Gymnázium a Jazyková škola s právem státní jazykové zkoušky Zlín
2016

Práce se souborovým adresářem pomocí PHP

Working with file directories using PHP

závěrečná práce ze semináře z informatiky

Filip Čižmář, IV.A

Prohlášení

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Poděkování

Rád bych touto cestou poděkoval panu učiteli Michalu Miklášovi za vstřícnost a cenné rady při tvorbě nejen této práce.

Abstrakt

V závěrečné maturitní práci jsem se zaměřil na tvorbu webové stránky sloužící uživatelům k ukládání obrázků. Uživatel má možnost třízení obrázků do složek, přepínání mezi nimi a zobrazování obrázků v galerii Lightbox. Pro praktické nasazení jsem také vyřešil registraci uživatelů a přihlášení. Celý web je postaven na jednoduchém PHP scriptu, který obsluhuje databázi a adresáře se samotnými obrázky. V tomto dokumentu není popsána problematika stylování.

Abstract

In this work, I focused on creating websites serving users to store images. The user has the option of sorting images into folders, switching between them and displaying images in gallery Lightbox. For practical deployment I also solved the user registration and login. The site is running on a simple PHP script that operate database and directories with the pictures. In this document is not described styling.

Contents

1 Úvod

1.1 Prolog

Do semináře z informatiky jsem vstupoval s vědomým, že mě jednou tento čas dožene a já budu muset tuto práci vytvořit a napsat. Když si vzpomenu jak jsme začínali s tvorbou webových stránek a programováním, připadal jsem si místy úplně ztracen a ani ve snu by mě nenapadlo, že jednoho dne budu schopný vytvořit to co jsem v rámci tohoto projektu vytvořil. Vlastně ani teď nedokáži pochopit, jak jsem to mohl vytvořit.

Naštěstí však celý seminář je velmi inspirativně vedený a i když se mi ze začátku zdálo něco jako nepřekonatelný problém, věřil jsem v nějaké zázračné řešení a ono nakonec přišlo. V tomto semináři jsem byl neustále nucen řešit problémy vlastní hlavou, nanejvýš s pomocí internetu. A díky těmto nabytým schopnostem se můžu vrhnout do práce.

1.2 Výběr práce

Téma této práce sice nepochází z mé hlavy, ale jsem velmi rád, že z nabízených možností jsem volil právě tuto. Mou původní představou bylo, že budu muset vyřešit pouze nahrávání obrázků na server, což se později ukázalo jako nejmenší problém.

1.3 Cíl práce

Jak jsem již naznačil, mé počáteční představy o výsledku byly poněkud odlišné od reality. Každopádně hned od začátku jsem si kladl za cíl vytvořit něco užitečného, praktického. Ať to bylo zprvu vylepšení galerie na mém starém webu, nebo později vytvoření komplexní služby pro uskladnění obrázků.

2 Příprava

2.1 Výpis HTML kódu pomocí PHP

Jako pokusnou stránku jsem si zvolil web www.beetroot.g6.cz, kterého jsem správce, a existuje zde jednoduchá fotogalerie. Jelikož jsem tento web vytvářel s minimální znalostí HTML a CSS, PHP jsem pak už neznal vůbec. Byla celá galerie tvořená tabulkou, ve které v každém prvku td byl ručně napsán kód pro zobrazení obrázku. Rozhodl jsem se tedy, tuto technologii změnit a použít PHP. Měl jsem výhodu v tom, že obrázky nesly jména podle přirozených čísel, takže pro vypsání HTML stačilo použít for cyklus a jeho inkrement pro pojmenování obrázků. Zdrojový kód této jednoduché funkce pak vypadal takto.

```
for($i=0;$i<10;$i++){
  print(„<div><a˜href=\\obrazky/\.$i.\.jpg\\>
  <img src=\\obrazky/nahled/\.$i.\.jpg\\/></a></div>\);
}
```

Přepracoval jsem také metodu zarovnání jednotlivých obrázků. Nyní využívám CSS pravidla flexbox místo zastaralých tabulek.

2.2 Upload obrázků na server

Zprvu jsem nepochopil zcela dobře zadání práce a zaměřil jsem se pouze na vkládání obrázků skrz uživatelské rozhraní. Dále jsem tedy pokračoval s úpravou galerie zmíněné výše. Přesněji jsem ji doplnil o formulář pro nahrání obrázků. Toto je příklad jednoduchého formuláře.

```
<form method=\post\ enctype=\multipart/form-data\>
  <input type=\file\ name=\obrazky[]\ multiple=\multiple\/>
  <input type=\submit\ value=\Nahrát\ />
</form>
```

Lépe řečeno jednalo se o dva formuláře, jeden pro nahrání obrázku v plné velikosti a druhý sloužící pro nahrání zmenšeného náhledového obrázku. Každý formulář měl jednoznačně definovanou cestu, kam má obrázek nahrát. Poměrně jednoduché řešení, které však naráží na neukázněnost správce. V případě, že by nedošlo k nahrání velkého či malého obrázku, by docházelo k chybám při vykreslování stránky. Na řešení tohoto problému se ale zaměřím až v další části práce.

Samotné nahrání obrázku na server se provádí odesláním formuláře. Nahraná data jsou pak uložená v proměnné `$_FILES`, v našem případě s indexem `obrazky`. Pomocí funkce `move_uploaded_file` provedeme přesun obrázků do určeného adresáře a tím se vlastně uloží na serveru. Je však důležité před tím než zavoláme tuto funkci ověřit, jestli se skutečně jedná o typ souboru, který chceme. V našem případě o obrázek.

2.3 Zobecnění výpisu HTML kóku

Jelikož jsem se již nemohl spolehnout na pojmenování obrázků vzestupně za sebou jdoucími přirozenými čísly a také jsem nemohl v PHP scriptu nastavit počet cyklů for cyklu na jednoznačnou hodnotu. Bylo tedy nutné naučit PHP nahlédnout do složky a podle toho co v ní najde teprve vypisovat HTML. K takovému nahlédnutí a uložení načtených dat do proměnné typu array slouží funkce `scandir`, jejíž argument je cesta ke složce, kterou chceme naskenovat. Pro zjištění počtu potřebných průchodů cyklu for použijeme funkci `count`, která vrátí počet prvků nalezených v adresáři, resp. v poli, které však v sobě nese obsah adresáře. Dále ještě provádím ověření, zdali se skutečně jedná o soubor typu jpg, png, nebo gif, pomocí funkce `is_image`. Tuto funkci mám definovanou ve scriptu výše, pro nás však není důležité ji popisovat.

```
$jmenaobrazku = scandir($path);
for($i=0;$i<count($jmenaobrazku);$i++){
  if(is_image($path.\/\.$jmenaobrazku[$i])===true){
    print(„<div><a˜href=\\\.$path.\/\.$jmenaobrazku[$i].\\\>
    <img src=\\\.$path.\/JPEG/\.$jmenaobrazku[$i].\\\></a></div>\);
  }
}
```

V této chvíli jsem již měl vyřešené dva nejpodstatnější problémy celé práce. Nahrání obrázku na server a výpis HTML pomocí PHP. Po konzultaci s panem Miklášem, kde mně bylo upřesněno, co je od mé práce očekáváno, jsem se rozhodl začít pracovat na komplexnějším užití těchto funkcionalit.

3 Tvorba webové stránky fotogalerie

3.1 Nápad

Po výše zmíněném upřesnění čím se mám v projektu zabývat, tedy rozdělením obrázků do složek za účelem větší přehlednosti, jsem přemýšlel co praktického udělat. Nakonec jsem se inspiroval sociální sítí rajče.net. Můj nápad na závěrečnou práci byl tedy vytvořit webovou stránku, kde se uživatel bude moci zaregistrovat, pochopitelně přihlásit, nahrávat vlastní fotogalerie a třídit je do složek.

Přirozeně bylo mi jasné, že asi nedokážu vyrobit tak komplexní řešení jako můžeme vidět právě na rajčeti, kde je mnohem více funkcí a možností pro uživatele. Mým cílem tak stalo alespoň se co nejvíce přiblížit této sociální síti.

3.2 Osnova

Když už jsem tedy věděl, co chci dělat, musel jsem si nastavit dílčí cíle pro dosažení úspěchu. Tyto cíle jsem již naznačil v předchozí části. Vypíši zde pouze problémy, o kterých jsem věděl před započetí práce, pochopitelně při rozšiřování projektu se ukazovalo kolik dalších věcí je potřeba řešit.

Služba, kterou poskytuje můj projekt, má být nasazena pro masivní použití a možnost využívat ji, by měl mít každý uživatel internetu. Jinými slovy umožnit vytvořit účet každého zájemce, tedy registraci.

Za prvé jsem musel vytvořit dva formuláře, jeden pro registraci a druhý pro přihlášení. Pochopitelně s rozsáhlou validací vstupních údajů. Když tedy uživatel zadá vše správně je nutné si tyto údaje někam uložit. V této chvíli je jasné, že se neobejdu bez zapojení databáze, data o každém uživateli je potřeba držet v tabulce. Obdobně při přihlášení již existujícího uživatele se musí nahlédnout do databáze a v případě zadání správných údajů zobrazit osobní stránku uživatele. Paradoxně se později ukázalo, že toto byl největší oříšek celé práce, ačkoli to nebylo součástí zadání.

Jestliže se tedy povedlo úspěšně přihlásit, musel jsem umožnit samotné nahrávání obrázků a jejich vykreslování na stránku uživatele. Naštěstí však jsem tuto funkci měl vytvořenou z předchozí práce. Nečekal jsem však, že ji budu muset ještě velmi rozšířit, aby komunikovala s databází a jak složité bude zapracovat do ní možnost vytváření nových adresářů.

Ačkoli jsem věděl, že bude potřeba řešit právě práci se složkami, postupoval jsem systematicky a tuto část jsem si nechal až na závěr mé praktické práce.

3.3 Registrace a přihlášení

3.3.1 Tvorba formuláře registrace a validace hodnot

Nejprv je nutné se zamyslet a říct si, jaké informace je dobré od uživatele požadovat. Pro jednoznačnou identifikaci je potřeba mít uživatelské jméno. Dlouho jsem zvažoval, jestli k přihlášení nepožadovat jen e-mail, ale rozhodl jsem se, že nechám pro uživatele možnost mít více účtů na jeden e-mail, tedy bude potřeba nějaký jiný identifikátor, kterým je právě uživatelské jméno. Dále pochopitelně heslo, aby se k souborům nedostal každý. Ještě vyžaduji zadání e-mailové adresy, sice ji k ničemu nepoužívám, ale v budoucnu se může hodit prostředek jakým kontaktovat uživatele. Samozřejmě můžete po uživateli vyžadovat zadání více informací, např. pokud byste byli velmi srdeční tak datum narození. Jediné využití vidím však při zasílání přání k narozeninám.

V hodinách semináře z informatiky jsme se naučili vytvářet HTML formuláře a validovat je pomocí PHP, to mi výrazně ulehčilo práci. Jelikož mám všechno z hodin zálohované, nahlédl jsem do dokumentu s formuláři a inspiroval jsem se jimi při tvorbě nového formuláře pro registraci. Zdrojový HTML kód formuláře pak vypadá takto.

```
<form action=\registrace.php\ method=\post\>
  <label for=\jmeno\>Uzivatelske jmeno:</label>
    <input name=\jmeno\ type=\text\>
  <label for=\heslo\>Heslo:</label>
    <input name=\heslo\ type=\password\>
  <label for=\potvrzenihesla\>Znovu heslo:</label>
    <input name=\potvrzenihesla\ type=\password\>
  <label for=\email\>E-mail:</label>
    <input name=\email\ type=\text\>
```

```
    <input name=\odeslat\ type=\submit\ value=\Zaregistrovat se\>
</form>
```

Mít hodnotu `action` nastavenou přímo na `registrace.php` není příliš praktické, a kdyby došlo ke změně názvu souboru, formuláře nebudou fungovat. Přichází tedy čas použít PHP a vypsat hodnotu `action` právě jím. PHP kód, který to zařídí, si jednoduše řečeno vyžádá jméno scriptu a ořeže z něj vše do posledního lomítka. Zapsáno kódem takto: `print(substr(strrchr($_SERVER[,SCRIPT_FILENAME‘],"/"),1));`.

Nyní již uživatel může formulář vyplnit a odeslat jej. Pro nás to znamená, že musíme zařídit kontrolu dat a podle toho buď zájemci o začlenění do naší komunity říct ať si opraví chyby, v případě špatně vyplněných údajů, nebo v případě správného vyplnění uložit záznam do databáze a oznámit úspěšné vytvoření účtu. Popíšeme si, co budeme ověřovat při zadání jména.

Rozhodně se nesmíříme se jménem o délce nula znaků, sice jména o délce jeden nebo dva znaky by už neměla dělat problém, ale ať to vypadá líp, dáme minimální délku 3 znaky. Protože myslíme dopředu a víme, že adresář kam se budou nahrávat obrázky nového uživatele, bude pojmenován právě po jeho jméně, zakážeme také v uživatelském jméně mezery. A protože se jméno bude odesílat skrz SQL dotaz, zakážeme také všechny zvláštní znaky, které by ho mohli narušit. K tomuto účelu je v PHP funkce `mysqli_real_escape_string`, avšak tato funkce vrací ořezaný řetězec o nepovolené znaky, já bych však raději, aby systém uživateli řekl, ať se přejmenuje, než změnil jeho jméno bez jeho vědomí. Proto jsem se rozhodl tuto funkci obejít a ověření udělat pomocí hledání nepovolených znaků v řetězci jména. Do proměnné `$zakazane_znaky` jsem tedy vypsal všechny znaky, které by mohli nějak narušit SQL dotaz nebo znaky, které se mi nelíbily. Pro případ pozitivního nálezu zakázaného znaku se hodí mít vytvořenou proměnnou např. `$spravne_jmeno` a přiřadit jí hodnotu `false`. Tuto proměnnou pak využijeme při vypisování chybových hlášek. Nesmíme zapomenou, že uživatelské jméno je pro každého uživatele jedinečné a tak musíme ověřit zda-li již v naší databázi není uživatel se stejným jménem. Tady si už nevystačíme jen s PHP, ale budeme muset poslat dotaz na databázi uživatelů. V případě nalezení uživatele se stejným jménem si opět tuto skutečnost uložíme do proměnné, abychom informovali žadatele, že si musí zvolit jiné jméno. PHP script pro ověření jména pak vypadá takto.

```
if (isset($_REQUEST[„jmeno\])){
  $jmeno = $_REQUEST[„jmeno\];
  $zakazaneznaky=\‘\\;:\\/‘*=@#$ %^&()+_-,.<>?|\;
  for($i=0;$i<strlen($zakazaneznaky);$i++){
    if(!strpos($jmeno,$zakazaneznaky[$i])===false){
      $spravne_jmeno=false;
    }
  }
  mysqli_select_db($conn, „uzivatelegalerie\);
  $sql=\SELECT ‘jmeno‘ FROM ‘uzivatel‘ WHERE ‘jmeno‘=‘\.$jmeno.\‘ LIMIT 1\;
  $jmenop=mysqli_query($conn, $sql);
  $jmenop=mysqli_fetch_array($jmenop);
  $jmenop=($jmenop[„jmeno\]);
  if($jmenop==$jmeno){
    $jmeno_pouzite=true;
  }
}
```

Pro e-mail je situace obdobná, jen nebudeme ověřovat existenci účtu se stejným e-mailem a povolíme zadání zavináče. Můžeme také použít funkci v PHP pro validaci e-mailu `filter_var($email, FILTER_VALIDATE_EMAIL)`. A pro heslo nebudeme ověřovat nic, protože jsme rádi, když uživatel v hesle použije speciální znaky. Uživatel heslo zadává dvakrát, takže jen ověříme, jestli se obě hasla, shodují, ale to je banalita. Všimněte si, že jsem v SQL dotazu použil klauzuli `LIMIT`, je to z důvodu šetření procesorů a hard disků při provádění dotazu. Jakmile se v tabulce narazí na uživatele se stejným jménem tak to SQL zabalí a vrátí výsledek, v případě kdyby tam tato klauzule nebyla by se zbytečně prohledávala celá tabulka i když by došlo k nalezení shody na prvním řádku.

3.3.2 Databáze uživatelů

Při tvorbě formuláře jsem neustále opakoval slova jako je databáze, tabulka nebo SQL. Avšak žádnou databází ani tabulkou zatím nedisponujeme.

Zapněme si tedy phpmyadmin a zde vytvořme databázi. Je jedno jaký název databáze zvolíte, já původně používal `uzivatele_galerie`, ale ve chvíli kdy jsem chtěl web spustit na hostingu, jsem byl nemile překvapen, že je zakázáno používat podtržítko v názvu databáze. Abych tedy nemusel neustále ve scriptu přepisovat název databáze, zvolil jsem i na mém osobním počítači název databáze bez podtržítka.

V naší nové databázi si vytvoříme tabulku, do které budeme ukládat data o uživatelích. Takovou tabulku si pojmenujeme `uzivatel`. Hned při vytváření je po nás požadováno říct kolik atributů v tabulce budeme chtít. Protože uživatel při registraci zadává jméno, heslo a e-mail, budeme určitě potřebovat tři atributy. Jelikož už máme nějaké zkušenosti s prací s databázemi, víme, že je velmi užitečné mít u každého záznamu identifikátor neboli primární klíč. Teoreticky můžeme říct, že jako primární klíč může sloužit i uživatelské jméno, jsem však názoru, že je lepší k tomuto účelu využívat číslo než změť znaků, které si vymyslí uživatel. Pokud se v budoucnu rozhodneme naši databázi obohatit o další tabulku se vztahem k uživateli, je vhodné používat pro relaci číslo typu integer už jen z důvodu obsazení méně místa na paměťovém médiu. V základu nám stačí čtyři atributy, pro chod našeho systému.

Jako první atribut bude id, pojmenujeme si ho ať je nám z názvu jasné, že se jedná a id a patří k tabulce `uzivatel`, tak třeba `id_uzivatel`. Nastavíme, že se jedná o primární klíč a zapneme auto increment, to zařídí, že s přidáním nového záznamu se jeho id zvedne o jedničku oproti záznamu s dosavadní nejvyšší hodnotou id. Nyní přichází nejtěžší část tvorby tabulky, zamyslíme se nad tím, co od našeho díla očekáváme a hlavně jak velký úspěch bude mít. V případě, že nestavíme datový typ id na integer tak do tabulky neuložíme více jak dvě miliardy uživatelů. Já však počítám s o něco menším vytížením, proto se nebojím zvolit jako datový typ integer. Další je jméno a e-mail, tady bych upozornil, že název atributu nesmí obsahovat spojovník, s těžkým srdcem použijeme jako jméno `email`. Poslední atribut bude obsahovat heslo, jak jste si mohli všimnou z předchozích kódů pro uložení hesla, používám šifru sha1, proto jsem přímo do jména napsal, že je heslo zaheshovaná tímto algoritmem, abych na to v budoucnu nezapomněl. Poslední tři atributy budou mít datový typ varchar, přičemž u hesla můžeme nastavit maximální délku na 40 znaků, protože vím, že sha1 produkuje vždy právě 40znakové řetězce.

3.3.3 Registrace

Máme hotový formulář s validací i databázi pro registraci. Ještě zbývá vysvětlit, jakým způsobem uložíme získané informace do naší databáze. K tomu využijeme SQL příkaz `INSERT INTO`, jak samotný název napovídá, příkaz vkládá něco do něčeho. Pro náš případ bude celý SQL příkaz vypadat takto.

```
INSERT INTO uzivatel (jmeno, heslo_SHA1, email)
VALUES (‚\.$jmeno.\‘, ‚\.sha1($heslo).\‘, ‚\.$email.\‘)
```

Přeloženo to znamená. Vlož do tabulky uživatel do atributů jmeno, heslo a email hodnoty, které jsou uloženy v proměnných jmeno, heslo, které zašifrujeme algoritmem sha1 a email. Pořadí vkládaných dat uvedených v závorkách je důležité.

V kapitole věnované tvorbě formuláře registrace jsem již zmiňoval, že adresář kam se budou nahrávat obrázky, bude pro každého uživatele separé. Proto jsme také zakázali mít ve jméně mezeru. Mějme tedy ve složce s indexem vytvořený adresář, ve kterém se budou vytvářet všechny složky uživatelů a do těchto složek se až budou nahrávat vlastní obrázky. Osobní složku každého uživatele musíme vytvořit ihned při registraci. V PHP na to máme funkci `mkdir`, do této funkce vložíme jako jediný argument cestu kde se má nová složka vytvořit zakončenou jejím názvem. Např. jestliže chceme v adresáři registrace vytvořit složku pojmenovanou po jméně uživatele, napíšeme do PHP `mkdir(„./registrace/\.$jmeno);`.

Příkazy pro vložení záznamu do databáze a vytvoření adresáře zapíšeme v PHP do podmínky, v případě kdy by došlo k selhání jedné operace například vlivem technické závady na serveru, se uživateli oznámí, že došlo k chybě. Celý zápis bude tedy vypadat takto.

```
if (($conn->query($sql) === TRUE) and (mkdir(„./registrace/\.$jmeno))) {
  print(„Účet vytvořen.\);
} else {
  print(„Chyba.\);
}
```

3.3.4 Formulář a ověření přihlášení

Tvorba formuláře pro přihlášení je v podstatě stejná jako pro registraci. Formulář ale bude obsahovat pouze dvě pole. A to jméno nebo e-mail a heslo. Uvědomujme si, že si nikdo nepamatuje jméno jaké použil k registraci a proto není špatné mít možnost přihlásit se pomocí e-mailové adresy.

Po stisknutí tlačítka pro přihlášení se informace opět odešlou na server, kde můžeme udělat základní ověření, např. pokud má jméno méně jak 3 znaky nebudeme dál pokračovat v ověřování a ihned napíšeme, že je zadáno špatné jméno. Jinak ovšem budeme muset poslat SQL dotaz, s žádostí o vrácení záznamu uživatele se jménem jaké nám přišlo prostřednictvím formuláře. V případě vrácení nulového výsledku, tedy špatně zadaného jména napíšeme ihned, že je zadáno neplatné jméno. Když se nám vrátí nějaká smysluplná hodnota, ověříme, zdali se shodují hesla a po té přihlásíme uživatele do systému. Kód k takovému ověření může vypadat takto.

```
if(isset($_REQUEST[„prihlasit\])){
  $sql=\SELECT „;
  $jmenoprihlaseni=$_REQUEST[„jmenoprihlaseni\];
  $hesloprihlaseni=$_REQUEST[„hesloprihlaseni\];
  if(!strpos($jmenoprihlaseni,\@\)===false){
    $sql=$sql.\‘id_uzivatel‘, ‘email‘, ‘jmeno‘, ‘heslo_SHA1‘ FROM ‘uzivatel‘
    WHERE ‘email‘=‘\.$jmenoprihlaseni.\‘ LIMIT 1\;
  }else{
    sql=$sql.\‘id_uzivatel‘, ‘jmeno‘, ‘heslo_SHA1‘ FROM ‘uzivatel‘
    WHERE ‘jmeno‘=‘\.$jmenoprihlaseni.\‘ LIMIT 1\;
  }
  $radekuzivatele=vytahni_z_databaze($sql, $conn);
  if(null==$radekuzivatele){
    $spravne_jmenoprihlaseni=false;
  }else{
    if($radekuzivatele[„heslo_SHA1\]==sha1($hesloprihlaseni)){
      $prihlaseniok=true;
    }else{
      $spravne_hesloprihlaseni=false;
    }
}
```

Všimněte si, že pro zjištění jestli se uživatel přihlašuje pomocí jména nebo e-mailu využívám jednoduchou podmínku. Když zadaný přihlašovací údaj obsahuje zavináč, jedná se o e-mail, samozřejmě ne vše co obsahuje zavináč je e-mail, ale protože je zakázané mít ve jméně zavináč je jasné, že se jedná o e-mail nebo je jméno zadáno chybně. Také abych nemusel psát složité funkce pro obsluhu databáze pořád dokola, vytvořil jsem si funkci `vytahni_z_databaze`. Opět pro případ špatně zadaných dat si vše ukládám do proměnných, abych mohl později uživatele informovat o nezdaru.

Řešení problematiky formulářů není předmětem této práce, proto jsem nastínil jen velmi stručně, jak se formuláře vytvářejí a validují.

3.4 Přihlášen

Uživatel se tedy úspěšně přihlásil. Ve výše uvedeném kódu můžete vidět, že v proměnné `$prihlaseniok` je uložená informace o úspěšném přihlášení. Nabízí se teď myšlenka celou stránku, kterou uživatel uvidí po přihlášení napsat do podmínky `$radekuzivatele[„heslo_SHA1\]==sha1($hesloprihlaseni)`. Co se ovšem stane, jestliže uživatel klikne třeba na některý obrázek, nebo na složku kterou si posléze vytvoří? Celý PHP script se spustí znova, ale nikde už nebude napsané, že je přihlášen právě ten uživatel. Zkrátka se načte opět úvodní stránka.

3.4.1 Jak si udržet informace o uživateli

V této chvíli jsem vůbec netušil, jak silný nástroj v sobě PHP ukrývá a tak jsem začal problém řešit pomocí skrytých formulářů. Všechna potřebná data se tedy odeslala podobně jako při přihlášení, ale je patrné, že se nejedná o příliš elegantní řešení. Hledal jsem dál na internetu, jak se přihlášení řeší, ale

vše mi přišlo moc složité. Když jsem však o tomto problému diskutoval s kolegy, bylo mi velmi důrazně doporučeno, ať použiji funkcionalitu `session`, čemuž jsem se ze strachu chtěl vyhnout.

Až jsem si možnosti `session` trochu osvojil, zjistil jsem, že se nejedná o vůbec nic složitého, právě naopak. Jediný zádrhel je, že úplně na začátku PHP scriptu musí napsané `session_start();`. Stačí to napsat o kousek níž a už to dělá problémy, proto zdůrazňuji ještě jednou, že je nutné to umístit na začátek celého sciptu. Ušetříte tak mnoho hodin při hledání chyby. Tato funkcionalita nám dává možnost vytvořit globální proměnné, ke kterým budeme mít přístup po celou dobu, než se uživatel odhlásí nebo nevyprší čas existence `session`, defaultně nastavený na 24 minut.

Když jsem měl tedy vytvořený `session`, nic mi nebránilo využívání jeho možností. Ve chvíli kdy se uživatel úspěšně přihlásil, se založí proměnné `$_SESSION[„jmeno\]` a `$_SESSION[„id_uživatel\]`. Do těchto proměnných se pak uloží jméno a id přihlášeného uživatele. Jakmile se ukončí ověřování přihlášení, program pokračuje dál a tady je pravá chvíle pro podmínku `isset($_SESSION[„jmeno\])`. Přeloženo do lidského jazyka, když existuje tato proměnná, udělej něco, v pro nás to něco bude vypsání osobní stránky uživatele, a pokud podmínka nebude platit, vypíše se přihlašovací stránka. Teď by náš kód měl vypadat nějak takto.

```
...
if($radekuzivatele[„heslo_SHA1\]==sha1($hesloprihlaseni)){
  $_SESSION[„jmeno\]=$jmenoprihlaseni;
  $_SESSION[„id_uzivatel\]=$radekuzivatele[„id_uzivatel\];
}
...
if(isset($_SESSION[„jmeno\])){
  //celá stránka po přihlášení
 }else{
  //celá přihlašovací stránka
}
```

Po uvedení této části scriptu do provozu si již uživatel může vesele klikat po stránce a při psaní HTML kódu službou Apache, víme, že je přihlášen pořád ten stejný uživatel.

Odhlášení Osobně jsem sice odhlášení řešil až později, ale hodí se to napsat právě sem. Nejedná se o nic složitého, ale nesmíme na to zapomenout.

Z pochopitelných důvodů musí mít uživatel, také možnost se odhlásit. Vyřešil jsem to opět pomocí tlačítka v formuláři. Jakmile dojde k přihlášení, tak se po celou dobu bude vypisovat toto tlačítko. Zde je kód pro ohlášení.

```
if(isset($_REQUEST["odhlasit"])){
  session_unset();
  session_destroy();
}
...
<form id="logout" action="registrace.php" method="post">
  <input name="odhlasit" type="submit" value="Odhlasit">
</form>
```

Jediné na co si musíme dát pozor je, aby jsme neopomněli umístit podmínku, tázající se zdali existuje proměnná `odhlasit`, ještě před podmínku, jestli je uživatel přihlášen. Funkce `session_unset` pak odstraní všechny proměnné, které jsme si v session vytvořili a následující funkce, ukončí celou session.

3.4.2 Nahrávání a zobrazování obrázků a tvorba nových adresářů

Co se týče nahrávání obrázků a vypisování HTML kódu tak tady nic převratného nevymyslíme. Jen zkopírujeme řešení z přípravy a zdánlivě máme funkční web.

S tak jednoduchým řešením se ovšem nesmíříme a chceme, aby uživatel mohl své obrázky třídit do složek. Teď se konečně dostáváme k tématu mé práce.

Vytvoření nové složky se může zdát jako naprostá banalita, když uživatel řekne, že chce novou složku, tak spustíme funkci `mkdir`.

Systém vytvoření nové složky založíme na jednoduchém formuláři, do textového pole se napíše jméno nové složky a požadavek se odešle stisknutím tlačítka.

Jak ale víme, tato funkce požaduje, jako argument cestu kde má složku vytvořit. Začněme tedy tvořit tuto cestu. Víme že kořenový adresář se jmenuje registrace, takže zapíšeme `$path = „./registrace/\`, v session proměnné máme uložené jméno uživatel a víme, že každá složka se jmenuje podle jména uživatele, můžeme tedy dále pokračovat řetězcem `.$_SESSION[„jmeno\]`. Na konec cesty se zapisuje jméno složky, kterou si přejeme vytvořit, toto jméno máme uložené v proměnné `$_REQUEST[„jmeno_nove_slozky\]`. Formulář a script pro vytvoření nové složky může vypadat takto.

```
if(isset($_REQUEST[„jmeno_nove_slozky\]){
  $path=‘./registrace/‘.$_SESSION[„jmeno\].\/\.$_REQUEST[„jmeno_nove_slozky\];
  mkdir($path);
}
<input name=\jmeno_nove_slozky\ type=\text\>
<input type=\submit\ value=\Vytvořit novou složku\>
```

Analogicky jako při ověřování uživatelského jména ověříme, jestli je uživatelem zadané jméno složky platné. Ještě bychom měli ověřit, zdali není už složka s tímto jménem vytvořená, toho však nejsme zatím schopni a o řešení tohoto problému si povíme níže.

Nová složka se úspěšně vytvořila. Aby však ji mohl uživatel používat, musíme zařídit její zobrazení ve webové stránce. Máme vytvořenou funkci, která vypíše HTML kód pro zobrazení obrázků ve složce, a tuto funkci jen rozšíříme. Uvnitř cyklu for zavoláme funkci `is_dir`. Nyní po naskenování složky dokážeme rozhodnout, jestli vypíšeme HTML pro obrázek nebo pro tlačítko s odkazem na složku.

Řekl jsem sice s odkazem na složku, ale přesměrování po kliknutí na jinou stránku není vůbec praktické, proto jsem raději zvolil řešení opět pomocí formuláře. Takže po kliknutí se načte znova celá stránka, jen se jednou podmínkou zeptáme, jestli uživatel klikl na složku resp. tlačítko formuláře, pod kterým se složka schovává. Kód je velmi podobný ukázce kódu pro vytvoření nového adresáře. Také do proměnné `$path` uložíme informaci o tom, že se nacházíme v jiné složce než je primární uživatelská složka. Podle hodnoty proměnné `$path` se totiž rozhodne obsah, které složky budeme zobrazovat pomocí HTML kódu.

V tuhle chvíli máme vytvořenou funkční fotogalerii, jakou jsme si představovali, jediný zádrhel může nastat, když budeme chtít vložit k obrázku popisek. Jelikož jediná informace, která říká, že ten obrázek existuje je jen jeho přítomnost ve složce. Takže jediné co o něm dokážeme zjistit je, je podle adresáře ke kterému uživateli patří a jeho jméno. Také musíme myslet do budoucna a náš systém připravit na možnost, že bude možné vytvářet složky ve složce. Za současného stavu by se to řešilo např. podmínkou, pokud jsem ve složce ve složce ve složce ve složce ve složce tak cestu nastav na všechny ty jména složek.

Problém vyřešíme tak, že všechna data o složkách a obrázcích budeme držet v databázi.

3.5 Optimalizace

Tady bych si dovolil zmínit, že není nic horšího než předělávat funkční řešení s vidinou drobného vylepšení. Jelikož jsem se již nemohl v kódu vyznat a po snaze naučit PHP práci s databázemi jsem se v něm nevyznal už vůbec. Hledání i té nejdrobnější chyby byl horor a proto jsem se rozhodl vytvořit nový soubor a začít psát script znovu. Samozřejmě, že jsem nepsal vše znovu, ale ze starého dokumentu jsem kopíroval části, o kterých jsem věděl, že fungují a pokoušel jsem se vše pečlivě komentovat. Také jsem udělal velmi důležitý čin a to psát jednotlivé funkcionality ve scriptu v takovém pořadí, že nedojde k žádnému nestandardnímu chování. Např. upload obrázku se provede před vypsáním HTML kódu, takže když je zavolána funkce pro vypsání obrázků ve složce, vypisuje i právě nahrané obrázky. Podobně u vytváření nové složky nebo funkcionality, která po úspěšném založení účtu ihned vypíše jméno do kolonky pro přihlášení.

3.5.1 Databáze složek a obrázků

Při tvorbě obou tabulek budeme postupovat podobně jako u tvorby tabulky uživatelů. Nejprv se tedy zaměřme na tabulku složek.

Po krátkém zamyšlení zjistíme, že budeme potřebovat u každé složky mít jednoznačný identifikátor neboli id, vědět ke kterému uživateli patří a také její jméno, které si uživatel zvolí při vytváření. Dalo by se říct, že pro funkčnost systému v současném stavu nám toto plně dostačuje. Avšak kdybychom se

rozhodli přidat možnost tvořit složky ve složce tak se nám bude hodit atribut, díky kterému určíme, ve které složce se tato složka nachází. Já jsem si ho pracovně nazval `nadslozka_id`. Dále jsem se ještě rozhodl zaznamenat si adresu každé složky, opět se tento atribut může jevit jako zbytečný, protože se dá vytvořit, když známe jméno uživatele a složky, ale zase narážíme na problém složky ve složce, v takovém případě bychom museli ještě při vytváření adresy vyžadovat z databáze jména všech složek, ve kterým je tato složka podsložkou.

U tabulky obrázků budeme potřebovat také id a složku, do které patří. Toto je pro nás dostačující, já jsem však ještě přidal pro lepší orientaci také atribut, který určuje, kterému uživateli obrázek patří a pro jednoduchý výpis výsledného HTML kódu také držím adresu ze stejného důvodu jaký je popsaný výše.

Nebudu zde rozmazávat použité datové typy a různá nastavení, je to v podstatě stejné jak jsem již zmiňoval u tabulky uživatelů.

3.5.2 Propojení tvorby složek s databází

Nové tabulky sice máme vytvořené, ale nejsou nám nic platné, když k nim nikdo nepřistupuje. Proto musíme provést patřičné úpravy v PHP scriptu a začít je využívat.

Úplně první složka se nám vytvoří při registraci nového uživatele. Takže upravíme kód tak, aby po vytvoření záznamu uživatele a jeho složky v kořenovém adresáři se nám vytvořil také záznam v tabulce složek. Kód upravíme do takové podoby.

```
$sql = „INSERT INTO uzivatel (jmeno, heslo_SHA1, email)
VALUES (‚\.$jmeno.\‘, ‚\.sha1($heslo).\‘, ‚\.$email.\‘)\;
if (($conn->query($sql) === TRUE) and (mkdir(„./registrace/\.$jmeno))) {
  print(„Účet vytvořen\);
  $sql=\SELECT id_uzivatel FROM uzivatel WHERE jmeno=‘\.$jmeno.\‘\;
  $id_uzivatele_pro_slozku=vytahni_z_databaze($sql, $conn);
  $id_uzivatele_pro_slozku=$id_uzivatele_pro_slozku[„id_uzivatel\];
  $sql=\INSERT INTO slozky (id_uzivatel, jmeno, adresa)
  VALUES (‚\.$id_uzivatele_pro_slozku.\‘, ‚\.$jmeno.\‘, ‚.\/registrace\/\.$jmeno.\‘)\;
  mysqli_query($conn,$sql);
}
```

Velkou nevýhodou takového řešení je, že v případě kdy vytvoření záznamu složky selže, tak ve skutečnosti složka i uživatel v databázi existovat bude. Avšak tento příkaz nemohu vložit do podmínky, jelikož neznám jméno uživatele, které je nutné zapsat do záznamu složky, protože uživatel ještě neexistuje.

Další případ vložení záznamu složky nastane pochopitelně ve chvíli, kdy uživatel vytvoří novou složku. Použijeme tedy obdobný kód jako pro vytvoření složky uživatele. Tady přichází chvíle, kdy jsme schopni ověřit, jestli už neexistuje složka se stejným jménem, jakou se uživatel právě pokouší vytvořit. K tomu použijeme jednoduchý SQL dotaz.

```
SELECT ‘jmeno‘ FROM ‘slozky‘ WHERE ‘jmeno‘=‘\.$_REQUEST[„jmeno_nove_slozky\].\‘
AND ‘nadslozka_id‘=\.$_SESSION[„id_uzivatel\].\ LIMIT 1
```

V dotazu se ptám také na atribut `nadslozka_id`, tím zajistím, že vyberu pouze záznam, který patří právě tomu jednomu uživateli nebo v budoucnu zajistí, že vybere pouze záznam složky, ve které se uživatel pokouší složku vytvořit. Jestliže však vše proběhne v pořádku a jako výsledek se nám vrátí nic, můžeme pokračovat dále v ověřování názvu. Podmínky pro název složky jsou v podstatě stejné jako pro uživatelské jméno. Jestliže je název validní, tak můžeme složku bez obav vytvořit. To můžeme udělat např. následujícím kódem.

```
$path=‘./registrace/‘.$_SESSION[„jmeno\].\/\.$_REQUEST[„jmeno_nove_slozky\];
$sql = „INSERT INTO slozky (id_uzivatel, jmeno, nadslozka_id, adresa) VALUES
(‚\.$_SESSION[„id_uzivatel\].\‘, ‚\.$_REQUEST[„jmeno_nove_slozky\].\‘, ‚\.
$_SESSION[„id_aktualni_slozky\].\‘, ‚\.$path.\‘)\;
if (!(($conn->query($sql) === TRUE) and (mkdir($path)))) {
  print(„Chyba.\);
}
```

Zde bych upozornil, že využívám proměnnou `$_SESSION[„id_aktualni_slozky\]`, tuto proměnou vždy obnovuji při načtení stránky. Pomocí podmínky se ptám, zdali je definována proměnná nesoucí jméno složky, na kterou uživatel klikl. Když není, vím, že jsem v hlavním adresáři, když ne tak `$_SESSION[„id_aktualni_slozky\]` nastavím na jméno složky, na kterou uživatel klikl. Zapsáno kódem takto.

```
if(isset($_REQUEST[„slozka\])){
  $sql=\SELECT ‘id_slozky‘, ‘jmeno‘, ‘adresa‘ FROM ‘slozky‘ WHERE
  ‘nadslozka_id‘=\.$_SESSION[„id_primarni_slozky\].\ AND
  ‘jmeno‘=‘\.$_REQUEST[„slozka\].\‘\;
  $aktualnislozka=vytahni_z_databaze($sql, $conn);
  $_SESSION[„id_aktualni_slozky\]=$aktualnislozka[„id_slozky\];
}
```

3.5.3 Propojení nahrávání obrázků s databází a jejich přejmenování

Funkci na upload obrázků máme hotovou, jediné co potřebujeme vyrobit je SQL příkaz na vložení záznamu do databáze. Před tím než začneme tento příkaz tvořit, bych upozornil, že se později ukázalo, že je praktičtější si obrázky pojmenovávat podle id každého obrázku. Zabráníme tak tomu že uživatel nahraje dva obrázky se stejným jménem a ten prvně nahraný se přepíše.

Vložení nového záznamu se dělá podobným příkazem, jaké jsme viděli již dříve. Zaměřil bych se spíše jakým způsobem obrázek přejmenovat.

Nejprve musíme vložit záznam do tabulky, aby bylo tomu obrázku přiděleno id, se kterým budeme dále pracovat. Po té si id vytáhneme z tabulky pomocí dalšího obdobného dotazu. Nesmíme opomíjet, že při přejmenování obrázku se tím změní i jeho adresa v adresářích, tuto změnu musíme zanést do databáze. Provedeme to dalším SQL příkazem, jež vysvětlím později. Nyní již disponujeme znalostí id obrázku, obrázek proto v klidu nahrajeme do adresáře a zavoláme funkci na přejmenování, tato funkce se jmenuje `rename` a jako argumenty vyžaduje zadat starou cestu a novou cestu k obrázku. Takto je to zapsáno ve scriptu.

```
$sql = „INSERT INTO ‘obrazky‘ (id_uzivatel, id_slozky, adresa) VALUES
(,\.$_SESSION[„id_uzivatel\].\‘, ,\.$_SESSION[„id_aktualni_slozky\].
„‘, ,\.$uploadDir.\/\.$fileName.\‘)\;
$conn->query($sql);
$sql = „SELECT ‘id_obrazky‘ FROM ‘obrazky‘ WHERE
‘adresa‘=‘\.$uploadDir.\/\.$fileName.\‘\;
$id_prave_ukladaneho_obrazku=vytahni_z_databaze($sql, $conn);
$nova_adresa=$uploadDir.\/\.$id_prave_ukladaneho_obrazku[„id_obrazky\].
strrchr($fileName,\.\);
$sql = „UPDATE ‘obrazky‘ SET ‘adresa‘=‘\.$nova_adresa.\‘ WHERE
‘id_obrazky‘=‘\.$id_prave_ukladaneho_obrazku[„id_obrazky\].\‘\;
$conn->query($sql);
if(move_uploaded_file($tmpName, „{$uploadDir}/{$fileName}\)) {
  rename(„{$uploadDir}/{$fileName}\,$uploadDir.\/\.
  $id_prave_ukladaneho_obrazku[„id_obrazky\].strrchr($fileName,\.\));
}
```

Objevuje se nám tu nový příkaz `UPDATE`, poslouží k přepsání atributu v tabulce, v našem případě přepisujeme zadanou adresu, protože došlo ke změně jména obrázku, adresa se musí změnit. Můžete namítat, že jsem si tu adresu nemusel vůbec ukládat do tabulky a vložit ji až teď, ale ve chvíli uložení záznamu, kdy jsme ještě neznali id, je právě adresa jediný primární klíč. Dále si můžete všimnou použití funkce `strrchr`, při přejmenování chceme změnit pouze jméno ale příponu nechat stejnou. Jednoduše řečeno nám tato funkce vrátí vše co je za poslední tečkou v řetězci názvu, Díky tomu si u obrázku zachováme informaci v jakém formátu je. Např. obrázek admin.jpg se změní na 1.jpg a obrázek admin.png se změní na 2.png.

3.5.4 Tvorba náhledových obrázků

Vzhledem k omezené rychlosti internetu a naší šetrností k hard diskům a počítačovým sítím je dobré pro náhledové obrázky na stránkách používat zmenšené kopie obrázků nahrané uživatelem. Ten kdo se na ty náhledové obrázky bude dívat z toho ani nebude nějak smutný, protože je uvidí ve velikosti nastavené parametrem height, v našem případe 60 px. V případě, že uživatel bude chtít obrázek vidět v plném rozlišení, klikne na něj a teprve teď se stáhne původně nahraný obrázek.

PHP má funkce, díky kterým jsme schopni tohoto dosáhnout, avšak nepodařilo se mi nalézt žádnou primitivní funkci. Jako jediná použitelná funkce pro tento účel se mi jevila funkce `imagecopyresized`, při čtení definice na webu php.net se určitě leknete, toho kolik argumentů tato funkce vyžaduje. Avšak po bližším zkoumání zjistíte, že to až takové drama není.

Jediný problém, který se může zdát jako obtížný a stojí za vysvětlení je, jak zjistit na kolik nastavit šířku a výšku náhledového obrázku, aby zůstal zachován poměr stran. Nejprve si zjistíme rozměry velkého obrázku pomocí funkce `getimagesize`, pak se zamyslíme nad tím jakou výšku zvolit pro náhledový obrázek. Výšku každého náhledového obrázku nastavíme napevno, jelikož obrázky vykreslujeme v řádcích, nechceme, aby nám obrázky skákaly, jak se jim zachce. Pokud vytváříte galerii pro nasazení ve východoasijských zemích, určitě stojí za uváženou, jestli nezvolit vykreslování do sloupců, v takovém případě nastavíte napevno šířku. Nové rozměry obrázku spočítáme snadno jako podíl šířky a výšky násobené novou výškou, tímto získáme šířku, výšku pak máme zadanou definováním proměnné.

Dále musíme deklarovat proměnnou typu obrázek, to uděláme funkcí `imagecreatetruecolor`, zde je již po nás požadováno zadat výšku a šířku. Po té si z adresáře vytáhneme původní obrázek, k tomu máme opět již předdefinované funkce v PHP, avšak tyto funkce umí vytáhnou obrázek pouze se známým formátem. Proto jsem je sloučil do jedné s názvem `imageCreateFromAny`. Teď už přichází ke slovu výše zmiňovaná funkce `imagecopyresized`, popíši zde pouze argumenty, které budeme využívat. Jako první zadáme proměnou, do které chceme uložit výsledek, jako druhý argument zadáme původní obrázek. Protože tato funkce slouží hlavně pro ořezání obrázku, následující čtyři argumenty slouží pro definování nových okrajů. My však nic ořezávat nechme tak je budeme ignorovat zadáním hodnoty `0`. Poslední čtyři argumenty jsou nová šířka, nová výška, původní šířka a původní výška.

Přijde mi ta funkce udělaná poměrně složitě, přece výšku a šířku nového i starého obrázku si může změřit sama. Stejně tak jsem očekával, že za tímto účelem bude v PHP lepší funkce. Na závěr nezapomeneme obrázek uložit do složky. Jestli jste se v mém výkladu poněkud ztráceli, uvedu pro přehlednost celý kód.

```
$height=70;
$informace=getimagesize(„$uploadDir}/\.$id_prave_ukladaneho_obrazku[„id_obrazky\].
strrchr($fileName,\.\));
$width = $informace[0]/$informace[1]*$height;
$maly=imagecreatetruecolor($width,$height);
$obr=imageCreateFromAny(„{$uploadDir}/\.$id_prave_ukladaneho_obrazku[„id_obrazky\].
strrchr($fileName,\.\));  imagecopyresized($maly,$obr,0,0,0,0,$width,$height,
$informace[0],$informace[1]);
imagejpeg ($maly, „{$uploadDir}/m\.$id_prave_ukladaneho_obrazku[„id_obrazky\].
strrchr($fileName,\.\), 70);
```

Možná vás napadá otázka, pod jakým názvem uložíme náhledový obrázek. Já jsem to vyřešil, asi ne úplně šťastně tak, že každý takový obrázek bude nést jméno podle mateřského obrázku, ale začínající na písmeno „m“. Složka s obrázky bude tedy obsahovat např. obrázek 1.jpg a m1.jpg. Ve funkci pro uložení obrázku do adresáře si všimněte hodnoty `70`, toto číslo určuje kvalitu obrázku v jakém se má uložit výsledný jpg soubor.

4 Závěrem

Doufám, že jsem nezapomněl zmínit nějakou stěžejní myšlenku, na kterou jsem musel přijít při tvorbě projektu.

4.1 Hodnocení práce

Ještě z toho moc nadšený nejsem. Myslím však, že v rámci mých znalostí nabytých v semináři z informatiky se jedná o dobrou práci. Když se zpětně dívám, co všechno jsem musel řešit za problémy a kolik nových technologií jsem si musel osvojit, nechápu, jak jsem dokázal něco takového vytvořit.

Nyní na téma, kterým jsem se zabýval, nahlížím jako na velmi perspektivní a myslím, že různé variace mého řešení najdou mnohá uplatnění v praxi. I když webová stránka pro správu a uchování obrázků je již v dnešní době spíše přežitkem. Velké využití vidím například při správě fotogalerií ať už na stránkách různých korporací nebo portfolia umělce či fotografa. Velkou výhodou je možnost úpravy obsahu webové stránky bez jediného zásahu správcem do HTML nebo PHP scriptu. Celá úprava se provádí v uživatelsky přívětivém prostředí a zvládne ji i člověk webovými technologiemi netknutý.

Doplnil bych, že jsem vlastně vůbec nevytvořil sociální síť, jak se mohlo po stanovení cíle zdát. Mé řešení totiž neobsahuje žádné nástroje pro sdílení. Avšak to nic nemění na tom, že web je možné použít pro zálohu obrázků a pomocí internetu dávat uživateli možnost mít své obrázky kdekoliv na světě a promítat je třeba známým u rodinné sešlosti.

4.2 Vize do budoucna

Je zde ještě mnoho co je potřeba vylepšit, určitě bych chtěl umožnit psát popisky k obrázkům, libovolně je přesouvat mezi složkami, nebo také neomezovat uživatele na pouze jednu vrstvu složek. Pochopitelně také přejmenovávání a mazání složek i obrázků. Když jsem dříve zmínil pojem sociální síť, tak si říkám, že bych to mohl, dotáhnou i do stádia propojení uživatelských účtů a tak sdílení obrázků. Zatím se mi to jeví jako velmi složitý problém, tak uvidím.

Asi bych byl šťastný, kdyby toto mé dílo našlo i využití v praxi. Zřejmě o službu jako takovou velký zájem nebude, ale třeba někdy využiji její část. Vzhledem k přihlédnutí k mým nepříliš velkým znalostem problematiky webových technologií jsem s prací spokojený a myslím, že ještě s drobnými úpravami, by byla schopná provozu v reálném světě.

4.3 Epilog

Možná se může zdát, že celá práce byla zbytečná, přeci na světě existují mnohem lepší řešení, a proto není důvod využívat právě to mé. To je skutečně pravda, a sice teď už vím, že jsem mohl vytvořit něco mnohem jednoduššího, např. již mnohokrát zmiňované portfolio umělce. Ale na druhou stranu je dobré, že jsem si díky této práci osvojil mnohé technologie pro tvorbu webových stránek, získal nové zkušenosti a to je to nejdůležitější co tato práce pro mě přinesla.

Vzpomínám si, jak jsem se jako čert kříži vyhýbal použití sessions a chtěl jsem celé přihlášení řešit nějak jinak. Nyní mi to připadá jako nejjednodušší věc z celého PHP, v podstatě jde jen o to zakládat nové a nové proměnné. Ačkoli se jedná o naprostou banalitu, trvalo mi hodiny, než jsem si nastudoval co to vlastně je a jak se sessions používají. To stejné mohu říci o používání nejrůznějších PHP funkcích atp.

Zjistil jsem, že tím největším motorem, který mě hnal dopředu je zvídavost a snaha dělat velké věci. Bez něho bych skončil s nějakým primitivním webem, ale šlo mi o víc. Aby to celé nějak fungovalo. A ono funguje. Snad zůstanu i nadále hladový po nových věcech a nesmířím se s tím, jak to teď vypadá.