Univerzita Pardubice

Fakulta ekonomicko-správní

Centrální evidence prostředků výpočetní techniky

Milan Mládek

Bakalářská práce

2010

Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Milan MLÁDEK**

Studijní program: **B6209 Systémové inženýrství a informatika**

Studijní obor: **Informatika ve veřejné správě**

Název tématu: **Centrální evidence prostředků výpočetní techniky**

Zásady pro vypracování:

Rozbor typů používaných prostředků VT v daném subjektu (hardware, software)
Určení atributů vhodných pro kategorizaci a vyhledávání a vazby na účetní doklady
Vypracování návrhu databáze (ER-model), Use-case rozbor, realizace a implementace

Rozsah grafických prací:

Rozsah pracovní zprávy: **30 - 40 stran**

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

Seznam odborné literatury:

WELLING, Luke, THOMSON, Laura. PHP a MySQL : Rozvoj webových aplikací. 3 : SoftPress, 2005. 830 s. ISBN 0-86497-83-6.

GUTMANS, Andi, SAETHER BAKKEN, Stig, RETHANS, Derick. Mistrovství v PHP 5. [s.l.] : Computer Press, 2007. 656 s. ISBN 978-80-251-1519-0.

LAVIN, Peter. PHP : Objektově orientované. 1. vyd. [s.l.] : Grada, 2009. 224 s. ISBN 978-80-247-2137-8.

Vedoucí bakalářské práce: **Ing. Oldřich Horák**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **5. října 2009**

Termín odevzdání bakalářské práce: **30. dubna 2010**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 5. října 2009

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 23. dubna 2010

Milan Mládek

Poděkování

Zde bych velice rád poděkoval Ing. Oldřichu Horákovi za čas, který mi při tvorbě práce věnoval, cenné připomínky a odborné rady, kterými přispěl k vypracování této praktické i textové části práce.

Anotace

Práce je věnována návrhu a realizaci webové aplikace pro evidování prostředků výpočetní techniky. V práci se vyskytuje také teoretická část, zabývající se technologiemi, které jsou využívány při tvorbě webových aplikací. V praktické části se nachází návrh datového modelu a realizace samotné aplikace.

Klíčová slova

evidence, výpočetní technika, hardware, software

Title

Central Evidence of Computer Engineering

Anotation

This project is devoted to design and implementation of web application for evidence of computer engineering. In excess there is a theoretical part which is dealt with technologies using in production of web applications. There is design of data model and realization of application itself in the practical part.

Keywords

evidence, computer technologies, hardware, software

Obsah

Úvod

Nástup informačních technologií a prostředků výpočetní techniky za posledních několik let zapříčinil mnohem rychlejší koloběh nákupu, nutnosti modernizace a s tím i souvisejícího vyřazování výpočetní techniky. Veškeré tyto změny ve strukturách prostředků výpočetní techniky, které v libovolné společnosti či organizaci nastávají, je nutné zaznamenat, tedy zaevidovat. Vzhledem k velké rozmanitosti typů výpočetní techniky a softwarových produktů je mnohdy náročné komplexně a korektně evidovat všechny potřebné vlastnosti a změny těchto prostředků v rámci jednotlivých subjektů. A právě této problematice se věnuje má bakalářská práce.

Pomyslnou pomoc při evidování prostředků výpočetní techniky mohou tvořit právě informační technologie, které tyto prostředky poskytují. Jelikož cílem informačních technologií je převážně ukládání, sběr a vyhodnocování informací, nemohly by se obejít bez prostředků, jež tyto požadavky splňují. V minulosti dané požadavky splňovaly kartotéky, které umožňovaly třídit a uspořádávat data. Operace s nimi však byly provozovány manuálně přímo člověkem. Souběžně s nástupem informačních technologií pak přišel mocný nástroj splňující dané požadavky na ukládání dat v mnohem širší míře. Tímto nástrojem jsou databáze, běžnému uživateli skrytý systém, „kdesi“ za rozhraním, se kterým komunikuje. Vyspělé společnosti si bez informačních technologií a především bez databází zpracovávajících data snad ze všech lidských oborů a činností, neumí představit svoji práci. A to obzvlášť v době, kdy informačním technologiím a databázím „přispěchal“ na pomoc třetí pomocník – internet. Ač by se internet dal sám o sobě považovat za informační technologii, v kombinaci s databázemi vytvořil nový rozměr možností práce s informacemi. Díky internetu je možné v podstatě odkudkoliv spravovat téměř jakákoliv data. Třeba i data o výpočetní technice využívané ve vlastní firmě či v jiné organizaci.

Za hlavní náplň práce jsem si však nepoložil sběr teoretických poznatků z této problematiky, ale praktické řešení, které by právě díky databázovým systémům a webovým technologiím umožnilo jednoduché evidování prostředků výpočetní techniky.

1 Vybrané vlastnosti PHP

Části této kapitoly nejsou věnovány základním informacím o programovacím jazyku PHP, ale věnují se především vybraným možnostem, které byly alespoň zčásti využity při realizaci praktické části práce. Důvodem k výběru těchto částí by také mohlo být nastavení vytvořené aplikace, které musí odpovídat požadavkům uvedeným v kapitole 2.2.3 nebo v nápovědě, kterou aplikace obsahuje. V této části jsou probrány tyto celky:

- superglobální proměnné
- definice vlastních funkcí a procedur
- práce s poli
- vybrané funkce pro práci s databázovým systém MySQL

1.1 Superglobální proměnné

S pojmem „superglobální“ proměnná se můžeme v PHP setkat od verze 4.1.0. Jedná se o několik předdefinovaných polí, které obsahují proměnné ze strany serveru nebo zajišťují pomoc při uživatelských vstupech.

V minulosti docházelo k automatické registraci proměnných posílaných například v URL adrese. Například z URL adresy `http://www.stranka.cz?var=1` byla ihned při vstupu na tuto stránku zaregistrována proměnná `$var` o hodnotě 1. Automatická registrace proměnných je velice snadno zneužitelná, a proto je od verze 4.2 vypnuta. Tuto službu však lze zpětně zapnout nastavením direktivy `register-globals`. Prakticky tedy přidáním řádky „`php_flag register_globals on`“ do souboru .htaccess, který musí být umístěn v adresáři, jež se vztahuje k webové aplikaci (nebo části aplikace), u které má být tato možnost povolena. [1]

Problém s bezpečným předáváním proměnných mezi jednotlivými částmi aplikace vyřešila definice superglobálních proměnných. Jak již bylo napsáno výše, jedná se o využití předdefinovaných polí, která jsou již zpočátku automaticky globální. Tato pole lze tedy zavolat ze skriptu odkudkoliv. Na následujících řádcích je vypsán výčet těchto polí.

$_GET – Metoda GET zpracovává všechny proměnné, které jsou poslány přes URL adresu. K obsahu jednotlivých proměnných je možné se dostat přes klíč pole `$_GET`. Tímto klíčem je

samotný název proměnné posílané přes URL adresu. Pro příklad u adresy `http://www.stranka.cz?var=1` je klíčem řetězec 'var' a obsah proměnné lze tedy vypsat následovně:

```
echo $_GET['var'];
```

$_POST – Touto metodou jsou zpracovávány formulářové prvky webových aplikací. Proměnné se z vyplněného či nastaveného formuláře posílají pomocí HTTP protokolu. Klíčem k proměnným v poli `$_POST` je obsah atributu `NAME` u odeslaných formulářových prvků. Pro vysvětlení lze uvést tento příklad:

```
<form method="post" action="skript.php">
  Místo narození: <input type="text" name="location">
  <input type="submit">
</form>
```

Na stránce skript.php je pak následujícím kódem vypsán vyplněný obsah formuláře.

```
echo $_POST['location'];
```

$_SESSION – Relace (sessions) jsou v PHP plně podporovány od verze 4.0. Jedná se o další možnost předávání proměnných mezi jednotlivými stránkami aplikace, přičemž tyto proměnné nemusí být součástí URL adresy nebo formuláře. Relace jsou v aplikaci spouštěny příkazem `session_start` (pokud není spouštění sessions automatické). Při vykonání tohoto příkazu interpret PHP zahájí novou relaci a přidělí jí jednoznačný identifikátor. V případě, že nějaká relace již existuje, tak se k ní připojí. Rozlišení jednotlivých relací probíhá na základě cookies nebo předáním identifikátoru relace URL adresou. Na každé stránce, na které jsou spouštěny relace, je pak možné využívat proměnné, jež uživatel uloží do globálního pole `$_SESSION`. Uložení proměnné do pole `$_SESSION` je naznačeno níže. [2]

```
session_start();
$_SESSION['promenna']='jmeno';
```

$_COOKIE – Pomocí metody cookies jsou na počítači, na kterém pracuje uživatel, ukládány soubory, jejichž obsah může být později využit. K obsahu těchto souborů má přístup jen samotný uživatel a jeho webový prohlížeč. S využitím této metody se většina uživatelů setkává velice běžně. Jedná se například o automatické vyplnění již dříve vyplněného formuláře nebo přenosu proměnných mezi stránkami. Ukládání těchto souborů musí být povoleno

v prohlížeči a samotné vytvoření probíhána základě příkazu `setcookie("nazev","obsah",doba_platnosti_cookie)`. Poslední parametr je nepovinný a udává, do jakého data a času bude daná cookie platná, tedy použitelná. [3] S jednotlivými cookies pak můžeme pracovat například takto:

```
echo $_COOKIE['promenna'];
```

$_FILES – Tato metoda slouží k získávání souborů od samotných uživatelů. Pole `$_FILES` není, na rozdíl od předchozích globálních polí, jednorozměrné, ale dvourozměrné. Jednotlivé možnosti tohoto pole jsou následující:

Tabulka 1: Superglobální pole $_FILES [zdroj: 4]

pole	význam
$_FILES["soubor"]["name"]	Jméno souboru
$_FILES["soubor "]["size"]	Velikost udávaná po bytech
$_FILES["soubor "]["type"]	MIME typ souboru
$_FILES["soubor "]["tmp_name"]	Dočasné jméno souboru
$_FILES["soubor "]["error"]	Chybová hláška

$GLOBALS – Do tohoto pole je možné ukládat proměnné, které mají být použity globálně. Tedy viditelně pro celý skript. Má ekvivalentní použití jako deklarace proměnné jako `global`. Klíčem tohoto pole je název proměnné. [4]

```
$x = 1;
function PrictiPet(){
  $vysledek = $GLOBALS['x']+5;
  return $vysledek;
}
```

Pokud by v předchozím příkladě byla proměnná `$x` nastavená pouze lokálně, tak by volání funkce skončilo chybou, jelikož proměnná `$x` není v této funkci definována.

$_REQUEST – Tato metoda pracuje se všemi vstupními proměnnými, respektive zdroji vstupů (GET, POST, COOKIES), najednou. V případě, že je použito více proměnných se stejnými názvy, tak je registrována pouze jedna proměnná. V direktivě `variables-order` je možné změnit pořadí zdrojů, ve kterém budou proměnné registrovány. [5]

$_SERVER – Pole proměnných, které obsahuje základní informace o prostředí, ve kterém je aktuální skript prováděn. [4]

$_ENV – Další pole, obsahující stejně jako pole `$_SERVER` serverem odeslané proměnné a informace o prostředí, ve kterém byl skript spuštěn. [4]

1.2 Definice vlastních funkcí a procedur

Při samotném programování se často může stávat, že je nutné některé části kódu vícekrát opakovat. U dlouhých a mnohdy složitých kódů by mnohačetná opakování přispěla pouze k horší čitelnosti a složitosti celého skriptu. Stejně jako většina programovacích jazyků má i PHP možnost definice vlastních funkcí a procedur.

Klíčovým slovem pro definování vlastní funkce či procedury je slovo `function`. Rozdíl mezi procedurou a funkcí je poté dán pouze v samotné definici, ve které se u funkcí vrací hodnota pomocí slova `return`. Jednoduchá ukázka definice vlastní funkce je níže. [6]

```
function VypisDatum(){
    $datum = StrFTime("%d.%m.%Y");
    echo ($datum);
}
```

Kdekoliv ve skriptu je poté možné zavolat proceduru `VypisDatum()`, která nám vypíše aktuální datum. V častých případech je ovšem nutné pro samotnou funkci uvést její vstupní vlastnosti, dle kterých bude pracovat nebo je využije. K tomuto účelu je možné s funkcí definovat i její vstupní parametry. Parametry funkcí lze pak předávat dvěma způsoby:

a) předávání parametru hodnotou,
b) předávání parametru odkazem.

Předávání parametru hodnotou je základním způsobem práce s parametry při definování vlastních funkcí. V této možnosti je brána v potaz samotná hodnota proměnné, jež vstupuje jako parametr do funkce. [6]

```
$hodnota = 0.5;
function Procenta($hodnota){
    $vysledek = $hodnota*100;
    return $vysledek;
}
```

V uvedeném příkladu je jako parametr brán obsah proměnné `$hodnota`, konkrétně číslo 0.5, přičemž se obsah této proměnné nezmění. V případě, kde by bylo nutné změnit i samotnou hodnotu proměnné, která vstupuje do funkce jako parametr, je využíván druhý způsob předávání parametru, a tím je **předávání parametru odkazem**. Při tomto typu předávání proměnné uvažujeme tedy i její změnu během vykonání samotné funkce. Před proměnné, které jsou předávané odkazem, je psán při výčtu parametrů znak „`&`". [6]

```
function Zmena($x){
    $x = 50;
}
$hodnota = 2500;
Zmena(&$hodnota);
echo $hodnota;
```

Po vypsání proměnné `$hodnota` je vypsáno číslo 50, protože je tato proměnná do funkce `Zmena(&$hodnota)` zavolána odkazem.

1.3 Pole v PHP

Jelikož v další podkapitole je rozebrána práce s MySQL, ve které jsou mnohé výběry z databáze realizovány pomocí polí, tak je tato podkapitola věnována základním mechanismům polí v PHP.

Práce s poli patří k nejzákladnějším možnostem většiny programovacích jazyků. V jazyce PHP možnost práce s poli samozřejmě nechybí a je pro ni v samotném jazyce předdefinováno několik desítek funkcí. O polích se dá zjednodušeně napsat, že se jedná o skupinu nebo množinu prvků spolu souvisejících. Polem by se tak dala označit například řada čísel, znaků nebo i řetězců. Jednotlivé prvky pole jsou dále identifikovány pomocí svého klíče

v konkrétním poli. V PHP nemusí být klíčem pouze číslo, ale třeba i znak nebo řetězec. Pole se stejně jako jiná data ukládají do proměnných, se kterými lze dále pracovat. Asi nejzákladnější způsob pro vytvoření pole je ten, ve kterém je použito klíčové slovo `array`. [7]

```
$pole = array("jaro","léto","podzim","zima");
```

V tomto krátkém kódu jsou do proměnné `$pole` uloženy názvy ročních období. Ve specifikaci daného pole však není žádná informace o klíčích k jednotlivým prvkům. Pro tento případ pak platí, že klíče takového pole jsou předdefinovány celým číslem. První položka pole má pak klíč o hodnotě 0, druhá 1 atd. Řádkem uvedeným níže lze pracovat s konkrétní položkou pole. V tomto případě bude položka vypsána. Kód konkrétně vypíše slovo „léto“.

```
echo $pole[1];
```

V případech, kde je potřeba mít klíč v jiném tvaru než číselném nebo kde je například třeba číslovat klíče jinak než od 0, je nutné tyto skutečnosti již ve specifikaci pole uvést a to například následujícími způsoby:

```
 $pole = array(5=>"jaro","léto","podzim","zima");
$pole2 = array("leden"=>1,"unor"=>2);
```

Klíče pole v proměnné `$pole` jsou tentokrát číslovány od čísla pět. Ve druhém příkladu jsou naopak klíči v proměnné `$pole2` řetězce „leden“ a „unor“.

K usnadnění procházení polí mohou sloužit cykly. Často využívaným cyklem pro práci s poli je cyklus `foreach`. Kód pro výpis pole by pak mohl vypadat například takto:

```
$pole = array("jaro","léto","podzim","zima");
foreach($pole as $hodnota){
     echo "Roční období: ".$hodnota."<br>";
}
```

Jak již bylo výše řečeno, PHP disponuje pro práci s poli několika desítkami funkcí. Následující funkce jsou vybrány zejména proto, že jsou použity i v praktické části práce.

Funkce **count()**, jejímž argumentem je pole, vrací počet prvků, ze kterých je dané pole složené. K tomuto účelu lze také použít funkci **sizeof()**. [8]

```
$pole = array("jaro","léto","podzim","zima");
echo count($pole); // vypíše číslo 4
```

V případě, kde je nutné zjistit na základě hodnoty pole její klíč, je možné použít metodu **array_keys()**. Jako parametry slouží proměnná pole a volitelně vyhledávaná hodnota, jejíž klíč v poli hledáme. Výsledkem je pak pole odpovídajících klíčů. Výpisem z následujícího kódu budou řetězce „pondeli“ a „utery“. [9]

```
$pole = array("pondeli"=>"prace","utery"=>"prace",
              "sobota"=>"volno");
$keys = array_keys($pole,"prace");
foreach ($keys as $hodnota){
     echo $hodnota. "<br>";
}
```

Dalšími užitečnými funkcemi by mohly být například **in_array()** zjištující, zda dané pole obsahuje hledanou hodnotu. Dále několik druhů třídění (sort) pole podle hodnot, klíčů a jiné. Asi nejvýznamnější využití polí v PHP nastává při práci s databázovými daty, která jsou v mnohých případech zpracována v poli. O této problematice pojednává další podkapitola. [10]

1.4 Práce s MySQL

Mnohé webové aplikace potřebují během svého provozu uchovávat veliké množství dat, která jsou přijímána od uživatelů prostřednictvím formulářových prvků. Tato data jsou dále zpracována programovacím jazykem PHP a jeho prostřednictvím použita v databázi. Nejoblíbenějším databázovým prostředkem, který je v PHP podporován, je databázový systém MySQL. PHP disponuje několika desítkami funkcí pro využití tohoto databázového systému. Na následujících řádcích je popsáno několik vybraných funkcí včetně příkladů práce s nimi. Byly vybrány především funkce, které jsou využity v praktické části práce.

mysql_connect otevírá spojení s databázovým serverem. Toto spojení je nutné pro veškerou další práci s databází. V případě, že se připojení k databázovému systému nezdaří, vrací tato funkce hodnotu FALSE.

mysql_select_db spojuje systém s konkrétní databází, která je v něm umístěna. Funkce vrací TRUE nebo FALSE v závislosti na úspěchu připojení. V případě nevyužití této funkce je nutné se na využívanou databázi odkazovat jiným způsobem, a to například jmenováním databáze v SQL dotazech, jak je uvedeno níže. [11]

```
SELECT * FROM database.table
```

mysql_close ukončí spojení s databázovým systémem. Jako parametr této funkce vystupuje konkrétní spojení, které má být uzavřeno. V případě, že není daný parametr uveden, je uzavřeno poslední použité spojení. Použití této funkce však není povinné.

Díky předchozím třem prostředkům je možné se připojovat a uzavírat spojení s databázovým serverem. Praktická ukázka spojení s MySQL by mohla vypadat například takto:

```
$connect = mysql_connect('localhost','user','password');
if (!$connect)
    die ('Nastala chyba při připojování k databázi.');
mysql_select_db('database',$connect);

//práce s databází

mysql_close($connect);
```

V proměnné `$connect` je uloženo připojení k databázovému systému. Parametry funkce `mysql_connect()` udávají adresu databázového systému, jméno uživatele a heslo. V případě, že se spojení nepodaří, je uživateli zobrazena hláška o této události. Následuje výběr databáze s názvem ‚database' a samotná práce s ní, po které je spojení uzavřeno.

Základem veškeré práce s databázovým systémem je zpracování SQL dotazů a tvorba výstupů z databázových tabulek do webového rozhraní. K těmto účelům mohou posloužit následující funkce.

mysql_query odesílá SQL dotaz databázovému serveru ke zpracování. Parametrem je tedy SQL dotaz a dále pak volitelně konkrétní připojení k databázi. U SQL dotazů typu SELECT, DESCRIBE atd. vrací funkce FALSE v případě neúspěšného provedení dotazu. V případě

úspěšného provedení dotazu vrací typ „`resource`“. U SQL dotazů typu INSERT, DELETE, UPDATE apod. vrací funkce hodnoty TRUE nebo FALSE. [12]

mysql_num_rows vrací počet záznamů, které odpovídají výsledku daného SQL dotazu. Využití této funkce má smysl pouze u SQL dotazů typu SELECT. Naopak u dotazů typu INSERT, DELETE, UPDATE se více vyplatí využívat funkci **mysql_affected_rows**, která vrací počet záznamů, jež byly ovlivněny posledním SQL dotazem tohoto typu.

mysql_insert_id vrátí skriptu poslední vygenerované id, respektive poslední hodnotu sloupce, který je automaticky inkrementován. Funkce může tedy sloužit například jako vazba pro zpětnou kontrolu posledního přidaného záznamu.

mysql_fetch_array je jedna z mnoha funkcí využitelných k výslednému zobrazení výsledků dotazu. Funkce sama o sobě sice nic „nezobrazuje“, ale načítá výsledek dotazu do asociativního pole. Asociativní pole je takové pole, jehož klíčem nemusí být celé číslo, ale například i řetězec. V případě neúspěchu vrací funkce hodnotu FALSE. [13]

Příklad s využitím všech předchozích funkcí by pak mohl vypadat následovně:

```
$sql = "INSERT INTO zamestnanci(jmeno,prijmeni,bydliste)
        VALUES ('Pavel','Novák','Pardubice')";
mysql_query($sql);
$id = mysql_insert_id();
$select = "SELECT * FROM zaměstnanci WHERE id = '$id'";
$result = mysql_query($select);
while ($zaznam = mysql_fetch_array($result)) {
     echo 'jméno: '.$zaznam['jmeno'].'<br>';
     echo 'příjmení: '.$zaznam['prijmeni'].'<br>';
     echo 'bydliště: '.$zaznam['bydliste'].'<br>';
}
```

Ve výše uvedeném příkladu je již předpokládáno vyřešené připojení k databázi. Nejdříve je do databáze přidán nový záznam, konkrétně se jedná o zaměstnance. V tabulce zaměstnanci je také uvažován „skrytý“ atribut id nastavený na autoinkrementaci a je i klíčem této tabulky. Pro zpětnou kontrolu je proveden výpis posledního přidaného záznamu. Klíč posledního záznamu je zjištěn pomocí funkce `mysql_insert_id()` a následně použit v SQL dotazu. Výsledek dotazu je v cyklu načten do pole `$zaznam` pomocí funkce

`mysql_fetch_array()` a poté jsou jednotlivé položky pole, jehož klíči jsou názvy sloupců tabulky, v tomto cyklu vypsány. Funkci `mysql_num_rows()` nemá smysl v daném příkladu využívat. V případě, že by na konci tohoto kódu byl uveden ještě řádek „`echo mysql_num_rows($result);`“, tak by na výslednou stránku bylo vypsáno číslo 1, protože ze základního předpokladu jedinečnosti atributu id by mohl být nalezen pouze jediný záznam.

Účelem této podkapitoly nebyl podrobný výpis všech možností a funkcí, které PHP pro databázový systém MySQL poskytuje, ale pouhé teoretické nastínění možností propojeni PHP a MySQL. Úplný výčet funkcí a možností v této oblasti je k nalezení na adrese http://php.net/manual/en/book.mysql.php, ze které jsou i informace pro tuto podkapitolu čerpány. [14]

2 Realizace

Tato kapitola je věnována praktické části bakalářské práce. Celá práce je nejdříve analyzována a poté dochází k samotnému řešení problému. Během postupu řešení tohoto problému je navržen databázový model na všech třech úrovních datového modelování. Daný model je implementován do konkrétního databázového prostředí a využíván praktickou aplikací. V textu se kromě textového popisu vyskytují i schémata, diagramy, úryvky kódu a obrázky z výsledné aplikace.

2.1 Analýza problému

Evidováním prostředků výpočetní techniky se rozumí evidence osobních počítačů, serverů, notebooků a k nim připojených zařízení včetně obsaženého hardwaru a softwaru v libovolné společnosti či organizaci.

V nejzákladnějším pojetí se jedná o evidenci osobních počítačů, notebooků a serverových stanic, jejichž asi nejpoužívanějším identifikátorem je inventární číslo, které je používáno ve formátu, jež je dán samotnou organizací. Způsob fyzického označení jednotlivých počítačů či stanic je záležitostí samotné organizace. Nejčastěji se jedná o nalepovací štítky, případně ručně psané označení na samotných stanicích. Seznamy těchto označení (inventárních čísel) mohou být dále k dispozici v jednotlivých místnostech či budovách formou vytištěného seznamu, tabulek v tabulkových procesorech (MS Excel, Quattro Pro, Gnumric aj.), v libovolných textových dokumentech (doc, docx, pdf aj.), případně mohou být uloženy formou webového dokumentu vytvořeného v nějakém značkovacím jazyce (html, xml) v interním sítovém systému v dané organizaci. Tyto seznamy mohou být dále hlavním zdrojem informací především pro účetní jednotky dané organizace, správce sítě, administrátory, případně jiné osoby, které mají výpočetní techniku v daném subjektu na starost. Inventární číslo ovšem nemusí být zdaleka jediným atributem, který je důležitý pro evidování. Účetní jednotky v dané organizaci může například zajímat datum nákupu, označení dokladu o nákupu či vyřazení, cena nebo informace specifikující odpovědnost konkrétní osoby za daný počítač či hardware. Naopak pro správce organizační sítě mohou být důležité informace o záruční době jednotlivých kusů hardwaru a počítačů, poznámky o změnách umístění či funkčnosti těchto jednotek.

U hardwaru a zařízení připojených k jednotlivým počítačům je fyzické označování inventárními čísly již různorodé, a to především z důvodů velké rozmanitosti těchto typů zaří-

zení. Například u velkých tiskáren či monitorů není problém označení nalepovacím štítkem, ovšem u fyzicky menších nebo elektronicky citlivějších typů hardwaru (procesor, datové kabely, základní desky) je fyzické označení štítkem či popiskou prakticky nemožné. Z těchto důvodů je lepší evidovaný hardware zobrazovat v k tomu určených seznamech, a to nejlépe v seznamech elektronických, které kromě lepší přehlednosti, mohou nabídnout v rámci databázových systémů i lepší přehled o fyzickém propojení například s počítači. Taktéž vzhledem k velkému množství kategorií a typů hardwaru je podstatné rozlišovat detailnost, s jakou budou kategorie hardwaru evidovány. Detailnost evidování závisí na pohledu samotného subjektu, který samotné evidenci provádí. Různé stupně detailnosti evidování hardwaru lze tedy chápat jako subjektivní stupně pohledu na to, co by měla evidence prostředků výpočetní techniky obsahovat v rámci konkrétních zařízení a hardwaru. Některým osobám, jež mají výpočetní techniku v organizacích na starost, tedy může vyhovovat evidování nejvýše jednotlivých osobních počítačů a stanic, jiné však mohou požadovat komplexní přehled o hardwaru v počítačích, například až do úrovně kabeláže. Takovéto komplexní informace však lze jen velmi náročně zobrazit na papír. Připočteme-li si k tomu množství atributů, jež mohou být velmi podobné těm požadovaným u osobních počítačů, které bychom chtěli o daném hardwaru a zařízení zaznamenávat, tak je nejvhodnějším řešením využití databázového systému (MS Access, MS SQL Server, MySQL, postgreSQL aj.) s propojením na nějaké rozhraní (například webové), které bude interně v rámci organizace k tomu určeným osobám k dispozici.

Dalším možným objektem vhodným k evidování v rámci výpočetní techniky jsou zakoupené softwarové produkty. Kromě označení softwaru (například opět inventárním číslem) je u většiny softwarových produktů podstatným atributem pro správce sítě počet zakoupených licencí, případně typ licence. Typem licencí nejsou v tuto chvíli myšleny jeho druhy distribucí (freeware, demo, shareware aj.), ale spíše možnosti rozdělování počtu zakoupených licencí určitého softwaru. Nejčastější formou jsou licenční klíče, jejichž zadání do příslušného vstupního formulářového prvku je podmínkou k využití daného softwarového produktu. Licenční klíč je přitom zakoupen přímo se samotným softwarovým produktem. V případě tzv. multilicence je jeden licenční klíč využíván k provozu zakoupeného softwarového produktu na více počítačích. Další možnou formou rozdělování licencí softwaru mezi jednotlivé stanice, by mohlo být přerozdělování licencí licenčním serverem. Jde o tzv. licence „per connection“, které skrz licenční servery povolují určitý počet aktivních připojení k určitému softwarovému produktu. Evidování softwaru může být tedy velice důležité, jak pro účetní jednotku (data

nákupů, ceny, doklady), tak i pro samotné správce sítě, kteří mohou využívat údaje o licencích a aktuální přehledy instalací softwaru na jednotlivých počítačích. Tyto informace by stejně jako u hardwaru bylo náročné přenést přehledně na papír, a proto je lepší využití informačních technologií. Právě možností ukládat informace o počítačích, hardwaru a softwaru pomocí informačních technologií (konkrétně formou webové aplikace) se zabývá praktická část mé práce.

2.1 Požadavky na aplikaci

Vzhledem k analýze problematiky evidování prostředků výpočetní techniky byly stanoveny následující požadavky, které budou podrobněji rozepsány.

Nejpodstatnějšími požadavky na aplikaci jsou její jednoduchost, přehlednost a nenáročnost na hardwarové prostředky. Vzhledem k povaze bakalářského studia oboru Informatika ve veřejné správě a absolvovaných předmětech Technologie internetu, Databázové systémy a Tvorba WWW stránek, byla jako forma praktické části zvolena webová aplikace tvořená za podpory skriptovacích jazyků PHP a Javascript, značkovacího jazyka HTML, kaskádových stylů CSS a databázového systému MySQL. Podmínkou pro provoz této aplikace je tedy pouze webový server podporující PHP a databázový systém MySQL a webový prohlížeč. Konkrétní verze webového serveru a webových prohlížečů jsou uvedeny v kapitole 2.2.3 případně v nápovědě k aplikaci.

Obsahem aplikace je souhrn základních funkcí pro evidování počítačů, k nim připojeného hardwaru a softwarových produktů.

Pro práci s počítači se jedná o možnosti přidávání počítače do evidence, přehledu a zadávání vstupních údajů počítače, úpravě těchto údajů a vyřazování počítačů z evidence. U správy počítačů v evidenci se dále jedná o jednoduché přehledy hardwaru, který je konkrétnímu počítači v současné době přiřazen, softwaru, který je na počítač nainstalován a přehledům změn v historii, jež se u konkrétního počítače v rámci přiřazeného hardwaru a nainstalovaného softwaru udály.

U evidování hardwaru a zařízení aplikace poskytuje možnosti přidávání hardwaru, zadání vstupních údajů, přiřazování a přeřazování k jednotlivým počítačům ve vazbě 1:N

(k počítačům může být přiřazeno libovolné množství zařízení, ale každé zařízení může být přiřazeno právě k jednomu počítači), vyřazování a přehled o současném začlenění a historii dané položky. V rámci hardwaru aplikace poskytuje i jednoduché členění typů hardwaru, které je jedním z atributů.

V rámci evidování softwaru je mezi možnostmi přidávání softwarových produktů, zadávání vstupních údajů o těchto produktech, úprava vstupních údajů, přiřazování (instalace) těchto softwarových produktů na jednotlivé počítače a přehledy o současném i minulém stavu vybraného softwarového produktu.

K ostatním možnostem patří jednoduché vyhledávání položek na základě vstupních atributů, zobrazování seznamů užívaných i již vyřazených položek, u počítačů a hardwaru jednoduchý systém kontroly záruční doby a u softwaru kontrola počtu užitých licencí.

2.2 Návrh databáze

Celý proces návrhu databázového modelu až k jeho realizaci prošel všemi úrovněmi, a to konkrétně:

- konceptuální úrovní
- technologickou úrovní
- implementační úrovní

V následujících třech podkapitolách je uveden popis návrhu všemi uvedenými úrovněmi.

2.2.1 Konceptuální úroveň

První částí návrhu databázového systému je určení entit a vztahů mezi nimi. V rámci evidování prostředků výpočetní techniky jsou určeny následující entity a jejich atributy. Podtržené atributy jsou jednoznačnými identifikátory (klíči) dané entity.

Počítač

Entita počítač je nejzákladnější entitou rámce evidování prostředků výpočetní techniky a tudíž i celé aplikace. Touto entitou jsou reprezentovány osobní počítače, notebooky, případně jiné počítačové stanice. Tato entita také vstupuje do vztahu s entitami software a hardware.

Atributy: id_pc, inventární číslo, název, cena, číslo dokladu, datum nákupu, záruční doba, odpovědná osoba, umístění, vyřazení, datum vyřazení, důvod vyřazení, číslo dokladu (k vyřazení), datum zápisu

Hardware

Entita hardware reprezentuje veškeré fyzické vybavení, které je přímo připojeno k počítači nebo s ním úzce souvisí. Prakticky se tedy může jednat například o tiskárny, síťové karty, procesory nebo o datovou kabeláž. Tato entita vstupuje do vztahu s entitou počítač.

Atributy: id_k, inventární číslo, název, typ, cena, číslo dokladu, datum nákupu, záruční doba, odpovědná osoba, umístění, vyřazení, datum vyřazení, důvod vyřazení, číslo dokladu (k vyřazení), datum zápisu

Software

Entita software reprezentuje v tomto datovém modelu všechny softwarové produkty, které mohou být nainstalovány na počítač. Tato entita tedy také vstupuje do vztahu s entitou počítač.

Atributy: id_sw, inventární číslo, název, cena, číslo dokladu, datum nákupu, počet licencí, typ licence, vyřazení, datum vyřazení, důvod vyřazení, číslo dokladu (k vyřazení), datum zápisu

Jelikož aplikace dovoluje uživateli přidávat položky daných entit se stejnými inventárními čísly, a to bez ohledu na druh entity, které se inventární číslo týká, nemohlo se vlastně inventární číslo stát jednoznačným identifikátorem. Je tedy nutné zavést uměle vytvořené identifikátory id_pc pro počítače, id_k pro hardware a zařízení a id_sw pro software. Podle těchto skrytých identifikátorů se aplikace rozhoduje mezi konkrétními položkami databáze. Tyto identifikátory pak slouží výhradně pro potřeby programových kódů a SQL dotazů a nemají tedy pro uživatele aplikace žádný význam. Pro uživatele podstatným identifikátorem, podle kterého vybírá konkrétní položky, je především inventární číslo, které ovšem nemusí být identifikátorem jednoznačným. Využití jednoho inventárního čísla pro více položek databáze však není doporučeno.

Výše definované entity vstupují mezi sebou do vzájemných vztahů. Tyto vztahy mezi entitami je nutné definovat, jelikož i zobrazení těchto vztahů poslouží k návrhu modelu reálného světa, v tomto případě k modelu evidování prostředků výpočetní techniky.

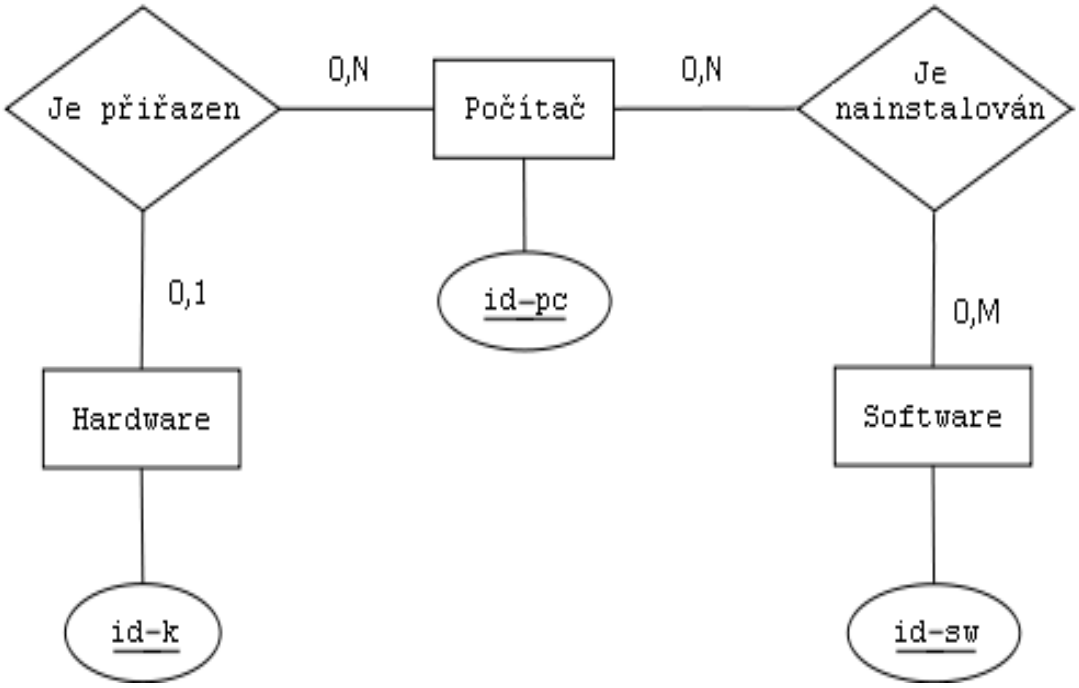
Přiřazen

Tato entita popisuje vztah mezi entitami hardware a počítač. Prakticky se jedná o fyzické připojení komponent či jiných zařízení k počítači. Entita počítač je s entitou hardware v nepovinném vztahu ve vazbě 1:N. Tedy na jeden počítač je možné přiřadit více kusů hardwaru. Ovšem jeden kus hardwaru může být v konkrétní době přiřazen právě k jednomu počítači.

Nainstalován

Entita nainstalován, která popisuje vztah mezi entitami software a počítač, reprezentuje fyzickou instalaci softwarových produktů na jednotlivé počítače. Entita software je, stejně jako entita hardware, v nepovinném vztahu s entitou počítač, ale ve vazbě M:N. Software, respektive konkrétní softwarové produkty, mohou být nainstalovány v konkrétní dobu na více počítačích a zároveň na jednom počítači může být nainstalováno více různých softwarových produktů.

V tuto chvíli jsou již všechny entity a jejich vzájemné vztahy definované. Je tedy možné vytvořit schéma takzvaného „Entity-Relationship modelu“ (E-R diagram).



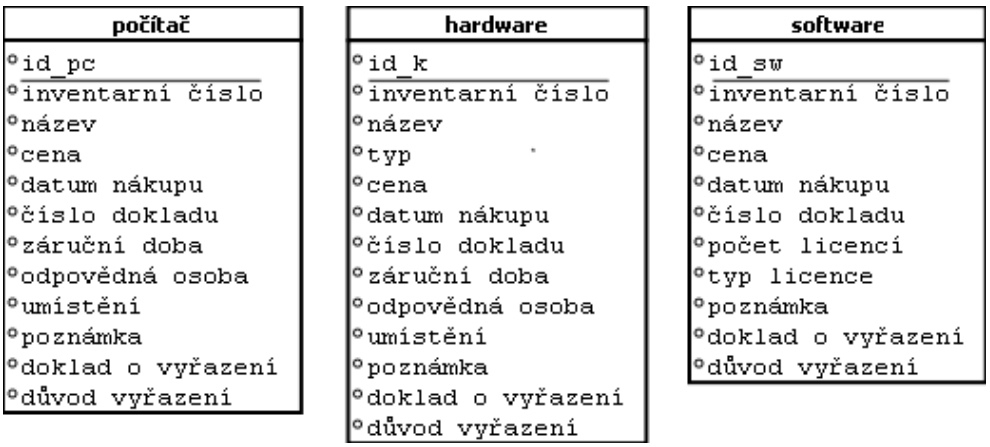
Obrázek 1: E-R diagram [zdroj: vlastní]

2.2.2 Technologická úroveň

Vzhledem k tomu, že na implementační úrovni návrhu databáze je konkrétně použit databázový systém MySQL, tak je na této úrovni nutné se zabývat relačním modelem dat.

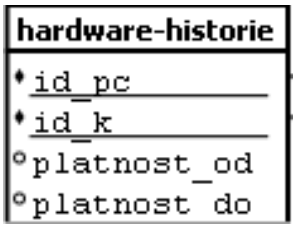
Relační model dat je v současné asi nejpoužívanějším modelem, a to především díky své přizpůsobivosti a efektivnosti. Tento model reprezentuje data jako takzvané relace. Relace je tabulka, která obsahuje n-tice, což jsou řádky, tabulky a sloupce, které reprezentují atributy dané relace. Relační model dat v současné době využívají například databázové systémy MS SQL Server, Oracle Database nebo již zmíněné MySQL. Výhodou relačních databází je využití SQL jazyka**.**

Transformací entit z konceptuální úrovně datového modelování vznikly následující relace:



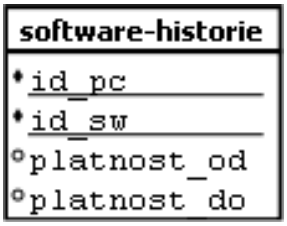
Obrázek 2: Základní relace [zdroj: vlastní]

Kromě transformace entit je dále nutná i transformace vztahů, které byly mezi těmito entitami. Vztah „přiřazen“ mezi entitami počítač a hardware je transformován do tří relací, protože jde o vztah s vazbou 1:N s nepovinným členstvím v tomto vztahu. Takto proběhnutá transformace vytvoří tedy tři relace, a to relaci počítač a hardware, které jsou již popsány výše a dále relaci hardware-historie. Tato relace je vztahová a obsahuje jednoznačné identifikátory relací hardware a počítače, tedy atributy id_k a id_pc. Tato relace dále obsahuje atributy platnost od a platnost do, které budou popsány části zabývající se implementační úrovní databázového modelu.



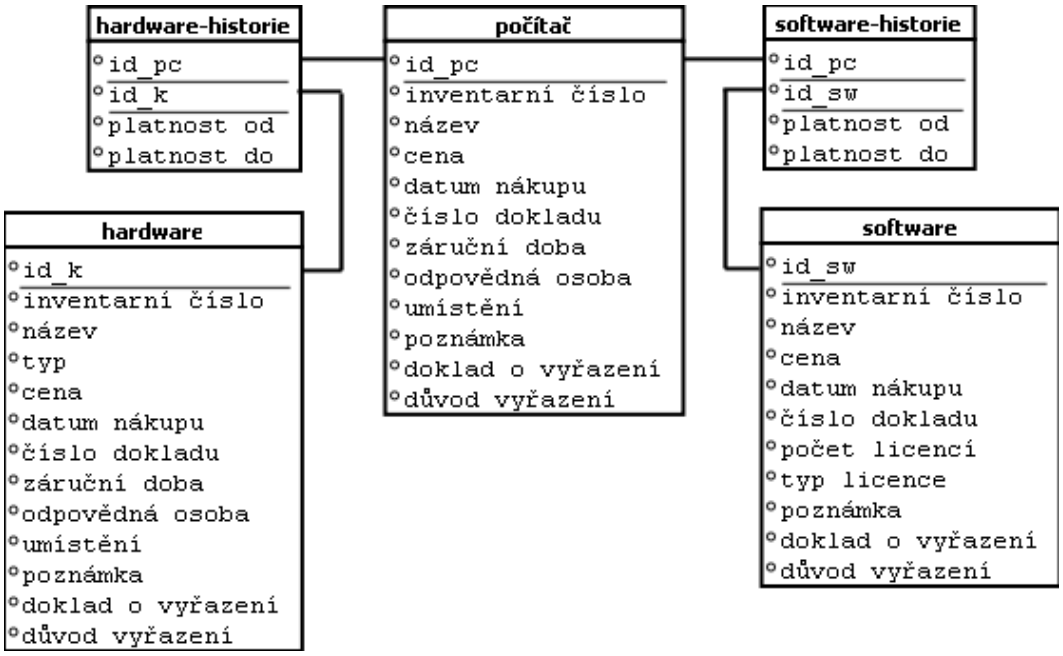
Obrázek 3: Relace hardware-historie [zdroj: vlastní]

Vztah nainstalován, který vznikl mezi entitami počítač a software, je na technologické úrovni datového modelování transformován do tří relací, jelikož jde vztah s vazbou M:N s nepovinnou účastí obou entit. Z této transformace vzešly konkrétně relace počítač, software, které byly již popsány a dále relace software-historie, která obsahuje jednoznačné identifikátory obou entit, jež vstupují do tohoto vztahu, tedy atributy id_pc a id_sw. Stejně jako u transformace předchozího vztahu přiřazení, obsahuje vztahová relace software-historie další atributy, a to platnost od a platnost do.



Obrázek 4: Relace software-historie [zdroj: vlastní]

Relační schéma

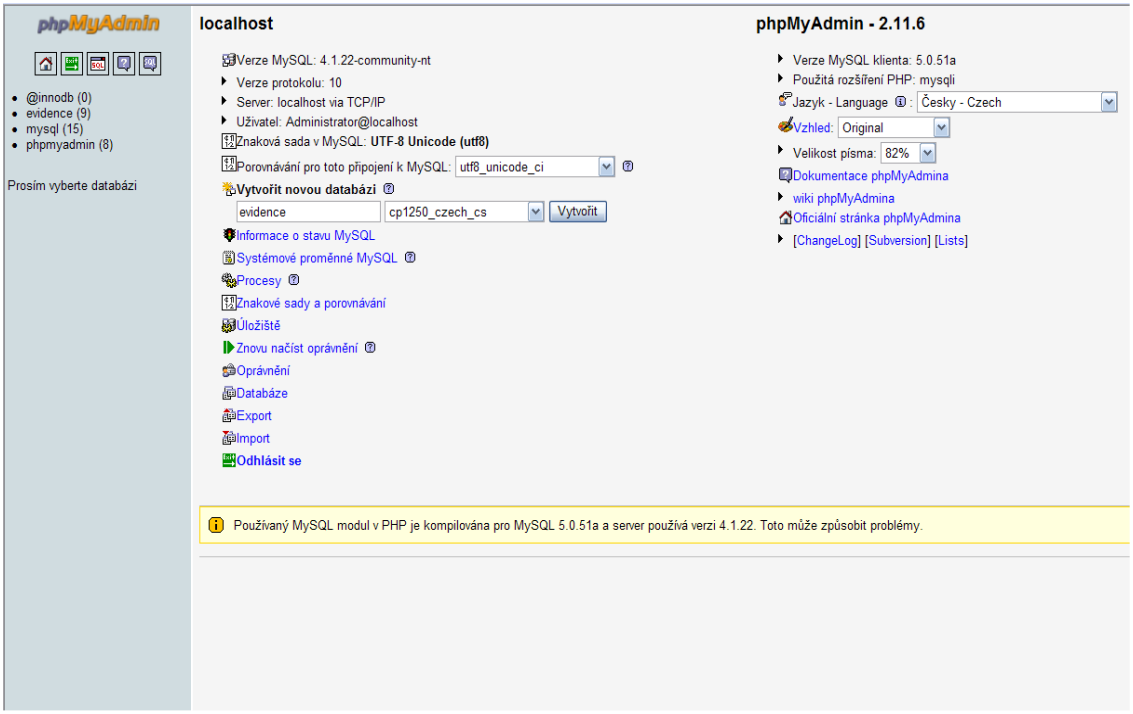


Obrázek 5: Relační schéma [zdroj: vlastní]

2.2.3 Implementační úroveň

Na této úrovni datového modelování jsou již popsány konkrétní nástroje a aplikace, které byly v rámci tvorby databáze využity.

Jak již bylo řečeno, k vlastní tvorbě a užití aplikace je použit hojně využívaný databázový systém MySQL. Při vlastní tvorbě byla využita verze MySQL 5.1.30. Vlastní instalace samotné aplikace není vůbec složitá. Aplikace si totiž databázové tabulky vytvoří sama již při prvním spuštění, tedy při prvním vstupu na stránku index.php. Před tímto krokem je však nejdříve nutné založit nebo vybrat databázi, do které se tyto tabulky implementují.



Obrázek 6: Práce v phpmyadmin [zdroj: vlastní]

Poté je potřeba otevřít soubor connect.php v kořenovém adresáři aplikace a nastavit přihlašovací údaje k databázovému systému. Tento soubor lze otevřít v Notepadu, případně v jakémkoliv jiném programu, který je určen na tvorbu webových aplikací. Obsah tohoto souboru vypadá následovně:

```
<?php
  $adr = "host";
  $user = "user";
  $pwd  = "password";
  $db  = "nazev_databaze";
?>
```

Slova host, user a password je nutné přepsat na skutečné hodnoty. Tedy namísto slova host se uvádí adresa webového serveru, slovo user se přepisuje uživatelským jménem, které je použito pro přístup k databázovému systému MySQL a slovo password je nutné přepsat heslem k tomuto účtu. K proměnné `$db` musí být přiřazen správný název vybrané databáze. Tento soubor je dále v programu volán pomocí PHP funkce `require_once()` na každou stránku, která vyžaduje spojení s databázovým serverem a samotné spojení je uskutečňováno za pomoci funkce `mysql_connect()`. Po tomto kroku, tedy po zadání vstupních údajů do souboru connect.php, je již možné v internetovém prohlížeči otevřít stránku index.php. Při

prvním pokusu o vstup na tuto stránku hrozí vteřinové „zaseknutí“ vstupu na stránku, jelikož se v tomto kroku v databázovém systému vytvoří databázová struktura, kterou samotná aplikace nutně využívá. V případě selhání této operace je uživatel upozorněn výstražnou hláškou a je nutné vstup na stránku index.php opakovat, dokud uživatel není vpuštěn na úvodní stránku aplikace. Po zbytek chodu aplikace by nemělo být nutné vstupovat manuálně do databáze (například pomocí databázového nástroje phpmyadmin).

K provozu aplikace je dále nezbyné splňovat tyto požadavky:

- webový prohlížeč Mozilla Firefox, Opera, IE 7 a vyšší (jiné nejsou otestovány),
- povolený Javascript,
- vypnutá automatická registrace proměnných (standartně vypnuto),
- MySQL 5.1.x,
- PHP 5.x.

Vytvoření celé databázové struktury je splněno na základě obsahu souboru create_db.php, který je opět funkcí `require_once()` otevřen na úvodní stránce aplikace. Během prvního spuštění aplikace tedy dojde k použití několika SQL dotazů. Například:

```
CREATE TABLE IF NOT EXISTS komponenty_zarizeni
         (id_k INT NOT NULL AUTO_INCREMENT,
          inventarni_cislo VARCHAR(25) NOT NULL,
          nazev VARCHAR(60) NULL,
          typ VARCHAR(25) NULL,
          cena INT NULL,
          cislo_faktury VARCHAR(25) NOT NULL,
          datum_nakupu DATE NULL,
          zaruka INT NULL,
          osoba VARCHAR(25) NULL,
          id_pc VARCHAR(15) NULL,
          umisteniINT NULL,
          poznamka VARCHAR(85) NULL,
          vyrazeni BIT DEFAULT 0 NULL,
          datum_vyrazeni DATETIME NULL,
          duvod_vyrazeni VARCHAR(65) NULL,
          c_dokladu VARCHAR(255) NULL,
          datum_zapisu DATETIME NOT NULL,
          PRIMARY KEY (id_k,id_pc))
```

Obsahem celé databáze je 9 tabulek, jejichž vytvoření, funkce a struktura jsou popsány níže.

Popis databázových tabulek a jejich atributů

Tabulka *komponenty_zarizeni* – Tato tabulka je přejmenovanou relací hardware, která je popsána v podkapitole technologické úrovně tvorby datového modelu.

Tabulka 2: Databázová tab. komponenty_zarizeni [zdroj: vlastní]

komponenty_zarizeni		
Atribut	**Datový typ**	**Popis**
id_k	INT	Identifikátor jednotlivých komponent a zařízení. Hraje roli v rámci programových kódů aplikace. Pro uživatele nemá význam.
id_pc	INT	Cizí klíč. Jedná se identifikátor počítače, ke kterému je hardware přiřazen. V případě nepřiřazeného hardwaru nabývá hodnoty nula. Pro uživatele nemá opět žádný význam.
inventární_cislo	VARCHAR(25)	Inventární číslo je pro uživatele zásadním atributem. Na základě tohoto atributu uživatel vyhledává jednotlivé komponenty. Atribut může být libovolného formátu, ale musí být u každé komponenty uveden.
nazev	VARCHAR(60)	Názvem je myšlen řetězec pro bližší specifikaci zařízení. V názvu může být uveden i výrobce. Např.: VGA Gigabyte GeForce 9800GT
typ	VARCHAR(25)	Atribut určený pro bližší specifikaci typu evidovaného hardwaru. Tento atribut má vazbu na tabulku *typy_komponent*.
cena	INT	Cena, za kterou byl daný hardware pořízen.
cislo_faktury	VARCHAR(25)	Atribut, který může být vazbou pro účetní jednotku v dané organizaci. Určuje konkrétní označení dokladu.
datum_nakupu	DATE	Konkrétní datum nákupu komponenty.
zaruka	INT	Záruční doba, která se vztahuje ke konkrétnímu zařízení. Pro správnou funkci v aplikaci by měla být zadávána v měsících.
osoba	VARCHAR(25)	Specifikuje dle jména a příjmení nebo osobního identifikátoru odpovědnou osobu za dané zařízení.
umisteni	VARCHAR(20)	Atribut upřesňuje fyzické umístění konkrétního kusu hardwaru.
poznamka	VARCHAR(85)	Dodatečné poznámky k danému zařízení.
vyrazeni	BIT	Bitový atribut určující, zda je konkrétní zařízení vyřazené. Nabývá hodnoty 0 pro zařízení užívané, hodnoty 1 pro zařízení vyřazené. Uživatel nemá k tomuto atributu přístup.
datum_vyrazeni	DATETIME	Určuje datum vyřazení daného hardwaru.
duvod_vyrazeni	VARCHAR(65)	Specifikuje důvod vyřazení daného hardwaru.
c_dokladu	VARCHAR(15)	Atribut může být vazbou na účetní jednotku. Určuje konkrétní doklad o vyřazení daného hardwaru

datum_zapisu	DATETIME	Atribut upřesňující zápis položky do evidence. Vzniká sám ihned při přidání položky.

Tabulka *komponenty_zarizen_historie* – Tabulka je přejmenovanou relací hardware-historie. Jejím obsahem jsou klíče tabulek komponenty_zarizeni a pc. Dále také obsahuje bližší informace pro určení platnosti záznamu. Tabulka je využívána pro určení historie a uživatel k této tabulce nemá přístup.

Tabulka 3: Databázová tab. komponenty_zarizeni_historie [zdroj: vlastní]

komponenty_zarizeni_historie		
Atribut	**Datový typ**	**Popis**
id_k	INT	Identifikátor hardwaru.
id_pc	INT	Identifikátor počítače.
poradi	INT	Pomocný atribut určující pořadí řádkových platností daného hardwaru.
platnost_od	DATETIME	Atribut upřesňující přesné datum a čas přiřazení hardwaru k počítači.
platnost_do	DATETIME	Atribut upřesňující přesné datum konce platnosti daného přiřazení. V případě platného (současného) přiřazení má atribut hodnotu „2999-12-31 00:00:00“.

Tabulka *pc* – Tato tabulka vychází z relace počítač z technologického modelu.

Tabulka 4: Databázová tab. pc [zdroj: vlastní]

pc		
Atribut	**Datový typ**	**Popis**
id_pc	INT	Jednoznačný identifikátor počítače.
inventární_cislo_pc	VARCHAR(25)	Na základě tohoto atributu, tedy inventárního čísla počítače, mohou být vykonávány určité funkce.
nazev	VARCHAR(60)	Analogicky daný atribut z tabulky komponenty_zarizeni.
cena	INT	Udává cenu počítače jako celku. Nikoliv jako součet cen jeho součástí.
cislo_faktury	VARCHAR(25)	Analogicky daný atribut z tabulky komponenty_zarizeni.
datum_nakupu	DATE	Analogický daný atribut z tabulky komponenty_zarizeni.
zaruka	INT	Udává záruční dobu počítače jako celku. Nijak se nevztahuje na záruční doby jeho součástí.
osoba	VARCHAR(25)	Analogický daný atribut z tabulky komponenty_zarizeni.

umisteni	VARCHAR(20)	Analogický daný atribut z tabulky komponenty_zarizeni.
poznamka	VARCHAR(85)	Analogický daný atribut z tabulky komponenty_zarizeni.
vyrazeni	BIT	Analogický daný atribut z tabulky komponenty_zarizeni.
datum_vyrazeni	DATETIME	Analogický daný atribut z tabulky komponenty_zarizeni.
duvod_vyrazeni	VARCHAR(65)	Analogický daný atribut z tabulky komponenty_zarizeni.
c_dokladu	VARCHAR(25)	Analogický daný atribut z tabulky komponenty_zarizeni.
datum_zapisu	DATETIME	Analogický daný atribut z tabulky komponenty_zarizeni.

Tabulka *selection* – Tato tabulka nijak nevychází z relačního modelu dat. Slouží pouze jako pomocná tabulka, do níž jsou během skriptu uloženy vybrané identifikátory položek, které jsou využívány náročnějšími SQL dotazy. Po provedení skriptu je obsah celé tabulky vymazán.

Tabulka 5: Databázová tab. selection [zdroj: vlastní]

selection		
Atribut	**Datový typ**	**Popis**
id	INT	Identifikátor vybrané položky databáze.

Tabulka *software* – Tabulka software vychází z relace Software z relačního modelu dat.

Tabulka 6: Databázová tab. software [zdroj: vlastní]

software		
Atribut	**Datový typ**	**Popis**
id_sw	INT	Jednoznačný identifikátor softwaru.
inventární_cislo_sw	VARCHAR(25)	Tímto atributem uživatel identifikuje jednotlivé softwarové produkty.
nazev	VARCHAR(60)	Analogicky daný atribut z tabulky komponenty_zarizeni.
cena	INT	Udává cenu softwarového produktu.
cislo_faktury	VARCHAR(25)	Analogicky daný atribut z tabulky komponenty_zarizeni.
datum_nakupu	DATE	Analogický daný atribut z tabulky komponenty_zarizeni.
licence	INT	Počet zakoupených licencí daného softwaru.
typ_licence	VARCHAR(25)	Bitový atribut zachycující, zda je zakoupená softwarová licence typu per connection.

		V případě splnění této podmínky atribut nabývá hodnoty 1.
poznamka	VARCHAR(85)	Analogický daný atribut z tabulky komponenty_zarizeni.
vyrazeni	BIT	Analogický daný atribut z tabulky komponenty_zarizeni.
datum_vyrazeni	DATETIME	Analogický daný atribut z tabulky komponenty_zarizeni.
duvod_vyrazeni	VARCHAR(65)	Analogický daný atribut z tabulky komponenty_zarizeni.
c_dokladu	VARCHAR(25)	Analogický daný atribut z tabulky komponenty_zarizeni.
datum_zapisu	DATETIME	Analogický daný atribut z tabulky komponenty_zarizeni.

Tabulka *software_historie* – Tabulka je převzatou relací Nainstalování z relačního modelu dat. Je tedy pomocnou tabulkou určující platnosti instalací softwarových produktů na jednotlivých počítačích.

Tabulka 7: Databázová tab. software_historie [zdroj: vlastní]

software_historie		
Atribut	**Datový typ**	**Popis**
id	INT	Identifikátor platnosti.
id_sw	INT	Identifikátor softwaru.
id_pc	INT	Identifikátor počítače.
platnost_od	DATETIME	Atribut upřesňující přesné datum a čas přiřazení softwaru k počítači.
platnost_do	DATETIME	Atribut upřesňující přesné datum konce platnosti daného přiřazení. V případě platného (současného) přiřazení má atribut hodnotu „2999-12-31 00:00:00“.

Tabulka *typy_komponent* – Pomocná tabulka, v níž jsou uloženy jednotlivé typy hardwaru.

Tabulka 8: Databázová tab. typy_komponent [zdroj: vlastní]

typy_komponent		
Atribut	**Datový typ**	**Popis**
typ	VARCHAR	Kategorie typu hardwaru.

Tabulka *sekce* – Pomocná tabulka, v níž jsou pro kontrolu uloženy informace o jednotlivých sekcích.

Tabulka 9: Databázová tab. sekce [zdroj: vlastní]

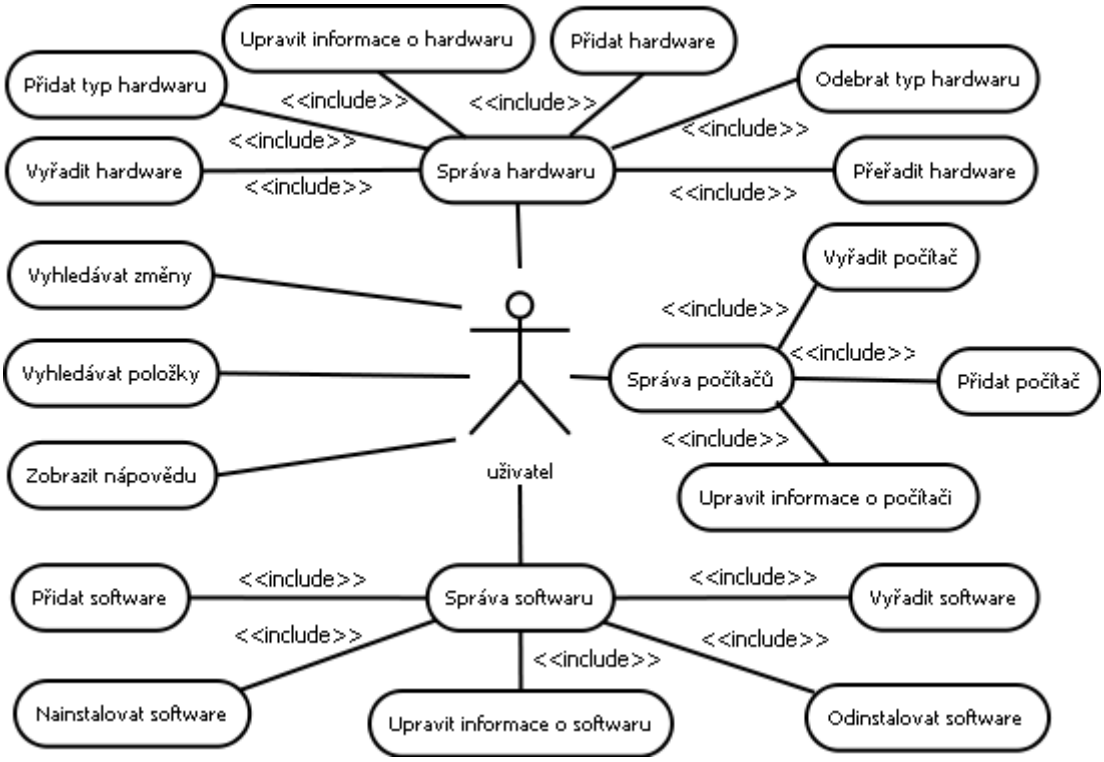
sekce		
Atribut	**Datový typ**	**Popis**
id	INT	Identifikátor sekce.
link	DATETIME	Pomocný atribut.

Tabulka *zmeny* – Pomocná tabulka, ve které je uložen seznam provedených procesů v databázi.

Tabulka 10: Databázová tab. zmeny [zdroj: vlastní]

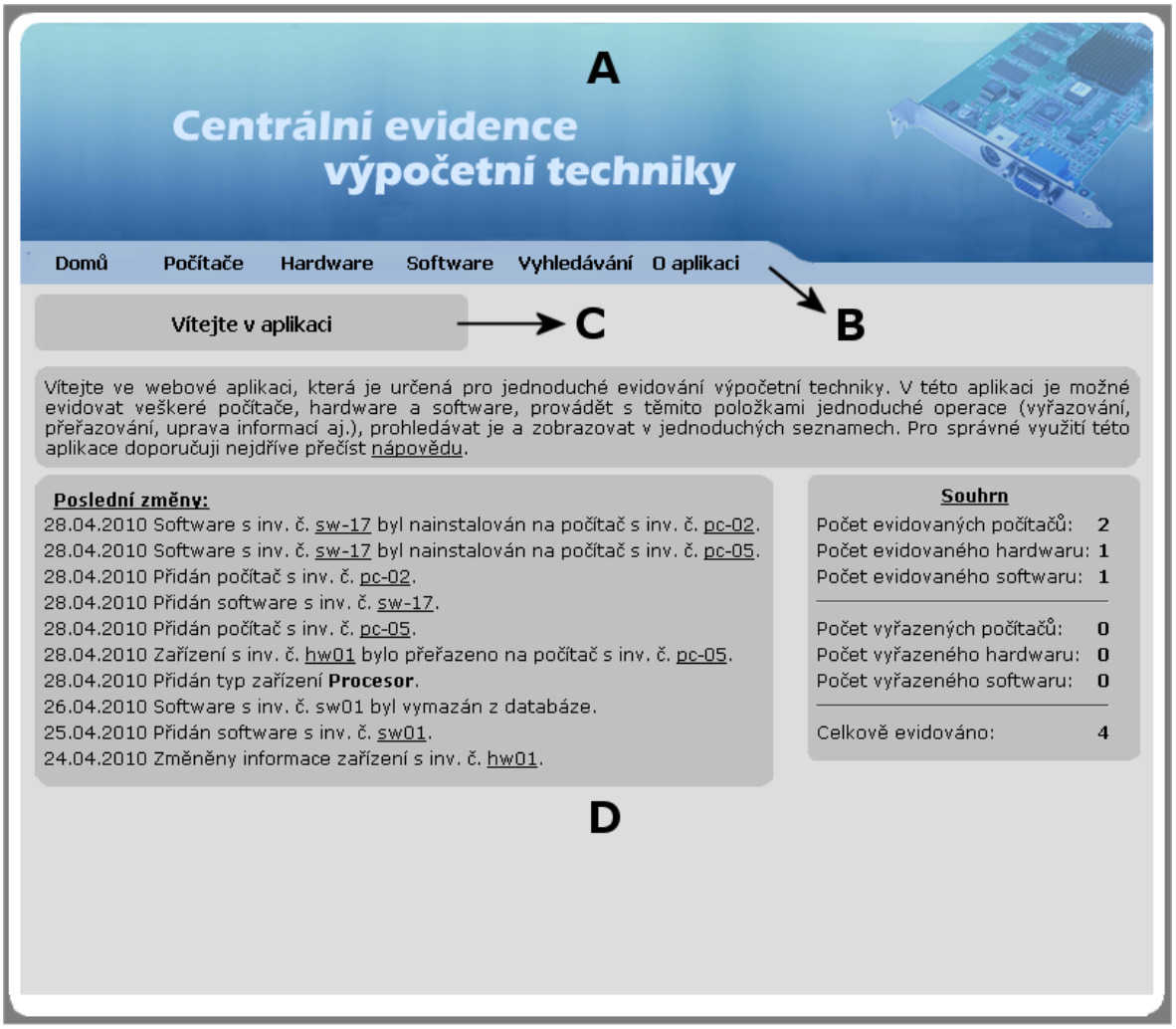
zmeny		
Atribut	**Datový typ**	**Popis**
id	INT	Identifikátor popisu procesu.
datum_zmeny	DATETIME	Atribut udávající, kdy k popisovanému procesu došlo.
zmena	VARCHAR(300)	Zkrácený zápis provedené změny.
co	VARCHAR(2)	Tento atribut upřesňuje kategorii databáze, které se popisovaný proces týká.

2.3 Use Case diagram



Obrázek 7: Use Case diagram [zdroj: vlastní]

2.4 Grafické rozložení aplikace



Obrázek 8: Grafické rozložení aplikace [zdroj: vlastní]

A – Logo

B – Horizontální vysouvací menu

C – Nadpis sekce

D – Obsah

Základní struktura všech částí webové aplikace je tvořena následujícím kódem:

```
<?php
     require_once('../head.html');
?>
<body>
<div class="zaklad">
  <div class="logo"></div>
```

```
    <?php
        require_once("../menu.html");
     ?>
  <div class="menu_pokr"></div>
  <div class="content_back">
    <div class="content_white">
      <div class="content">
        <div class="nadpis_sekce_top"></div>
         <div class="nadpis_sekce"><!-NADPIS-!></div>
           <div class="nadpis_sekce_bottom"></div>
            <br>
               <!-OBSAH-!>
            <br>
      </div>
    </div>
    <div class="bottom_screen"></div>
  </div>
</div>
</body>
</html>
```

V připojených souborech head.html a menu.html, které jsou připojeny ke kódu pomocí PHP funkce `require_once()`, jsou uloženy zdrojové kódy pro hlavičku a menu.

2.5 Souborová struktura aplikace

Celá webová aplikace je tvořena 61 soubory (včetně obrázků), které jsou uspořádány do 4 složek dle jejich funkce a obsahu. Konkrétní uspořádání je uvedeno v tabulce níže.

Tabulka 11: Souborová struktura aplikace [zdroj: vlastní]

Přípona	Počet souborů	Umístění	Funkce
.php	21	Kořenový adresář, složka „/other“	Obsah aplikace, funkce, pomocné soubory
.html	2	Kořenový adresář	Menu a hlavička
.png	31	Složka „/pictures“	Obrázky, ikonky, logo
.css	4	Složka „/css“	Třídy kaskádových stylů
.js	3	Složka „/js“	Funkce pro úpravu formulářů a vysouvacího menu

Jelikož všechny části webové aplikace mají dynamicky měnící se obsah, je nutné, aby soubory, ve kterých se definuje struktura webu, měly příponu .php. Konkrétně se jedná o všechny soubory kromě souborů head.html a menu.html. V každém PHP souboru se většinou ještě před samotným výpisem stránky, který je zobrazený výše, vyskytuje prvotní načtení globálních proměnných poslaných metodami GET, POST či SESSION. Dále se zde objevuje

kontrola podmínek, při jejichž nesplnění se vypíše chybová stránka. Podrobnější funkce této kontroly bude popsána v podkapitole 2.7**.** Před samotným výpisem obsahu aplikace se též ve většině případů vyskytují funkce generující proměnné, jejichž obsahem je dynamicky generovaný nadpis sekce, úvodní potřebné SQL dotazy (například pro stránkování) a celkové určení procesu, ve kterém se v dané chvíli aplikace nachází. Tyto informace jsou dále důležité pro samotný výpis stránky a pro určení vstupních parametrů funkcí, jež generují obsah.

2.6 Výpisy a historie položek

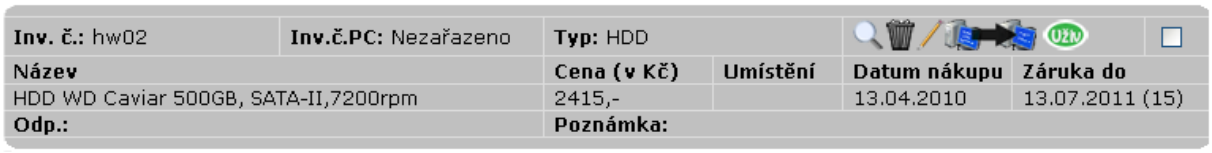
Hlavní funkcí celé aplikace je přehledný výpis evidovaných položek včetně jejich základních atributů a historických změn v jejich umístění. Jelikož šířka základního obsahu stránky je nastavena na 800 pixelů a pro přehledný výpis či třídění je nutné vidět většinu atributů zobrazovaných položek, tak zobrazení jedné položky s jejími atributy tvoří její karta. Tyto karty lze zobrazovat v kategorických seznamech či výběrech. Zjednodušený HTML kód karty vypisující jednu položku hardwaru vypadá následovně:

```
<table class="tab">
   <tr class="tab_top">
     <td colspan="7"></td>
   </tr>
   <tr class="tab_content">
     <td><strong>Inv.č.:</strong><!-INV. ČÍSLO-!></td>
     <td><strong>Inv.č. PC:</strong><!-INV. ČÍSLO PC-!></td>
     <td colspan="2"><strong>Typ:</strong><!-TYP-!></td>
     <td colspan="2"><!-IKONKY-!></td>
   </tr>
   <tr class="tab_content_header">
     <td>Název</td>
     <td>Cena</td>
     <td>Umístění</td>
     <td>Datum nákupu</td>
     <td width="160">Záruka</td>
   </tr>
   <tr class="tab_content">
     <td width="25%"><!-NÁZEV-!></td>
     <td><!-CENA-!></td>
     <td><!-UMÍSTĚNÍ-!></td>
     <td><!-DATUM NÁKUPU-!></td>
     <td><!-ZÁRUKA-!></td>
   </tr>
```

```
    <tr class="tab_content">
      <td colspan="2">
           <strong>Odp.:</strong><!-ODPOVĚDNOST-!>
      </td>
      <td colspan="5">
           <strong>Pozn.:</strong><!-POZNÁMKA-!>
      </td>
    </tr>
    <tr class="tab_bottom">
      <td colspan="7"></td>
    </tr>
</table>
```

Karty položek pak vypadají takto:

- hardware



Obrázek 9: Karta hardwaru [zdroj: vlastní]

- počítače

Obrázek 10: Karta počítače [zdroj: vlastní]

- software

Obrázek 11: Karta softwaru [zdroj: vlastní]

V částech aplikace, kde se může vyskytovat zobrazení většího množství položek (karet), je nastaven jednoduchý systém stránkování, který zobrazuje karty položek po třiceti na jednu stránku. Tento počet je však možné jednoduše nastavit na jinou hodnotu zásahem do proměnné `$pocet` ve skriptu, na které se dané stránkování nachází. Stránkování se nachází ve všech

seznamech užívaných a vyřazených položek, jež jsou přístupné z menu. Všechny tyto seznamy jsou zpracovány v souboru seznam.php. Ke stránkování také dochází při výběru položek, jež se účastní probíhajícího procesu. Jedná se například o přiřazení hardwaru k právě vytvářenému počítači. Všechny tyto výběry probíhající během procesů jsou naskriptovány v souboru selection.php. Poslední částí aplikace, kde dochází ke stránkování výsledků SQL dotazů, je přehled veškerých změn. Soubor, který výpis všech změn v rámci systému zpracovává, je zmeny.php umístěný v kořenovém adresáři aplikace.

Kromě výpisu atributů položek na jejich kartách je podstatnou součástí aplikace i možnost zpětné kontroly historie přiřazení hardwaru a softwaru k jednotlivým počítačům. Základem této funkce jsou tabulky komponenty_zarizeni_historie a software_historie. Každým přiřazením hardwaru a softwaru je ukončena platnost řádku současného stavu položky (konkrétně z hodnoty „2999-12-31 00:00:00“ na hodnotu současného data a času) a přidána řádka, která vypovídá o současném stavu. Platnost na novém řádku je nastavena od současného data a času opět do data „2999-12-31 00:00:00“. Díky této jednoduché technice je snadné u každé položky stanovit její historické změny v přiřazení k jednotlivým počítačům. Ve výpisu detailu počítače jsou pak zjištěny hodnoty hardwaru a softwaru, který počítačem v minulosti prošel a dále hodnoty aktuální. Prakticky to znamená, že například během fyzické operace, při níž je složen počítač z evidovaných komponent, které byly součástí již vyřazených počítačů, je uživatel schopen zpětně dohledat počítače, kde se v minulosti vyskytovaly komponenty z nově sestaveného počítače. Historické i aktuální záznamy platností je možné zobrazit na detailu každé položky, přičemž na tento detail se lze dostat z karty této položky (ikonka lupy). Zobrazení detailu položky je za pomoci PHP a SQL dotazů vytvořeno v souboru detail.php. Jeden z mnoha SQL dotazů, které jsou v tomto souboru zapotřebí, vypadá takto:

```
SELECT komponenty_zarizeni_historie.id_k,
       komponenty_zarizeni_historie.id_pc,
       komponenty_zarizeni_historie.poradi,
       komponenty_zarizeni_historie.platnost_od,
       komponenty_zarizeni_historie.platnost_do,
       komponenty_zarizeni.inventarni_cislo,
       komponenty_zarizeni.typ,
       komponenty_zarizeni.nazev
FROM komponenty_zarizeni_historie
LEFT JOIN komponenty_zarizeni
ON komponenty_zarizeni_historie.id_k=komponenty_zarizeni.id_k
```

```
WHERE komponenty_zarizeni_historie.id_pc='$id' AND
      komponenty_zarizeni_historie.platnost_do!='2999-12-31
      00:00:00'
ORDER BY komponenty_zarizeni_historie.platnost_do ASC
```

Tento dotaz pro vysvětlení vyhledá inventární čísla, názvy a typy všech komponent, které již nejsou součástí počítače, jehož identifikátor je uložen v proměnné `$id`. Identifikátor položky, jejíž detail chceme zobrazit, je na soubor detail.php poslán metodou GET a následně po kontrole jeho existence uložen do proměnné `$id`. Výsledná stránka detailu položky je ukázána na obrázku níže. Obrázek je vystřiženou částí detailu položky počítače, k němuž aktuální přiřazené položky se nacházejí v tabulkách `hardware` a `software`. Položky historické (již z počítače odebrané) jsou v tabulkách `software_historie` a `komponenty_zarizeni_historie`.

Hardware

Inv. č.	Typ	Název
hw02	HDD	HDD Seagate Barracuda 7200.10 ...
hw03	Procesor	AMD Athlon 3000+
hw08	Klávesnice	Genius KB-350e

Software

Inv. č.	Název	Nainstalován dne
sw01	SW Norton 360 v3 CZ upgrade	26.03.2010

Historie - hardware

Inv. č.	Od:	Do:	Typ	Název
hw05-x	27.03.2010	27.03.2010	LCD	LG Flatron Wide 19

Historie - software

Inv. č.	Od:	Do:	Název

Nenalezen žádný záznam historie

Obrázek 12: Detail položky [zdroj: vlastní]

2.7 Bezpečnost

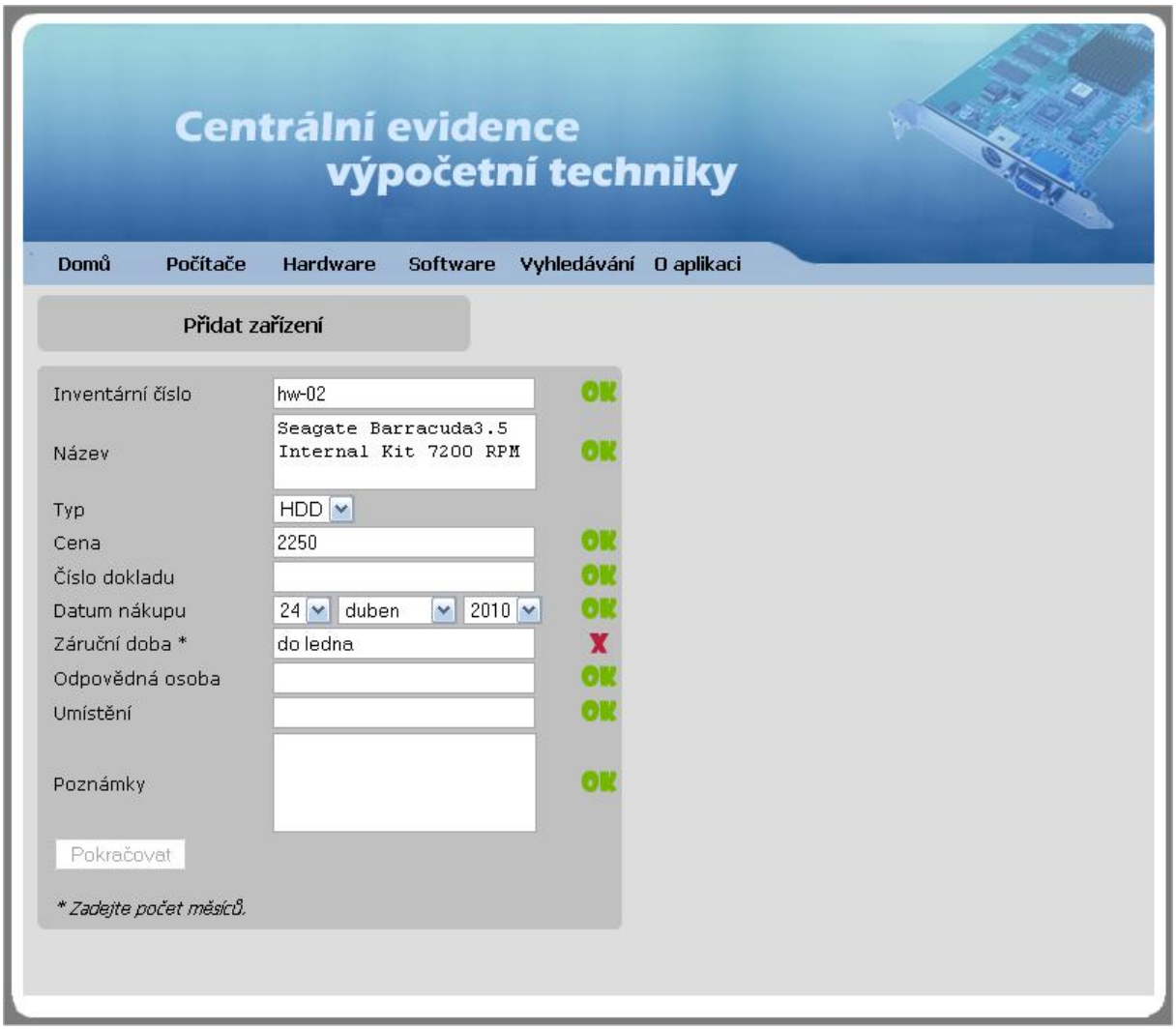
I přes předpoklad spouštění aplikace z interního serveru subjektu, jsou v jádru samotné aplikace vytvořeny alespoň minimální základy pro její bezpečnost. Tyto základy se týkají zejména:

- kontroly vstupních formulářových prvků,
- zásahů do URL adresy,
- redukce operací s položkami na základě jejich vybraných atributů.

2.7.1 Kontrola vstupních formulářových prvků

Kontrola vstupních formulářových prvků je v aplikaci vytvořena hlavně pomocí skriptovacího jazyka JavaScript. Tímto jazykem jsou na základě události `OnKeyUp` a `OnChange` kontrolovány veškeré vstupní formulářové prvky, které se vyskytují při přidávání, updatu či vyřazování položek. Každý formulářový prvek je prakticky kontrolován k němu přidělenou funkcí, jež na základě vepsané či vybrané hodnoty a daných podmínek rozhoduje o pokračování v probíhajícím procesu. V případě, že zadaná vstupní hodnota podmínky nesplňuje, je uživatel upozorněn pomocí dialogového okna nebo symbolu. Konkrétní podmínky všech formulářových prvků, které jsou kontrolovány, jsou uvedeny v nápovědě umístěné přímo v aplikaci. Posledními kontrolními prvky formulářů je převedení zvláštních znaků, jako jsou například hranaté závorky, na HTML entity, a to pomocí PHP funkce `htmlspecialchars()` a odstranění mezer na začátku a konci řetězců pomocí metody `trim()`.Tento převod probíhá ve většině případů těsně před přidáním řetězce do databáze.

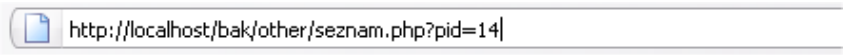
Jako příklad kontroly formuláře je možné uvést zadání záruční doby slovem při přidávání hardwaru. Jelikož atribut obsahující záruční dobu pracuje s atributem data nákupu, tak jej není možné zadat slovně, ale pouze pomocí čísel. Proto není uživateli umožněno pokračovat dále v procesu přidávání hardwaru. Ukázkovou situaci zachycuje následující obrázek.



Obrázek 13: Vstupní formulář [zdroj: vlastní]

2.7.2 Zásahy do URL adresy

Kontrola URL adresy se orientuje zejména na kontrolu proměnných posílaných metodou GET a dále na kontrolu (ne)přeskakování některých částí procesu změnou URL adresy.



Obrázek 14: URL adresa [zdroj: vlastní]

Proměnné, které jsou v aplikaci posílány metodou GET, jsou nazvány pid, id a stranka. Proměnná s označením „pid“ v URL adrese rozhoduje o druhu procesu, nadpisech či konkrétním výpisu, který se v daném souboru může vyskytovat, a proto je nutné zabezpečit, aby hodnota této proměnné byla zadaná, existující a využívaná konkrétním souborem. Například sou-

bor seznam.php, jehož obsahem jsou veškeré seznamy položek, musí na svém vstupu přijmout proměnnou pid s hodnotami v rozmezí 12-17. V případě, že tuto proměnnou nepřijme nebo přijme s hodnotou, která není stanovena v daném rozmezí, skončí aplikace chybovou hláškou. Pro kontrolu této proměnné je většinou využíváno konstrukce switch – case. Proměnná „id“ označuje identifikátor konkrétní položky evidence. Pro tuto proměnnou je rovněž nutné, aby existovala v dané kategorii (hardware, software, počítače) a zároveň, aby s takto identifikovanou položkou byla umožněna konkrétní operace na základě jejích atributů. Tomuto problému je věnováno více pozornosti v následující kapitole 2.7.3. U proměnné „stranka“, která označuje současnou pozici stránkování, jde především o kontrolu existence dané stránky tak, aby nevznikl přesah stránkování.

Jelikož některé procesy v aplikaci probíhají ve více krocích, je nutné zajistit, aby uživatel některé kroky nemohl přeskočit pouhým přepsáním názvu souboru, který skripty provádí. Kroky většiny procesů v aplikaci lze popsat následovně:

a) nastavení procesu -> výběr položek -> uložení do databáze
b) nastavení procesu -> uložení do databáze

K nastavení procesu je buď přiřazen vlastní soubor (add.php, delete.php aj.) nebo v některých procesech je tento krok pouhým ověřením vybraných položek. V tomto případě ověření probíhá v souboru actions.php. Výběr položek u všech procesů je základem souboru selection.php. Samotné uložení údajů do databáze probíhá v souboru process.php, ve kterém jsou na základě SESSION proměnné „proces“, jejíž hodnotou je označen druh procesu, volány příslušné funkce vykonávající vhodné SQL dotazy. Kontrola spočívá v tom, aby nebylo možné se bez odeslání formulářového tlačítka dostat na stránky selection.php nebo process.php. Problém je vyřešen přidáním skrytého formulářového prvku, jehož hodnota je v těchto souborech ověřena.

2.7.3 Redukce operací s položkami na základě jejich atributů

Tato kontrola se ve své podstatě týká obou předchozích podkapitol a zabývá se „smysluplností“ některých operací při hodnotách určitých atributů vybraných položek. Asi nejpodstatnějším atributem, na jehož základě jsou omezeny některé operace, je atribut obsahující stav vyřazení či užívání dané položky. S vyřazenými položkami nemohou být uskutečňovány některé operace. Konkrétně se jedná o operace přeřazování hardwaru na počítač

a (od)instalací softwaru na počítač. V případě adekvátního postupu nebude daná vyřazená položka vůbec nalezena nebo bude daný proces znepřístupněn. V případě změny URL adresy, kde by například u operace přeřazení hardwaru došlo ke změně identifikátoru položky na položku již vyřazenou, dojde k vypsání chybové hlášky. Atribut vyřazení má dále vliv na samotné vyřazování, protože vyřazenou položku již nelze znovu vyřadit, ale pouze úplně odstranit z databáze.

Atribut, který má ještě menší vliv na samotné operace, je atribut identifikátoru počítače na položkách hardwaru. Tento atribut má vliv na operaci přeřazování hardwaru, ve kterém při nulové hodnotě tohoto atributu (hardware je nezařazený) není uživateli umožněno opětovně zařadit hardware do nezařazeného.

2.8 Další funkce

Kromě základních funkcí vytvářejících předpoklady pro evidování prostředků výpočetní techniky, aplikace disponuje i dalšími jednoduchými funkcemi, které využívá v současné době téměř každá webová aplikace. Jedná se především alespoň o základní možnosti ve vyhledávání položek, přehledu operací, které byly v aplikaci provedeny a dále je aplikace vybavena drobnými funkcemi, jež souvisí s přehledností a některými atributy položek.

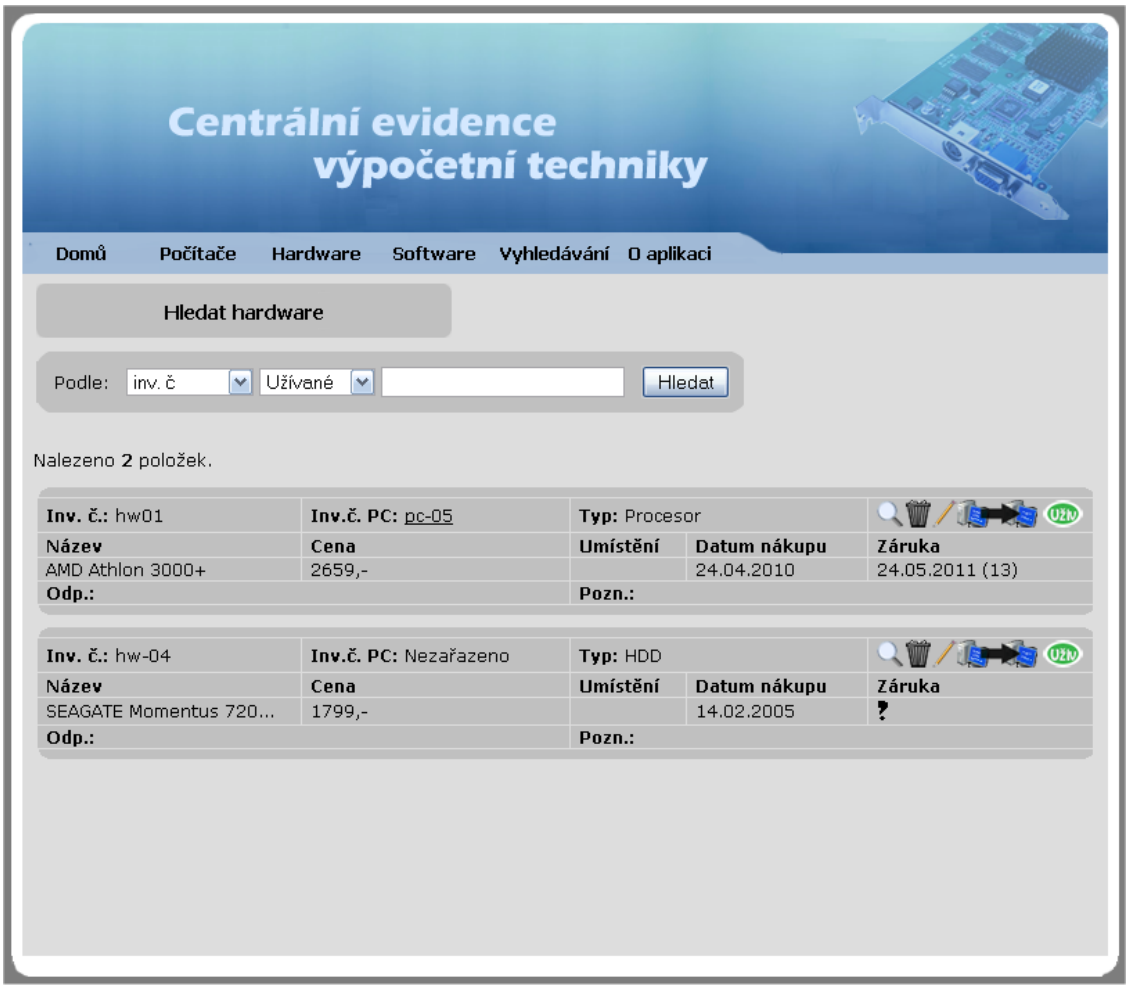
2.8.1 Vyhledávání

Při větším množství evidovaných položek se kromě vyhledávání v setříděných seznamech vyplatí vyhledávat konkrétní položky či více položek dle jejich atributů. Právě proto disponuje samotná aplikace alespoň jednoduchým vyhledávacím systémem. Tento systém je do praxe uveden s pomocí několika formulářových prvků, PHP a několika SQL dotazy. Celý proces vyhledávání je v aplikaci implementován do souboru hledat.php. Na základě proměnné „pid“, která je na tento skript poslána metodou GET, je rozhodováno, o jaké vyhledávání se jedná. Tedy o vyhledávání mezi počítači, hardwarem či softwarem. Tímto rozhodovacím procesem je formulářovým objektům dán obsah v podobě vybraných atributů a dále je rozhodnuto o SQL dotazu, který bude při vyhledávání použit. Tento SQL dotaz je dále zpracován funkcí `mysql_query()` a na základě jeho výsledků jsou vypsány karty odpovídajících položek. Část této konstrukce pro vyhledávání hardwaru je zobrazena níže.

```
$sql='SELECT komponenty_zarizeni.id_k,
              komponenty_zarizeni.inventarni_cislo,
              komponenty_zarizeni.nazev,
              komponenty_zarizeni.typ,
              komponenty_zarizeni.cena,
              komponenty_zarizeni.cislo_faktury,
              komponenty_zarizeni.datum_nakupu,
              komponenty_zarizeni.zaruka,
              komponenty_zarizeni.osoba,
              komponenty_zarizeni.id_pc,
              komponenty_zarizeni.poznamka,
              komponenty_zarizeni.vyrazeni,
              komponenty_zarizeni.datum_zapisu,
              pc.inventarni_cislo_pc
     FROM komponenty_zarizeni
     LEFT JOIN pc ON komponenty_zarizeni.id_pc = pc.id_pc
     WHERE '.$pole_db[$vyber].' LIKE \'%'.$search.'%\'
     AND '.$pole_db2[$vyber2].'';

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

SQL dotaz uložený v proměnné `$sql` vybere všechny atributy (včetně inventárního čísla počítače, ke kterému je zařízení přiřazeno), které jsou využívány při zobrazení karty, z tabulky obsahující veškerý hardware a následně vybírá položky, které odpovídají nastavenému vyhledávání. Toto nastavení je uloženo v proměnných `$pole_db[$vyber]`, `$pole_db2[$vyber2]` a `$search` a je přijímáno na začátku souboru metodou POST. Obrázek níže zachycuje ukázkovou situaci.



Obrázek 15: Vyhledávání mezi položkami [zdroj: vlastní]

2.8.2 Přehled všech operací v databázi

Jednoduchý přehled vykonaných operací je jen doplněním k celému systému. V praxi může mít pak uživatel aplikace alespoň částečný přehled o všech změnách, které se během celého běhu systému uskutečnily.

K tomuto účelu je využívána tabulka s názvem „zmeny“. Do této tabulky jsou po každé operaci jednoduchými dotazy přidávány krátké a stručné informace o změně. Součástí této informace jsou také odkazy na detaily příslušných položek, se kterými operace proběhla tak, aby bylo možné po zobrazení těchto informací ihned identifikovat danou položku. Vzhledem k tomu, že těchto změn se bude odehrávat v aplikaci velké množství, je uživateli umožněno změny vyhledávat. Pro pomoc při vyhledávání je vytvořen v tabulce „zmeny“ také atribut udávající, jaké kategorie položek se operace týkala.

2.8.3 Doplňující funkce

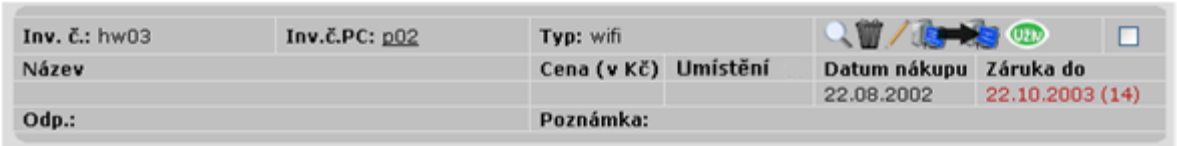
Mezi krátké funkce, které uživateli dají trochu větší přehled o evidovaných položkách, jsou v aplikaci řazeny funkce pro výpočet záruční doby u hardwaru a počítačů a zobrazení počtu užitých licencí u položek softwaru. Pro obě možnosti jsou napsány jednoduché funkce v PHP.

Funkce pro výpočet záruční doby:

```
function Zaruka($datum_db,$doba){
  if ($doba == 0 or $datum_db == '0000-00-00')
     return '<font face="Showcard Gothic" size="4"
              color="#000000">?</font>';
  else  {
     $den = StrFTime("%d", StrToTime($datum_db));
     $mesic = StrFTime("%m", StrToTime($datum_db));
     $rok = StrFTime("%Y", StrToTime($datum_db));
     $timestamp = mktime(0, 0, 0, $mesic+$doba, $den, $rok);
     $datum = date("M-d-Y", $timestamp);
     if (time()>($timestamp))
          return '<font color="#C00000">'.VypisDatum($datum).'
                 ('.$doba.')</font>';
      else
          return VypisDatum($datum).' ('.$doba.')'; } }
```

Tato funkce pro výpočet data, do kterého platí záruční doba zadaná v měsících, potřebuje mít zadané na vstupu dva parametry. Prvním je datum nákupu dané položky a druhou právě zadaná záruční doba. V případě, že jeden z parametrů není zadán, tak funkce vrátí otazník. Jestliže jsou oba parametry zadány, je nutné rozdělit datum nákupu zvlášť na dny, měsíce a roky, a to pomocí funkce `StrFTime()`. Dále je v příkladu použita funkce `mktime()`. Ta vrací počet vteřin uběhlých od 1.1.1970, tzv timestamp. Timestamp doby, kdy uběhne záruční doba, je poté porovnán s hodnotou timestamp aktuálního data a dle výsledku je vypsáno samotné datum. V případě, že daná položka je již po záruce, je výpis této doby vypsán červeně.

Výsledek pak vypadá na kartě položky takto:



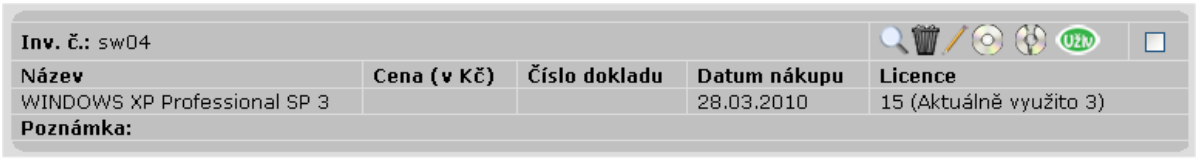
Obrázek 16: Ukázka záruční doby na kartě [zdroj: vlastní]

Funkce pro zjištění aktuálního počtu využitých licencí:

```
function Licence ($id_sw,$pocet_licenci,$typ_licence) {
    $return_pocet = mysql_query("SELECT id_pc FROM
            software_historie WHERE id_sw='$id_sw' AND
            platnost_do='2999-12-31 00:00:00' AND id_pc!=0");
    if ($typ_licence == 0){
        if ($pocet_licenci == '')
           return '<font face=\'Showcard Gothic\' size=\'4\'
                  color=\'#000000\'> ? </font> (Aktuálně
                  využito:
                  '.mysql_num_rows($return_pocet).')';
        else {
           return $pocet_licenci.' (Aktuálně využito
                  '.mysql_num_rows($return_pocet).')';
        }
    }
    else
       if ($pocet_licenci == '')
          return '<font face=\'Showcard Gothic\' size=\'4\'
                 color=\'#000000\'>?</font>
                 (<strong>S</strong>)</font>';
       else
          return $pocet_licenci.'(<strong>S</strong>)';
}
```

Tato funkce potřebuje na vstupu všechny údaje o samotné licenci. Tedy její typ, počet udělených licencí a také id konkrétního softwarového produktu. V prvé řadě je stanoven dle SQL dotazu počet počítačů, ke kterým je vybraný software v tu chvíli přiřazen. Jestliže se jedná o klasický typ licence, kde hlavní úlohu hraje počet platných instalací na jeden software, je následně podle zadaných údajů proveden výpis využitých licencí, a to pomocí funkce `mysql_num_rows()` a dále jejich počet (pokud je zadán). V případě licence typu „per connection“ je místo aktuálního počtu použitých licencí vrácen funkcí symbol „(S)“.

Výsledek vypadá na kartě softwaru následovně:



Obrázek 17: Ukázka počtu licencí na kartě [zdroj: vlastní]

Závěr

Hlavním cílem bakalářské práce bylo navrhnout funkční aplikaci umožňující evidovat veškeré prostředky výpočetní techniky. Kromě praktické části, která je ovšem v této práci stěžejní, je část věnována také teoretickým základům několika oblastí programovacího jazyka PHP.

Jak již bylo zmíněno, teoretická část je věnována především vybraným možnostem programovacího jazyka PHP, bez kterých si tvorbu webových aplikací lze jen těžko představit, včetně tvorby praktické části práce. Jelikož popisované tematické celky jsou velice obsáhlé, jsou vybrány pouze nejzákladnější informace, které jsou doplněny jednoduchými příklady.

Praktická část, tvořící větší část práce, je celkovým popisem činností, které bylo nutno vykonat během celého procesu realizace. V této části je popsán celý problém od prvotní analýzy, výběru vhodného nástroje pro realizaci, návrhu databáze až po popis možností výsledného produktu a prostředků, kterými byl tento produkt vytvořen. V textu nechybí doporučené diagramy (E-R diagram, relační schéma, Use Case diagram), množství obrázků přímo z vytvořené aplikace a úryvky zdrojových kódů a funkcí, kterými je tato aplikace tvořena. Výsledná aplikace je k této práci přiložena na CD.

Použitá literatura

[1] *edrive hosting* [online]. 2008, [cit. 2010-04-04]. nápověda - register globals v PHP. Dostupné z WWW: <http://www.edrive-hosting.cz/napoveda/php/register-globals>.

[2] *Linuxsoft.cz* [online]. 2004, [cit. 2010-04-06]. PHP - Sessions. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=440>.

[3] *Linuxsoft.cz* [online]. 2004, [cit. 2010-04-06]. PHP - Cookies. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=436>.

[4] *Interval.cz* [online]. 2003, [cit. 2010-04-06]. Superglobální proměnné v PHP. Dostupné z WWW: <http://interval.cz/clanky/superglobalni-promenne-v-php/>.

[5] *Programujte* [online]. 2005, [cit. 2010-04-06].PHP. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2005042302-php-2-lekce>.

[6] GUTMANS, Andi; BAKKEN, Stig Saether; RETHANS, Derick. *Mistrovství v PHP5*. [s.l.] : Computer Press, 2007. 656 s. ISBN 978-80-251-1519-0.

[7] *Tvorba-webu.cz* [online]. 2006, [cit. 2010-04-15]. PHP pole (Array). Dostupné z WWW: <http://www.tvorba-webu.cz/php/pole.php>.

[8] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-15]. count. Dostupné z WWW: <http://php.net/manual/en/function.count.php>.

[9] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-15]. array_keys Functions. Dostupné z WWW: <http://php.net/manual/en/function.array-keys.php>.

[10] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-15]. in_array. Dostupné z WWW: <http://php.net/manual/en/function.in-array.php>.

[11] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-10]. mysql_select_db. Dostupné z WWW: <http:// php.net/manual/en/function.mysql-select-db.php>.

[12] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-10]. mysql_query. Dostupné z WWW: <http:// php.net/manual/en/function.mysql-query.php>.

[13] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-10]. mysql_fetch_array. Dostupné z WWW: <http://php.net/manual/en/function.mysql-fetch-array.php>.

[14] *PHP Manual* [online]. 2001, last updated 2010-04-23 [cit. 2010-04-10]. MySQL. Dostupné z WWW: <http://php.net/manual/en/book.mysql.php>.

Seznam obrázků

Seznam tabulek