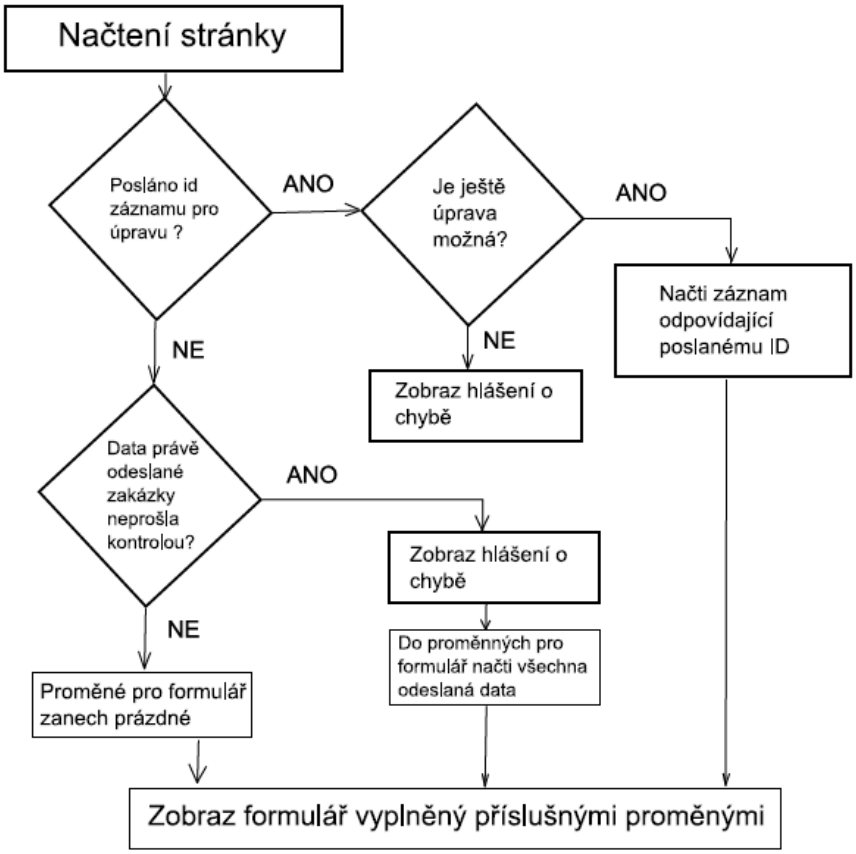
10.6 Řešení odhlašování

Odhlašování bude mít díky předešlému mechanizmu přihlašování jednoduchý princip. Bude vytvořena funkce, která bude obsahovat formulář s jedním tlačítkem „*Odhlásit*“. Tuto funkci pak můžeme vložit kamkoliv do libovolné stránky. Stisknutím tlačítka formulář odešle proměnnou "*odhlasit*". Při každém načtení stránky *index.php* se kontroluje, zda byla tato proměnná nastavena. Pokud ano, samotné odhlášení se provede nastavením *session proměnné „prihlásit“* na nulu. Celý proces může být doplněn o oznámení o úspěšném odhlášení. Dále již bude aplikace postupovat jako při prvním vstupu na stránky. Pokud proměnná nebude mít nastavenou jednu ze tří úrovní práv, pak bude přístup na chráněné stránky zamítnut.

10.7 Zadávací formulář

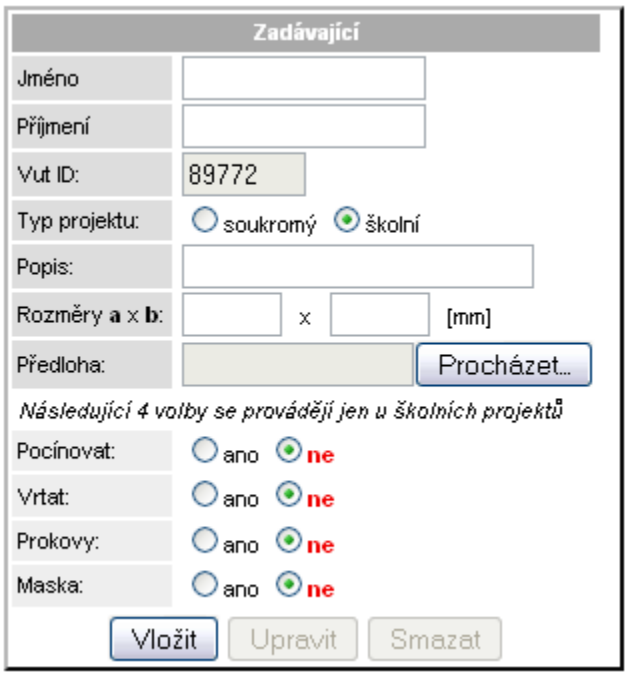
Nyní se již lze do systému přihlásit i odhlásit. Návrh může pokračovat souborem *zadani.php.* Výstup tohoto souboru se bude vkládat do šablony a bude obsahovat především formulář, do kterého budou uživatelé zapisovat parametry nové zakázky. Tento formulář bude současně sloužit i pro úpravu dat již zapsané zakázky, či pro znovu zobrazení zakázky, kterou se pokusil uživatel uložit, ale která neprošla kontrolou vstupních dat. Vzhledem k tomu, že ke každému účtu přísluší jiná práva, bude muset být pro každý účet i částečně odlišný zadávací formulář, tak aby byly každému typu účtu nabídnuty k zapsání právě ty hodnoty, které má právo zapisovat.

Právě z důvodů použití formuláře pro různé účely, předchází zobrazení formuláře řídící logika, jenž se stará o obsah polí, které bude mít formulář předvyplněné po načtení stránky. Princip řízení obsahu jednotlivých polí zjednodušeně zachycuje Obrázek 10.7



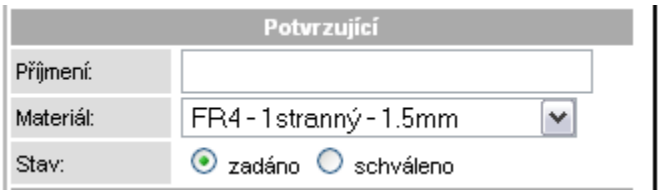
Obrázek 10.7 Vývojový diagram řízení obsahu zadávacího formuláře

Vzhled samotného formuláře ukazuje Obrázek 10.8



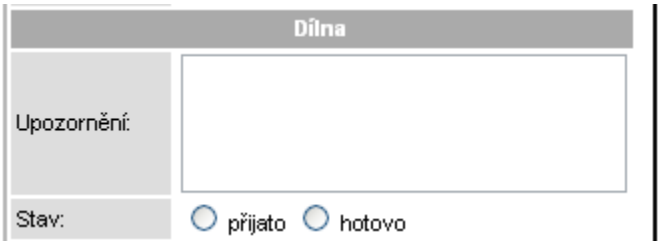
Obrázek 10.8 Formulář pro zadání zakázky - účet *student*

Přihlášení uživatelé účtu *zaměstnanec* mají tento formulář rozšířen o další část, viz Obrázek 10.9. V této části schvalují zakázku, vloží své jméno a z rozvinovacího seznamu vyberou požadovaný materiál.



Obrázek 10.9 Rozšíření zadávacího formuláře pro účet *zaměstnanec*

Uživatelé účtu *dílna* mají navíc ještě jednu část, viz. Obrázek 10.10. Zde vkládají případné upozornění a mění stav zakázky.

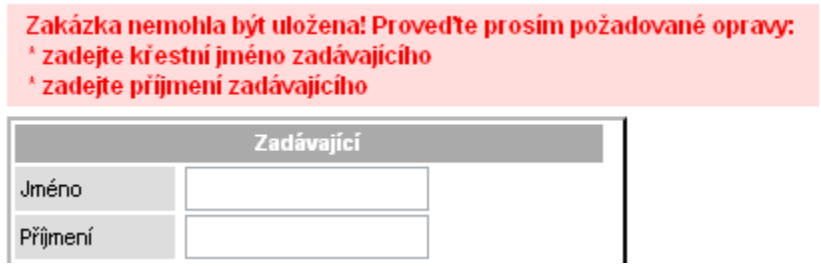


Obrázek 10.10 Rozšíření formuláře pro uživatele typu dílna

10.7.1 Zpracování odeslaných dat

Aby mohla být odeslaná data vložena do databáze, musíme je nejprve zpracovat. Do souboru *index.php* bude vložen další soubor s názvem *logika_ukladani.inc.php*. Obsah tohoto souboru je navržen dle potřebných funkcí. Skript uvnitř souboru, při každém načtení kontroluje, zda byla odeslána nějaká data a pokud ano, tak za jakým účelem.

Buď mohou být data odeslána za účelem uložení, nebo za účelem upravení. Po té co skript získá data načtením ze superglobální proměnné `$_POST` provádí jejich kontrolu po obsahové stránce. Zda uživatel nezapomněl vyplnit některá pole, či zda nepřekročil některé limity zadávaných parametrů. Proměnná *`$data_check`* je nastavena na „fail“, pokud jsou data z jakýchkoliv důvodů shledána neplatnými. Kontrolou této proměnné na konci skriptu se rozhoduje, jaký obsah bude načten do HTML šablony. Pokud kontrola bude hlásit chybu, načte se do šablony znovu zadávací formulář i s odeslanými daty a uživateli bude oznámena příčina chyb.



Obrázek 10.11 Oznámení o nesprávně vyplněných polích

Jestli-že data prošla kontrolou je možné je uložit do databáze. Jelikož se nelze spoléhat na to, že byla skutečně přijata jen ta data, které má právo daný uživatel vkládat/upravovat. Z tohoto důvodu bude vhodné, aby pro každý typ účtu byla vytvořena vlastní funkce pro ukládání/úpravu dat. Po zkontrolování úrovní práv (uložených v session proměnné) se vybere pro danou operaci funkce příslušná té které úrovni práv. Jen touto dvojitou kontrolou (při odeslání dat a při jejich přijetí) je možno předejít různým druhům napadení systému.

Samotné vložení zakázek do systému je realizováno pomocí SQL dotazu s klíčovými slovy INSERT INTO. V PHP se sestavený SQL dotaz vloží do funkce *mysql_query()*, která provedení dotazu vykoná.

```
mysql_query ("INSERT INTO zakazka (datum, vut_id, …)
                              VALUES (CURDATE(), '89772',…)");
```

Nejprve je však nutné se do databáze připojit - příkaz `mysql_connect(SERVER, UZIVATEL, HESLO)`, a vybrat pracovní databázi - příkaz `mysql_select_db(DATABASE)`. K tomuto účelu bude vytvořena funkce *openDB(*SERVER, UZIVATEL, HESLO, DATABASE DATABASE *)*, která se postará o oboje najednou. Jejími parametry jsou název serveru, ke kterému se připojuje – SERVER, jméno pod kterým se připojuje – UZIVATEL, přístupové heslo – HESLO a název databáze – DATABASE. Tyto čtyři parametry jsou uloženy jako konstanty na začátku souboru *index.php* a je potřeba je nastavit před prvním spuštěním systému.

Soubor *logika_ukladani.inc.php* také zajišťuje mazání zakázek, či výpočet plochy objednávané DPS a ceny, pokud je zakázka soukromá.

10.8 Výpis objednávek

Soubor *vypis.php* bude další ze souborů, jejichž zpracovaný výstup se bude zobrazovat přímo na stránky v rámci použité šablony. Soubor bude mít v podstatě pouze dvě funkce. První funkce bude výpis všech objednávek (výjimkou je účet „student“). Druhá funkce je filtrování načtených dat, lépe řečeno vybírání z databáze již vyfiltrovaných dat.

Možnost filtrování byla zvolena z důvodů snadnějšího nalezení konkrétní zakázky při jejím potvrzování zaměstnancem UREL. Původně byl filtr navržen pouze pro filtrování dle „stavu“ zakázky a VUT ID, po konzultaci s dílnou byla přidána možnost filtrovat data dle příjmení zadávajícího. Vyplněná data jsou načtena a použita při sestavení SQL dotazu.

Výběrový SQL příkaz SELECT, který zajišťuje načítání dat z formuláře, má nepřeberné množství variant zadávaných syntaxí. Pro jednoduchou filtraci postačí přidat klauzuli `WHERE *název_sloupce = hodnota*`. Tento příkaz načte z databáze všechny záznamy, u nichž sloupec *název_sloupce* bude přesně roven parametru *hodnota*. Jayzk SQL ale nabízí zajímavější variantu. Bude-li místo rovnítka použito slovo LIKE budou z databáze načteny všechny záznamy, u nichž bude sloupec *název_sloupce* obsahovat *hodnotu*. V tomto případě se nemusí obě položky přesně shodovat, Což se hodí zejména při prohledávání řetězcových sloupců.

SQL dotaz použitý pro filtraci dle jména, VUT ID a stavu bude mít tedy tvar:

```
$sql = "SELECT * FROM zakazka WHERE vut_id LIKE '$filtr_id' AND zadav_pr LIKE
'$filtr_pr' AND stav LIKE '$filtr_stav' ORDER BY $order DESC";
```

Klíčové slovo ORDER určuje, dle kterého sloupce se budou výsledná data řadit, DEST označuje, že toto řazení bude sestupné. Podrobnější popis SQL příkazů lze nají například v [6]. Provedení tohoto sql příkazu opět obstará funkce

```
$vysledek_dotazu = mysql_query ($sql);
```

Načtená data, budou uložena jako asociativní pole v proměnné `$výsledek_dotazu`. K vypsání všech dat slouží skript uvedený níže

```
include_zahlavi_tab ();
while ($radek = mysql_fetch_assoc($vysledek_dotazu))
       { include "one_row.inc.php"; }
echo '</table>';
```

První řádek vloží záhlaví tabulky, následuje cyklus, který z proměnné `$vysledek_dotazu` načte vždy jeden záznam do proměnné `$radek`. Tato proměnná je typu pole a je použita na vyplnění jednoho řádku tabulky v souboru one_row.inc.php.

1. část výpisu objednávek zobrazuje Obrázek 10.12, druhou pak Obrázek 10.13

Datum poslední změny	Zadávající		Rozměry *a* x *b* [mm]		Popis objednávky	Schválil	Typ projektu	Stav
	Jméno Příjmení	VUT ID						
2009-05-26	Jiří Procházka	89772	100	13	regulovatelný zdroj	Ostřížek	školní	schváleno
2009-05-26	Jiří Procházka	89772	100	13	NF zesilovač	Kučera	školní	přijato
2009-05-26	Jiří Procházka	89772	70	13	blikacka	Novotný	školní	hotovo
2009-05-26	Jiří Procházka	89772	12	14	zkušební DPS		soukromý	zadáno
2009-05-26	Jiří Procházka	89772	67	45	nová deska	Karský	soukromý	hotovo

Obrázek 10.12 1.část výpisu objednávek

Materiál	Cín	Vrtat	Prokov	Maska	Cena [Kč]	Možnosti
FR4 1stranný	ano	ano	ano	ano	0.0	-upravit-
FR4 1stranný	ano	ne	ano	ne	0.0	-upravit-
Arlon 350 2stranný	ne	ano	ano	ne	0.0	-upravit-
FR4 2stranný	ne	ne	ne	ne	35.0	-upravit-
FR4 1stranný	ne	ne	ne	ne	35.0	-upravit-

Obrázek 10.13 2. část výpisu objednávek

Soubor one_row.onc.php obsahuje několik skriptů pro řízení formátování zobrazovaných dat dle jejich obsahu. Z Obrázek 10.12 je patrné, že sloupec „stav“ je podbarvován třemi typy barev dle momentálního stavu zakázky. Přičemž první stav „zadáno“ pouze přebírá barvu aktuálního řádku. Skript také rozhoduje o tom, zdali bude popis desky pouhý text, či se stane odkazem na uložený soubor předlohy. Právě tak se stará i o vytváření odkazu ve sloupci „VUT ID“ a v posledním sloupci „Možnosti“. Ten se pro uživatele (kromě účtu dílna) stává pouhým textem, pokud je již stav zakázky „přijato“ nebo vyrobeno. V ostatních případech se chová jako odkaz a po kliknutí způsobí, že se aktuální záznam načte do zadávacího formuláře a je umožněna jeho editace.

10.9 Stránka nastavení

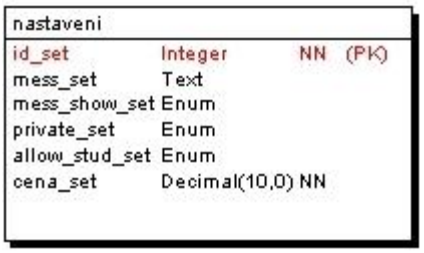
Pracovníci dílny vyslovili požadavky a nové náměty ohledně různých funkcí systému funkcí systém, které by uvítali.

- Možnost nastavit výrazné upozornění zobrazované na všech stránkách
- Při velkém vytížení možnost blokování zápisu soukromých desek
- Možnost vkládání a editace používaných materiálů

Původně už bylo počítáno s nastavováním položek:

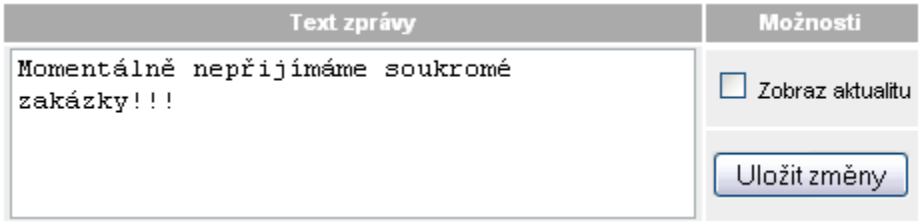
- Cena výroby DPS u soukromých projektů v Kč/dm^2
- Možnost změnit přístupová hesla

Všem výše uvedeným požadavkům se podařilo vyhovět. Navíc byla s výhledem do budoucna přidána volba umožňující povolení studentům osobně zadávat do systému DPS. Veškerá nastavení se budou provádět pomocí dalšího pluginu v podobě souboru *nastaveni.php*. Je však třeba vytvořit další tabulku v databázi, ve které budou nastavené parametry uchovány. Tabulka bude mít strukturu zobrazenou v Obrázek 10.14.



Obrázek 10.14 Přehled sloupců tabulky *nastaveni*

Sekce výrazné zprávy využívá sloupec *mess_set* pro uložení konkrétní zprávy a sloupec *mess_show_set* pro uložení, zda se má zpráva zobrazit či nikoliv. Právě takto se zpráva ovládá, viz Obrázek 10.15.



Obrázek 10.15 Nastavení výrazné zprávy

Sloupec *cena_set* uchovává cenu a sloupce *private_set, allow_stud_set* uchovávají buď hodnotu „ano“ ve smyslu povoleno, nebo „ne“ zakázáno.

- *private_set* – blokace/povolení soukromých desek
- *allow_stud_set* – blokace/povolení zadávání objednávek samotnými studenty

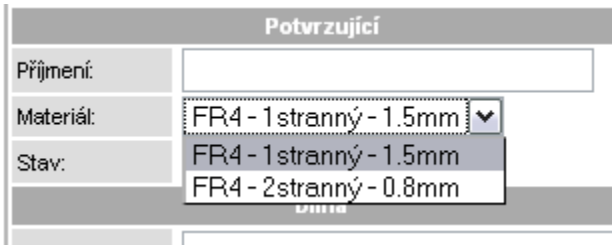
Tabulka pro uložení přístupových hesel a tabulka pro uchování všech materiálů již byla vytvořena dříve. Výpis tabulky materiálů na stránky vypadá následovně, viz. Obrázek 10.16.

Počet druhů materiálů: 3

Název	Typ	Tloušťka	Dostupný	Možnosti
FR4	1stranný	1.50	ano	-upravit-
FR4	2stranný	0.80	ano	-upravit-
Arlon 350	2stranný	0.79	ne	-upravit-

Obrázek 10.16 Ukázka výpisu tabulky materiálů

Výpis každého materiálu, je podobně jako výpis objednávek, zakončen odkazem *upravit*. Klinutím na odkaz se parametry materiálu načtou do formuláře, ve kterém lze tyto parametry měnit nebo lze celý záznam materiálu zcela smazat. Přidáním sloupce „dostupný“, by měli být pracovníci dílny ušetřeni od mazání a opětovného vložení materiálu, který např. není momentálně skladem. Po změně sloupce na „ne“ se příslušný materiál přestane objevovat v nabídce materiálů v zadávacím formuláři. (Obrázek 10.17)



Obrázek 10.17 Výběr materiálů v zadávacím formuláři

10.10 Vytvoření tabulkové struktury databáze

Během návrhu jednotlivých částí systému bylo zapotřebí postupně vytvořit čtyři tabulky. K vytvoření bylo použito webové aplikace phpMyAdmin, kde byly tabulky ručně zadány a postupně zadány i všechny jejich sloupce včetně určení typů vkládaných dat, defaultních hodnot a jiných voleb.

Pokud má být předkládaný systém uživatelsky přívětivý, nemůže chtít po případném provozovateli, aby před jeho spuštěním musel zdlouhavě nastavovat databázi s vědomím, že jediná chyba v nastavení může znásobit čas strávený při zakládání databáze.

Proto je systém doplněn dalším pluginem – soubor *create_db.php*. Ten obsahuje přednastavené SQL příkazy, které po spuštění vytvoří požadovanou

strukturu v zlomku vteřiny. Jediné co musí provozovatel předkládaného systému zadat, jsou parametry připojení k databázi MySQL. Tyto parametry se nastavují jako konstanty příkazem *define* hned na prvních řádcích souboru *index.php*.

Bylo by vhodné, aby byl PHP kód ošetřen pro případ, že by se nemohl připojit k databázi. V takovém případě by volání funkcí, které databázi využívají, skončilo v lepším případě bílým místem v šabloně, v horším případě vypsáním hrozivě vypadajících chybových hlášení vygenerovaných serverem PHP či databází MySQL.

Systém tedy bude vybaven mechanizmem, který toto ošetří. Při navazování spojení s databází se bude testovat výsledek tohoto pokusu. Pokud se spojení nezdaří, bude potlačeno generování chybových hlášení a místo toho se nastaví proměnná *$error*, která v tomto případě slouží jako jakýsi ukazatel. Po jeho nastavení se zruší vkládání pluginů do šablony a místo nich se vypíše hlášení o selhání spojení. Mohou nastat tři varianty selhání spojení, buď se vůbec nepodaří přihlásit k serveru MySQL nebo se nepodaří najít na serveru zvolenou databázi a nastavit ji jako pracovní anebo se v nastavené databázi nenajdou potřebné tabulky.

Výjimku tvoří právě plugin *create_db.php,* ten jediný má možnost se načíst i v případě chyby spojení. Je to z toho důvodu, že při prvním spuštění systému nejsou v databázi žádné tabulky, které má tento plugin za úkol vytvořit (pokud nebyly tabulky vytvořeny jinak). Po jeho spuštění bude uživatel požádán o zadání autorizačního hesla, jež se také zadává jako konstanta v souboru *index.php* a následně mu bude zobrazena tabulka s parametry, se kterými se systém pokusil připojit k databázi. To se buď nezdaří, pak je uživatel požádán o kontrolu těchto parametrů nebo se přihlášení zdaří a pak je nabídnuto vytvoření struktury tabulek. S vytvořením struktury, jsou zároveň do tabulek *nastaveni* a *tab_hesla* uložena defaultní data. Do tabulky *material* jsou uloženy tři vzorové materiály.

11 Bezproblémový chod databázové aplikace

S ohledem na zadání diplomové práce i s ohledem na možnost ostrého provozu této webové aplikace, bylo potřeba prozkoumat bezpečnostní „díry“ popisované databázové aplikace. Jedním z největších nebezpečí pro databázi je útok nazývaný SQL injection popisovaný v kapitole 6.1. Problém spočívá v tom, že syntaxe SQL dotazů nerozděluje data a příkazy zvlášť.

Bylo představeno několik možností jak tomuto útoku zabránit. Některé z nich může programátor aplikace používat sám, jiné může nastavovat správce webhostingu. Aplikace by na to měla být připravena.

Direktivu `magic_quotes_gpc` nemůže tvůrce skriptu nijak ovlivnit (pokud zároveň není i správcem webhostingu). Pokud je nastavena na „on“ escapují se všechny potenciálně nebezpečné znaky v globálních polích GET, POST a COOKIES (odtud zkratka *gpc* na konci názvu direktivy) zpětnými uvozovkami.

Pokud je tato direktiva nastavena na „off“, musíme tato data ošetřit některou ze zmíněných funkcí, například funkce addslashes(). Pokud by ovšem správce webhostingu najednou nastavil direktivu `magic_quotes_gpc` na „on“, escapovala by se data dvakrát a byla by tak znehodnocena. Je proto vhodné vytvořit si vlastní skript, který by toto ošetřoval. Nastavení zmiňované direktivy sice změnit ze skriptu nelze, ale lze se alespoň dotázat na jeho nastavení pomocí funkce get_magic_quotes_gpc(). Skript by mohl obsahovat ještě kontrolu délky ošetřovaného řetězce, aby se zamezilo vkládání delších řetězců, než je nastavena velikost sloupce v databázi (což by mohlo v některých případech generovat chybu). Následuje ukázka funkce, která předešlé realizuje.

```
function checkString ($data, $max_length = 20)
{
  if (!get_magic_quotes_gpc())   // pokud magic_quotes_gpc vyplé
      { $data = addslashes($data);}       // pouzij addslashes

, pak ho zkrati na $max_length
 if ( strlen($data)>$max_length)  //pokud je retezec delsi nez $max_length
      { $data = substr ($data ,0,$max_length);} //zkrat retezec

  return $data;
}
```

Takto mohou být ošetřeny proměnné, v nichž se nacházejí řetězce. Pro ošetření číselných hodnot musíme použít jiné mechanizmy. Bude též výhodné naprogramovat vlastní skript, který by nejen kontroloval obsah proměnné, ale opět by mohl kontrolovat i délku. Navíc i to, zda je číslo záporné či kladné a v případě nesplnění některých podmínek by funkce vrátila předem nastavenou defaultní hodnotu.

```
function checkNumber ($prom, $max_value, $def, $unsig)
{
 if (!is_numeric($prom))    // pokud proměnná není číslo
      { return $def;          // vrať defaultní hodnotu
      }
            // ošetření záporné hodnoty při vkládání do nezáporného typu
 if (($unsig == true)&&($prom < 0))
      { $prom = abs($prom);               // vratí absolutní hodnotu čísla
      //$prom = $def;         /  /2.možnost -vrátí defaultní hodnotu čísla
      }

         // kontrola maximální hodnoty
 if ($max_value!="nekontrolovat") // pokud se má kontrolovat
      {
            //--- ověření, zda není číslo mimo přípustný rozsah
        if ( abs($prom)>$max_value ) // pokud je číslo příliš velké
            { return $def;
            }
      }
return $prom;    // pokud vše v pořádku, vrátí původní číslo
}
```

Další problém nastane, pokud někdo do databáze uloží například značku <H1> ta se při výpisu z databáze vepíše do vygenerované stránky a rozhodí celý

layout stránky. Toto narušení stránky bývá označováno jako HTML injection. Proto je nutné všechny proměnné, které se vypisují na obrazovku nejprve ošetřit. Php pro tyto případy přímo nabízí funkci htmlspecialchars(), která znaky, které mají v hml speciální význam, nahrazuje jejich entitami.

- '<' je nahrazeno '<'
- '>' je nahrazeno '>'

:

Atd.

Obecné pravidlo pro udržení bezpečnosti na co nejvyšší úrovni, je důsledná kontrola všech vstupů a proměnných od uživatele.

12 Závěr

Cílem této diplomové práce bylo prozkoumat možnosti provozu a správy webového serveru, který by umožňoval provoz databázové aplikace umožňující snadnou možnost objednání výroby desky plošných spojů. A konkrétní realizace takového systému.

V této oblasti se dostává velké obliby především těm aplikacím, které jsou nezávislé na použitém operačním systému a které jsou volně šiřitelné. Většina z nich vznikla původně jako soukromé projekty, které se časem rozšířily v profesionální aplikace.

V oblasti databázových aplikací je to především databáze MySQL, založená právě na dotazovacím jazyku SQL. Mezi skriptovacími jazyky, které slouží k tvorbě dynamických www stránek, vede jazyk PHP, který též patří do rodiny volně šiřitelného softwaru a umožňuje poměrně snadnou spolupráci s databází MySQL. Tyto dvě aplikace pak velice často doplňuje http server Apache, který umožňuje celý databázový systém zpřístupnit na internetu či lokální síťi, prostřednictvím protokolu http. Kombinace těchto tří aplikací se velmi často distribuuje v rámci jednoho softwarového balíku.

Důležitou roli ve webových systémech hraje zabezpečení, jelikož s webovými aplikacemi se dostávají do styku nejen uživatelé, kterým mají webové aplikace především zjednodušit život ale i „uživatelé“ kteří chtějí případné bezpečnostní díry využít ve svůj prospěch, ať už pro získání citlivých informací, či k finančnímu prospěchu.

Na základě rozboru problematiky byla navržena tabulková struktura databázového systému. Pro vytvoření profesionálnějšího vzhledu byla použita grafická šablona a pro lepší pochopení problematiky a vytvoření uživatelsky co nejpřívětivější aplikace byl celý systém konzultován přímo s pracovníky dílny pro výrobu desek plošných spojů. Vytvořený databázový systém je schopen plnit účel, pro který byl vytvořen. HTML stránky, které jsou generovány, prošly kontrolou standardu W3C. Nicméně i nadále je co zlepšovat, inovovat a rozšiřovat jeho možnosti, tak jako ostatně u každého projektu.

.

13 Seznam použité literatury

[1] Bejček, V. (1992). *Databázové systémy.* Brno: VUT.

[2] Bráza, J. (2003). *PHP4 Učebnice základů jazyka.* Praha: Grada Publishing a.s.

[3] Broža, P., & Buranský, I. (2003). *Programování WWW stránek.* Brno: Computer Press.

[4] DeLisle, M. (2004). *phpMyAdmin efektivní správa MySQL.* Brno: Zoner Press.

[5] Lacko, L. (2001). *Web a databáze.* Praha: Computer Press.

[6] *PHP5, MySQL, Apache Vytváříme webové aplikace.* (2006). Brno: Computer Press, a.s.

[7] Jak psát web o tvorbě internetových stránek [online] dostupné z http://www.jakpsatweb.cz

[8] MySQL (1) - pestrý svět databází [online] dostupné z
http://www.linuxsoft.cz/article.php?id_article=731

[9] SQL Injection [online] dostupné z http://www.security-portal.cz/clanky/sql-injection.html

[10] Zabezpečení PHP / SQL – 2. Díl [online] dostupné z
http://www.acunetix.cz/websitesecurity/php-security-2.htm

14 Seznam obrázků a tabulek

OBRÁZEK 7.1 OKNO APLIKACE PHPMYADMIN 21
OBRÁZEK 8.1 OKNO APLIKACE CASE STUDIO 22
OBRÁZEK 9.1 PRVNÍ ČÁST PŮVODNÍ ZÁHLAVÍ PŮVODNÍ TABULKY 25
OBRÁZEK 9.2 DRUHÁ ČÁST ZÁHLAVÍ PŮVODNÍ TABULKY 25
OBRÁZEK 9.3 TABULKA NAVRŽENÁ DLE STARÉ STRUKTURY 26
OBRÁZEK 9.4 NOVĚ NAVRŽENÁ STRUKTURA TABULEK 27
OBRÁZEK 10.1 STRUKTURA SYSTÉMU NAVRŽENÁ KLASICKÝM ZPŮSOBEM 28
OBRÁZEK 10.2 VKLÁDÁNÍ SPOLEČNÉ HLAVIČKY. 29
OBRÁZEK 10.3 SYSTÉM VYUŽÍVAJÍCÍ HTML ŠABLONY 29
OBRÁZEK 10.4 ŠABLONA "BINARYNEWS" 30
OBRÁZEK 10.5 PŘIHLAŠOVACÍ FORMULÁŘ 32
OBRÁZEK 10.6 TABULKA HESEL ZOBRAZENÁ V PROGRAMU PHPMYADMIN 33
OBRÁZEK 10.7 VÝVOJOVÝ DIAGRAM ŘÍZENÍ OBSAHU ZADÁVACÍHO FORMULÁŘE 35
OBRÁZEK 10.8 FORMULÁŘ PRO ZADÁNÍ ZAKÁZKY - ÚČET *STUDENT* 35
OBRÁZEK 10.9 ROZŠÍŘENÍ ZADÁVACÍHO FORMULÁŘE PRO ÚČET *ZAMĚSTNANEC* 36
OBRÁZEK 10.10 ROZŠÍŘENÍ FORMULÁŘE PRO UŽIVATELE TYPU DÍLNA 36
OBRÁZEK 10.11 OZNÁMENÍ O NESPRÁVNĚ VYPLNĚNÝCH POLÍCH 36
OBRÁZEK 10.12 1.ČÁST VÝPISU OBJEDNÁVEK 38
OBRÁZEK 10.13 2. ČÁST VÝPISU OBJEDNÁVEK 38
OBRÁZEK 10.14 PŘEHLED SLOUPCŮ TABULKY *NASTAVENI* 39
OBRÁZEK 10.15 NASTAVENÍ VÝRAZNÉ ZPRÁVY 39
OBRÁZEK 10.16 UKÁZKA VÝPISU TABULKY MATERIÁLŮ 40
OBRÁZEK 10.17 VÝBĚR MATERIÁLŮ V ZADÁVACÍM FORMULÁŘI 40

TABULKA 4.1 RELACE 1:N 11
TABULKA 4.2 RELACE N:M 12
TABULKA 4.3 NULTÁ NORMALIZOVANÁ FORMA 12
TABULKA 4.4 PRVNÍ NORMALIZOVANÁ FORMA 13
TABULKA 4.5 ŠPATNĚ NAVRŽENÁ 2. NORMALIZOVANÁ FORMA 13
TABULKA 4.6 SPRÁVNĚ NAVRŽENÁ TABULKA V 2. NF 13
TABULKA 6.1 MOŽNÉ ATRIBUTY PRO ULOŽENÍ BINÁRNÍCH SOUBORŮ 17

15 Seznam zkratek

URL - Uniform Resource Locator – definuje adresu serveru a umístění požadovaného zdroje na serveru
HTML - HyperText Markup Language – značkovací jazyk pro vytváření webových stránek
DPS - Deska plošného spoje
UREL - Ústav Radioelektroniky
PHP - Hypertext preprocesor
SQL - Strukturovaný dotazovací jazyk (*Structured Query Language*)
CSS – Složky kaskádních stylů (*Cascading Style Sheets*)

16 Seznam příloh

A. Manuál k aplikaci ……..49

A. MANUÁL K APLIKACI

Obsah přílohy A. Manuál k aplikaci

1 Úvodní informace 51
1.1 Vložení objednávky desky plošných spojů 52
1.2 Výpis objednávek 54
1.2.1 Filtrování dat 54
1.2.2 Úpravy 54
1.2.3 Upozornění 55
1.2.4 Maximální a minimální rozměry 55
1.2.5 Předloha s motivem tištěného spoje 56
1.2.6 Odkaz VUT ID 56
1.2.7 Barevnost výpisu 56
1.3 Nastavení systému 56
1.3.1 Aktuální upozornění 56
1.3.2 Soukromé DPS 57
1.3.3 Druhy používaných materiálů 57
1.3.4 Obecné nastavení 57
1.3.5 Přístupová hesla 57
1.3.6 Statistika materiálů 58

1 Úvodní informace

Tato webová aplikace slouží jako nástroj pro efektivní správu objednání výroby desek plošných spojů (DPS). Je specificky upravený pro potřeby dílny Ústavu radioelektroniky (UREL) Fakulty elektrotechniky a komunikačních technologií (FEKT) Vysokého učení technického v Brně (VUT).

Výroba DPS je určena primárně pro zaměstnance pracující na různých projektech nebo pro studenty, jenž DPS potřebují v rámci svých semestrálních, bakalářských či diplomových prací. Pro podporu studentské tvořivosti je také dovoleno vyhotovovat DPS i pro soukromé účely. Taková zakázka je poté zpoplatněna. Je-li však dílna zaneprázdněna, může zadávání soukromých desek blokovat.

Aby ze stran studentů nedocházelo k přílišnému zneužívání služeb dílny, mají pracovníci dílny možnost blokovat zadávání objednávek samotnými studenty. Ti pak v případě potřeby řeší vložení své objednávky do systému s pomocí vedoucího své práce, či přímo s pracovníkem dílny.

Princip objednání DPS pro studenty je následující. Student v závislosti na aktuálním nastavení, sám zapíše, nebo nechá zapsat parametry požadované desky do databáze. Tuto objednávku mu v systému musí schválit (potvrdit) některý ze zaměstnanců UREL. Dále již objednávku přebírá dílna. Poté co student doručí dílně předlohu DPS vytištěnou na pauzovací papír, může být přistoupeno k výrobě.

U zaměstnanců probíhá zadání DPS k výrobě obdobně, s tím rozdílem, že schválení objednávky si provedou sami.

Z výše uvedeného vyplívá, že systém obsluhují tři typy uživatelů – studenti, zaměstnanci a dílna. Pro odlišení svých práv v systému má každý z nich svůj specifický účet s vlastním heslem.

Pro výrobu DPS si každý zájemce musí nachystat následující údaje:

- Rozměry DPS v milimetrech (o maximálních možných rozměrech se informujte přímo v dílně).
- Stručný popis/název projektu.
- Případně obrázek předlohy pro ve formátu JPG, PNG nebo GIF (není bezpodmínečně nutný).

Pokud se zadává školní projekt, pak je možno DPS vyhotovit v tzv. rozšířené variantě. To znamená, že je možno vyrobenou DPS:

- Pocínovat
- Vyvrtat díry
- Osadit prokovy
- Nanést nepájivou masku

Procedury při výrobě rozšířených desek jsou relativně zdlouhavé, proto by se jejich použití mělo volit v rozumné míře i u školních projektů.

Dále se v každé objednávce vyplňuje jméno a příjmení objednavatele a pokud možno i VUT ID (unikátní identifikační číslo každého studenta VUT). Zadáním tohoto čísla se urychlí vyhledávání záznamů v databázi a v případě výrobních problémů, také snazší kontaktování zadávajícího. Studenti své VUT ID zadávají povinně.

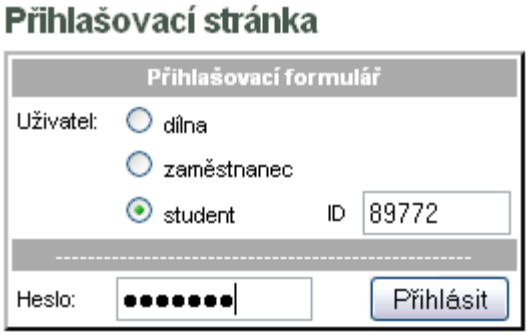
Nyní je namístě shrnout několik informací týkajících se objednávky.

- Zakázka může být soukromá nebo školní, oba typy musí někdo potvrdit:
  - školní schvaluje vedoucí práce
  - soukromou schvaluje dílna nebo jiný zaměstnanec UREL
- Soukromá zakázka je zpoplatněna.
- Zakázka je v systému vždy označena jedním ze čtyř stavů:
  1. „zadáno“ - zakázka jen vložena do systému
  2. „schváleno“ - zakázka je potvrzena zaměstnancem
  3. „přijato“ – dílna přijala zakázku a začíná její výrobu
  4. „hotovo“ – zakázka je připravena k vyzvednutí

1.1 Vložení objednávky desky plošných spojů

Po vstoupení na stránky se ocitnete na přihlašovací stránce. (případně klikněte v menu na tlačítko „Přihlášení“)

Na přihlašovací stránce je formulář jako na obr. 1

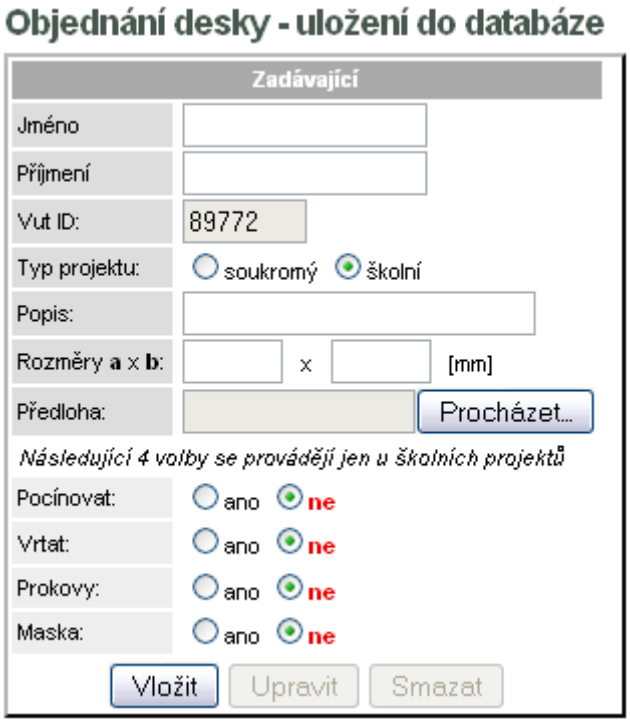


obr. 1 Formulář pro přihlášení do systému

Zatrhneme typ účtu a zadáme příslušné heslo. Student ještě vyplní své VUT ID. Při špatně zadaných hodnotách se objeví chybové hlášení.

V případě úspěšného přihlášení se vypíše oznámení a systém Vás přesměruje na stránku, kde je výpis všech objednávek. Je-li přihlášen student, zobrazí se mu pouze objednávky odpovídající jeho ID. V menu úplně napravo se objeví tlačítko „odhlásit“ spolu s informací o typu používaného účtu. Studentům se vedle tlačítka objeví jejich VUT ID.

Pro objednání nové DPS klikněte v menu na odkaz „Vložit objednávku“. Objeví se formulář, který je na obr. 2.



obr. 2 Formulář pro zádání objednávky

Chce-li si DPS objednat student, jsou dvě možnosti jak toto provést a to v závislosti na momentálním nastavení aplikace, které provádí pracovníci dílny.

Pokud je povoleno, aby student mohl parametry své zakázky vkládat do systému osobně, zadá požadované parametry a stiskne tlačítko uložit. Stav zakázky je v systému automaticky uložen jako „zadáno“. Poté společně s vedoucím své práce, případně přímo s pracovníkem dílny vyhledá v seznamu svoji zakázku, z rozevíracího seznamu vybere vhodný typ materiálu, ze kterého se DPS vyrobí a objednávku potvrdí, což dotyčný zaměstnanec provede překlopením jejího stavu na „schváleno“ a stisknou tlačítko „upravit“.

Pokud je možnost vkládání zakázek studentům blokováno, student vyhledá vedoucího svého práce a ten mu parametry do systému vloží. Soukromé zakázky vkládají do systému pracovníci dílny. Stav pak rovnou označí jako „schváleno“ a uložení zakázky provedou stisknutím tlačítka „uložit“.

Stav „schváleno“ je pokynem pro dílnu, že může přistoupit k výrobě. Poté kdy zadávající dodá předlohu plošného spoje (obrázek plošného spoje vkládaný spolu s ostatními parametry, je čistě pro kontrolu a nestačí k započetí výroby), označí dílna zakázku jako „přijato“. Následně už není možné parametry zakázky změnit.

V tento moment již zadávající jen čeká, až se mu ve výpisu objednávek ukáže u dané objednávky stav „hotovo“.

Ještě je třeba podotknout, že systém se snaží hlídat zadávání vstupních parametrů a v případě opomenutí některého z nich, nabídne zadávací formulář předvyplněný Vámi zadanými parametry s upozorněním, kde je potřeba provést opravu dat. Pokud systém shledá zadaná data jako platná, žádné chybové hlášení se neobjeví a uživateli se zobrazí výpis všech objednávek.

1.2 Výpis objednávek

Kliknutím v menu na odkaz „Výpis objednávek“ se v aplikaci objeví tabulka s výpisem objednávek, viz obr. 3. Dle typu účtu, pod kterým je uživatel přihlášen, se může lišit množství takto zobrazených objednávek.

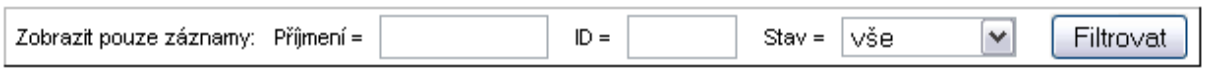
Datum poslední změny	Zadávající		Rozměry *a* x *b* [mm]		Popis objednávky	Schválil	Typ projektu	Stav	Materiál	Cín	Vrtat	Prokov	Maska	Cena [Kč]	Možnosti
	Jméno Příjmení	VUT ID													
2009-05-26	Jiří Procházka	89772	100	13	regulovatelný zdroj	Ostřížek	školní	schváleno	FR4 1stranný	ano	ano	ano	ano	0.0	-upravit-
2009-05-26	Jiří Procházka	89772	100	13	NF zesilovač	Kučera	školní	přijato	FR4 1stranný	ano	ne	ano	ne	0.0	-upravit-
2009-05-26	Jiří Procházka	89772	70	13	blikacka	Novotný	školní	hotovo	Arlon 350 2stranný	ne	ano	ano	ne	0.0	-upravit-
2009-05-26	Jiří Procházka	89772	12	14	zkušební DPS		soukromý	zadáno	FR4 2stranný	ne	ne	ne	ne	35.0	-upravit-
2009-05-26	Jiří Procházka	89772	67	45	nová deska	Karský	soukromý	hotovo	FR4 1stranný	ne	ne	ne	ne	35.0	-upravit-

obr. 3 Výpis objednávek

Zatímco uživateli typu student se vypíší pouze objednávky odpovídající jeho VUT ID, uživateli s účtem zaměstnanec a dílna budou vypsány zakázky všechny.

1.2.1 Filtrování dat

Pro snazší hledání je nad výpisem objednávek umístěn filtr.



obr. 4 Filtr výpisu objednávek

Z obr. 4 je patrno, že můžeme objednávky filtrovat dle tří parametrů: *příjmení*, *VUT ID* a *stavu* každé objednávky. Nastavením parametrů a stisknutím tlačítka „Filtrovat“ se záznamy znovu načtou z databáze, tentokráte již s aplikovaným filtrem dle nastavených parametr. Počet vypsaných záznamů se vždy zobrazuje vpravo nad tabulkou. Uživateli s účtem student je filtrování dle příjmení a VUT ID blokováno, jelikož se mu zobrazují jen jeho zakázky.

1.2.2 Úpravy

Každý řádek výpisu objednávek je zakončen kolonkou, v níž je odkaz „upravit“. Po kliknutí na něj se zobrazí stejný formulář, který byl vyplňován již při zadání zakázky a uživatel tak dostává možnost svoji zakázku dodatečně upravit. Po uložení oprav, stačí stisknout tlačítko „upravit“.

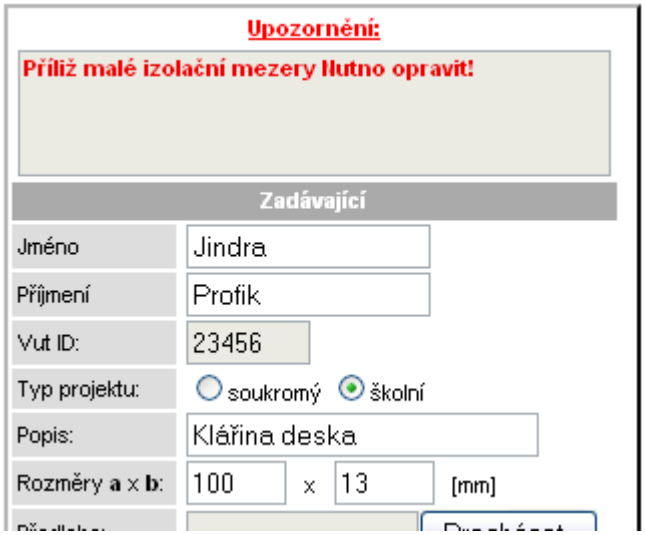
Jestli-že uživatel není přihlášený pod účtem dílna, nelze upravovat zcela všechny objednávky. Není umožněno upravovat zakázku, která již je označena stavem „přijato“ či „hotovo“. Úprava zakázky, která se právě vyrábí nebo už je hotová, nemá význam.

Dále je třeba pamatovat na to, že pokud uživatel student změní parametry své zakázky, je tato uložena opět jen jako „zadaná“ a je potřeba znovu zajistit její schválení.

1.2.3 Upozornění

Pokud dílna dodatečně zjistí nesrovnalosti nebo nečekané problémy při výrobě zadané DPS, má možnost ke každé jednotlivé zakázce připsat upozornění. Upozornění se uživateli zobrazí ve výpisu objednávek pod zadanými parametry jeho zakázky (viz obr. 5) a také v horní části zadávacího formuláře (obr. 6) po kliknutí na odkaz „upravit“ na konci výpisu objednávky.

obr. 5 Zobrazení upozornění ve výpisu objednávek



obr. 6 Zobrazení upozornění v zadávacím formuláři

1.2.4 Maximální a minimální rozměry

Pokud jsou při zadání překročeny maximální povolené rozměry DPS, které dílna vyrábí, systém tuto objednávku sice umožní uložit, ale rozměr, který byl překročen, podbarví červeně a do kolonky upozornění, o které byla řeč v předchozím odstavci, vypíše zprávu: „*Překročeny maximální rozměry. Nicméně konzultujte s dílnou možnost výroby této desky*“. Je to z toho důvodu, že dílna za určitých okolností dovede vyrobit DPS, která překračuje rozměrové limity. Je proto potřeba se na tyto okolnosti informovat. Pokud se pracovníci dílny rozhodnou, že se pokusí tuto DPS vyrobit, je po nastavení stavu objednávky na „přijato“ hlášení o překročených rozměrech smazáno.

Naopak existuje minimální plocha materiálu, která se při výrobě spotřebuje. Plocha je počítána automaticky ze zadaných rozměrů a pokud je menší než dané minimum, pak se pro případný výpočet ceny použije právě tato minimální plocha i když je skutečná plocha výsledné DPS menší.

1.2.5 Předloha s motivem tištěného spoje

Byla-li při uložení do databáze vložena i předloha dané DPS, bude zobrazena po kliknutí na „popis objednávky“, který se v případě přítomnosti předlohy zobrazí jako odkaz. Předloha je zobrazena v dalším okně, nerespektuje skutečné měřítko, jelikož má jen čistě kontrolní charakter.

1.2.6 Odkaz VUT ID

Je-li u objednávky vloženo i VUT ID, zobrazí se ve výpisu jako odkaz. Po kliknutí na něj se přesměrujeme na stránky VUT, kde se zobrazí profil daného uživatele. Dílna tak má v případě potřeby možnost zjistit např. emailovou adresu.

1.2.7 Barevnost výpisu

Je třeba dodat, že podbarvení *stavu* zakázky ve výpisu je pouze pro zpřehlednění výpisu a neznamená žádnou výstrahu.

- Stav „zadáno“ zůstává podbarven stejně jako ostatní políčka řádku.
- Stav „schváleno“ je podbarven červeně.
- Stav „přijato“ je podbarven žlutě.
- Stav „hotovo“ je podbarven zeleně.

Stejně tak barevné písmo parametrů „Cín“, „Vrtat“, „Prokov“ a „Maska“ má čistě zpřehledňovací charakter, aby bylo na první pohled zřetelné, které procedury se budou vykonávat.

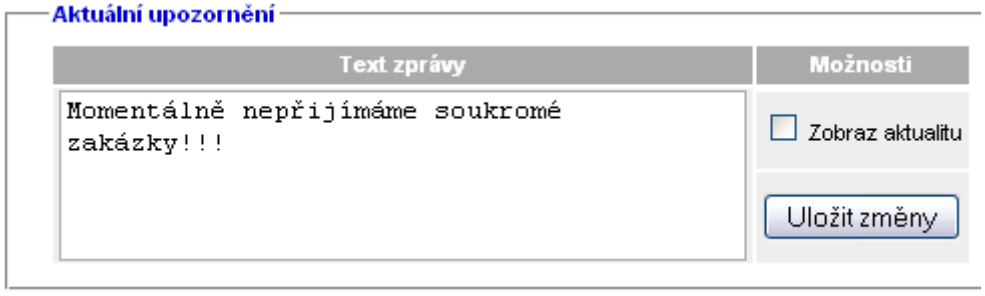
1.3 Nastavení systému

Pracovníci dílny (obecně uživatelé systému s účtem „dílna“) mají nejvyšší úroveň práv. Mohou tedy měnit parametry celého systému v závislosti na aktuální situaci.

Pro tyto účely se účtu „dílna“ zobrazí po přihlášení do systému ještě další odkaz v menu s názvem „Nastavení“. Tato stránka obsahuje různá nastavení rozdělená do šesti sekcí.

1.3.1 Aktuální upozornění

V této sekci (obr. 7) mají pracovníci dílny možnost zobrazit důležité oznámení či upozornění. Toto upozornění se zobrazí v horní části každé stránky. Pro zobrazení je potřeba vepsat sdělení do připraveného textového pole a zároveň zatrhnout políčko „zobrazit aktualitu“. Pro zrušení zobrazení stačí toto políčko odznačit, přičemž text upozornění zůstane uložen pro případné další použití.



obr. 7 Aktuální upozornění

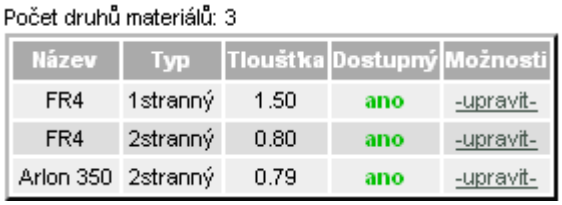
1.3.2 Soukromé DPS

První nastavení v sekci soukromé DPS se týká situace, kdy je dílna zavalena spoustou zakázek, školní zakázky mají v tomto případě přednost, dílna proto může zablokovat zadávání soukromých projektů.

Jako druhá volba v této sekci je nastavení ceny soukromých zakázek v korunách za 1 dm^2.

1.3.3 Druhy používaných materiálů

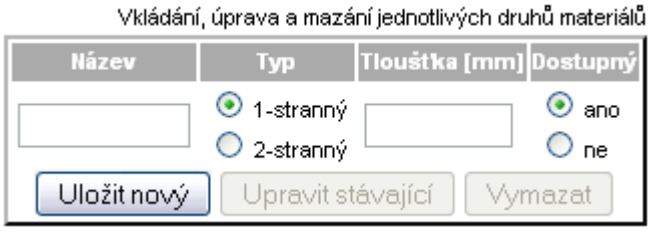
V levé části je seznam všech materiálů (obr. 8) vložených do systému spolu s jejich základními parametry.


Počet druhů materiálů: 3

Název	Typ	Tloušťka	Dostupný	Možnosti
FR4	1stranný	1.50	ano	-upravit-
FR4	2stranný	0.80	ano	-upravit-
Arlon 350	2stranný	0.79	ano	-upravit-

obr. 8 Vzhled tabulky materiálu

Po kliknutí na odkaz „upravit“ na konci každého řádku se vybraný materiál načte do formuláře napravo (obr. 9). Zde jej můžeme upravit či smazat. Vhodnější než smazat je ale označit ho jako nedostupný volbou „ne“ v sloupci „Dostupný“. Takovýto materiál již dále nebude nabízen jako jedna z možností při vkládání nových zakázek, ale zároveň zůstane zobrazován u již vyřízených objednávek. Formulář pro úpravu jednotlivých materiálů, zároveň slouží i pro vkládání nových.



obr. 9 Formulář pro úpravu parametrů materiálů

1.3.4 Obecné nastavení

V této sekci se nastavuje zda-li mohou uživatelé účtu *student* samostatně vkládat parametry svých zakázek.

1.3.5 Přístupová hesla

V této sekci se mění přístupová hesla jednotlivých účtů. Systém je nastaven tak, že minimální délka vkládaného hesla je 6 znaků.

Tlačítko „Ukázat hesla“ zobrazí v malé tabulce typy účtů a jednotlivá hesla.

1.3.6 Statistika materiálů

Poslední sekce neslouží pro nastavení, ale poskytuje základní informace o použití jednotlivých druhů materiálů, počet desek, celkovou plochu spotřebovaného typu materiálu, atd.