VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

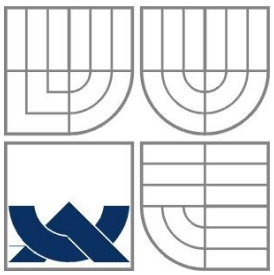
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÉ ROZHRANÍ PRO ZPRACOVÁNÍ OBRAZU

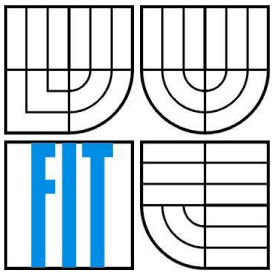
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PAVEL WOLLNÝ
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÉ ROZHRANÍ PRO ZPRACOVÁNÍ OBRAZU

WEB INTERFACE FOR IMAGE PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL WOLLNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VÍTĚZSLAV BERAN

BRNO 2009

Abstrakt

Práce pojednává o tvorbě webového rozhraní pro zpracování obrazu. Definuje základní požadavky na výslednou aplikaci. Zabývá se návrhem klientského rozhraní, serveru a jejich vzájemné komunikace. Následně se věnuje implementaci těchto částí. Systém je postaven na technologiích HTML, CSS, JavaScriptu, PHP a AJAXu. Vlastní zpracování fotografií pak zajišťuje GD knihovna PHP a externí aplikace napsaná za pomoci OpenCV.

Klíčová slova

Obrázek, fotografie, PHP, GD knihovna, JavaScript, AJAX, OpenCV

Abstract

The bachelor's thesis treat of creation web interface for image processing. The basic demands for final application are defined here. Work is concerned with design client interface, server and their communication. After that implementation of these parts are described. System is based on technologies HTML, CSS, JavaScript, PHP and AJAX. Processing of photos is provided by GD library PHP and extern application, which is write with the help of library OpenCV.

Keywords

Picture, photo, PHP, GD Library, JavaScript, AJAX, OpenCV

Citace

Wollný Pavel: Webové rozhraní pro zpracování obrazu. Brno, 2009, bakalářská práce, FIT VUT v Brně.

Webové rozhraní pro zpracování obrazu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Wollný
19. května 2009

Poděkování

Rád bych poděkoval svému vedoucímu, panu Ing. Vítězslavovi Beranovi za jeho trpělivost, čas, cenné připomínky a rady.

Obsah

1 Úvod

Mým úkolem je nastudovat technologie týkající se tvorby webových rozhraní a následně navrhnout systém, který umožní uživateli definovat operace nad obrazem, umožní mu měnit parametry operací a průběžně bude zobrazovat jejich výsledek. Tento systém jsem se rozhodl demonstrovat pomocí jednoduchého editoru obrázků.

Programy jako jsou například Adobe Photoshop, Gimp, Corel Photo-Paint jistě znáte, slouží uživateli pro profesionální úpravu fotografií. Jsou to velice komplexní a robustní aplikace, které poskytují obrovskou škálu funkcí. Jinými slovy, výsledkem mé práce, bude jednodušší verze takového programu, která bude volně přístupná uživateli na internetu. Ten si tak bude moci svou fotografii upravit v okně svého prohlížeče a nebude nucen si instalovat některý z těchto programů do svého počítače. Tedy v konečném součtu, bude výsledkem webová stránka, na které si bude moci uživatel snadno a rychle upravovat své fotografie.

Tato myšlenka mě velice zaujala, myslím si, že je to dobrý nápad. Nemuset instalovat žádný program kvůli jednoduché úpravě obrázku, ale jednoduše zadat webovou adresu do prohlížeče a tam si fotku upravit a nakonec si ji uložit zpět do počítače.

V úvodní kapitole si nastíníme, co je úkolem této práce a zmíním se také o důvodech, které mě vedly k výběru tohoto tématu. Ve druhé kapitole si něco řekneme o již existujících aplikacích a detailně si vymezíme cíle práce. Třetí kapitola se zpočátku věnuje mým prvotním představám o implementaci výsledné aplikace, ale její hlavní částí je samotný návrh aplikace. Ve čtvrté kapitole se pak můžeme dozvědět detaily samotné implementace. Další kapitola je pak věnována testovaní a v šesté poslední kapitole se nachází shrnutí celé této práce.

2 Zamyšlení se nad problémem

V této kapitole se budu věnovat již existujícím aplikacím, zmíním se o jejich výhodách či nevýhodách, o tom jestli mě něčím zajímavým zaujali, či nikoliv a v neposlední řadě si také vymezím cíl mé práce.

2.1 Rozbor podobných existujících aplikací

Zpočátku, kdy jsem začal studovat zadání a přemýšlel nad jeho realizací, jsem si myslel, že nic takového zatím neexistuje, proto mě tohle téma na první pohled zaujalo a chtěl jsem se pokusit o jeho realizaci. Ale po chvilce hledání na internetu jsem narazil na několik zajímavých odkazů, které mě vyvedli z omylu. Zjistil jsem, že existuje několik webových stránek, které tyto služby a funkce poskytuji. Podíval jsem se teda na ně detailněji, zkusil jsem si různé úpravy s fotografií, sledoval, jak se systém chová a pracuje. Snažil jsem se všímat detailů a vlastností jakož jsou například tyto: jak rychle reagují na akce a požadavky od uživatele, na intuitivnost ovládání, vzhled a jednoduchou orientaci v grafickém prostředí.

Vyzkoušel jsem si práci s pěti takovými online editory obrázků. Některé byli alespoň podle mého názoru hodně povedené a moc se mi líbili. Od jejich úvodní obrazovky mě bylo zřejmé, co musím udělat pro to, abych nahrál svůj obrázek na server a mohl tak začít s jeho úpravou. I samotná úprava fotografie byla intuitivní a efektní a to vše bylo vsazeno do velice příjemného grafického prostředí. Tyto editory jsou ale udělané ve Flashi. Je to grafický vektorový program, který slouží k tvorbě interaktivních prezentací, animací či her. Více se o tomto programu se můžete dozvědět zde, viz [1]. Ostatní editory jsou založené na mě známějších technikách a to za pomocí PHP, Javaskriptu a dalších technologií. Ale takto vytvořené editory mě přišli takové nepřehledné a těžkopádné. Než jsem se v něm zorientoval a přišel na to jak nahrát svou fotografii na server, chvíli to trvalo. Navíc rozhraní, ve kterém samotná editace fotky probíhala, mě přišlo zbytečně složité a nepřehledné a někdy i odpudivé svým vzhledem. A ten je určitě dobré při tvorbě nepodceňovat, myslím si, že má svou určitou váhu a důležitost.

2.1.1 Editory implementované pomocí Flashe

Musím se přiznat, že o tomto programu toho příliš nevím, tudíž detailněji se k takto vypracovaným editorům vyjádřit nemůžu. Ono to ani nebylo předmětem mé práce pochopit, jak tento program funguje a naučit se s ním pracovat. Ale když se na tyto editory podívám jako na celek, tak musím říct, že se mi velice líbí, jejich design je jednoduchý přehledný a vypadá moc pěkně. Navíc i samotná

práce s fotografii je velice příjemná, rychlá. Jednotlivé operace s obrázkem se provádějí v reálném čase, bez nějakého čekání či zdržení. Také tyto editory působí robustněji, i komplexněji, daleko více se přibližují klasickým programům pro úpravu fotografií, které máme nainstalované přímo na svém počítači.

2.1.2 Editory implementované pomocí JavaScriptu a PHP

Jazyky, technologie jakož jsou například Javascript, PHP, AJAX aj. jsou mě daleko známější než výše zmíněný Flash. Navíc také i v zadání mé práce bylo využít tyto technologie a implementovat můj systém s jejich pomocí. Hlavním rozdílem, mezi editorem implementovaným ve Flashi a pomocí těchto jazyků JavaScriptu, PHP, či technologie AJAX je ten, že úprava fotografií se už neprovádí v reálném čase. Tudíž existuje větší, či menší časová prodleva při provádění různých akcí. Nevím jak tomu je u Flashe, ale u těchto technologií se dá využít mnoho metod, jak samotné úpravy nad obrázkem provádět. Od úprav samotným JavaScriptem na straně klienta, nebo využití GD knihovny PHP na straně serveru, tak také použití různých externích binárních programů napsaných s pomocí různých knihoven jakou je také například OpenCV.

Vyzkoušel jsem si práci se třemi editory zrealizovanými pomocí těchto technologií. Snad jen jeden z nich, se dokázal přiblížit a dal srovnávat s editory udělanými ve Flashi. Samotná práce s obrázkem se prováděla relativně rychle, rozhraní bylo pěkné, jednoduché a přehledné. Jelikož zdrojové kódy nejsou z pochopitelných důvodů k dispozici tak předpokládám, že zde byla využita také technologie AJAX, pro přenos dat mezi serverem a klientem. Bez ní bych si ani takový editor nedokázal moc představit. Práce s obrázkem by byla asi taková těžkopádná a pro uživatele ne moc příjemná. Jak už jsem řekl, tak jen jediný editor se mi takto udělaný líbil. Zbylé dva, byly o poznání horší, jednotlivé úpravy s obrázkem trvaly někdy až příliš dlouho, nebylo ani poznat, zda editor ještě stále obrázek zpracovává či nikoliv. Mnohdy jsem si myslel, při zkoušení nějaké nové funkce, u které jsem nevěděl, co provede za změnu, že už se operace provedla a tak jsem začal pracovat dál a najednou se mi obrázek z ničeho nic změnil. Navíc také samotné rozhraní bylo takové nepřehledné, možná až zbytečně složité a překombinované. Ale asi jako největší nedostatek považuji již zmíněnou délku prováděných operací. Hodně to působilo a dělalo samotnou práci takovou neplynulou a rozkouskovanou.

2.2 Vymezení cíle práce

Cílem této práce je vytvoření webového rozhraní pro zpracování obrazu. Ze začátku bylo spoustu otázek, na které jsem se snažil postupně nacházet odpověď. Co se zobrazí uživateli v okně prohlížeče, co bude moci dělat za funkce, jak to bude vše vypadat a fungovat? O co se bude starat server, jaká

bude jeho funkce, co potečou za data mezi klientem a serverem? A spousta dalších a dalších otázek. Cílem práce tudíž není jen vytvoření samotného webového rozhraní, ale také rozdělit práci mezi klienta a server. Navrhnout způsob jak budou mezi sebou komunikovat a zpracovávat data. Na začátku jsem si také řekl, že by bylo dobré stanovit si kritéria, které by měla aplikace splňovat. Za takové hlavní podmínky považuju efektivnost a intuitivnost.

- **Efektivnost –** Chtěl bych se pokusit docílit takového výsledku, že by se uživatelem zadané operace prováděly co nejefektivněji. Pod tímto pojmem si představím také slova, jako jsou například rychlost, nebo úspora přenášeních dat. Myslím si, že rychlost zpracování jednotlivých operací, nebo aplikace jakož to celek, je určitě důležitá vlastnost. Pochybuji, že by se uživateli líbilo, kdyby mezi jednotlivými operacemi byla dlouhá a nudná pauza. Tohohle všeho bych chtěl dosáhnout také s minimalizováním toku přenášených dat, a to z toho důvodu, aby poté nároky na síť byly co nejmenší. Výsledkem by tedy měla být aplikace, která bude pracovat rychle a efektivně.

- **Intuitivnost –** Pod tímto pojmem si také představím slovo jednoduchost, či přehlednost. Chtěl bych vytvořit uživatelské rozhraní, které by bylo jednoduché, přehledné a intuitivní. Také bych se chtěl vyhnout a zabránit dlouhému bádání uživatele nad tím, jak se s aplikací pracuje. Nechtěl bych, aby musel zdlouhavě hledat akci, kterou chce provést a pak následně přemýšlel nad tím jak ji použít. Chtěl bych se alespoň trošičku přiblížit k takovému řešení, že bude uživateli jasné vše na první pohled.

Po rozboru již existujících aplikací jsem si promyslel, jak by mělo vypadat mé řešení. Určitě důležitou součástí webových aplikací je jejich design. Ačkoliv má práce není jen navrhnout grafické uživatelské rozhraní, dal jsem si na něm záležet. Myslím si totiž, že samotný design je neodmyslitelnou součástí webu a mnohdy je to hlavní kritérium, zda jej uživatel bude hodnotit kladně či záporně. Určitě dobře navrhnuté rozhraní přispěje také k tomu, aby bylo přehledné a intuitivní. Jak už jsem psal výše, výsledkem a cílem této práce by tedy měla být webová aplikace, stránka, na které si bude moci uživatel snadno a rychle upravovat své fotografie. Nechtěl bych ale jen přepracovat již udělané aplikace, ale přidat ji nějakou novou funkčnost, nový prvek. Tím by měla být historie zpracovaných obrázků. Kde se v ní bude moci uživatel libovolně pohybovat a dívat se jak se obrázek postupně mění, případně se pomocí ní vracet zpět či vpřed.

3 Analýza a návrh

Níže v této kapitole si popíšeme návrh systému, jak se postupně vznikal, až do výsledné podoby. Budu se také snažit popsat své myšlenky a představy jak by výsledná aplikace mohla a vypadat.

3.1 Prvotní návrh a představy

Při návrhu aplikace jsem chtěl jít cestou jednoduchosti, přehlednosti, efektivnosti a zaměřit se také na samotný design stránky. Celou aplikaci jsem si představoval jako celek asi takto.

Po zadání webové adresy by se načetla úvodní strana, kde bude logo a základní informace a popis k jakému účelu je tato stránka určena. Na této straně bude také panel s možností nahrání fotografie na server. Pomocí kterého si budeme moci vybrat obrázek, který máme uložen na svém počítači a po stisknutí tlačítka jej odeslat na server. Poté co se nahrávání úspěšně dokončí, bude uživatel přesměrován na další stranu, kde se zobrazí rozhraní pro samotnou úpravu fotografií. Jednotlivé akce nad obrázkem by pak mohli vypadat tak, že po stisknutí tlačítka požadované akce by se objevilo nové okno, kde by byly vidět dvě miniatury fotografie, jedna by odpovídala aktuálnímu stavu a druhá by ukazovala obrázek po aplikování efektu.

Když jsem se prvně nad tímto problémem zamyslel, jak by se to dalo realizovat, tak jsem se domníval, že hlavní roli v tomto projektu sehraje PHP (*Hypertext Preprocessor*). Je to skriptovací programovací jazyk, pracující na straně serveru. Jinak řečeno, všechny požadavky a akce se provádějí na serveru a klientovi je poslán až samotný výsledek. Pomocí něj můžeme zpracovávat data z formulářů, validovat tyto formuláře, můžeme také nahrávat a ukládat soubory na server a také pracovat s databází. Toto je jen nastínění funkcí, které pomocí PHP můžeme dělat, více se ale o tomto programovacím jazyce se můžete dozvědět zde, viz [2]. Představoval jsem si to tak, že s pomoci PHP si nadeklaruji třídy a metody, které budou pracovat s obrázkem a jakmile si uživatel nahraje fotografii na server, vytvořím si objekt a přiřadím mu tento obrázek a pak s ním budu nadále pracovat.

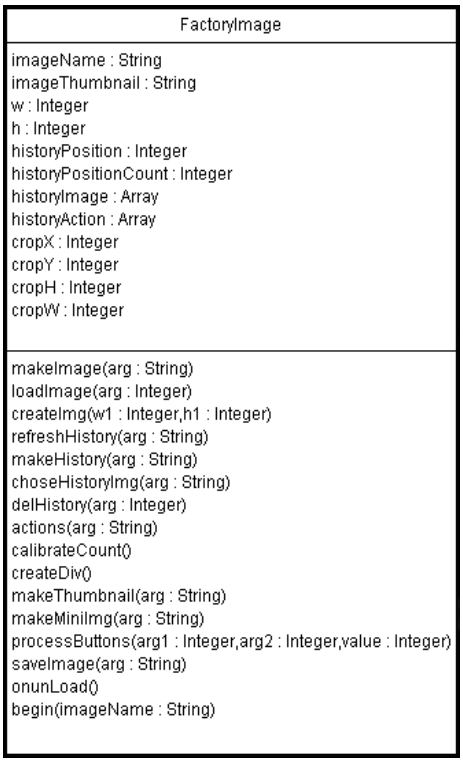
Jelikož jsem nevěděl, jak by se tento návrh přesně realizoval, neboť mé znalosti nebyli na dostačující úrovni, tak mi nezbylo nic jiného, než začít tuto problematiku více zajímat a studovat. Již dříve jsem se chtěl více dozvědět a blíže se seznámit s tvorbou webových rozhraní a teď k tomu byla výborná příležitost. Tudíž začal jsem hledat na internetu různé techniky, technologie a postupy jak se tyto rozhraní tvoří. Narazil jsem na pojmy, jako jsou JavaScript, PHP, DOM, AJAX a další. Když jsem se s těmito pojmy seznámil blíže a této problematice jsem více porozuměl, tak jsem zjistil, že mé původní představy o realizaci byli zcela milné. Jsou totiž daleko efektivnější metody jak dosáhnout tíženého výsledku. Ale jak toho dosáhnout, co použít za technologie či postupy si detailněji povíme níže v jednotlivých částech návrhu.

3.2 Klient

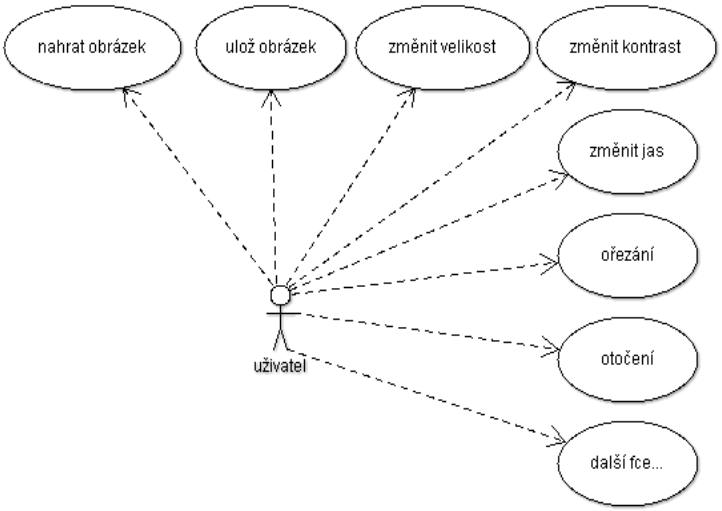
Jaké jsou požadavky na klienta, co by měl umět, umožňovat, zprostředkovávat? Toto byli mé další otázky, na které jsem se snažil odpovědět, než jsem se pustil dál do práce. Měl by v prvé řadě poskytnout samotné rozhraní, ve kterém se bude obrázek zpracovávat. Také by se měl starat a zajišťovat interakci s uživatelem, reagovat na něho požadavky, umožnit mu nastavení parametrů a hodnot požadovaných akcí. Tyto akce pak bude následně předávat serveru ke zpracování. A jejich výsledek pak opět zobrazovat uživateli. Tudíž by se měl starat a zajišťovat právě tuto komunikaci mezi samotným klientem a serverem.

Na základě nastudovaných materiálů jsem se rozhodl využít výhod jazyka JavaScript. Ten pracuje na straně klienta, a tudíž zpracovávání různých akcí nezatěžuje server, je rychlejší a snižuje objem přenesených dat na síti. Dosažení rychlé interakce rozhraní s uživatelem, toto bylo jedno z kriterií a z toho důvodu jsem si klienta navrhl v plné režii tohoto jazyka. Jestliže se chcete dozvědět více o tomto programovacím jazyku, můžete se podívat a něco si o něm přečíst zde, viz [3]. Pod takovým klientem si můžeme představit klasickou webovou stránku vytvořenou pomocí HTML a CSS. HTML (*Hypertext Markup Language*) je základní značkovací jazyk pro vytváření internetových stránek, umožňuje nám umístit a prohlížet různé dokumenty a prezentace. CSS (*Cascading Style Sheets*) je to jazyk, který nám umožňuje definovat, jak bude výsledná stránka vypadat. Umožňuje oddělit vzhled od obsahu stránky. Díky této vlastnosti je tvorba webových stránek daleko přehlednější. Více se můžete o těchto jazycích dozvědět například zde, viz [4], [5].

Společně s načtením stránky se také načtou a stanou se součástí stránky i tzv. skripty, které jsou napsané pomocí CSS a nebo také pomocí jazyka JavaScript. Obecně těchto typů souborů mohou být do obsahu stránky načteno i více. Jinými slovy klient bude klasická internetová stránka, tak jak ji známe s prvky HTML, kde se o jeho správnou funkčnost budou starat skripty napsané pomocí jazyka JavaScript. V něm jsem si navrhl třídu FactoryImage, která obsahuje celou řadu atributů jako např. název obrázku, jeho šířku, výšku, historií obrázků, akcí a jiné. Třída obsahuje také metody, které s obrázkem pracují a také metody, které zajišťují správnou funkčnost klienta. Na kompletní třídu se můžete podívat na *Obrázek **1***. A pro rychlý přehled akcí, které bude moci uživatel provádět, si můžete povšimnout diagramu případu užití *Obrázek **2***.



Obrázek 1: Diagram tříd



Obrázek 2: Diagram případu užití

3.3 Historie obrázků

Tato historie by měla přispět uživateli k zpřehlednění práce, kterou vykonává nad jednotlivým obrázkem. Bude si v ní moci listovat, vracet se k libovolné operaci, kterou provedl. Urychlí a z efektivní mu práci například v tomhle tom případě. Rozhodne se, že chce fotografii ořezat

a následně provede nad tímto ořezaným obrázkem několik dalších operací a najednou si uvědomí, že to vlastní ořezání neprovedl správně a že by ho chtěl provést znovu. V klasickém případě, jak tomu je u standardních editorů by se musel vrátit na začátek tlačítkem zpět a začít svou práci znovu od začátku. V tomto případě může využít výhod právě této historie. Vrátí se do posledního kroku před tím vlastním ořezáním a to provede znovu. Následně se pak na všechny efekty a změny, které následovali, provedou na aktuálně ořezaném obrázku.

Zvažoval jsem jak tuto historii navrhnout a realizovat, napadlo mě několik možností, které jsem ale díky získávání znalostí a studiu postupně zavrhoval. Nakonec mě napadlo toto řešení. K jeho realizaci si budu muset zapamatovat informace o názvech jednotlivých obrázků a akcí, které nad nimi byly prováděny. K tomu mi budou sloužit dvě pole. Kde právě v jednom si budu uchovávat názvy a ve druhém akce. Dále také bude potřeba mít nějaký ukazatel, který mi bude ukazovat na aktuální pozici, na které se nacházím. Uživateli by se tato historie mohla zobrazovat například ve třech divech, ve kterých by byly načteny obrázky podle názvů v poli. Takto by nějak mohla výsledná historie obrázků vypadat.

3.4 Server

Ten by měl v prvé řadě zajišťovat akce, které po něm skrze klienta uživatel požaduje. Od nahrání samotného obrázku na server, přes požadavky na jeho úpravu (ořezání, změna kontrastu, světlosti, apod.), až po uložení fotografie zpět na disk.

Na serveru bude uložen skript napsaný v jazyce PHP. Tento bude sloužit jako jakýsi rozhodovací mechanismus při zpracovávání a změně obrázků. Na základě požadované akce se server rozhodne, jakou metodou bude obrázek zpracovávat. Chtěl bych totiž rozhraní navrhnout a realizovat tak, aby bylo možné používat různých technik pro samotné zpracování fotografií. Nechtěl bych, aby výsledkem mé práce byl jen editor s konečným počtem funkcí, ale chtěl bych docílit toho, aby o další funkce mohl být rozšířen nějakým modulem. Například pro klasické a jednodušší změny obrázků budeme používat GD knihovnu PHP a pro náročnější výpočty by se mohla použít nějaká aplikace napsaná v jazyce C/C++ s podporou OpenCV.

3.5 Vzájemná komunikace klient/server

O požadavcích na klienta a server, o tom jak by měli vypadat, jsme si již něco málo řekli výše. Ale nic jsme si ještě neřekli o jejich vzájemné komunikaci, jak si mezi sebou budou posílat data a jaký budou mít tyto data podobu. Ke komunikaci mezi klientem a serverem, budu využívat technologii AJAX (*Asynchronous JavaScript and XML*). Ta nám umožní asynchronní přenos dat mezi

serverem a klientem. Zpracovávání různých akcí bude tedy rychlejší. Nečeká se na to, až uživatel dokončí určitou sadu instrukcí a ty pak odešle stisknutí nějakého tlačítka na server. Ten je následně zpracuje a vrátí zpět klientovi. Tomuto se říká tzv. synchronií přenos dat. AJAX nám právě umožní to, že jsou data zpracovávána průběžně a na stránce se aktualizuje právě ta část, ve které byla provedena změna. Není nutné tak znovunačtení a překreslení cele stránky po každé požadované akci. Díky tomuhle by také měl být za důsledek snížení datového toku a zátěže na síti. Více si o této technologii můžete dozvědět také zde, viz [6].

Při tvorbě tohoto rozhraní jsem pracoval s technologiemi, se kterými jsem se setkal v podstatě poprvé a neměl jsem s nimi příliš velké zkušenosti. Tudíž ze začátku když jsem začal využívat AJAX a experimentovat s ním, tak jsem vše dělal pomocí čistého JavaScriptu. Bylo třeba ošetřit spoustu věcí a to především práci v různých verzích webových prohlížečů. Teprve později, kdy už jsem do této problematiky pronikl daleko víc, tak jsem zjistil, že existuje knihovna jQuery a díky ní, už spoustu problémů a úskalí nemusím ošetřovat sám. Je to velice obsáhlá JavaScriptová knihovna, která podstatně usnadňuje a urychluje práci. Existuje do ní spousta přídavných pluginu, které ji využívají. Určitě seznámit se s „čistým“ JavaScriptem je zajímavá a cenná zkušenost, ale nakonec jsem se jednotlivé funkce rozhodl předělat a začít více využívat této knihovny a jejích výhod.

3.5.1 Popis protokolu

	Název obrázku	Název funkce	Šířka	Výška	Ukazatel
Odesílaní dat	imageName	arg			
Příjem dat	imageName		w	h	imageExist

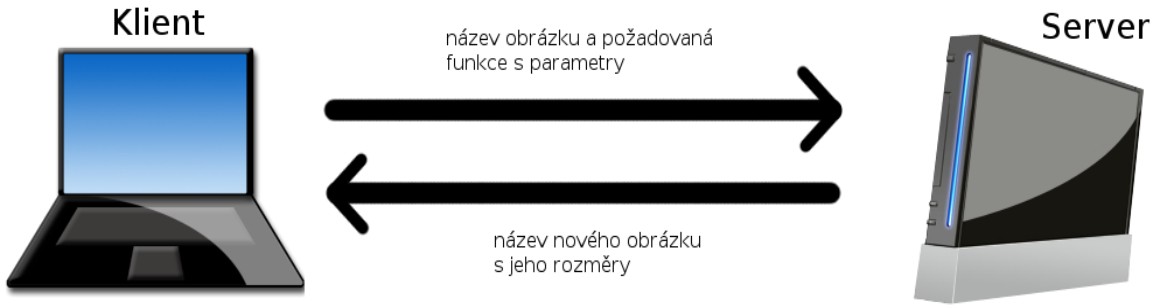
ImageName, *arg*, *w*, *h* jsou proměnné, které obsahují důležitá data, které si mezi sebou předávají server a klient.

Při odesílání dat, klient odesílá obsahy proměnný *imageName* a *arg*. Jak už název napovídá, tak u prvně jmenované proměnné je obsahem název obrázku, na kterém chceme provést nějakou akci. Druhá obsahuje řetězec, který se skládá s názvu požadované akce společně s jejími parametry například (*"action=rotate&param="* + *value)*.

Oproti datům při odesílání se přijímaná data trošičku liší. Klient obdrží od serveru data ve formátu JSON (JavaScript Object Notation), je to určitý formát určený pro výměnu dat. Více se můžete dozvědět o něm zde, viz [7]. Nejprve klienta zajímá jaký je obsah proměnné *imageExist*, která udává informaci, zda server obrázek nalezl a provedl na něm požadovanou změnu. Dále také obdrží název nového obrázku s jeho rozměry (šířku a výšku).

3.5.2 Předávání dat mezi serverem a klientem

Server a klient tedy spolu komunikují pomocí technologie AJAX a s pomocí výše popsaného protokolu. Vlastní komunikace pak vypadá asi takto. Jakmile je požadována nějaká akce na změnu fotografie, tak se zavolá metoda *makeImage*. Je to asi nejdůležitější metoda celé třídy, zajišťuje komunikaci mezi serverem a samotným klientem. Stará se o to, aby nově získaný název obrázku byl uložen v poli na správné pozici, a následně volá metody pro jeho zobrazení a přidání do historie obrázků. Tato metoda odešle serveru aktuální název obrázku, název požadované akce společně s jejími parametry. Informace odchytí PHP skript na serveru, kterému jsou data poslána pomocí metody GET. Ten má poté k těmto datům přístup přes globální proměnné *$_GET["nazevPromenne"]*, se kterýma pak může bez problému libovolně pracovat. Klient mezitím vyčkává na server, až obdržená data přečte a zpracuje. Podle požadované akce a jejích parametru vytvoří nový přepracovaný obrázek a následně odešle zpět klientovi název tohoto nově vytvořeného, přepracovaného obrázku a údaje o jeho šířce či výšce. Pro větší názornost, jak tato komunikace vypadá a funguje, se můžete podívat na *Obrázek **3***.



Obrázek 3: Komunikace mezi serverem a klientem

4 Implementace klienta

Jak už jsem naznačil výše, tak hlavní funkce klienta bude zajišťovat skript napsaný v jazyce JavaScript. Tudíž bez zapnuté této podpory v prohlížeči webová aplikace nebude fungovat. Skript se v podstatě stará o vše důležité co se děje u klienta, zajišťuje správné odezvy tlačítek, bezproblémový chod historie obrázků apod. Jinými slovy tvoří jádro celého systému a práce s obrázkem na klientovi.

Níže v této kapitole bych chtěl nastínit, jak vznikali a jak fungují některé dílčí části na klientovi a také bych rád nastínil, jak vznikla výsledná podoba grafického návrhu.

4.1 Grafický návrh

Při samotné implementaci klienta jsem si dal také práci s rozvržením samotné stránky a jejich prvků. Za úkol jsem si totiž kladl, aby výsledná aplikace byla jednoduchá, přehledná a intuitivní a tak bylo důležité jak jednotlivé objekty na stránce umístit. Rozhodl jsem se, že stránky optimalizuji pro širokoúhlé monitory. K tomuto rozhodnutí mě vedlo to, že jsou čím dál tím více rozšířenější a populárnější, ať už co se týče notebooků tak LCD monitorů. Díky tomuto kroku je k dispozici také daleko větší plocha pro samotnou úpravu obrázků.

Nejprve jsem měl představu takovou, že v záhlaví bude logo, dále pak ve středu stránky by v levé části bylo menu s akčními tlačítky pro úpravu obrázků, v pravé části by byla načtena samotná fotografie a v záhlaví by byla historie obrázků. Nejdříve jsem takto své rozhraní implementoval, ale po chvilce práce s takto navrženým rozhraním jsem usoudil, že tento návrh nesplňoval mé kriteria. Asi největší problém byl v tom, že při nahrání obrázku o větších rozměrech nebyla historie na první pohled vůbec vidět. Ani samotná práce s ní nebyla příliš pohodlná, uživatel by byl nucen pořád se posouvat dolů na zápatí, aby mohl historií obrázků využít.

Na základě tohoto zjištění jsem se rozhodl historii přesunout. Využil jsem toho, že jsem optimalizoval a dělal stránky pro širokoúhlé rozlišení a tak jsem historii umístil ne pod načtený obrázek, ale vedle, svisle vedle něj. Tím jsem docílil toho, že je historie „pořád“ na očích a řekl bych, že teď se s ní bude i lépe pracovat. Na výslednou podobu designu stránek se můžete také podívat na přílohu 1, která se nachází na samotném konci dokumentu.

4.2 Realizace historie obrázků

V této podkapitole se nejdříve zmíním, jak vznikala výsledná podoba historie obrázků a poté se také zmíním, jak tato historie pracuje a jak funguje.

4.2.1 Grafický návrh historie obrázků

Nejdříve jsem tuto historii chtěl prezentovat a také zrealizoval způsobem, jakým jsem si ho prvně navrhl. Ve spodní části jsem si vytvořil tři divy, mezi nimi byli šipky, které sloužili posunování se v historii. V každém z těchto divu byl vytvořen element *img*, kterému byl pomocí JavaScriptu vždy změněn atribut *scr* podle aktuální pozice v poli a přiřazen název obrázků, který se následně zobrazil. Jak tento princip zobrazování mohl vypadat, se můžete podívat na *Obrázek 4*.

Obrázek 4: Ukázka prvotního návrhu historie obrázků

Tento systém plnil svou funkci dobře, na první pohled nevypadal ani špatně, ale samotné posouvání v historii bylo takové strohé a obyčejné. Řekl jsem si, že by to chtělo něčím oživit a vylepšit. Začal jsem tedy hledat na internetu techniky, jak bych toho mohl docílit. Díky tomuhle jsem se v podstatě seznámil s již zmíněnou knihovnou jQuery.

Díky získaných znalostí o jQuery a nastudování dokumentace nalezeného pluginu, jsem se rozhodl, že výše popsaný systém zobrazování historie přepracuji. Musel jsem tomu podřídit zpracování a zobrazování jednotlivých obrázků, neboť plugin nepracuje s divy, ale čistě s posloupností elementu *img*. Musel jsem tudíž statické divy nahradit dynamicky se tvořícími obrázky. Toho jsem docílil pomocí technologie DOM, viz [8]. Ta mi umožní, že jakmile vytvořím nový obrázek, můžu jej dynamicky vložit do obsahu stránky, případně jej opět odstranit. Díky této změně a použití pluginu pro jQuery se mě povedlo webové rozhraní trošičku oživit. Docílil jsem toho, že už posouvání v historii nevypadá tak jednoduše a obyčejně jako v předešlém případě.

4.2.2 Popis vlastní implementace historie obrázků

Jak už jsem naznačil v dřívější kapitole tak pro uchování informací, které budu potřebovat k vytvoření historie, budu mít dvě pole *historyImage*, *historyAction* a ukazatel na aktuální pozici *historyPosition*. Po provedení každé operace nad obrázkem si do polí uložím aktuální název a akci, která byla provedena. Takto si uchovávám potřebné informace, se kterými pak dále pracuji.

Celý princip zobrazování obrázků v historii pracuje pak asi takto. Vlastní plugin se v podstatě jen stará o animované posouvání fotografií v rámci jednoho divu o dané velikosti. Ale to jakým způsobem tyto obrázky do tohoto divu dostanu, nebo co se stane po kliknutí na jednotlivé fotografie, tyto věci jsem už musel implementovat. Přidávání fotografií do historie obrázků probíhá tímto způsobem. Jakmile je zpracována požadovaná akce od uživatele, tedy nový přepracovaný obrázek je už uložen fyzicky na serveru. Klient poté od něj obdrží jeho název a vytvoří nový element *img* s jednoznačným *id* a přiřadí atributu *src* název nového obrázku. Ten se poté zobrazí v historii. Samotné *id* má velice důležitou vlastnost, je číselného tvaru a odpovídá číslu aktuální pozice v poli, kde jsou uloženy názvy obrázků, tedy hodnotě *historyPosition*. Toto *id* mě pomůže k určení o který obrázek se jedná, když si jej uživatel v historii obrázků vybere. Také mi pomůže k určení, na jaké pozici se v ní nacházím, neboť současně s *historyPosition* mám také proměnnou *historyPositionCount*, která mě udává aktuální počet fotografií zobrazených v historii obrázků.

Hodnoty těchto dvou proměnných jsou velice důležité pro speciální funkčnost zpětného přepracování fotografií. O jeho hlavním využití jsem se již zmínil v kapitole 3.3. Nyní bych chtěl blíže osvětlit, jak tento princip funguje a pracuje. Po zavolání metody *makeImage* se na jejím konci volá metoda pro zobrazení nového obrázku do hlavního okna, kde pak můžeme s tímto obrázkem následně dále pracovat. V této metodě se pak mimo jiné volá i metoda *createImg*, která nám zajistí vytvoření nového elementu *img* a zobrazení nové fotografie také v samotné historii obrázků. Při volání této metody mohou nastat jen dvě situace. Buď se nacházíme na konci historie a tudíž vytvoření nového elementu *img* nic nebraní, anebo se nacházíme na jiné pozici v historii. V takovém případě je pak zbytečné tvořit tento element nový, stačí změnit jen jeho atribut *src*. Jelikož teď vím, že se nenacházím na konci historie obrázků, ale někde v ní, tak následně zavolat metodu *refreshHistory*, která mi zajistí přepracování všech fotografií, které následují za aktuálně zpracovaným.

4.3 Editační okno a práce s miniaturami

Při návrhu této aplikace jsem také přemýšlel nad tím, jak uživateli umožnit měnit různé parametry akcí, které bude chtít nad obrázkem vykonat. Jednoduché to bylo u akcí typu převod do stupnice šedi. Tam nebylo co řešit, neboť u této akce nepotřebuji nějak měnit její parametry a tak mi stačí pro její

vykonání jedno tlačítko v menu. Ale problém nastal u jiných funkcí, jako jsou například, změna kontrastu, nebo zesvětlení či ztmavení a další. Bylo to třeba udělat tak, aby to permanentně nezabíralo prostor na webové stránce, říkal jsem si, že by to mohlo být například někde skryté, neviditelné pokud to zrovna uživatel nepotřebuje.

Nakonec mě napadlo to udělat tímto způsobem. Při stisknutí tlačítka požadované akce vytvořím dynamicky pomocí DOM div a v něm potřebné prvky pro danou akci. Obsahem tohoto divu pak můžou být například dvě miniatury obrázku, kdy jedna bude znázorňovat aktuální obrázek a druhá nám bude ukazovat obrázek po aplikování chtěného efektu. Dále pak může být součástí tohoto divu například posuvník, pomocí kterého budeme stanovovat hodnoty, které budou ovlivňovat danou funkci. Součástí tohoto divu také bývá tlačítko *ok* pro potvrzení dané změny nad obrázkem a *křížek*, který tento vytvořený div skryje a zruší ho i všechny prvky které obsahoval. V praxi to pak vypadá tak, že po stisknutí tlačítka se tento div animovaně pomalu zobrazí a po jeho zavření se opět skryje. Docílit tohoto efektu mě pomohli dvě funkce jQuery a to *show()* a *hide()*.

V tomto divu, který slouží k ovlivňování parametrů funkcí, pracuji s miniaturami původního obrázku a to z důvodu, abych snížil datový tok a urychlil tím tak zobrazování přepracované fotografie. Tímto divem jsem chtěl totiž docílit urychlení práce nad obrázkem, aby uživatel nemusel například změnit kontrast o 5 procent, následně zjistil, že to bylo málo a tak dal zpět a celý proces tak opakoval znova a znova. Toto prostředí má právě urychlit a umožnit uživateli nalézt rychleji tíženou hodnotu, která pak ovlivní původní obrázek. Na základě této úvahy právě předpokládám, že jednotlivé funkce se budou provádět několikrát a proto v tomto prostředí pracuji s miniaturami obrázků.

5 Implementace serveru

Zjednodušeně princip úprav obrázků na serveru je přibližně takový. Rozhodl jsem se, že jednotlivé obrázky budu uchovávat přímo na fyzickém disku serveru po celou dobu, co se s nimi bude pracovat. A to z důvodu, že chci implementovat historii obrázků, a kdybych musel pokaždé obrázek přepracovávat znova, zbytečně by to příliš zatěžovalo server. Uživatel si tedy nahraje svou fotografii na server. Ta se uloží například pod takovýmto názvem (*2009-04-04_00-12-34_ukazka_0.jpg*). Řetězec začíná aktuálním datem a časem, následuje název fotografie a na konci je číslice, která se postupně inkrementuje. Tento formát jsem zvolil z toho důvodu, abych co nejvíce eliminoval možnost, že dva uživatele budou chtít pracovat s obrázkem stejného názvu v tutéž dobu. Po nahrání na server je vše připraveno k použití. Každá změna, úprava nad tímto obrázkem bude mít za důsledek vytvoření nového obrázku s inkrementovanou číslicí na konci řetězce. Po skončení práce a ukončení okna prohlížeče se pak všechny tyto fotografie na serveru vymažou.

Hlavním úkolem serveru tedy je, aby správně zajišťoval a reagoval na požadavky od uživatele, který s ním komunikuje skrze klienta. Jinými slovy, uživatel bude chtít provést ořezání svého obrázku. Klient mu slouží jako jakési rozhraní a prostředek, pomocí kterého provede výběr oblasti, kterou chce ořezat, ale samotné ořezání provede až server. Ten pak v konečné fázi odešle klientovi nově vytvořený a správně ořezaný obrázek.

Na serveru je uložen PHP skript, který přijímá data od klienta a to název obrázků, akci kterou chce uživatel provést a její parametry. Na základě požadované akce rozhodne, jakou metodou se tento obrázek bude zpracovávat. Pro samotné zpracování obrázků využívám dvou technologií, dvou principů jak tyto obrázky přetvářet. Jedním je GD knihovna PHP a druhým je aplikace implementovaná s pomocí OpenCV. Po zpracování a vytvoření nového obrázků tento skript odešle název nového obrázku společně s jeho rozměry zpět klientovi.

5.1 GD knihovna PHP

Je to knihovna PHP, která obsahuje spoustu funkcí, které nám umožní provádět různé změny s obrázkem. Ať už se jedná o jeho zmenšení, ořezání, nastavení nějakého barevného filtru, tak také lze i pomocí této knihovny vytvořit a nakreslit i vlastní obrázky, nebo přidat do samotné fotografie nějaký text. Něco málo v češtině si o GD knihovně můžete přečíst zde viz. [9], kde najdete i odkaz na různé praktické ukázky.

Ukázka funkcí, které tato knihovna poskytuje:

- **imagecreatetruecolor** (int *$sirka*, int *$vyska*) – Funkce slouží k vytvoření prázdného obrázku.
- **imagecreatefromjpeg** (string *$nazev_souboru*) – Tato funkce vytvoří nový obrázek z jpg fotografie.
- **imagejpeg** (zdroj *$obrazek* [, string *$nazev_souboru* [, int *$kvalita*]]) – Funkce, která uloží výsledný obrázek fyzicky na disk se zadanou kvalitou.
- **imagecolorset** (zdroj *$obrazek*, int *$index*, int *$cervena*, int *$zelena*, int *$modra*) – Tato funkce aplikuje barevný filtr nad obrázkem se zadanými hodnotami červené, zelené a modré barvy.
- **imagerotate** (zdroj *$obrazek*, float *$uhel*, int *$barva_pozadi*) – Funkce, která nám umožní pootočit obrázek o zadaný počet stupňů.

Při implementací funkcí pomocí této knihovny jsem se nesetkal s většími problémy. Podmínkou pro využívání této knihovny je, že musí být na serveru nainstalována a v souboru *php.ini* povolena její podpora. Většinou ale bývá tato knihovna součástí každého serveru, který nabízí na internetu prostor pro vlastní prezentace. Funkce této knihovny se používají velice jednoduše.

Ukázka jednoduché funkce pro převod do stupnice šedi:

```
// nejdříve vytvoříme nový obrázek z původního
$novyObrazek = imagecreatefromjpeg ($puvodniObrazek);
// poté aplikujeme filter pro převod do stupnice šedi
imagefilter($novyObrazek,IMG_FILTER_GRAYSCALE);
// a nakonec obrázek uložíme fyzicky na disk
imagejpeg($novyObrazek, $nazevObrazku, 100);
```

5.2 OpenCV

OpenCV (Open Source Computer Vision) je knihovna funkcí zaměřená na zpracování obrazu a počítačového vidění v reálném čase. Existuje spousta odvětví, kterými se tato knihovna zabývá, a pro které obsahuje řadu funkcí. Jakož jsou například identifikace objektů, rozpoznávání obličeje, gest a další.

Aplikace vytvořena pomocí této knihovny, je napsaná v jazyce C/C++ a uložena na serveru v podobě binárního spustitelného souboru. Při jeho spuštění, se mu jako parametr předá jednoznačný

identifikátor funkce, kterou požadujeme, její parametry, název aktuálního obrázku, na kterém chceme tuto změnu aplikovat a název nového obrázku, který bude výstupem této aplikace. Tento název bude poté opět pomocí technologie AJAX poslán klientovi a ten novou fotografii uživateli zobrazí. Tento soubor se spouští pomocí PHP a to funkce *exec()*.

Nejdříve jsem si myslel, že budu daleko více využívat výhod této knihovny a implementuji pomocí ní daleko více funkcí než s pomocí GD knihovny. Ale postupně, jak jsem začal aplikaci implementovat, tak jsem zjistil, že je zde takový větší problém. Internetové servery, které poskytují zdarma web hosting, tak mají standardně zapnutý *safe_mode*, tudíž mají mimo jiné zakázáno spouštět z internetových stránek pomocí příkazu *exec* externí binární soubory. Toto omezení se týká i školního serveru Eva, na kterém mají studenti VUT možnost nahrát své webové stránky. Takže nakonec jsem se rozhodl pro jednu demonstrační funkci, kterou budu zpracovávat pomocí této aplikace, ale bude bohužel fungovat jen na localhostu, kde bývá *safe_mode* standardně vypnutý. Problém s internetovým web hostingem by se nejspíš vyřešil při přechodu na placený, kde by se určitě dala deaktivace tohoto omezení domluvit.

6 Testování

Důležitou součástí vývoje nějaké aplikace je určitě i její testování, zjištění zda vše pracuje a funguje tak jak má. Nikdy nelze opravit všechny chyby, neboť opravením jedné může mít za důsledek vznik nových, ale testováním lze spousta těchto chyb odhalit a opravit. Myslím si, že vývojář sám nikdy svou aplikaci dobře neotestuje. On sám může přijít na spoustu chyb, ale myslím si, že nikdy neodhalí tolik co třeba nějaký jiný znalý uživatel, který s touto aplikací přijde poprvé do styku. Programátor, který vyvíjí nějakou aplikaci, totiž zná svůj program a ví, co po něm chce, co mají jednotlivé funkce dělat, jaký má být jejich výsledek. A tak při jeho testování a práce s ním, to vždy provádí určitým způsobem, tak nějak stejně a proto případné skryté chyby se nemusejí projevit. Naopak právě, když nějakému uživateli, který tuto aplikací vidí poprvé, zadáte nějaký testovací úkol, je u něj velká pravděpodobnost, že k tíženému výsledku dojde jiným způsobem než samotný programátor. Může se třeba několikrát splést, vrátit změny zpět a tak podobně. Samozřejmě pokud tento úkol zadáte většímu počtu uživatelů, tak také můžete odhalit více chyb a nedostatků. Tento způsob testování má mimo toto ještě jedno obrovské pozitivum, dozvíte se také různé postřeny, námitky, nedostatky či nápady na vylepšení od samotných uživatelů, pro které je ve většině případů samotná aplikace určena.

6.1 Praktický test

Na základě předešlých úvah jsem se rozhodl tímto způsobem aplikaci otestovat. Chtěl bych s vybranou skupinou deseti lidí udělat pár testů a současně provést i nějaká měření této aplikace.

Způsob jakým jsem testery vybíral:

- Měli by se s touto aplikací setkat a pracovat poprvé.
- Snažil jsem se o to, aby v této skupině byli uživatelé od těch zkušenějších až po ty méně zkušené v práci s počítačem.
- Také, aby zde byli uživatelé, kteří upravují své fotografie často a ti, kteří je upravují zřídka kdy.

Tento test se skládal ze třech praktických úkolů. U prvních dvou se jednalo o to, že uživatel podle poskytnutého seznamu napsaných bodů si měl vyzkoušet a provést nějaké úpravy s obrázkem. První úkol byl kratší a jednoduchý, měl sloužit k seznámení s prostředím a vyzkoušení si pár jednoduchých funkcí. Druhý byl o něco delší a ten měl za úkol také poukázat na zajímavou funkci historie obrázků a to na již výše zmíněné zpětné přepracování fotografií. Posledním úkolem byla

tzv. „volná zábava“, kde si měl uživatel libovolně vyzkoušet různé funkce a práci s touto aplikací. Poté jsem každého z nich požádal o vyplnění krátkého formuláře.

Po dobu provádění úkolu jsem si chtěl měřit čas, jak dlouho bude uživateli, který tuto aplikací uvidí poprvé, trvat než tyto úkoly splní. Poté chci časy zkušenějších uživatelů a těch méně zkušených mezi sebou porovnat. Tímto bych mohl a dostat jakýsi obrázek toho jestli je aplikace intuitivní a přehledná, nebo ne. Dále bych si chtěl změřit objem přenesených dat a porovnat ho s hodnotou, kde bych provedl stejný úkol, ale nevyužil při práci miniatur obrázku. Tímto testem bych si chtěl ověřit, jestli jsou tyto miniatury přínosem v aplikaci či nikoliv. Také bych chtěl pozorováním zjistit, jak často a pokud vůbec budou využívat historii obrázků při těchto jednoduchých úkolech. Toto by měl být také jakýsi ukazatel toho, jestli je rozhraní přehledné. Úkoly jsou totiž jednoduché, a proto si myslím, že by v těchto případech měla být historie využívána co nejméně. Udělat aplikaci přehlednou a jednoduchou na ovládání, to byl totiž jeden z mých hlavních cílů.

6.2 Zhodnocení výsledků praktického testu

Test č.1:

- Testovaná fotografie měla velikost 77,7 KB
- Test začal nahráním fotografie na server, poté uživatel provedl výřez detailu na obrázku, následně zvýšil jeho kontrast, udělal z něj černobílou fotografii, o něco málo zesvětlil a nakonec takto upravený obrázek uložil na disk svého počítače.
- Průměrná hodnota datového toku byla 572KB.
- Průměrný čas, který uživatelé nad tímto úkolem strávili, se pohyboval okolo 160 sekund. A rozdíl mezi zkušenějším uživatelem a tím méně zkušeným byl okolo 43 sekund.
- Historie obrázků se u tohoto úkolu nevyužívala skoro vůbec. Byli případy, kdy ji uživatelé vyzkoušeli, ale to bylo spíše ze zvědavosti, aby zjistili, jak pracuje.

Test č.2:

- Testovací fotografie měla velikost 213 KB
- Test začal opět nahráním fotografie na server, opět byl uživatel vyzván k tomu, aby udělal určitý detail obrázků, poté použil sepia filtr, následně ztmavil obrázek, pak otočil o 90° a zabarvil jej do modra. Poté zjistil, že se mu takto ořezaný obrázek nelíbí a byl požádán, aby výřez znovu opakoval. Po vykonání této akce, byl následně upozorněn na změny co se postupně projevili v celé historii obrázků. A na konci byl opět požádán, aby výslednou fotografii opět uložil na disk počítače.

- Průměrná hodnota datového toku byla 350KB. Tato hodnota je ovlivněna také tím, že se pracovalo po celou dobu pouze s malým detailem fotografie.
- Průměrný čas se pohyboval okolo 141 sekund a rozdíl mezi zkušenějším a méně zkušeným uživatelem už nebyl tak vysoký, pohyboval se okolo 25 sekund.

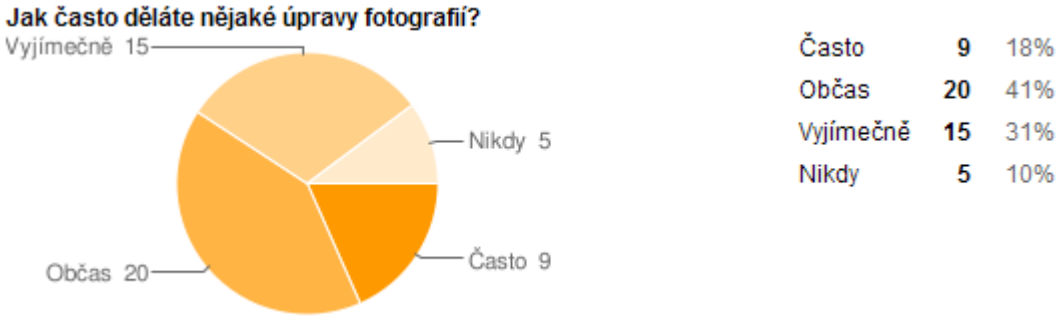
Zhodnocení naměřených výsledků:

Výsledkem datového toku naměřeného v prvním testu je číslo 572KB, tato hodnota je ovlivněna tím že uživatelé při zvyšování kontrastu, či zesvětlení využili tuto funkci několikrát. Tudíž se tak datový tok navýšil. Ale díky tomu, že při těchto funkcí využívám miniatury původního obrázku, tak jsem povedlo datový tok rapidně snížit. Provedl jsem totiž také test bez použití těchto miniatur a dostal jsem se na hodnoty přes 2MB i více, záleželo na počtu pokusů zvýšení kontrastu a světlosti. Z naměřených výsledků tedy vyplývá, že v tomto případě se mi povedlo snížit datový tok až čtyři krát. Ono toto číslo je ale hodně relativní, závisí totiž na tom, jaké operace s obrázkem bude uživatel dělat, takže ve výsledku může být naměřená hodnota nižší, ale taky podstatně vyšší.

Co se týče naměřených času, tak u zkušenějších uživatelů šlo vidět, že upravují fotografie častěji, orientace v rozhraní i samotná úprava fotografie jim nedělala sebemenší problémy. U méně zkušených uživatelů šlo naopak vidět, že takovéto operace neprovádějí příliš často, ale po chviličce zkoumání kde se co nachází, už také upravovali fotografii bez problému. V druhém testu, už pak časový rozdíl mezi jednotlivými uživateli nebyl tak vysoký. Myslím si, že to bylo dáno tím, že už s touto aplikací nepracovali poprvé a také tím, že zkušenější uživatelé jsou možná o něco málo zručnější, ale hlavně mají v tomto směru větší praxi.

6.3 Vyhodnocení odpovědí formulářových otázek

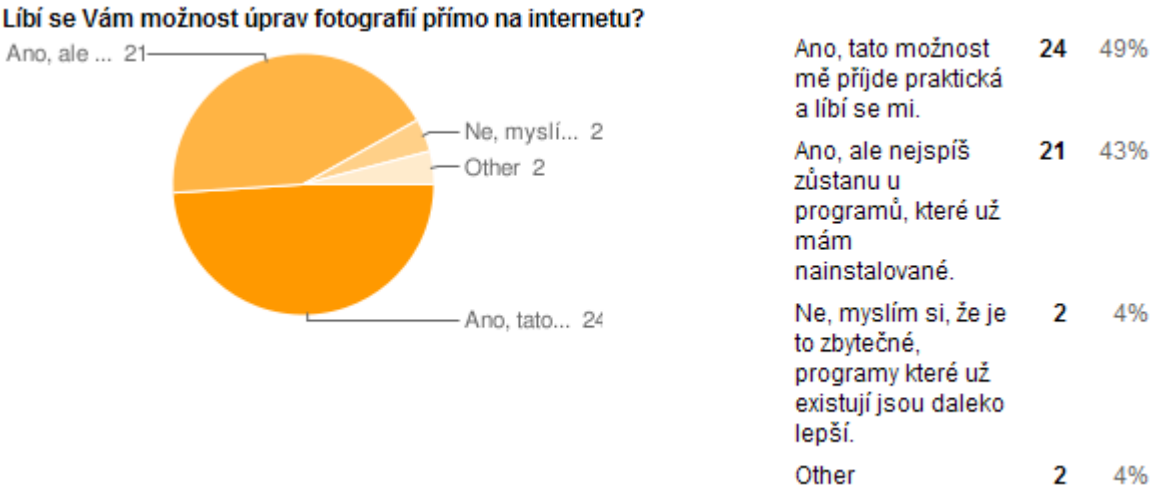
Praktické testy jsem prováděl jen s malou skupinou uživatelů, což o aplikaci samo o sobě mnoho nepoví. A proto jsem ještě vytvořil formulář, se kterým jsem se snažil o to, aby ho vyplnil daleko větší počet uživatelů.



Obrázek 5 : Otázka č.1

Touto otázkou jsem se snažil zjistit, jaká skupina uživatelů tento formulář vyplňovala. Jestli se nejedná například jen o nějakou úzkou skupinu uživatelů, kteří fotografie ještě nikdy neupravovali nebo naopak.

Výsledek: Z grafu můžeme vyčíst, že tuto aplikaci otestovala pestrá množina uživatelů.



Obrázek 6 : Otázka č.2

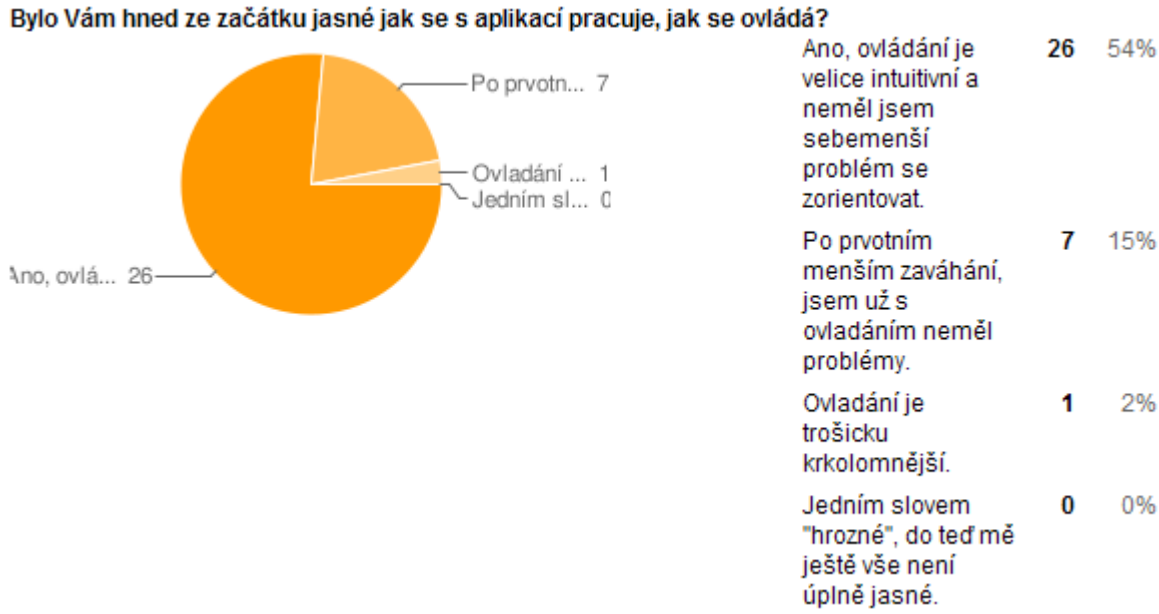
Touhle otázkou jsem chtěl zjistit, jaký mají uživatelé na tuto možnost úpravy fotografií názor. Jestli se jim to líbí a rádi by toho využívali či nikoliv.

Výsledek: Velké většině dotázaných uživatelů se tato možnost líbí, ale využívat by ji využívala asi jen polovina. Zbytek by používala nadále převážně aplikace, které mají nainstalované fyzicky na svém disku.

Obrázek 7 : Otázka č.3

Také jsem chtěl zjistit, jaký mají uživatelé názor na design stránek.

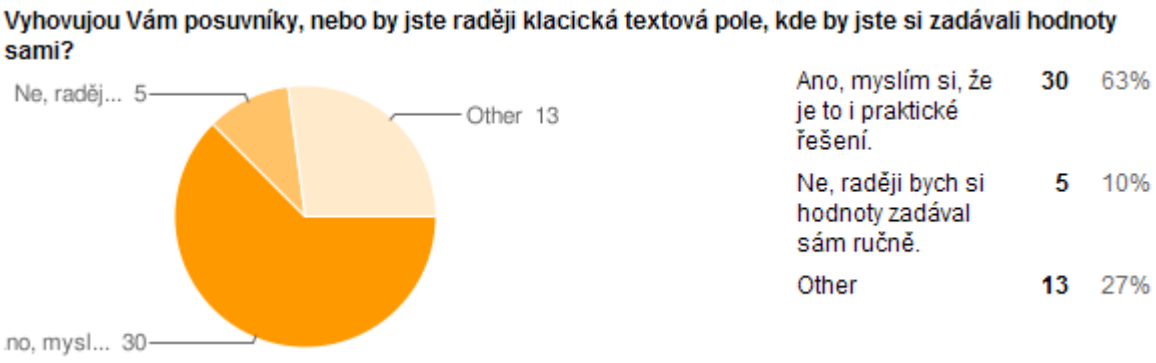
Výsledek: Upřímně mám z tohoto výsledku velikou radost, Jsem rád, že se jim tento design zalíbil.



Obrázek 8 : Otázka č.4

Touto otázkou jsem chtěl ověřit a zjistit, jestli se mi povedlo vytvořit jednoduché, přehledné a intuitivní webové rozhraní pro zpracování obrazu.

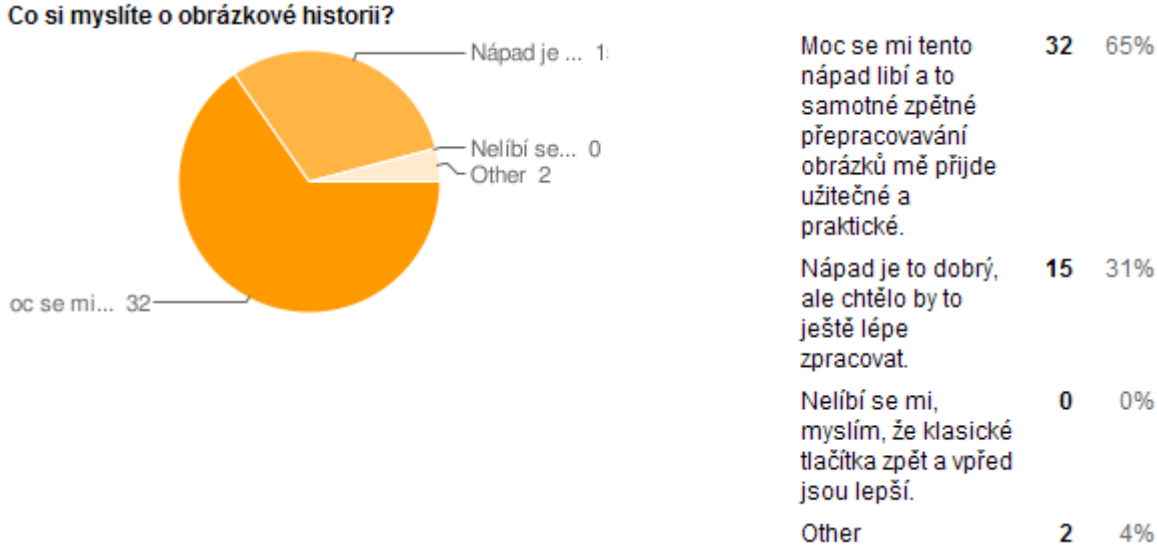
Výsledek: Zdá se, že se mě tohoto cíle povedlo dosáhnout, většina uživatelů se neměla problém v tomto rozhraní zorientovat.



Obrázek 9 : Otázka č.5

Otázka měla za úkol zjistit, jak se uživatelům líbí zvolené řešení jak ovlivňovaní hodnoty a parametry prováděných operací.

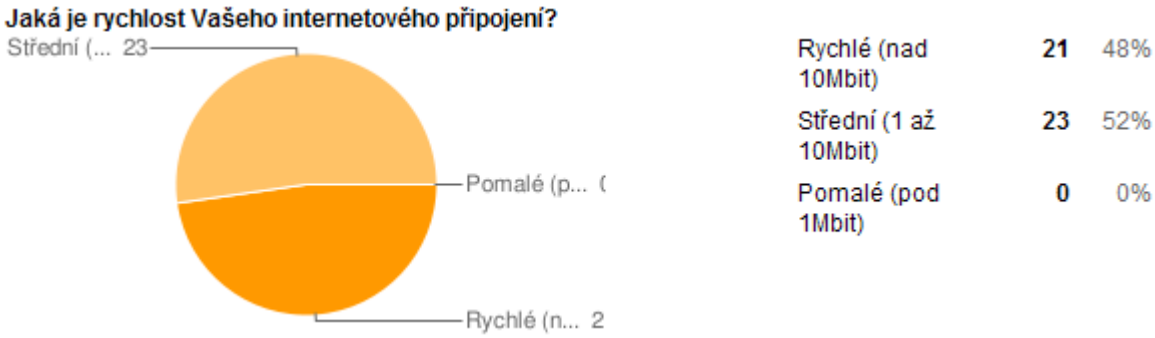
Výsledek: Většině uživatelů se toto řešení zamlouvá, ale spousta uživatelů by uvítala obojí řešení. Jak posuvníky, tak i textová pole, kde by hodnoty zadávali sami. Já jsem chtěl implementovat ovlivňování hodnot funkcím co nejpohodlněji pro uživatele. Z tohoto důvodu jsem implementoval posuvníky, aby uživatel mohl pracovat jen jednou rukou a nemusel tak přehmatávat z myši na klávesnici při vyplňování textových polí. Ale jak z výsledků dotazu vyplývá, tak asi implementace obou variant by vyřešila veškeré rozpaky.



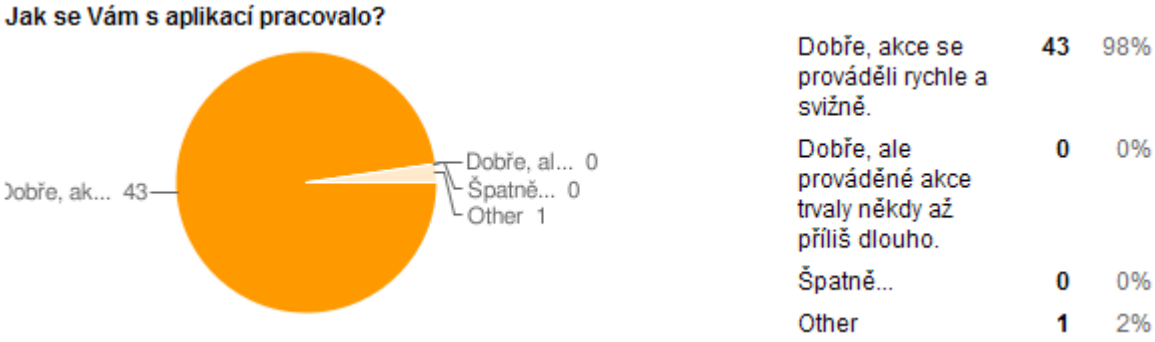
Obrázek 10 : Otázka č.6

Také mě velice zajímal názor na historii obrázků. Jestli se jim tento nápad líbí či nikoliv.

Výsledek: Velké většině se tento nápad a samotné zpracování historie obrázků líbí. Někteří si ale myslí, že by ji chtělo lépe zpracovat.



Obrázek 11 : Otázka č.7



Obrázek 12 : Otázka č.8

Tyto poslední dvě otázky spolu souvisí, zajímalo mě, jestli uživatelům aplikace pracovala rychle, svižně a bez problémů v závislosti na rychlosti připojení.

Výsledek: Uživatelům, kteří mou aplikaci testovali, se rychlost internetového připojení pohybovala od 1Mbit/s a výše. Na takovémto připojení aplikace pracovala bez věších problémů rychle a svižně.

V tomto formuláři jsem měl také možnost pro uživatele, napsat mě nějakou připomínku či nápad. O pár zajímavých nápadů se semnou uživatele podělili. Prvním, který mě přišel zajímavý, poukazoval na historii obrázků. Týkalo se to možnosti tyto obrázky během samotné práce také umět vymazat. Tohle je určitě velice užitečná a důležitá funkčnost, která by neměla v aplikaci chybět. Dále se pak objevila zajímavá připomínka k funkci ořezání, že zde chybí zobrazené rozměry prováděného výběru. Uživatel si tak přesně nedokáže oříznout fotografii v určitém rozměru, ve kterém ji potřebuje. Také by se mohlo přidat nastavení zachování určitého poměrů stran při ořezávání samotného obrázků. Toto byli asi ty nejzajímavější nápady a připomínky od samotných uživatelů.

Později mě také napadlo zeptat se jich, jaký mají typ monitoru, jestli širokoúhlý, nebo klasický. Ověřil bych si tak, zda byli mé úvahy správné, že většina uživatelů už vlastní širokoúhlé monitory. Otázka mě ale napadla později a tak výsledky v době psaní této práce nejsou zatím k dispozici.

Aplikaci jsem vyvíjel lokálně na svém počítači pomoci programu VertigoServ, pod operačním systémem Windows XP. Následně jsem pak tuto aplikaci testoval na školním serveru Eva.

7 Závěr

Cílem této práce byla tvorba webového rozhraní pro zpracování obrazu. S tím bylo spojeno nastudování různých metod a technik jak tohoto cíle dosáhnout. Důležitou částí práce bylo navrhnout klienta, server a hlavně jejich vzájemnou komunikaci. Součástí bylo také i vytvoření samotné demonstrační aplikace, jednoduchého online editoru obrázků. Před samotným návrhem a implementací jsem si vymezil hlavní cíle práce. Těmi byli jednoduchost, přehlednost, intuitivnost rozhraní, také efektivnost a rychlost prováděných operací. Na základě vyplněných dotazníků od uživatelů se zdá, že se mi tyto cíle povedlo do jisté míry splnit. Výsledkem tedy je ono webové rozhraní, ve kterém si může uživatel snadno a rychle editovat své fotografie a to za pomocí několika základních funkcí.

Díky této práce jsem si značně rozšířil své znalosti s tvorbou webových rozhraní. Detailněji jsem se seznámil s JavaScriptem, jQuery, AJAXem, OpenCV a dalšími technologiemi a jazyky. Už dříve jsem se chtěl o této problematice dozvědět více, ale nenašel jsem tu správnou motivaci. Tou bylo až zadání této práce, které mě na první pohled zaujalo, přišlo mi to jako velmi dobrý nápad umožnit uživatelům upravovat své fotografie přímo online na internetu bez nutnosti instalovat cokoliv na svůj počítač. Jedinou podmínkou tohoto řešení je mít v prohlížeči zapnutou podporu JavaScriptu.

Možných dalších rozšíření a vylepšení této práce je určitě mnoho. Počínaje přidání více funkcí, možností upravování více fotografií najednou, uchovávání obrázků na serveru i po ukončení práce k pozdějšímu editování, nebo také například nahrání celého archívu s fotografiemi. Při testování jsem se o toto zajímal a zeptal jsem se uživatelů, jaké by se jim líbila případná další rozšíření. Právě výše zmíněná vylepšení byly nejčastěji jmenovanými, které by si uživatelé přáli.

Citovaná literatura

1. Adobe Flash. *Wikipedie: Otevřená encyklopedie.* [Online] 3. Květen 2009. http://cs.wikipedia.org/wiki/Adobe_Flash.

2. PHP. *Wikipedie: Otevřená encyklopedie.* [Online] 15. Duben 2009. http://cs.wikipedia.org/wiki/Php.

3. JavaScript. *Wikipedie: Otevřená encyklopedie.* [Online] 3. Duben 2009. http://cs.wikipedia.org/wiki/Javascript.

4. HyperText Markup Language. *Wikipedie: Otevřená encyklopedie.* [Online] 27. Duben 2009. http://cs.wikipedia.org/wiki/HyperText_Markup_Language.

5. Cascading Style Sheets. *Wikipedie: Otevřená encyklopedie.* [Online] 2. Duben 2009. http://cs.wikipedia.org/wiki/Cascading_Style_Sheets.

6. Asynchronous JavaScript and XML. *Wikipedie: Otevřená encyklopedie.* [Online] 16. Duben 2009. http://cs.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML.

7. JavaScript Object Notation. *Wikipedie: Otevřená encyklopedie.* [Online] 1. Květen 2009. http://cs.wikipedia.org/wiki/JavaScript_Object_Notation.

8. Document Object Model. *Wikipedie: Otevřená encyklopedie.* [Online] 3. Duben 2009. http://cs.wikipedia.org/wiki/Document_Object_Model.

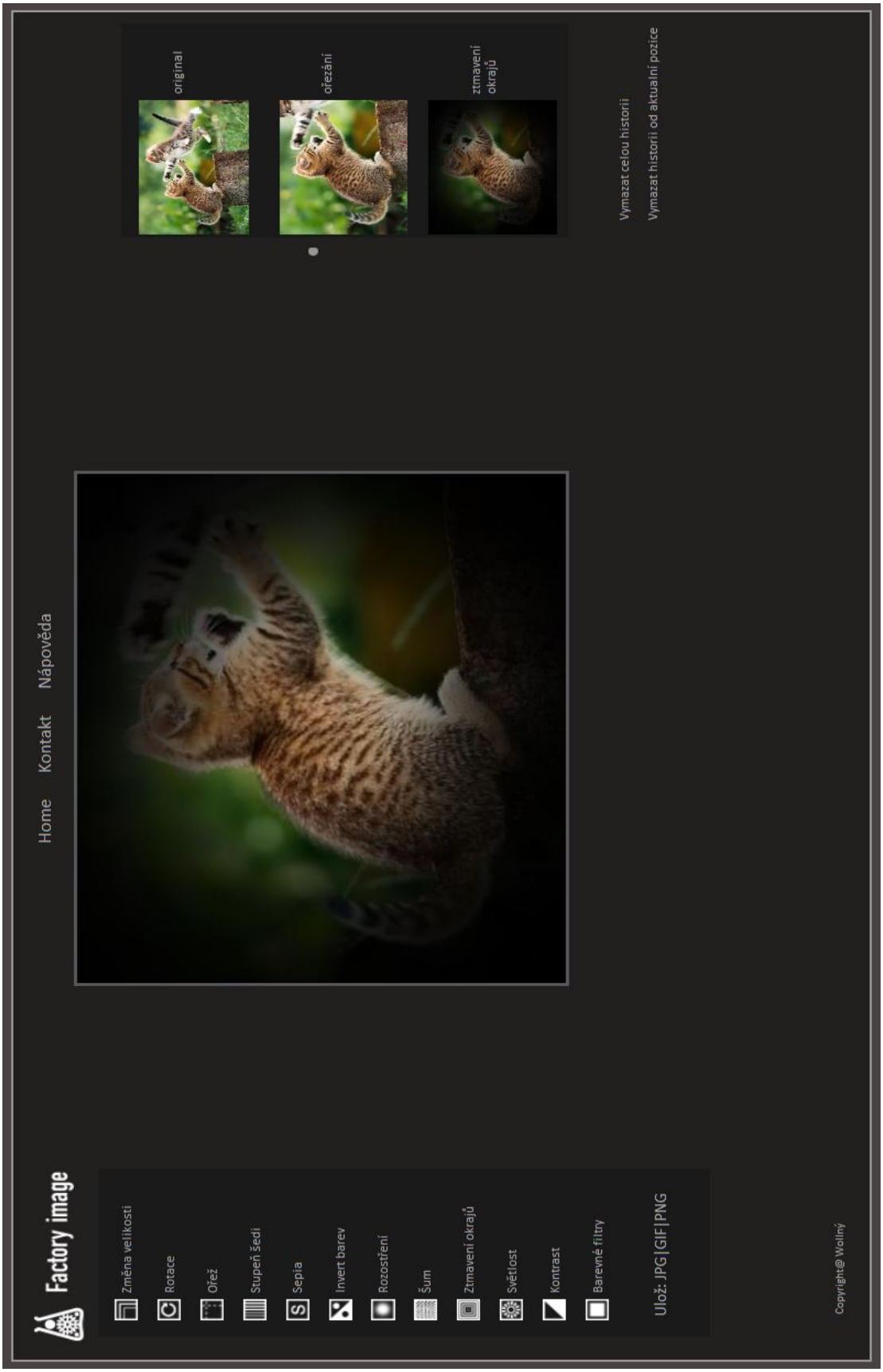
9. **Burel, David.** Úvod do práce s Knihovnou GD v PHP. [Online] 12. Leden 2008. http://programujte.com/index.php?akce=clanek&cl=2008010402-uvod-do-prace-s-knihovnou-gd-v-php.

Seznam příloh

Příloha 1. Ukázka vzhledu aplikace

Příloha 2. CD se zdrojovými kódy aplikace a s elektronickou verzí této práce

Příloha 1


Factory image
Home
Kontakt
Nápověda
Změna velikosti
Rotace
Ořež
Stupeň šedi
Sepia
Invert barev
Rozostření
Šum
Ztmavení okrajů
Světlost
Kontrast
Barevné filtry
Ulož: JPG | GIF | PNG
Copyright@ Wollný
original
ořezání
ztmavení okrajů
Vymazat celou historii
Vymazat historii od aktualní pozice