VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÉ ROZHRANÍ PRO ZPRACOVÁNÍ OBRAZU

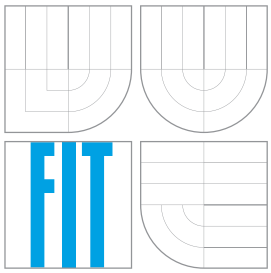
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MILAN BERAN

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÉ ROZHRANÍ PRO ZPRACOVÁNÍ OBRAZU

WEB INTERFACE FOR IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MILAN BERAN
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Beran Milan, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Webové rozhraní pro zpracování obrazu**
Web Interface for Image Processing
Kategorie: Web

Pokyny:

1. Prostudujte literaturu týkající se tvorby webových aplikací s využitím AJAX technologie. Seznamte se nástrojem OpenCV pro zpracování obrazu.
2. Na základě nastudovaných informací navrhněte systém s webovým rozhraním, který umožní uživateli definovat operace nad obrazem, měnit parametry operací, provede zadané operace a průběžně bude zobrazovat výsledek.
3. Implementujte navržený systém.
4. Proveďte experimenty zkoumající výkonnost a uživatelskou přístupnost navrženého systému. Experimenty pečlivě zdokumentujte a okomentujte.
5. Diskutujte omezení a nedostatky navrženého řešení a navrhněte případná vylepšení. Vytvořte plakát reprezentující Vaše řešení.

Literatura:

- Hlaváč, V.: Zpracování signálů a obrazů, Praha, ČVUT, 2005.
- Rafael C. Gonzalez, Richard E. Woods: Digital image processing, Prentice-Hall, 2002.
- Galbiati, L. J.: Machine vision and digital image processing fundamental, Prentice-Hall, 1990.
- Dále dle pokynu vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1, 2 a částečně 3.

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

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

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

Vedoucí: **Beran Vítězslav, Ing.**, UPGM FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práce se zabývá návrhem a implementací systému, který slouží k jednoduššímu ovládání konzolových aplikací pro zpracování obrazu. Práce je postavena na třech oblastech informačních technologií: distribuovaných systémech, zpracování obrazu a webových technologiích. Systém se skládá z několika samostatných prvků, které spolu komunikují při zpracování zadaných úkolů. Řídící rozhraní a daemon, přijímající požadavky s úkoly, jsou implementovány v jazyce PHP. Programy pro zpracování obrazu jsou naprogramovány v jazyce C s použitím knihovny OpenCV. Ovládání systému je řešeno za pomoci webového rozhraní, které využívá dynamické ovládací prvky, implementované pomocí JavaScriptu, knihovny jQuery a rozhraní jQueryUI. Součástí práce je také popis nasazení systému v rámci dvou prostředí, experimenty zkoumající výkonnost systému a testování webového rozhraní z hlediska uživatelské přístupnosti.

Abstract

This paper concerns design and implementation of a system which provides easier control of digital image processing console applications. The work is based on threes information technology domains: distributed systems, image processing and web technologies. The system consist of number of separated components communicating with each other in order of processing desired tasks. Control interface and the task daemon are implemented in PHP language. Image processing programs are implemented in C language using OpenCV graphic library. Control of the system is carried out through web graphical interface using dynamic control components implemented in Javascript language, jQuery library and jQueryUI interface. Part of the work is also a description of employment of the system in practical use in two environments, experiments concerning system performance and web interface user acceptance testing.

Klíčová slova

Ajax, Cloud Computing, CSS, detekce hran, distribuované systémy, Houghova transformace, JavaScript, jQuery, MySQL, OpenCV, PHP, Rich Internet Application, Web 2.0, webové služby, XHTML, XML, zpracování obrazu

Keywords

Ajax, Cloud Computing, CSS, Distributed systems, Edge detection, Hough transform, Image processing, JavaScript, jQuery, MySQL, OpenCV, PHP, Rich Internet Application, Web 2.0, Web services, XHTML, XML

Citace

Milan Beran: Webové rozhraní pro zpracování obrazu, diplomová práce, Brno, FIT VUT v Brně, 2010

Webové rozhraní pro zpracování obrazu

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Vítězslava Berana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Milan Beran
24. května 2010

Poděkování

Rád bych poděkoval vedoucímu práce za cenné rady a připomínky při psaní této práce. Dále bych rád poděkoval lidem, kteří mi pomohli s provedením testů uživatelského rozhraní a poskytli mi svůj názor na moji práci.

Obsah

Kapitola 1

Úvod

V dnešní době existuje mnoho různých nástrojů a aplikací, které se stále ovládají pouze za pomoci příkazové řádky. Nástroje pro zpracování obrazu nejsou v této oblasti výjimkou. Vetšina těchto aplikací má navíc nejednotné příkazy a spoustu různých parametrů a nastavení. Pokud uživatel pracuje s více takovými nástroji, může pro něj být složité všechny efektivně ovládat.

Cílem této diplomové práce je navrhnout a implementovat systém s webovým rozhraním, který umožní uživateli definovat operace nad obrazem, měnit parametry operací, provede zadané operace a bude průběžně zobrazovat výsledky uživateli. Před započetím řešení je potřeba určit jakou cestou se vydat. Autor práce se tedy nejvíce zaměřil na návrh a implementaci komunikace mezi jednotlivými prvky systému tak, aby bylo zpracování zadaných úkolů co nejefektivnější. Výsledný návrh systému by měl být univerzální, aby umožnoval použití libovolného konzolového programu. V implementaci se však již systém více profiluje a je zaměřen na programy pro zpracování obrazu. Jako ukázkové příklady byly použity zejména programy pro detekci hran v obraze. Velká pozornost byla také věnována webovému rozhraní, u kterého byly využity dynamické ovládací prvky a komunikace se serverem pomocí technologie Ajax.

Po první uvodní kapitole následuje kapitola, která je teoretická a věnuje se třem oblastem informačních technologií, na kterých je práce založena. Tyto oblasti jsou distribuované systémy, zpracování obrazu a webové technologie. U každé z oblastí jsou uvedeny základní pojmy a termíny, které se dané problematiky týkají.

Hlavní část práce je rozdělena do tří kapitol. První část (kapitola 3) se věnuje obecnému návrhu systému, kdy jsou zkoumány různé možnosti rozdělení systému na jednotlivé komponenty a vzájemné zapojení těchto komponent. Jsou zde popsány tři hlavní etapy vývoje tak, jak vznikaly návrhy postupně za sebou, od začátku až po finální návrh. Poslední verze návrhu je poté v další části (kapitola 4) detailněji popsána. Kapitola mimo jiné definuje důležité požadavky a omezení na výsledný systém. Poslední část (kapitola 5) popisuje samotnou implementaci systému, nasazení systému v rámci dvou prostředí a experimenty s implementovaným systémem.

Předposlední kapitola (kapitola 6) obsahuje testování systému zejména z hlediska uživatelské přístupnosti spolu se zhodnocením tohoto testování. Poslední kapitola je Závěr, obsahující shrnutí výsledků diplomové práce. Poslední kapitola také obsahuje možnosti pokračování práce do budoucna a návrhy na možná vylepšení.

Kapitola 2

Teoretická část

Systém pro zpracování obrazu popisovaný v této diplomové práci je založen na třech oblastech z oboru informačních technologií. Tato kapitola se zabývá teorií a pojmy spojenými s každou z těchto tří oblastí a snaží se objasnit jejich úlohu v této práci.

2.1 Distribuované systémy

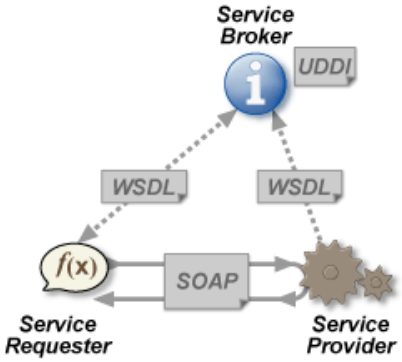
Jednou z oblastí, které se tato diplomová práce bude týkat, je oblast distribuovaných systémů. Distribuovaný systém je takový systém propojení množiny nezávislých uzlů, který poskytuje uživateli dojem jednotného systému. V práci se zaměříme hlavně na distribuované webové aplikace. V této podkapitole je proto uvedeno vysvětlení dvou pojmů, které můžeme v souvisloti s distribuovanou webovou aplikací slyšet.

Webové služby

Služba WWW (World Wide Web) nabízí možnost prezentovat informace, databáze, výsledky výpočtů ovlivněných uživatelem nebo čímkoliv jiným. Nutnou součástí webu je člověk, který vykonává interaktivitu vedoucí k zobrazení požadovaných dat (hledá v dokumentech, formuluje dotazy atd.). Webové služby mohou být tedy chápány jako obdoba webu pro stroje. Bez přítomnosti člověka lze tedy zajistit spolupráci dvou programů, které si navzájem vyměňují data, zpracovávají a dále je využívají. Existuje mnoho druhů webových služeb od těch nejjednodušších až po komplexní. Příkladem webové služby může být zobrazení náhodného čísla či zobrazení statistik získaných od mnoha milionů zkoumaných objektů.

Na internetu lze nalézt mnoho definic pojmu webová služba. Jednu z nich nabízí například mezinárodní organizace W3C (World Wide Web Consorcium): "*Webová služba je softwarový systém navržený pro podporu interakce mezi stroji na síti. Je popsána ve formátu zpracovatelném strojem (typicky WSDL). Ostatní systémy komunikují se službou způsobem předepsaným jejím popisem pomocí SOAP zpráv, typicky přenášených použitím HTTP protokolu s XML zápisem ve spolupráci s ostatními webovými standardy.*" [10] Architekturu takovéto webové služby lze vidět na obrázku 2.1. Ostatní definice se mohou mírně lišit, ale všechny se shodují ve vyjádření podstaty webových služeb. Jde o dosažení kooperace mezi dvěma a více systémy pracujícími na různých platformách za použití standardizovaných komunikačních rozhraní, jednotných protokolů a sjednocených formátů přenášených dat.

Webové služby jsou jednou z implementací architektury orientované na služby [13].



Obrázek 2.1: Architektura webové služby - zasílání zpráv pomocí SOAP (převzato z [14])

Webová služba je služba typu klient-server, kdy uživatel posílá na server požadavky, server požadavek zpracuje a pošle klientovi odpověď. Kromě SOAP zpráv se dají využívat i jiné protokoly pro komunikaci - například XML-RPC a REST.

V rámci této diplomové práce bude implementováno předávání zpráv mezi prvky systému založené právě na myšlence webových služeb. K zasílání zpráv však bude využíváno zasílání POST požadavků protokolu HTTP.

Cloud computing

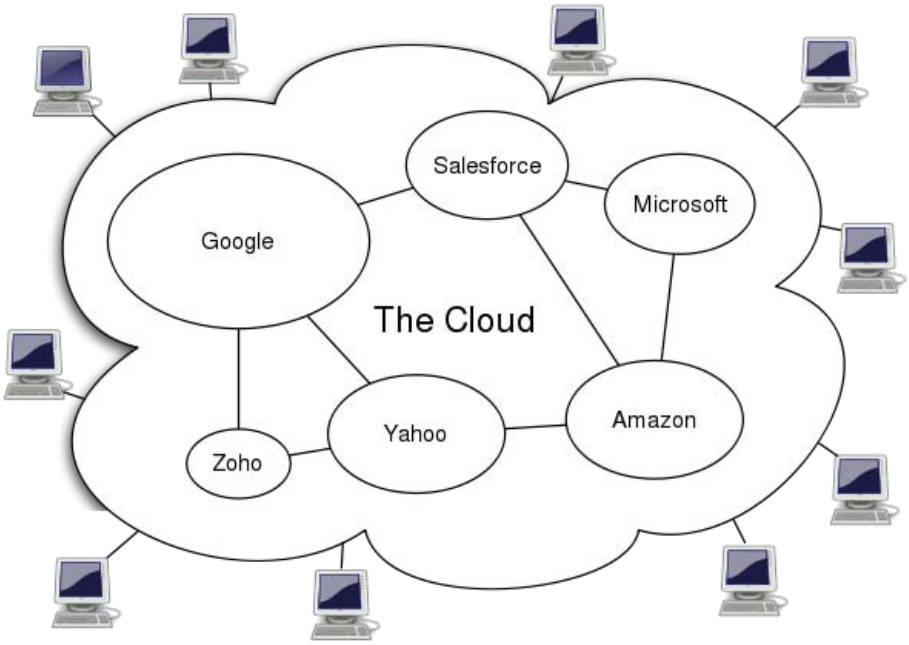
Pojem Cloud computing se v poslední době na internetu objevuje stále častěji, ale najít jeho přesnou definici je velmi těžké. A co si vlastně lze pod termínem Cloud computing představit? Jedna z kratších definic říka: "*Cloud computing je sdílení hardwarových i softwarových prostředků pomocí sítě.*" [20]. Pod touto definicí si lze však představit jakékoliv sdílení v rámci informačních technologií.

Wikipedie definuje Cloud computing následovně: "*Na internetu založený model vývoje a používání počítačových technologií. Lze ho také charakterizovat jako poskytování služeb či programů uložených na serverech internetu s tím, že uživatelé k nim mohou přistupovat pomocí webového prohlížeče a používat prakticky odkudkoliv. Uživatelé neplatí (za předpokladu, že je služba placená) za vlastní software, ale za jeho užití. Nabídka aplikací se pohybuje od kancelářských aplikací, přes systémy pro distribuované výpočty, až po operační systémy provozované v prohlížečích.*" [15] Tato definice nám již dává trochu lepší vysvětlení, ale stále se její znění ještě formuje do konečné podoby.

Poslední definice, která zde bude zmíněna je ze stránek Abakowiki [19] a zní: "*Termín označuje souhrnně technologie a postupy používané v datových centrech a firmách pro zajištění snadné škálovatelnosti aplikací dodávaných přes Internet. Cloud Computing je zastřešující pojem pro pojmy a přístupy jako SaaS, PaaS, IaaS, MaaS, CaaS.*"

- SaaS - software jako služba (*Software as a Service*) - Uživatel si nekupuje software, ale pronajímá si ho. SaaS je typicky webová aplikace. Pokud je schopen kupující zobrazit stránky s SaaS produktem , je schopen ho používat. Není třeba žádné dodatečné instalace. Stačí webový prohlížeč. SaaS aplikací je celá řada, počínaje emailem až po řešení pro podniky jako intranety či CRM systémy. Příkladem může být sada nástrojů Google Apps.

- PaaS - platforma jako služba (*Platform as a Service*) - Uživateli je poskytována platforma pro provoz jeho aplikací. Programátor s nápadem na aplikaci využije k její implementaci a provozu PaaS, kde si platí výpočetní výkon vzdálené platformy.
- IaaS - infrastruktura jako služba (*Infrastructure as a Service*) - Pod tímto pojmem se skrývá pronájem infrastruktury, tedy počítačů. Typicky to probíhá tak, že dojde k virtuálnímu pronájmu počítače. Na tento počítač si uživatel může nahrát operační systém dle svého výběru a platí se za hodiny, kdy je tento počítač pronajatý.
- CaaS - komunikace jako služba (*Communication as a Service*) - Pod tímto pojmem si můžeme představit služby poskytující možnost komunikace, jako například VoIP nebo videokonference.
- MaaS - monitorování jako služba (*Monitoring as a Service*) - Monitorovací služby jsou důležité pro všechny předchozí služby k jejich sledování a vyhodnocování.



Obrázek 2.2: Znázornění Cloud computingu (převzato z [15])

Cloud computing má také řadu možností poskytování. To jak bude poskytován nám říká model nasazení:

- Veřejný cloud computing (*Public cloud computing*) - Někdy také označován za klasický model cloud computingu. Široké veřejnosti je poskytována výpočetní služba.
- Soukromý cloud computing (*Private cloud computing*) - Služby jsou v tomto případě poskytovány pouze pro organizaci, a to buď samotnou organizací nebo třetí stranou.
- Hybridní cloud computing (*Hybrid cloud computing*) - Hybridní cloud computing kombinuje jak veřejné, tak soukromé cloudy. Navenek vystupují jako jeden celek, ale jsou propojeny pomocí standardizačních technologií.

- Komunitní cloud computing (*Comunity cloud computing*) - Jedná se o model, kdy je infrastruktura sdílena mezi několika organizacemi, skupinou lidí, kteří ji využívají. Tyto skupiny může spojovat bezpečnostní politika, stejný obor zájmu nebo třeba pracovní zaměření.

Abychom mohli dojít k nějakému závěru, tak by se dalo říci, že Cloud computing je, více než co jiného, jakési markentigové označení pro poskytování služeb po síti. Tyto služby jsou poskytovány uživateli (skupině uživatelů), kterému je jedno, jakým způsobem služby pracují a kde jsou uložena jeho data. Uživatele zajímá pouze funkčnost poskytovaných služeb. Vše ostatní je pro něj skryto "za mraky".

2.2 Zpracování obrazu

Zpracování obrazu se zabývá použitím různých algoritmů pro zpracování obrazových dat. Jednotlivé algoritmy mohou sloužit jak k provádění různých úprav obrazových dat, tak také k získávání nejrůznějších informací.

Obrazová data

Pod pojmem obrazových dat budeme v této práci chápat data, která obsahují číselné hodnoty získané digitalizací obrazu. Tento obraz je nejdříve potřeba nějakým způsobem převést do digitální podoby (pomocí skeneru, digitální fotoaparátu atd.). Digitalizace obrazu je poté převod analogového signálu do diskrétního tvaru. Získaný digitální obraz je následně uložen do počítače.

Vstupní signál je popsán funkcí $f(x, y)$ dvou proměných - souřadnic bodů v obraze. Funkční hodnota odpovídá například jasu nebo hodnotám spektrálních složek signálu při barevném snímání. Vstupní signál je kvantován a vzorkován. Výsledkem je matice čísel popisující digitální obraz. Jeden prvek matice představuje jeden obrazový element, nazývaný pixel.

Knihovna OpenCV

Knihovna OpenCV [18] (Open Computer Vision library) je volně dostupná grafická knihovna původně vyvíjená společností Intel. Má bohatou zásobu funkcí zaměřených převážně na zpracování obrazu, ale i tvorbu uživatelského rozhraní, práci s grafickými formáty a mnoho dalších. Knihovna je určena především pro programovací jazyky C a C++.

Operace nad obrazem

V této části bude následovat vysvětlení metod pracujících s obrazovými daty, které budou následně použity pro ukázkové programy v rámci diplomové práce.

Detekce hran

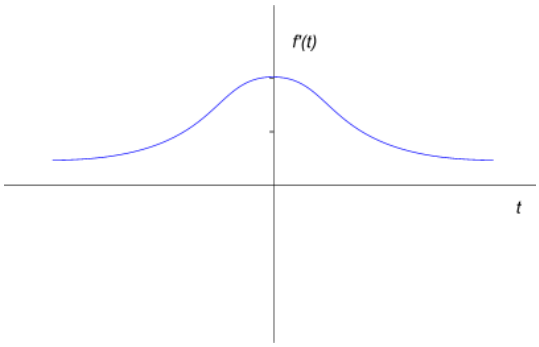
Detekce hran je postup v digitálním zpracování obrazu, sloužící k nalezení oblastí bodů, ve kterých se podstatně mění jas. Hrany jsou místa v obraze, které lidské oko vnímá jako hranici objektů nebo rozhraní světla a stínu. Model ideální hrany může být skoková funkce (*step*). V reálných obrazech je změna jasu postupná, nikoli skoková, proto je vhodnější

použít šikmou funkci (*ramp*). Pokud se obě definované funkce objeví v obraze vedle sebe, vznikají ještě další dva typy hran: čára (*line*) a střecha (*roof*) - viz obrázek 2.3. K provedení hranové detekce se používají hranové detektory (*edge detector*).

(a) step (b) ramp (c) line (d) roof

Obrázek 2.3: Typy hran v obraze (převzato z [6])

Existuje celá řada hranových detektorů, které se mohou lišit jak v metodách vedoucích k nalezení hran, v reprezentaci hran, tak také v citlivosti rozpoznání . S citivlostí rozpoznání také souvisí odolnost proti šumu a jiným nežádoucím efektům zpracovávaného obrazu. Základní metody detekce hran v obraze se dají rozdělit do dvou hlavních skupin. Jedná se o metody využívající první derivaci nebo druhou derivaci jasové funkce. U první skupiny je výsledný hranový gradient porovnán s prahem, určujícím, zda se jedná o hrany či nikoli. U metod využívajících druhou derivaci je výskyt hrany detekován, je-li prostorová změna v polaritě druhé derivace dostatečně významná.



Obrázek 2.4: První derivace funkce intenzity (převzato z [4])

Detekce hran pomocí první derivace Jestliže je hrana definována jako relativně velká změna hodnoty jasové funkce, pak bude v místě hrany velká hodnota derivace této funkce (obrázek 2.4). Maximální hodnota derivace (gradientu) bude ve směru kolmo na hranu. Kvůli zjednodušení výpočtu se ale hrany detekují pouze ve dvou, resp. ve čtyřech směrech. K výpočtu gradientu můžeme přistupovat jako ke konvolučnímu filtrování obrazu. Jednotlivé hranové detektory se pak liší jádrem konvolučního filtru. Jádro filtru, reprezentované maticí určené ke konvoluci, udává, které body se použijí pro výpočet gradientu a jaké váhy tyto body budou mít. Vlastnosti filtru, velikost a hodnoty významně ovlivňují výsledky detekce. Obecně platí, že čím je matice větší, tím je detektor odolnější vůči šumu, protože okolí použité k aproximaci je větší. Tzv. gradientní obraz získáme konvolucí mezi originálním obrazem a jádrem filtru. Konvoluce se provádí po řádcích, kdy se používá jedna matice a po sloupcích, kdy se používá transponovaná matice. Jádra některých známých filtrů mohou vypadat takto:

Operátor	G_x	G_y
Robertsův	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Prewittové	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobelův	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Robinsonův	$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
Kirschův	$\begin{bmatrix} 3 & 3 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & -5 \end{bmatrix}$	$\begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}$
Frei-Chenův	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Z výsledků detekce hran v horizontální a vertikálním směru - G_x a G_y lze spočítat sílu hrany:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

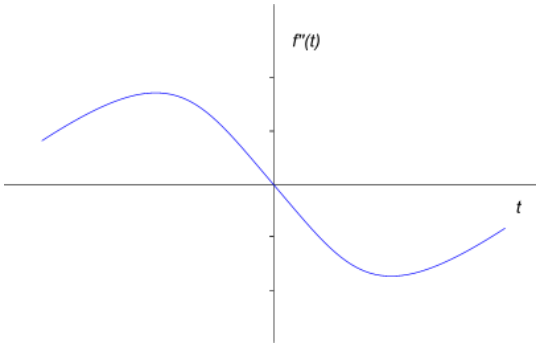
nebo lze hodnotu kvůli výpočetní složitosti následně aproximovat:

$$|G| = |G_x| + |G + y| \tag{2.2}$$

Orientaci gradientu vzhledem k vodorovné ose lze vypočítat jako:

$$\theta = arctan\frac{G_x}{G_y} \tag{2.3}$$

Detekce hran pomocí druhé derivace Metody využívající druhou derivaci jasové funkce se snaží nalézat průchody této derivace nulou (obrázek 2.5). Využívají toho, že je jednodušší nalézt průchod nulou, než extrém funkce. Bohužel metody druhé derivace jsou ještě více citlivé na šum, než metody první derivace. Je proto výhodné výpočet kombinovat s takovým vyhlazením, které odstraní maximální množství šumu, ale nepoškodí hrany. Tyto metody proto většinou pracují s Laplaceovým operátorem - Laplacianem Δf, který aproximuje druhou všesměrovou derivaci jasové funkce. K výpočtu se používá pouze jedna matice. Jeho nevýhodou je, že má dvojité odezvy na hrany odpovídající tenkým liniím v obraze.



Obrázek 2.5: Druhá derivace funkce intenzity (převzato z [4])

Příklady normalizovaného Laplacianu:

$$\frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{8}\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

Marrův filtr (Marr-Hildrethův) se snaží odstranit šum použitím Gaussovské funkce (rozmazání) a následnou aplikací Laplaciánu (LoG = Laplacian of Gaussian).

Cannyho hranový detektor

Cannyho algoritmus se považuje za optimální hranový detektor. Je navržen tak, aby splňoval tři základní požadavky, které dříve zmíněné detektory zajistit nedokázaly. Tyto požadavky jsou:

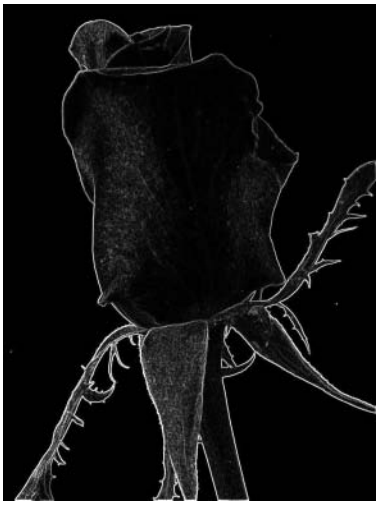
- Minimální počet chyb - Všechny významné hrany musí být detekovány a nesmí být žádná odevza v místě, kde hrana není.
- Přesnost lokalizace - Rozdíl mezi skutečnou a nalezenou polohou hrany musí být minimální.
- Jednoznačnost - Každá hrana může být detekována pouze jednou. Tento požadavek je zaměřen zejména na zašuměné a nehladké hrany.

Postup detekce se skládá z několika kroků. Prvním krokem je eliminace šumu Gaussovým filtrem. Doporučuje se Gaussův filtr realizovat pomocí konvoluce užívající masku. Poté následuje aplikace Sobelova operátoru pro nalezení velikostí gradientů a jejich směrů. Po nalezení hran následuje proces ztenčení (*thinning*), který spočívá ve výběru lokálních maxim nalezených gradientů. Princip této fáze spočívá v tom, že jako bod hrany je označen jen takový bod, jehož sousední body v okolí kolmém na směr gradientu (směr známe z předchozí fáze) mají hodnotu gradientu nižší. Posledním krokem je prahování s hysterezí (*tresholding*). Účel tohoto kroku spočívá v ohodnocení významu nalezených hran (splnění kritéria jednoznačnosti). V této chvíli jsou všechny hrany označeny, včetně těch nejmenších, které mohou pocházet například ze šumu. Proto jsou tedy zvoleny minimální (T_1) a maximální (T_2) hodnoty prahů. Hodnoty nalezených gradientů jsou potom porovnávány s těmito prahy. Pokud hodnota gradientu daného pixelu je vetší než prah T_2, pak je přímo označen jako hranový. Pokud je jeho hodnota menší než T_1, pak je označen jako nehranový. A pokud

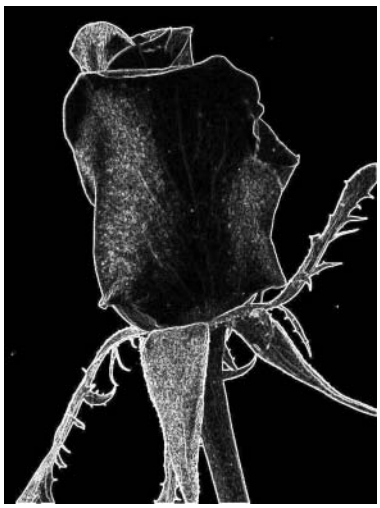
(a) Původní obrázek

(b) Robertsův

(c) Prewittové



(d) Sobelův



(e) Kirschův

(f) Cannyho

Obrázek 2.6: Aplikace vybraných hranových detektorů

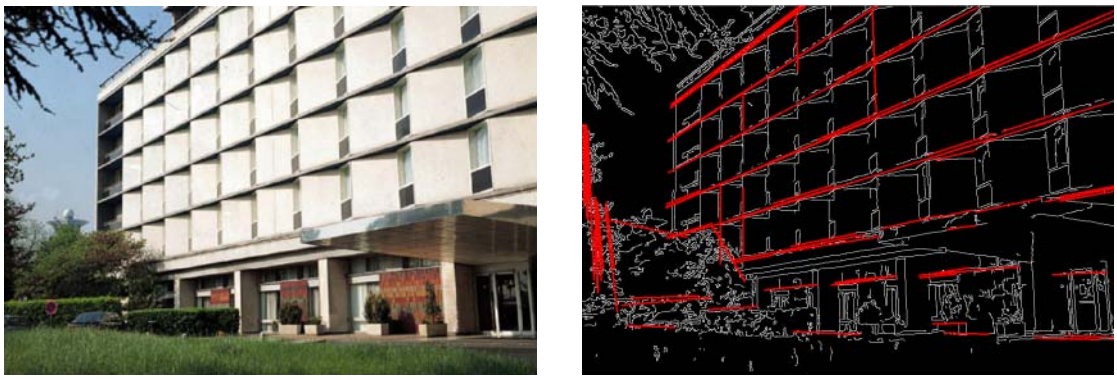
hodnota gradientu daného bodu leží mezi těmito dvěma prahy, pak je označen jako hrana pouze v případě, že sousedí s bodem, který už jako hranový byl uznán.

Houghova transformace

Houghova transformace je metoda pro nalezení parametrického popisu objektů v obraze. Používá se především pro segmentaci objektů, jejichž hranice lze popsat jednoduchými křivkami. Hlavní výhodou této metody je robustnost vůči nepravidelnostem a přerušení hledané křivky. Houghova transformace je využívána k nalezení parametrů matematického modelu hledaného objektu za předpokladu známých vstupních dat (pixelů). Přímku v rovině lze popsat několika způsoby. Jednou z možností je vzorec:

$$xcos\theta + ysin\theta = r, \tag{2.4}$$

kde r je délka normály od přímky k počátku souřadnic, $theta$ je úhel mezi normálou a osou x. Jestliže do předchozí rovnice dosadíme souřadnice nějakého bodu (x_i, y_i), pak množina všech dostupných řešení (r, θ) vytvoří v Houghově prostoru spojitou křivku. Jestliže si tímto způsobem promítneme do Houghova prostoru všechny body ležící na nějaké přímce p, pak uvidíme, že křivky odpovídající jednotlivým bodům x_i, y_i se protnou v jediném bodě (r_{max}, θ_{max}). Tato dvojice jsou ve skutečnosti hledané parametry přímky p [11]. Příklad aplikace Houghovy transformace můžete vidět na obrázku 2.7.



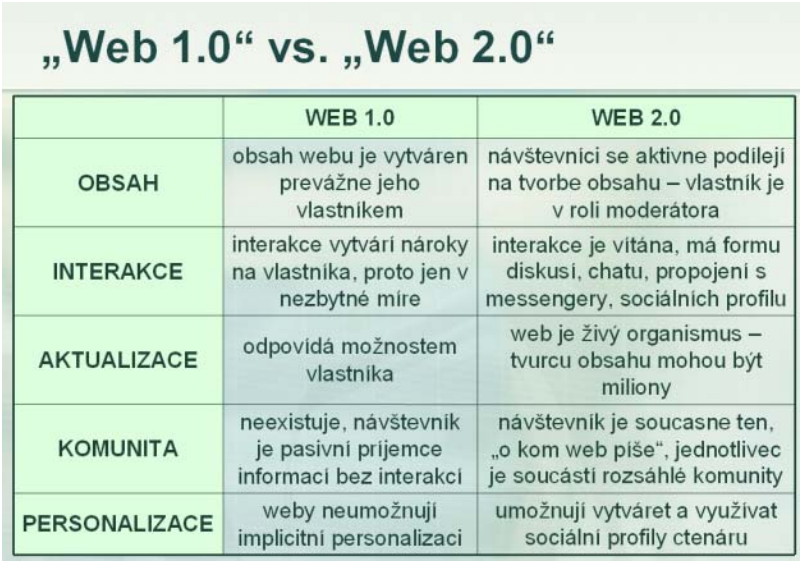
Obrázek 2.7: Příklad použití Cannyho hranového detektoru a Houghovy transformace (převzato z [11])

2.3 Webové technologie

Součástí této diplomové práce je také webové rozhraní využívající moderní techniky tvorby webových aplikací. V této podkapitole si tedy přiblížíme tyto moderní techniky, které mají sloužit především k lepší přístupnosti. Lepší přístupnosti se snaží dosáhnout zejména využitím nejrůznějších dynamických ovládacích prvků usnadňujících práci uživatele s webovou aplikací. Mezi nástroje, které tyto prvky dokáží implementovat patří zejména skriptovací jazyk na straně klienta a technologie Ajax.

Web 2.0

Pojem Web 2.0 je označením pro etapu vývoje webu, v níž byl pevný obsah webových stránek nahrazen prostorem pro sdílení a společnou tvorbu obsahu. Pojem Web 2.0 se poprvé objevil v roce 1999, kdy ho použila Darcy DiNucci ve svém článku Fragmented Future [2]: "*Web, jak ho známe teď, který se jako statický text načte do okna prohlížeče, je jen zárodek webu, který přijde. První záblesky Webu 2.0 se již začínají objevovat a my sledujeme, jak se toto embryo začíná vyvíjet. Web bude chápán ne jako obrazovky plné textu a grafiky, ale jako prostředí, jako éter, jehož prostřednictvím dochází k interaktivitě. Objeví se na obrazovce počítače, na televizním přijímači, na palubní desce, na mobilním telefonu, na herní konzoli, a možná, že i na vaší mikrovlnné troubě.*" Poždeji se termín objevuje v roce 2004, kdy Tim O'Relly a zástupci Medialive international jednali o názvu konference a pojem Web 2.0 jim připadal jako vhodná metafora pro "druhý dech" internetového podnikání a vývoje.



Obrázek 2.8: Porovnání Web 1.0 a Web 2.0 (převzato z [3])

Na internetu je velmi těžké najít přesnou definici tohoto pojmu. Je to způsobeno tím, že každý si tento pojem vykládá jinak a není žádná oficiální definice, které by se dalo držet. Web 2.0 je tedy spíše označení pro změnu chápání internetu a jeho účelu za pomoci využití nových trendů a technologií. Rozdíl oproti původnímu konceptu webu lze vidět na obrázku 2.8. V dnešní době jen těžko budeme hledat stránky, které by nedodržovaly principy Web 2.0. Mezi charakteristické rysy patří:

- interakce s uživatelem a jeho vtažení do tvorby obsahu
- otevřená komunikace, sdílení a znovuvyužívání informací
- lépe organizovaný a roztříděný obsah spolu s lepší strukturou odkazů
- změna webových stránek z uzavřených informačních portálů na stránky poskytující webové aplikace koncovému uživateli

Charakteristické aplikace a zástupci trendu Web 2.0 jsou:

- Wiki - Wikipedia, ...
- Sociální sítě - Facebook, Last.fm, ...
- Blogy - Blogger.com, Blog.cz, Bloguj.cz, ...
- Sdílení dat - YouTube, Stream.cz, Flickr, ...

Web 3.0

V posledních pár letech se také mluví o termínu Web 3.0, který by měl představovat novou generaci webu. Web 3.0 zatím nemá příliš dlouhou historii a v jeho vývoji jde spíše o hledání hranice konce Webu 2.0. V aplikacích spadajících pod novou verzi webu by měly být využívány zejména prvky sémantického webu, kdy budou počítače moci lépe zpracovávat informace díky novým formátům uložení. Web 3.0 je také často spojován s Cloud Computingem. Mezi jeho základní prvky patří [7]:

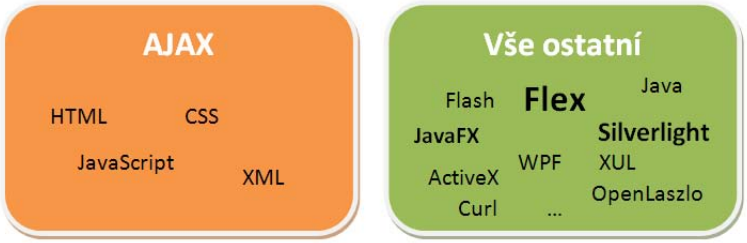
- Propojení multimediálního obsahu s obsahem textovým (vkládání hudby, videa, animací) nejlépe nezávisle na zařízení, z kterého je přistupováno (notebook, mobil, terminál, TV, mp3 přehrávač, fotoaparát, GPS, lednička).
- Geografické vyhledávání pomocí významu vět, tedy rozpoznání podmětu, předmětu, jejich vzájemného vztahu se zřetelem na případné určení času a místa.
- Personalizace (bude potřeba znát informace o uživateli), tedy např. filtrování obsahu podle osobních zájmů konkrétního uživatele.
- Znalost polohy, tedy např. filtrování obsahu podle aktuální polohy nebo podle místa rozpoznaného v dotazu.

Rich Internet Application

Dalším pojmem z oblasti webových technologí je zkratka RIA (*Rich Internet Application*)[1]. Jedná se o označení nového směru vývoje, kterým by se měla vydat další generace aplikací běžících na standardech internetu.

Bylo nutné najít nové cesty, jak obejít omezení stávajících internetových technologií. Tyto technologie jsou limitovány zejména bezstavovým protokolem HTTP a jeho modelem žádost/odpověď. Každá změna stavu u klienta (odeslání formuláře, přidání zboží do košíku atd.) vyvolává požadavek na znovunačtení všech dat na stránce. Stále se zvyšující nároky na uživatelskou přístupnost aplikací požadují nový prostor pro vývoj potřebných řešení a RIA toto řešení poskytuje hlavně v těchto oblastech:

- komplexnost grafického rozhraní a práce s více dokumenty současně (MDI koncept[12])
- uživatelsky přívětivější chování (odstínění od modelu žádost/odpověď)
- funkce odpovídající klasické desktopové aplikaci (drag&drop, klávesové zkratky, kontextová nápověda atd.)



Obrázek 2.9: Základní dělení RIA technologií (převzato z [1])

V posledních letech proto proběhly velké změny v možnostech tvorby webových aplikací a došlo k rozdělení přístupu RIA na dvě velké skupiny (obrázek 2.9). První skupinou jsou aplikace založené na Ajaxu a druhou jsou aplikace založené na pluginech.

Mezi největšího propagátora přístupu RIA a aplikací na něm založených je firma Google. Ta se zasloužila o silné rozšíření mezi programátory a uživatele. Mezi aplikace postavené na RIA se řadí například Gmail[1] a Google Maps[2] využívající Ajax. Naopak aplikace jako Photoshop Express[3] a SlideRocket[4] jsou založené na využití ostatních technologií.

RIA založené na Ajaxu

RIA založené na technologii Ajax využívají zavedené standardy internetu a přidávají k nim nové možnosti. Hlavní podporou pro tuto cestu je asynchronní vzdálené volání klientskými skriptovacími jazyky, nástup CSS2 (*Cascading Style Sheets*), vývoj nových knihoven pro jednodušší práci s JavaScriptem a mnoho dalších.

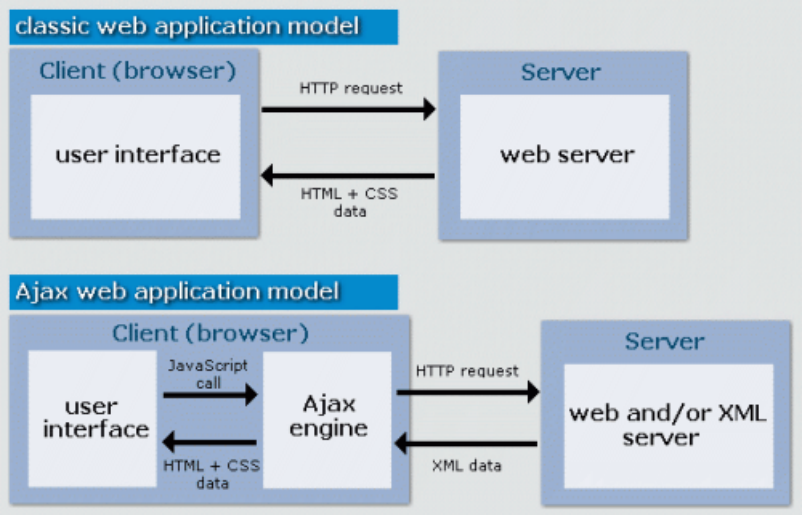
V této diplomové práci se bude pracovat právě s touto cestou tvorby webových aplikací.

Ajax Ajax[5] (*Asynchronous JavaScript and XML*) je technologie postavená na JavaScriptu, umožňující zasílat HTTP požadavky bez nutnosti načítání celé WWW stránky. Využití Ajaxu je především v moderních WWW stránkách pro vylepšení funkčnosti a zjednodušení ovládání.

Ajax se skládá především z prezentace tvořené standardním XHTML a CSS, dynamického zobrazování pomocí DOM, výměny dat se serverem pomocí XML (lze použít i jiný formát včetně XHTML) realizované asynchronně pomocí objektu XMLHttpRequest a JavaScriptu.

JavaScript JavaScript[9] je multiplatformní, objektově orientovaný skriptovací jazyk, který se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky. Často se vkládá přímo do HTML kódu stránky. Skript napsaný pomocí JavaScriptu se obvykle spouští na straně klienta (po stažení WWW stránky). Z toho plynou jistá bezpečnostní omezení. JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí

[1] http://www.gmail.com
[2] http://maps.google.com/
[3] https://www.photoshop.com/
[4] http://www.sliderocket.com/index_b.html



Obrázek 2.10: Porovnání klasického konceptu s Ajaxem (převzato z [8])

uživatele. Jeho nejčastější použití je pro ovládání nejrůznějších prvků stránky, jejich animaci a efekty.

jQuery jQuery[16] je jednoduchá, přehledná a rychlá knihovna JavaScriptových funkcí, která umožnuje vyhledávat, modifikovat a vytvářet elementy DOM. jQuery také umožnuje práci s Ajaxovými komponenty, možnosti animace prvků a jejich pohyby v rámci webového prohlížeče. Jedná se o otevřenou knihovnu, a tak lze přidávat funkce, které programátorovi schází. Narozdíl od programování pouze v JavaScriptu je kód s touto knihovnou stručnější, přehlednější a celkově jednodušší. Knihovna obsahuje oproti JavaScriptu funkce pro provádění animací, práci s událostmi a vyhledávání elementů v DOM struktuře HTML dokumentu. Jednotlivé elementy lze jednoduše vytvářet, editovat a mazat.

jQueryUI jQueryUI[17] je JavaScriptový framework využívající funkcí jQuery knihovny. Obsahuje v sobě widgety, které lze snadno přidávat do webového rozhraní a využívat jejich funkčnost. Framework pracuje pouze ve svém jmenném prostoru a nemodifikuje nic co mu nepatří[5]. Obrovskou výhodou je možnost přidávání nejrůznějších pluginů a rozšíření. jQueryUI také umožnuje použití předdefinovaných CSS stylů pro tvorbu vzhledu výsledné stránky. Framework je zaměřen hlavně na tvorbu grafického uživatelského rozhraní pro RIA.

RIA založené na pluginech

Druhou cestou, kterou se člověk tvořící webovou aplikaci s přívlastkem RIA může vydat, je použití pluginové technologie. Takto vytvořená aplikace potřebuje pro svůj chod přítomnost pluginu v uživatelském prohlížeči. Programátor má usnadněnou práci v tom, že se vyhýbá použití HTML a CSS, a tak nemusí řešit problémy s různým zobrazením aplikace v různých typech prohlížečů. Pokud má prohlížeč nainstalovaný potřebný plugin pro zobrazení, tak se aplikace zobrazí všude stejně. Mezi nejznámnější zástupce patří Adobe Flex, Silverlight a JavaFX.

[5]Narozdíl od agresivnějších konkurenčních frameworků jako MooTools (http://mootools.net/) nebo Prototype (http://www.prototypejs.org/).

Adobe Flex Adobe Flex[21] je sada technologií, vytvořená společností Adobe Systems pro vývoj multiplatformních RIA aplikací založených na Adobe Flash. V dnešní době je považován za jednu z nejlepších pluginových technologií, a to především díky možnosti běhu aplikace v prostředí Flash Playeru. Kromě velmi rozšířeného prostředí pro běh aplikací je Flex také přívětivý pro vývojáře, zejména díky několika faktorům:

- definice uživatelských rozhraní pomocí MXML se podobá HTML
- k vytvoření vzhledu aplikace lze využít podmnožinu CSS
- ActionScript 3 kombinuje rysy JavaScriptu a Javy, takže jsou zmenšeny nároky na počáteční znalosti programátorů přecházejících z jiných jazyků

Silverlight Technologie Silverlight[23] je projektem společnosti Microsoft. Základní princip je stejný jako u předchozí technologie. Uživatelské rozhraní je definováno ve značkovacím jazyku (XAML), o výkonnou část se stará objektově orientovaný programovací jazyk a ke spuštění aplikace je také potřebný plugin v prohlížeči. Aby měl Microsoft šanci uspět v konkurenci proti rozšířenému Flash Playeru, tak nabízí několik výhod pro vývojáře:

- využívá stejné výkonné jádro jako .NET
- lze používat více programovacích jazyků (C#, Visual Basic, Ruby, Python nebo PHP)
- je zde lepší podpora objektově orientovaného programování
- serverovou i klientskou část lze naprogramovat za pomoci jednoho jazyka a tedy nejsou kladeny vysoké nároky na programátory

JavaFX Poslední z nejvýznamnějších technologií je JavaFX[22] od firmy Sun Microsystems. Java byla jedna z prvních technologií pro vývoj RIA aplikací, která si však vybudovala špatnou pověst kvůli velkému instalačnímu balíku aplikace a pomalému startu aplikací. Proto firma Sun vytvořila JavaFX, která má být konkurencí pro předchozí dvě technologie. Výhoda je programování v Javě, která je mezi programátory velmi rozšířená. Také si lze stáhnout webovou aplikaci vytvořenou v JavaFX do počítače a využívat ji jako desktopovou aplikaci bez připojení k internetu.

Kapitola 3

Návrh řešení

Tato kapitola se zabývá návrhem systému, který prošel několika fázemi. Jednotlivé fáze návrhu počítají s různým přístupem k řešení umístění a propojení jednotlivých částí systému. Možností propojení jednotlivých komponent je velké množství, a tak bylo potřeba postupně vyzkoušet různé možnosti a navrhnout co nejoptimálnější řešení, které by nejlépe splňovalo požadavky autora práce na výsledný systém. Součástí kapitoly je také neformální specifikace, od které se návrh systému vyvíjel.

3.1 Neformální specifikace

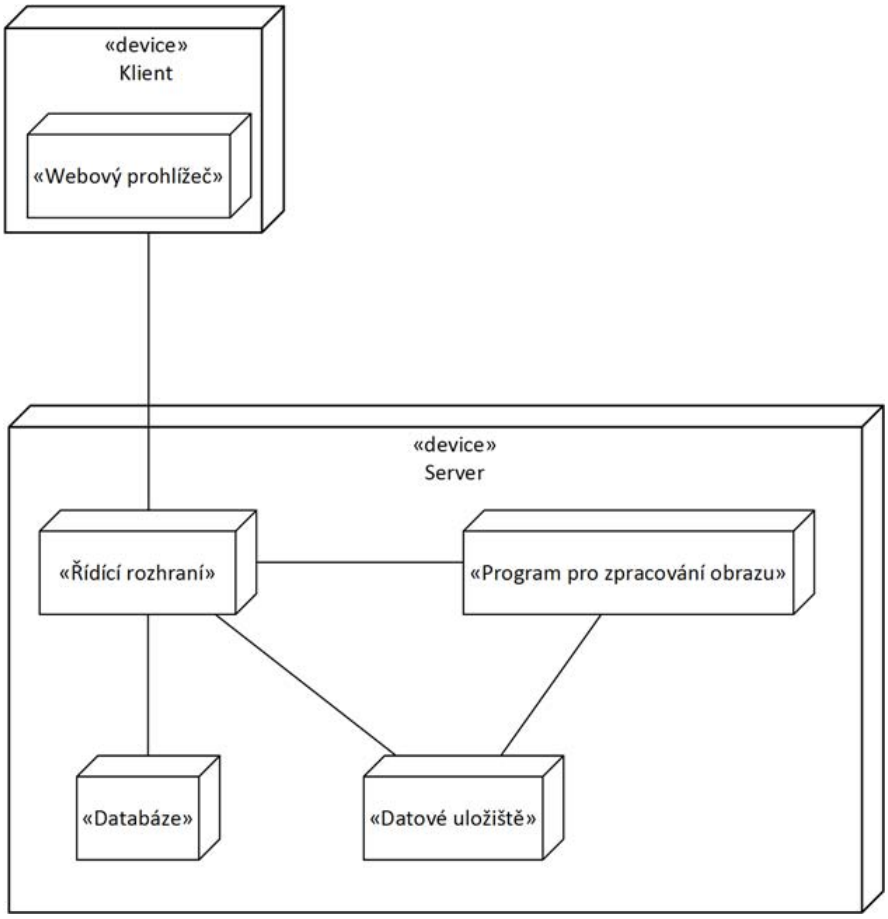
Navrhovaný systém by měl sloužit k jednoduššímu ovládání konzolových programů pro zpracování obrazu. Pomocí webového rozhraní (klienta) si uživatel zvolí obrazová data a operace nad těmito daty. U jednotlivých operací nastaví parametry. Následně uživatel pomocí klienta odešle požadavek na zpracování úkolů, které chce provést. Požadavek se předá řídícímu rozhraní, které se postará o provedení požadovaných operací. Řídící rozhraní spustí potřebné programy, kterým předá obrazová data. Programy zpracují data a výsledek vrátí řídícímu rozhraní. Toto rozhraní poté výsledek úkolu zobrazí uživateli.

3.2 Nedistribuovaný systém

Nejjednodušším řešením systému je umístění jednotlivých částí systému v rámci jednoho serveru (obrázek 3.1). Klient komunikuje s řídícím rozhraním, které se stará o veškeré věci spojené se zpracováním úkolu. Rozhraní komunikuje s databází a spouští potřebné programy, které provádí operace nad obrazovými daty. Data jsou umístěna v lokálním datovém úložišti, které je dostupné pro všechny prvky systému.

Výhody Největší výhoda tohoto řešení spočívá v jednoduchosti nasazení systému, kdy není potřeba řešit problémy s komunikací jednotlivých částí systému, protože se programy pro zpracování obrazu mohou volat přímo z řídícího rozhraní. Díky umístění komponent v rámci jednoho serveru a přímému volání se tedy nemusí brát ohled na zabezpečení přenosu dat a práva přístupu k jednotlivým datům spojeným s úkoly.

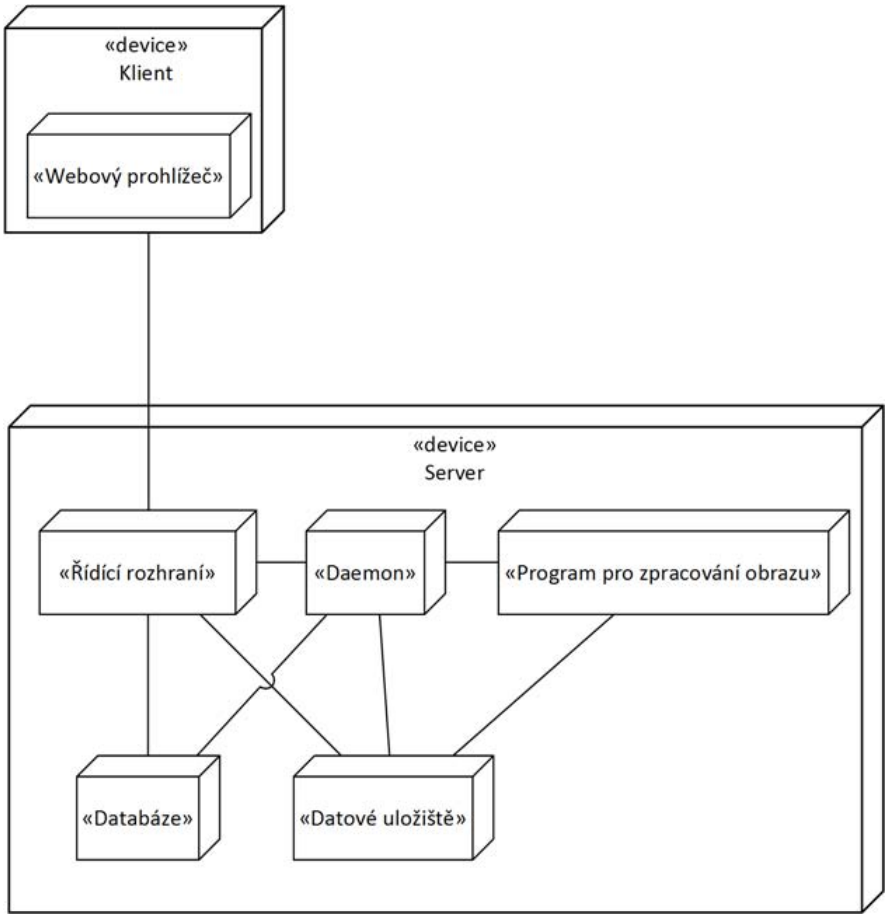
Nevýhody Nevýhoda nedistribuovaného systému může vzniknout pokud potřebujeme provádět náročné operace nad obrazem. Webové servery jsou zpravidla provozovány na méně výkonných počítačích, a proto nejsou vhodné pro náročné výpočty. V tomto případě



Obrázek 3.1: Diagram nasazení - Nedistribuovaný systém

tedy musí být celý systém umístěn na výkonném stroji, který bude tyto operace zvládat. Tento návrh také příliš nepodporuje provádění více operací zároveň, protože řídící rozhraní může v jednom momentě efektivně obsluhovat pouze jeden spuštěný program pro zpracování obrazových dat.

Optimalizace Jisté zlepšení by mohlo přinést zavedení prvku obsluhujícího spouštění programů (daemon). Daemon by byl program na serveru (mohl by existovat ve více instancích). Tento program by byl obsluhován z řídícího rozhraní a staral se o volání jednotlivých programů nad zadanými daty. Daemon by měl přístup k databázi a datovému úložišti, zapisoval by hlášení o ukončení provádění úkolu a řídící rozhraní by pouze o ukončení úkolu informoval, aby mohlo následně dojít k zobrazení výsledku na straně klienta. Toto řešení (obrázek 3.2) je ovšem také velmi omezováno výkonem serveru, na kterém bude systém umístěn.



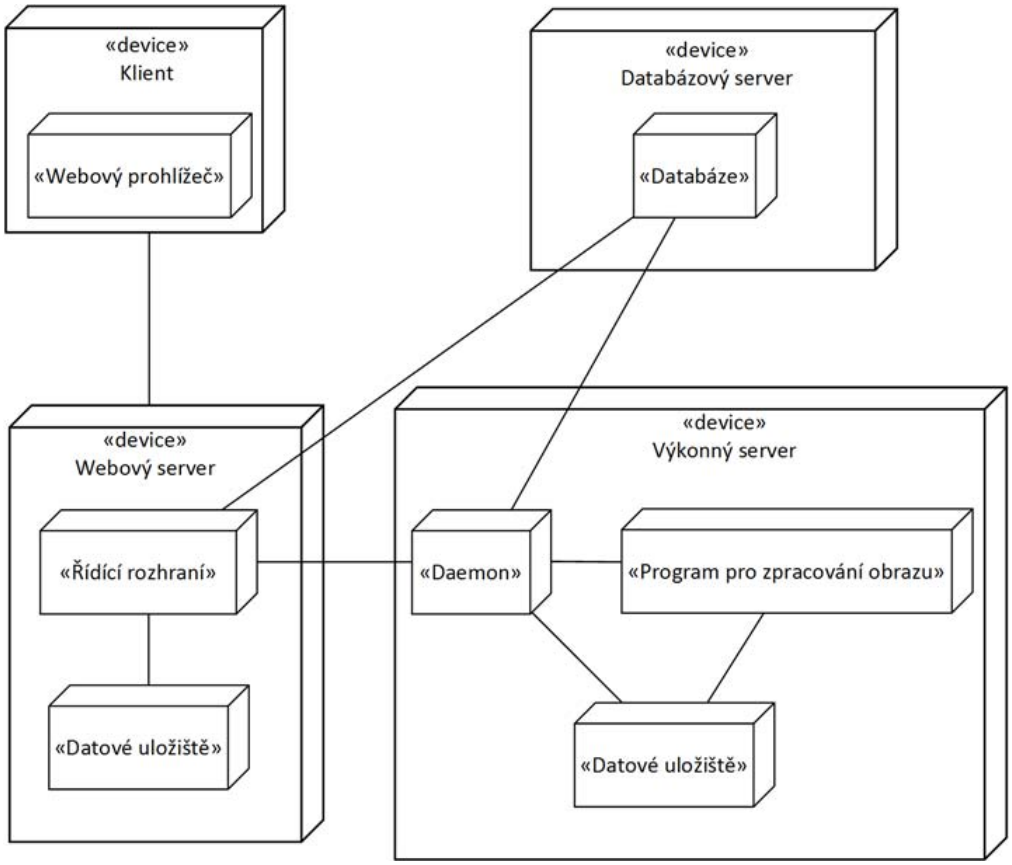
Obrázek 3.2: Diagram nasazení - Optimalizovaný nedistribuovaný systém

3.3 Distribuovaný systém

Problém s výkonem může vyřešit distribuce jednotlivých částí systémů. Druhá verze systému (obrázek 3.3) s distribucí jednotlivých částí mezi různé servery počítá. Řídící rozhraní je umístěno na webovém serveru spolu s lokálním datovým úložištěm. Databáze je umístěna na databázovém serveru a daemon je umístěn spolu s programy a svým lokálním datovým úložištěm na výkonném externím stroji. Řidící rozhraní a daemon komunikují s databází a předávají si pomocí ní požadavky a informace o stavu úkolů. Data, nad kterými se operace mají provést, se přenáší mezi řídícím rozhraním a potřebným daemonem.

Výhody Distribuovaný přístup přináší možnost existence libovolného množství daemonů, kteří mohou zpracovávat úkoly paralelně. Tento přístup také řesí problém s požadavkem na vysoký výkon webového serveru, protože výpočetní síla je umístěna na specializovaných serverech.

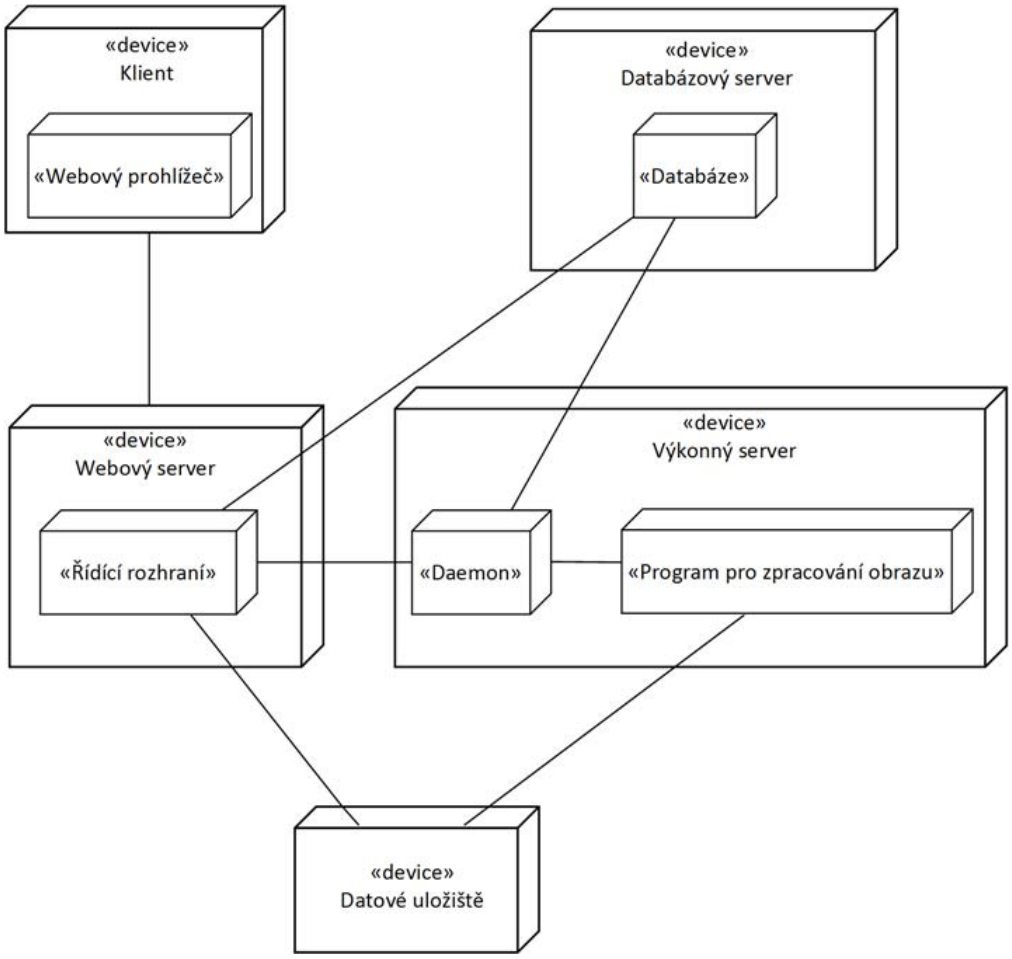
Nevýhody Značným nedostatkem tohoto návrhu jsou datová úložiště a předávání potřebných dat mezi jednotlivými částmi systému. Obrazová data bývají ve většině případů objemná, a proto by jejich přenos mezi řídícím rozhraním a daemonem mohl trvat déle než samotné provádění operací nad nimi. Další problém také může být při řešení dostupnosti



Obrázek 3.3: Diagram nasazení - Distribuovaný systém

databáze z řídícího rozhraní a zároveň z libovolného počtu daemonů. Při takovémto požadavku na dostupnost databáze se značně zvyšují nároky na její zabezpečení.

Optimalizace Problém s datovými úložišti by mohl být vyřešen pomocí jejich sloučení do jednoho a jeho zpřístupnění všem potřebným částem systému (obrázek 3.4). Touto optimalizací by došlo ke snížení množství přenášených dat a zrychlení provádění úkolů, protože by se čas provádění snížil o čas potřebný k přenosu dat. Na druhou stranu se zvýší požadavky na zabezpečení datového úložiště, podobně jako u společné databáze.



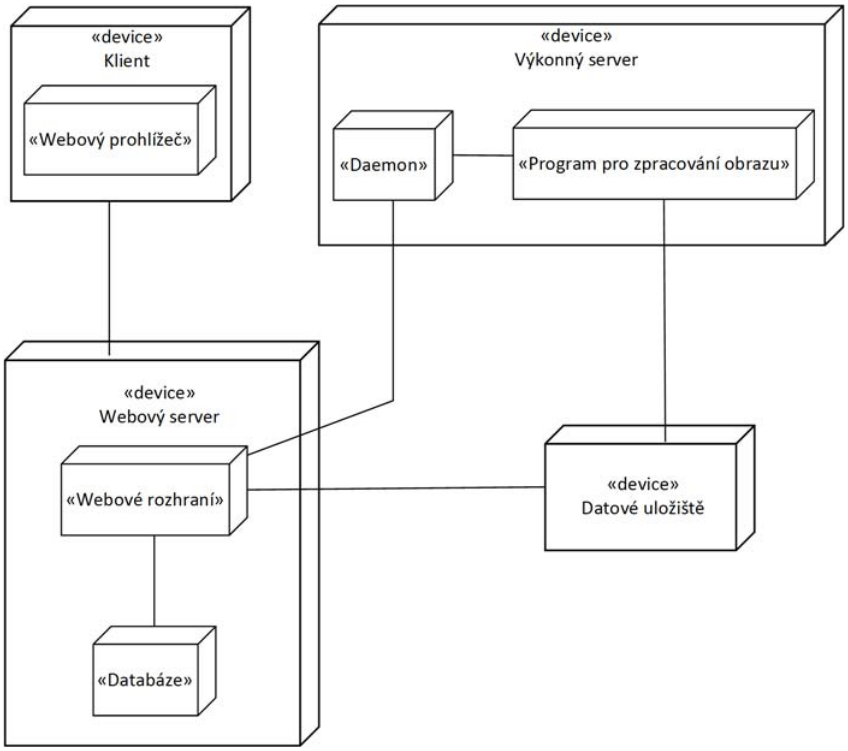
Obrázek 3.4: Diagram nasazení - Optimalizovaný distribuovaný systém

3.4 Finální verze

Tato verze systému (obrázek 3.5) je kompromisem mezi prvními dvěma přístupy. Zpracování operací nad obrazovými daty zůstává distribuováno pomocí daemonů na výkonných serverech. Datové úložiště také zůstane na serveru, který bude přístupný všem prvkům systému. Zásadní změnou tedy je zpřístupnění databáze pouze pro řídící rozhraní. Řízení bude tedy spolu s databází umístěno na webovém serveru a změny od daemonů se do databáze budou zapisovat pomocí zpráv posílaných řídícímu rozhraní. Dále si také požadavky na nové úkoly nebude daemon načítat přímo z databáze, ale budou mu předávány zprávami od řídícího rozhraní.

Výhody Díky přístupu k databázi pouze z řídícího rozhraní se lze vyhnout problémům se zabezpečením databáze pro přístup z více serverů. Také se zamezí problémům s konzistencí dat v databázi, které by mohly vznikat při současném zápisu.

Nevýhody V tomto řešení by se dalo najít také několik nevýhod, jako například navýšení množství komunikace mezi řídícím rozhraním a jednotlivými daemony. Tyto nevýhody jsou



Obrázek 3.5: Diagram nasazení - Finální verze

však již přijatelné jako výzvy pro realizaci výsledného systému a více se jejich popisu a řešení věnuje následující kapitola.

Kapitola 4

Specifikace návrhu

Tato kapitola se zabývá bližší specifikací výsledného návrhu systému pro zpracování obrazu. Z neformální specifikace, z návrhu finální verze a z diagramu nasazení finální verze plynou jisté požadavky a omezení na jednotlivé části systému a také na komunikaci mezi těmito prvky.

4.1 Uživatelské rozhraní

Webové rozhraní (klient), které slouží především k nastavení jednotlivých kroků před zpracováním dat a k zobrazování výsledků zadaných úkolů. Jedná se o výstup produkovaný řídícím rozhraním a zobrazovaný v prohlížeči uživatele.

- Úkoly - Uživatelské rozhraní bude umožnovat správu jednotlivých úkolů, jejich vytváření a zobrazování. Bude prováděna automatická aktualizace stavů u jednotlivých úkolů. Uživatel vybere data ke zpracování, vybere operace nad daty, nastaví potřebné parametry operací a odešle požadavek na zpracování. Řízení se předá rozhraní na webovém serveru.
- Uživatelé - Součástí klienta je také možnost registrace a přihlašování jednotlivých uživatelů. Každý uživatel bude mít svůj privátní seznam úkolů. Budou existovat dvě uživatelské role, a to uživatel a administrátor.
  - Uživatel - Bude mít možnost vytvářet a zobrazovat vlatní úkoly.
  - Administrátor - Bude mít stejné pravomoce jako uživatel a navíc bude mít možnost vytvářet, editovat a mazat ostatní uživatele.
- Ovládací prvky - Pro zprovoznění nejrůznějších dynamických ovládacích prvků jsou využity technologie Ajax, JavaScript, jQuery a jQueryUI. Díky těmto technologiím by měl klient poskytovat jednoduché a intuitivní ovládání celého systému.
- Výstup - Výstup systému zobrazovaný klientem bude ve formátu XHTML 1.0 Strict a bude formátován pomocí CSS.

4.2 Webový server

Webový server, na kterém bude umístěno řídící rozhraní a databáze systému. V rámci této diplomové práce bude pro implementaci použito prostředí hostované na serveru Apache HTTP Server.

Řídící rozhraní

Rozhraní, které se stará o komunikaci jednotlivých částí systému mezi sebou. Jedná se o nejdůležitější část systému, která se stará o aplikační logiku.

- Vytváření úkolů - Rozhraní přijímá požadavky na vytvoření úkolů od klienta, ukládá je do databáze a předává informace o úkolech daemonovi.
- Správa úkolů - Rozhraní bude automaticky zjišťovat stavy úkolů a informace o nich ukládat do databáze a zobrazovat klientovi.
- Registrace daemonů - Rozhraní přijímá požadavky na registraci nových daemonů. Od daemona získá konfigurační XML soubor s potřebnými údaji o daemonovi a seznamem všech dostupných programů a operací daemona.
- Komunikace s daemonem - Řídící rozhraní se dále stará o veškerou komunikaci s daemonem. Předává mu úkoly ke zpracování a přijímá od něj informace o ukončení úkolů. Také má možnost zjistit aktuální stav daemona.
- Implementace - Řídící rozhraní bude implementováno pomocí jazyka PHP. Jazyk PHP byl vybrán kvůlí své rozšířenosti a dostupnosti. Velkou roli ve výběru tohoto jazyka hrála také předchozí zkušenost autora této práce.

Databáze

Databáze pro uložení všech potřebných dat pro chod systému. S databází bude komunikovat pouze řídící rozhraní. Databáze bude implementována na databázovém systému MySQL. Systém MySQL byl vybrán hlavně pro svoji rozšířenost, dostupnost a celkovou jednoduchost použití při spolupráci s PHP.

Následují komentáře k jednotlivým tabulkám, které jsou zobrazeny v ER diagramu na obrázku 4.1:

- **users** - Uživatelé

  Tabulka uživatelů, kde má každý uživatel jedinečné číslo `id`, uživatelské jméno `username`, e-mail `email`, heslo `password`, úroveň oprávnění `level` a čas poslední aktivity `timestamp`.

- **daemons** - Daemoni

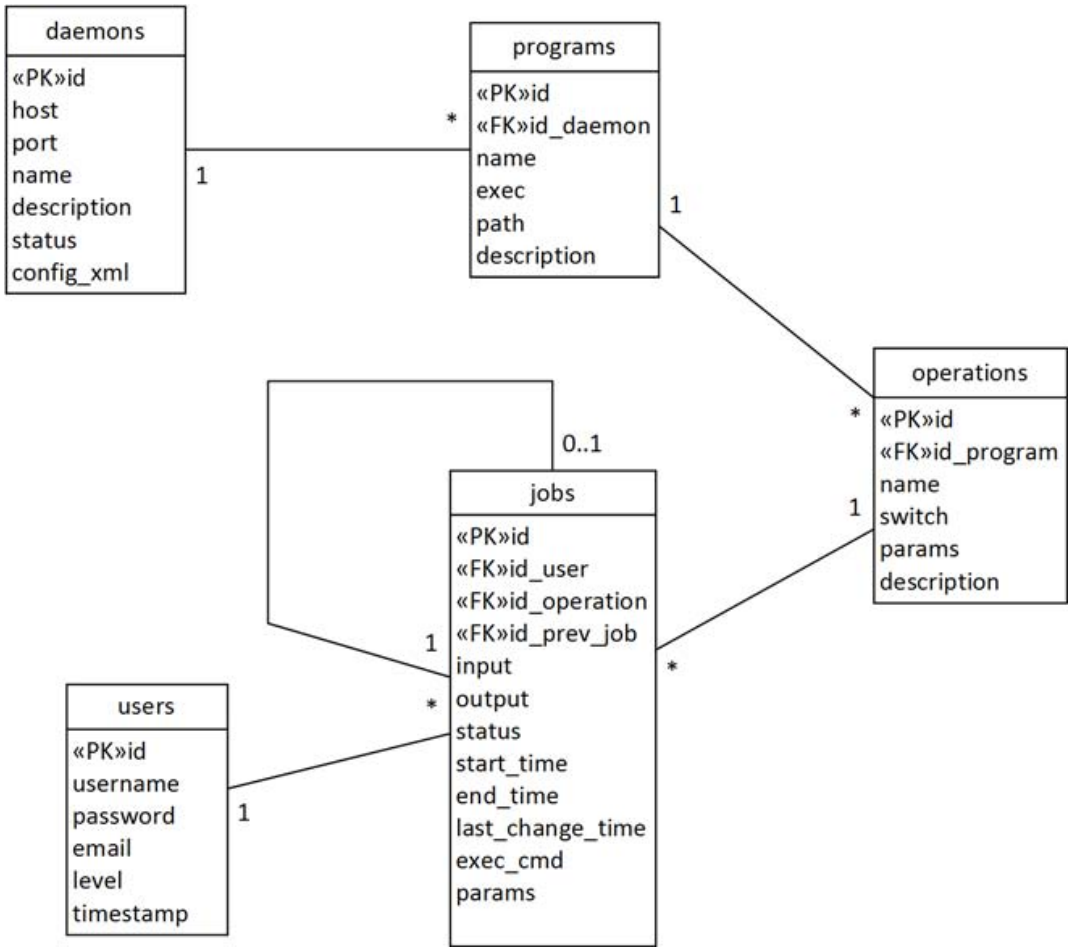
  Tabulka daemonů, která obsahuje jedinečné číslo daemona `id`, adresu daemona `host`, port pro TCP spojení `port`, jméno `name`, popisek `description`, stav `status` a název konfiguračního souboru `config_xml`.

- **programs** - Programy

  Tabulka se seznamem programů obsahující jedinečné číslo programu `id`, číslo daemona `id_daemon`, jméno `name`, název spustitelného souboru `exec`, cestu k datovému úložišti `path` a popisek `description`.

- **operations** - Operace

  Tabulka operací. Každá operace má jedinečné číslo `id`, číslo programu `id_program`, jméno `name`, přepínač `switch`, seznam parametrů `params` a popisek `description`.



Obrázek 4.1: ER diagram

- **jobs** - Úkoly

  Tabulka úkolů, kde je každý úkol reprezentován jedinečným číslem `id`, číslem majitele úkolu `id_user`, číslem operace `id_operation`, číslem předcházejícího úkolu `id_prev_job`, vstupem `input`, výstupem `output`, stavem úkolu `status`, počátečním časem `start_time`, konečným časem `end_time`, časem poslední změny `last_change_time`, příkazem pro spuštění operace na výkonném serveru `exec_cmd` a parametry operace `params`.

  Číslo předcházejícího úkolu je nutné v případě, že se bude provádět více na sebe navazujících operací.

4.3 Výkonný server

Výkonným serverem je myšleno takové místo v síti, které je dostatečně výkonné pro běh programů provádějících zpracování obrazu. Na výkonném serveru je umístěn daemon a programy. Výkonných serverů může být v rámci celého systému libovolné množství.

Daemon

Aplikace pro zpracování požadavků od řídícího rozhraní. Stará se o spouštění programů pro zpracování obrazu a odesílání informací o stavu úkolu zpět řídícímu rozhraní.

- Zpracování úkolů - Daemon bude informace o úkolech získávat od řídícího rozhraní a na základě nich spouštět potřebné programy.
- Externí programy - Daemon bude ve svém konfiguračním souboru mít informace o programech, které může spouštět. U jednotlivých programů nadále budou informace o poskytovaných operacích a jejich parametrech.
- Implementace - Daemon může být implementován pomocí libovolného jazyka. Musí pouze splňovat požadavky pro komunikaci, které budou popsány až dále v textu (podkapitola 4.5). Ukázkový daemon v této práci bude implementován pomocí jazyka PHP.

Informace o daemonovi budou uloženy v konfiguračním XML souboru, ze kterého tyto informace bude zpracovávat řídící rozhraní. Formát XML dokumentu je předem definován:

```
<?xml version="1.0" encoding="utf-8"?>
<daemon>
  <name>[name]</name>
  <host>[host]</host>
  <port>[port]</port>
  <description>[description]</description>
  <programs>
    ...
  </programs>
</daemon>
```

- `name` - Jméno daemona.
- `host` - Adresa serveru, na které je daemon spuštěn.
- `port` - Port, na kterém daemon běží.
- `description` - Popis daemona.
- `programs` - Seznam programů, které má daemon na starosti. Formát zápisu programů je uveden v následující podkapitole.

Programy pro zpracování obrazu

Návrh systému, který je v této práci popisován, je velmi univerzální a měl by umět pracovat s libovolným programem pro zpracování dat. Jediná podmínka kladená na program je, aby mohl být ovládán pomocí příkazové řádky. Nejsou kladeny žádné další specifické požadavky na to, co musí program umět. Toto poskytuje naprostou volnost při využití celého systému a mohlo by se jednat o univerzální ovládací prostředí pro konzolové aplikace.

V rámci diplomové práce se však autor zaměřil na zpracování obrazových dat, a proto je návrh uzpůsoben pro programy provádějící operace nad obrazem. V systému implementovaném v rámci práce se proto pracuje pouze s rastrovými grafickými soubory. Také je definován předpokládaný formát vstupních parametrů programu.

Formát vstupu je následující:

```
[program] [operation_switch] [param_switch] [param_value] ...
```

- `[program]` - Název spustitelného souboru.
- `[operation_switch]` - Přepínač operace.
- `[param_switch]` a `[param_value]` - Přepínač a hodnota parametru operace. Každá operace má dva povinné parametry a těmi jsou názvy vstupního a výstupního souboru. Nelze tedy použít program, který by používal vlastní pojmenování výstupních souborů.

Vstup do programu se bude sestavovat automaticky za pomoci konfiguračního XML souboru daemona. Formát XML zápisu se proto musí řídit předem definovanou strukturou:

```
...
<program exec="[program_exec]" path="[program_path]" name="" description="">
  <operations>
    <operation switch="[operation_switch]" name="" description="">
      <param switch="[param_switch]" type="input" name="" description="" />
      <param switch="[param_switch]" type="output" name="" description="" />
      <param switch="[param_switch]" type="[param_type]" name=""
          description="" />
      ...
    </operation>
    ...
  </operations>
</program>
...
```

- `program_exec` - Název spustitelného souboru.
- `program_path` - Cesta z programu k datovému úložišti.
- `operation_switch` - Přepínač operace. Pokud program obsahuje pouze jednu operaci, která se spouští automaticky bez přepínače, tak hodnota bude `null`.
- `param_switch` - Přepínač parametru. Paramatry se na vstupu řadí podle pořadí v XML dokumentu. Pokud parametr přepínač nemá, tak hodnota bude `null`.
- `param_type` - Typ parametru. Parametry lze rozdělit na několik typů:
  - `input` - Povinný parametr definující vstupní soubor.
  - `output` - Povinný parametr definující výstupní soubor.
  - `slider` - Parametr, který dovoluje nastavit rozsah hodnot. Parametr tohoto typu má ještě atributy `default` (výchozí hodnota), `from` (minimální hodnota), `to` (maximální hodnota) a `step` (velikost kroku).
  - `select` - Parametr, který dovoluje jednu z možností nastavení. Parametr tohoto typu má ještě atributy `default` (výchozí hodnota) a `array` - pole hodnot ve tvaru `[hodnota]=[název]#`.
- Všechny tagy také obsahují atributy `name` (název) a `description` (popis), které mají pouze informační hodnotu pro koncového uživatele.

4.4 Datové úložiště

Datové úložiště pro vstupní data, výsledná data po zpracování a XML konfigurační soubory daemonů.

- Dostupnost - Datové úložiště musí být dostupné z obou serverů (z webového serveru i z výkonného serveru, na kterém je umístěn daemon a programy).
- Adresář `XML` - Adresář, který bude obsahovat konfigurační XML soubory daemonů. V adresáři budou moci být pouze soubory s příponou `*.xml`.
- Adresáře s daty - Adresáře obsahující obrazová data budou pro webový server dostupné pouze pro čtení. Pro externí server budou dostupné pro čtení i zápis.
- Adresář `OUTPUT` - Adresář, do kterého se budou ukládat výsledná data po provedení zadaných operací.

4.5 Komunikace

Návrhu komunikace mezi jednotlivými komponenty systému byla věnována zvláštní pozornost, protože se jedná o jeden z hlavním pilířů celé funkčnosti. Mezi hlavní požadavky patří jednoduchost a bezpečnost. Nejdůležitějším procesem v komunikaci systému je posloupnost, která vede k vytvoření a zpracování úkolu. Celý tento proces se dá rozdělit na několik komunikačních bloků, které na sebe navazují.

- Uživatel pomocí klienta vybere data ke zpracování, zvolí požadované operace, nastaví potřebné parametry operací a odešle požadavek na vytvoření úkolu.
- Řídící rozhraní přijme požadavek na zpracování úkolu, uloží jej do databáze a informuje daemona o novém úkolu.
- Daemon požádá o informace o úkolu, zpracuje jej a předá informace o ukončení zpracování zpět řídícímu rozhraní.
- Řídící rozhraní zapíše detaily o ukončení úkolu do databáze a zobrazí výsledek úkolu klientovi.

Veškeré komunikace se tedy bude vždy účastnit řídící rozhraní, které bude komunikovat s klientem, databází a daemonem. V komunikaci mezi řídícím rozhraním a klientem a řídícím rozhraním a databází jde o standardní komunikaci, a proto zde budou popsány pouze požadavky na komunikaci mezi řídícím rozhraním a daemonem.

Řídící rozhraní a daemon

Díky výměně dat mezi těmito dvěma prvky lze zajistit pohodlné zpracování zadaných úkolů, a proto lze komunikaci mezi řídícím rozhraním a daemonem považovat za klíčovou pro běh celého systému. Veškerou komunikaci, až na jeden případ, zahajuje daemon zasláním POST požadavku na řídící rozhraní, které následně zašle HTTP odpověď. Jediná výjimka je při vytvoření nového úkolu, kdy řídící rozhraní probouzí spícího daemona pomocí zprávy zaslané přes soketové TCP spojení. Zahajování komunikace bylo svěřeno daemonovi hlavně kvůli bezpečnosti, aby nedocházelo k neoprávněnému přístupu z webového rozhraní na

externí server, který může být jinak chráněný. Také se díky tomuto způsobu komunikace zamezí zbytečnému dotazování řídícího rozhraní na výsledky zpracování úkolů.

Typy komunikace mezi daemonem a řídícím rozhraním se tedy dají rozdělit do tří skupin:

- Daemon zasílá řídícímu rozhraní zprávy za pomoci POST požadavků protokolu HTTP. Zprávy se posílají na stránku `daemonHandler.php`, která je umístěna v kořenovém adresáři řídícího rozhraní. Formát zprávy je následující:

```
POST /daemonHandler.php HTTP/1.1
Host: [adresa serveru]
Content-length: [délka odesílaných dat]
Content-type: application/x-www-form-urlencoded
Connection: close

code=[kód požadavku]&[atributy]...
```

- Řídící rozhraní odpovídá pomocí HTTP odpovědi:

```
HTTP/1.1 200 OK
Date: Tue, 11 May 2010 14:56:41 GMT
Server: Apache/2.2.11 (Win32) PHP/5.3.0
X-Powered-By: PHP/5.3.0
Set-Cookie: PHPSESSID=d1c5elsjut1qndfkkoll14r6o1; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
  pre-check=0
Pragma: no-cache
Content-Length: [délka odpovědi]
Connection: close
Content-Type: text/html; charset=utf-8

[kód odpovědi]
```

- Jako poslední možnost komunikace je soketové TCP spojení pro informování daemona o novém úkolu.

Zpracování úkolu

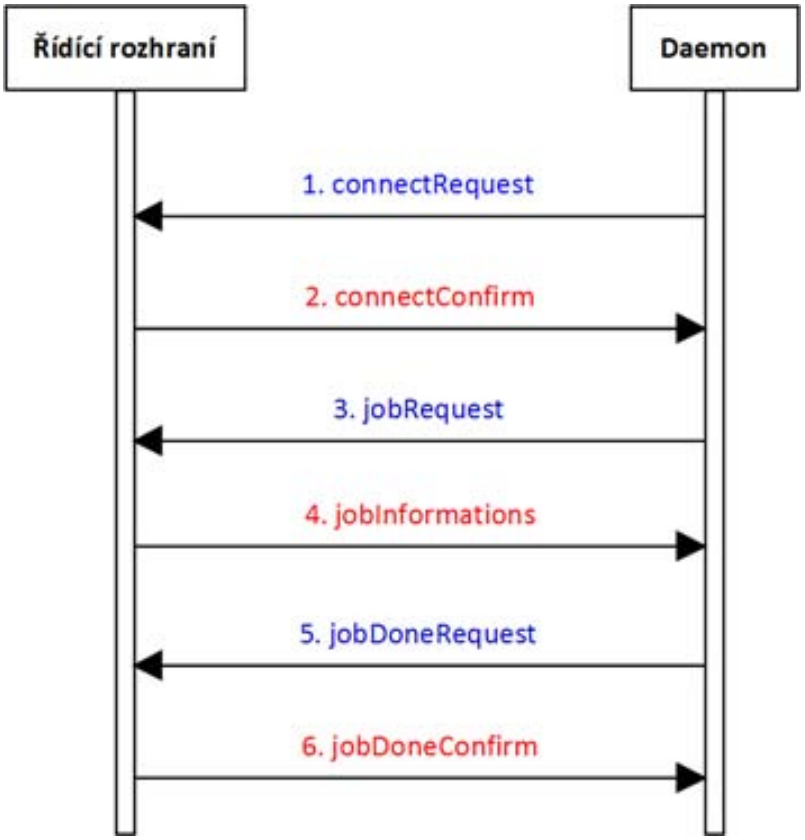
Nejčastější komunikací mezi daemonem a řídícím rozhraním bude samotné zpracování úkolů, které se bude skládat ze získání informací o úkolu a oznámení o provedení úkolu. Posloupnost jednotlivých zpráv bude následující:

1. Po svém spuštění zašle daemon požadavek na připojení k řídícímu rozhraní.

   ```
   code=connectRequest&host=[host]&port=[port]
   ```

2. Rozhraní v databázi zkontroluje, zda daemona zná. Pokud ano, tak změní jeho stav na připraven (*Ready*) a odešle mu v odpovědi potvrzení o připojení.

   ```
   connectConfirm
   ```



Obrázek 4.2: Posloupnost zpráv - Zpracování úkolu

3. Daemon požádá o informace k úkolu, který je určen ke zpracování.

   `code=jobRequest&host=[host]&port=[port]`

4. Řídící rozhraní zašle daemonovi informace a změní jeho stav na zaneprázdněn (*Busy*).

   `jobInformations | [id úkolu] | [příkaz pro spuštění operace nad daty]`

5. Daemon spustí program s potřebnými parametry a informuje řídící rozhraní o provedení úkolu.

   texttt`code=jobDoneRequest&id=[id úkolu]&host=[host]&port=[port]`

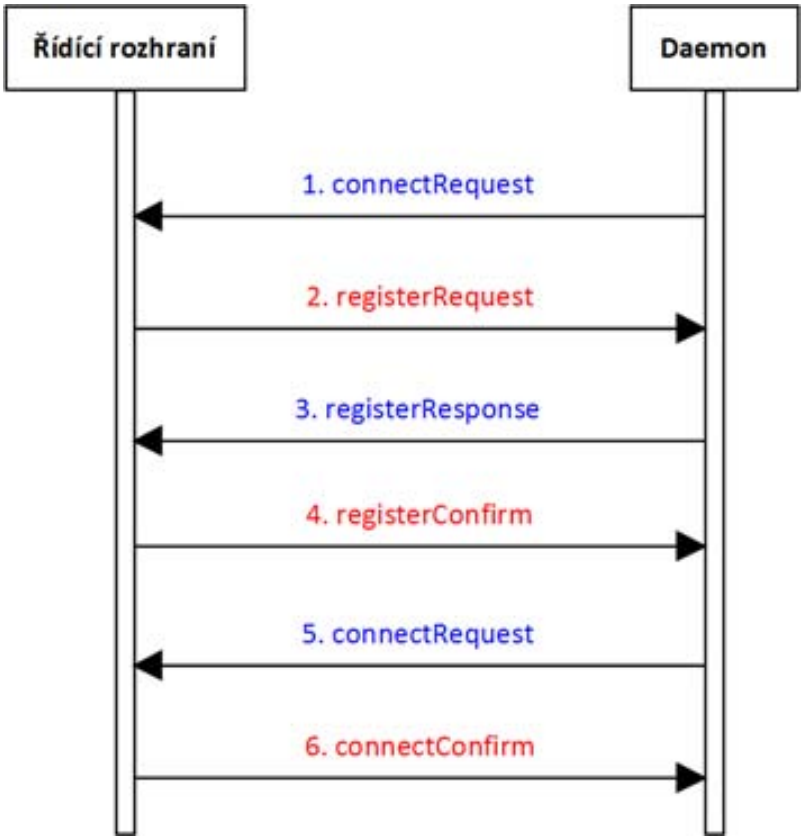
6. Rozhraní zapíše informaci o ukončení úkolu do databáze, změní stav daemona zpět na připraven (*Ready*) a zašle daemonovi potvrzení.

   `jobDoneConfirm`

7. Daemon pokračuje krokem 3.

Registrace daemona

Další možností komunikace mezi řídícím rozhraním a daemonem je registrace daemona, který ještě není řídícímu rozhraní znám.



Obrázek 4.3: Posloupnost zpráv - Registrace daemona

0. Majitel daemona vytvoří a nahraje XML konfigurační soubor na datové úložiště do složky `XML`.
1. Po svém spuštění zašle daemon požadavek na připojení k řídícímu rozhraní stejně jako v prvním případě.
2. Rozhraní v databázi zjistí, že daemona nezná a zažádá o jeho registraci.

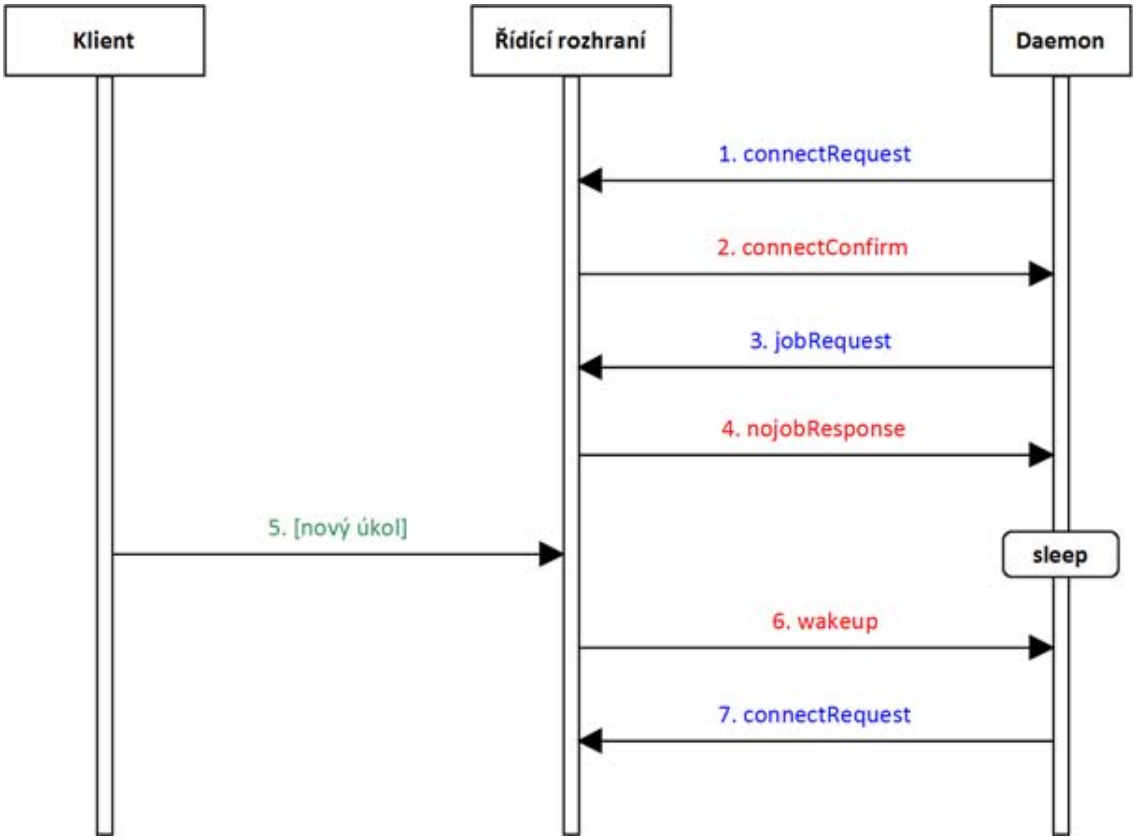
   `registerRequest`
3. Daemon zašle odpověď na registraci spolu s cestou ke svému konfiguračnímu souboru.

   `code=registerResponse&config=[config]`
4. Rozhraní načte konfigurační soubor, uloží informace o daemonovi do databáze a odešle potvrzení o registraci.

   `registerConfirm`
5. Daemon zopakuje krok 1 s tím, že už je registrován a dostane potvrzení o připojení k řídícímu rozhraní.

Probuzení daemona

Pokud je daemon aktivní, ale nezpracovává žádný úkol, tak při vytvoření nového úkolu může řídící rozhraní poslat dameonovi zprávu, aby ho informoval o existenci tohoto úkolu.



Obrázek 4.4: Posloupnost zpráv - Probuzení daemona

1. Daemon je úspěšně připojen k řídícímu rozhraní a zažádá si o informace k úkolu.
2. Řídící rozhraní zjistí, že aktuálně neexistuje žádný úkol, který by daemon mohl zpracovávat a odešle mu o tom informaci.

   `nojobResponse`
3. Daemon se přepne do pasivního režimu, kdy naslouchá na svém portu a čeká na příchozí soketové TCP spojení.
4. Po zapsání nového úkolu do databáze vytvoří řídící rozhraní TCP spojení s daemonem na jeho portu. Pomocí tohoto spojení pošle daemonovi zprávu. Pokud se nepodaří vytvořit spojení, tak rozhraní změní stav daemona na nepřihlášený (*Offline*).

   `wakeup`
5. Daemon se přepne zpět do aktivního řežimu a zopakuje žádost o připojení k řídícímu rozhraní.

Ostatní zprávy

Mimo uvedené tři možnosti, které jsou nejčastější, existuje ještě několik dalších zpráv, které si daemon s řídícím rozhraním může vyměňovat. Jedná se o příkazy pro aktualizaci údajů daemona, odregistraci daemona a dále o chybové zprávy. Přehled jednotlivých kódů pro tyto zprávy je uveden v následující tabulce.

Dotaz	**Odpověď**	**Popis**
`registerRequest`	`registerFailed`	Chyba při registraci daemona
`updateRequest`	`updateConfirm`	Aktualizace údajů daemona
`updateRequest`	`updateFailed`	Chyba při aktualizaci údajů daemona
`deleteRequest`	`deleteConfirm`	Smazání daemona z databáze
`deleteRequest`	`deleteFailed`	Chyba při mazání daemona z databáze

Zpráva s kódem `updateRequest` obsahuje parametry `host`, `port` a `config`. Zpráva s kódem `deleteRequest` obsahuje parametry `host` a `port`.

Kapitola 5

Implementace

V této kapitole je podrobněji popsána vlastní implementace celého systému. Jsou zde vysvětleny použité funkce, postupy a algoritmy. V závěru kapitoly je rozebráno nasazení systému v rámci lokální sítě a v rámci internetu.

5.1 Uživatelské rozhraní

Uživatelské rozhraní je výstup generovaný řídícím rozhraním jako XHTML dokument. Tento dokument je poté formátován za pomoci CSS stylů. Takto naformátovaný výstup lze poté oživit přidáním dynamického ovládání jednotlivých prvků. Následující text se proto věnuje zejména implementaci těchto dynamických prvků. K tomu je využit jazyk JavaScript, knihovna jQuery verze 1.4.2 a framework jQueryUI verze 1.8.

Vzhled uživatelského rozhraní

Framework jQueryUI poskytuje předpřipravené sady CSS stylů, díky kterým lze jednodušeji vytvářet vzhled stránky. V rámci implementace byl využit styl *Redmond*. Mimo tento CSS styl byly použity také styly jednotlivých pluginů a konečně také styl vytvořený autorem práce, který definuje zejména rozložení jednotlivých elementů a detaily, které ostatní styly neřeší.

Přístup k elementům

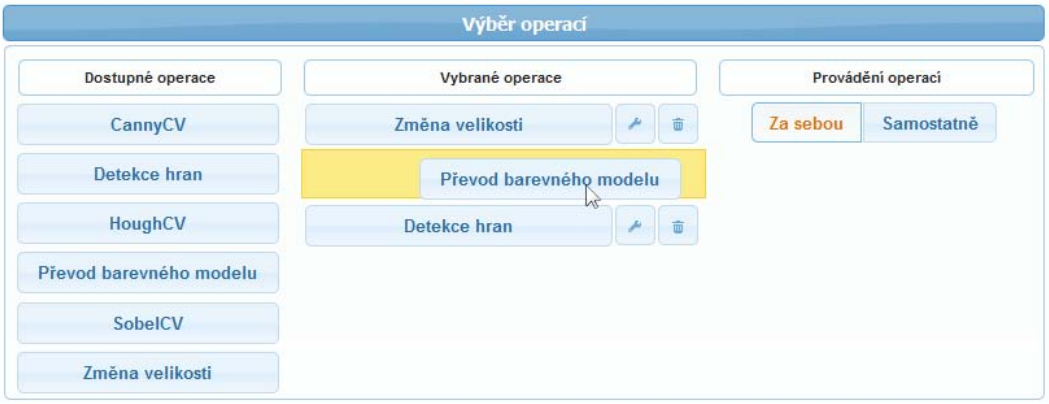
K XHTML elementům stránky se dá přistupovat dvěma způsoby a to buď přes identifikátor, nebo přes třídu. Pokud tedy máme například prvek `<div id="testId"class="testClass">test</div>`, tak ke všem jeho atributům lze přistoupit přes identifikátor `$("#testId").nazevMetody();` nebo pomocí třídy `$(".testClass").nazevMetody();`.

Vytvoření úkolu

Stránka sloužící k vytvoření úkolu je základem celého webového rozhraní a poskytuje nejvíce nastavení pro uživatele. Z tohoto důvodu je na této stránce použito mnoho dynamických ovládacích prvků, které by měly usnadnit uživateli práci.

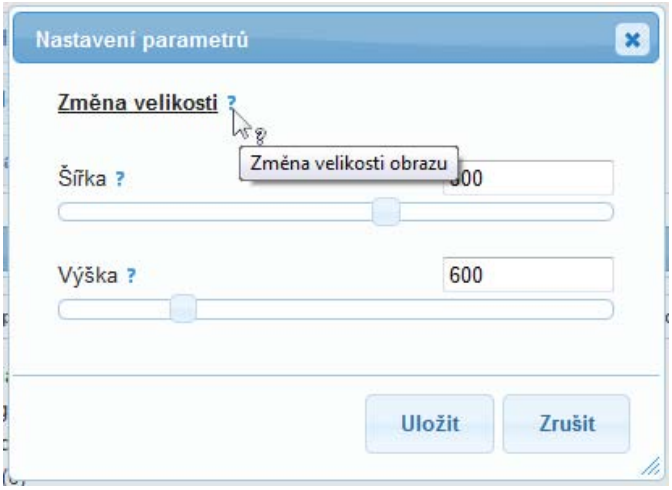
Výběr operací Výběr operací je realizován jako dva propojené seznamy. První seznam obsahuje názvy dostupných operací. Druhý seznam je prázdný. Operace z prvního seznamu

se pomocí *Drag&Drop* kopírují do druhého seznamu. Toto je realizováno pomocí metody `draggable()` zavolané na první seznam a metody `droppable()` použité na druhý seznam (obrázek 5.1). Po upuštění přetahované operace nad druhým seznamem dojde k vyvolání dialogového okna pomocí metody `dialog()`. Obsah dialogového okna se načte pomocí Ajaxového volání `ajax()`. V dialogu má uživatel možnost nastavit jednotlivé parametry operace. Pro nastavování číselných hodnot jsou použity prvky typu *Slider* volané pomocí metody `slider()` (obrázek 5.2). Po nastavení žádaných hodnot se dialog odešle a operace se uloží do seznamu. U operace se objeví tlačítka umožnující změnu parametrů a odstranění operace. Pořadí jednotlivých operací lze měnit díky použití metody `sortable()` na jejich seznam.



Obrázek 5.1: Výběr operací - Ukázka s Drag&Drop

Největší problém představovala realizace předávání identifikátoru a parametrů jednotlivých operací do formuláře, který se odešle při vytvoření úkolu. Při zkopírování operace do seznamu k provedení se proto vždy musí nastavit nové `id` pro prvek seznamu a identifikátor operace musí být uložen v atributu `name` tohoto prvku. Pod hodnotou `id` prvku se poté parametry operace uloží do pomocného pole a při odeslání se vygeneruje seznam operací ve správném pořadí.



Obrázek 5.2: Dialogové okno s nastavením parametrů operace

U výběru operací si lze pomocí přepínače, realizovaného použitím metody `buttonset()` na `input` prvky typu `radio`, zvolit zda se mají operace provádět v návaznosti za sebou, nebo každá samostatně.

Výběr obrázků Výběr obrazových dat je realizován pomocí pluginu `jsTree`[1]. Po zavolání funkce `tree()` na strukturu netříděných seznamů umožnuje tento plugin vytvořit strom s možností výběru jednotlivých listů. Kořeny tohoto stromu jsou složky v datovém úložišti a jednotlivé soubory jsou reprezentovány jako listy stromu. Při kliknutí na název se soubor označí jako zvolený ke zpracování. Při najetí na název souboru se ukáže náhled obrázku (obrázek 5.3).

Obrázek 5.3: Výběr obrázků

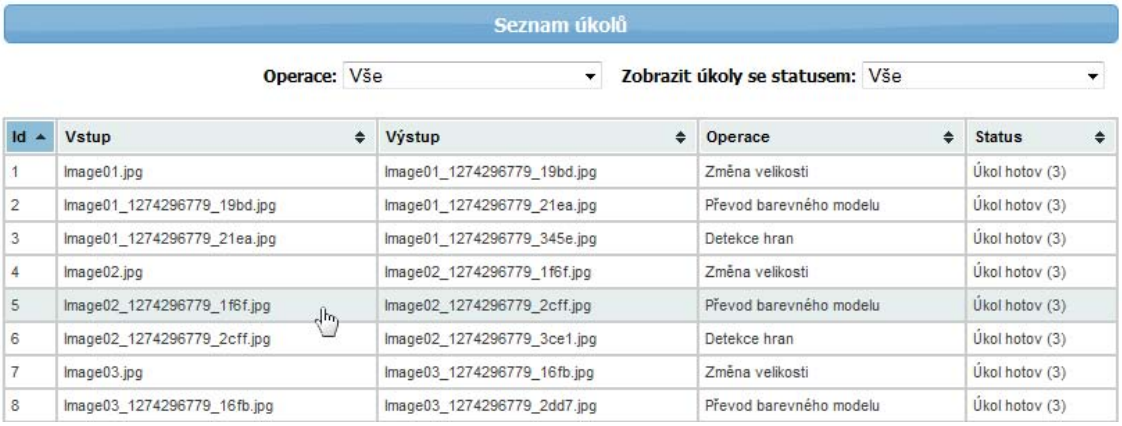
Přehled úkolů

Stránka s přehledem úkolů je tvořena tabulkou vyplněnou informacemi o jednotlivých úkolech. Pro efektivnější práci s tabulkou je použit plugin *tablesorter*[2], který poskytuje možnost řazení jednotlivých sloupců tabulky. Pro inicializaci pluginu je potřebné na element <`table id="listTable"`> zavolat metodu `tablesorter()`. O aktualizaci dat v tabulce se stará funkce `getListTable()`. Největším problémem bylo zajistit, aby se řazení při automatickém načtení hodnot nezměnilo. Tento problém řeší funkce `getSorting()`, která před načtením dat zjistí aktuální řazení tabulky a po aktualizaci dat nastaví řazení stejné (obrázek 5.4). Součástí přehledu úkolů jsou i dva filtry a to filtr operací a filtr stavu úkolu.

Detail úkolu

Stránka zobrazující detail úkolu umožnuje automatickou aktualizaci stavu úkolu pomocí technologie Ajax. Jsou zde zobrazeny informace o vstupním a výstupním obrázku, informace o zadané operaci a informace o stavu provádění úkolu. Součástí této stránky je také seznam všech souvisejících úkolů. To znamená, že jsou zde zobrazeny úkoly se stejným zdrojovým

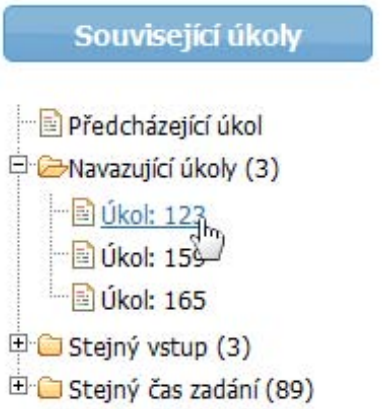
[1] http://www.jstree.com/demo
[2] http://tablesorter.com/docs/



Id	Vstup	Výstup	Operace	Status
1	Image01.jpg	Image01_1274296779_19bd.jpg	Změna velikosti	Úkol hotov (3)
2	Image01_1274296779_19bd.jpg	Image01_1274296779_21ea.jpg	Převod barevného modelu	Úkol hotov (3)
3	Image01_1274296779_21ea.jpg	Image01_1274296779_345e.jpg	Detekce hran	Úkol hotov (3)
4	Image02.jpg	Image02_1274296779_1f6f.jpg	Změna velikosti	Úkol hotov (3)
5	Image02_1274296779_1f6f.jpg	Image02_1274296779_2cff.jpg	Převod barevného modelu	Úkol hotov (3)
6	Image02_1274296779_2cff.jpg	Image02_1274296779_3ce1.jpg	Detekce hran	Úkol hotov (3)
7	Image03.jpg	Image03_1274296779_16fb.jpg	Změna velikosti	Úkol hotov (3)
8	Image03_1274296779_16fb.jpg	Image03_1274296779_2dd7.jpg	Převod barevného modelu	Úkol hotov (3)

Obrázek 5.4: Přehled úkolů

obrázkem, úkoly se stejným začátkem provádění a úkoly, které předchází a navazují na právě zobrazený úkol. Seznam souvisejících úkolů je realizován pomocí pluginu `Treeview`[3], který je svojí funkčností podobný pluginu `jsTree`, ale poskytuje trochu jinou funkčnost. Pro vytvoření stromu je potřeba zavolat metodu `treeview()` (obrázek 5.5).



Obrázek 5.5: Detail úkolu - Seznam souvisejících úkolů

Registrace uživatele

Pro registrační formulář je využit plugin *Validation*[4], který umožnuje pokročilou validaci prvků formuláře pomocí metody `validate()` zavolané na požadovaný formulář (obrázek 5.6). Stejný plugin je použit i na formulář, který slouží pro změnu údajů o uživateli.

[3] http://bassistance.de/jquery-plugins/jquery-plugin-treeview/
[4] http://bassistance.de/jquery-plugins/jquery-plugin-validation/

Obrázek 5.6: Registrace uživatele

Ostatní dynamické prvky a funkce

Mezi ostatní použité dynamické prvky z frameworku jQueryUI patří prvek *Accordion* použitý na úvodní stránce. Tento prvek se volá pomocí metody `accordion()` a umožnuje vytvořit posouvací menu, kdy po kliknutí na záložku vyjede požadovaný obsah. Mezi další použité prvky patří efekty a metody umožnující animaci prvků stránky a přístup k jednotlivým vlastnostem elementů XHTML.

5.2 Řídící rozhraní

Řídící rozhraní obsahuje aplikační logiku celého systému. To v sobě zahrnuje zejména třídy pro obsluhu daemonů a třídy pro generování výstupu zobrazené uživatelským rozhraním.

Komunikace s databází

Pro komunikaci s databází MySQL slouží třída `Database`, která rozšiřuje funkčnost PHP třídy `Mysqli`. Za zmínku zejména stojí funkce `query($q)`, která vrací výsledek v závislosti na typu dotazu `$q`. Pokud se jedná o dotaz na více prvků tabulky, tak vrátí pole objektů. Pokud se jedná o dotaz na jednu buňku tabulky, tak vrátí pouze hodnotu této buňky. Pokud se jedná o dotaz `INSERT`, `UPDATE` nebo `DELETE`, tak nevrátí nic.

Vytváření úkolů

Vytváření úkolů má na starost třída `Jobs`. Pro samotné vytváření záznamů v databázi o úkolech jsou použity dva různé algoritmy, protože při výběru operací může být zadáno provádění za sebou nebo samostatně.

Provádění operací za sebou znamená, že se zvolené operace provedou sériově. Nad vstupním obrázkem se provede první operace a nad jejím výstupem se provede další zadaná operace. Algoritmus pro sériové uložení úkolů je vytvořen ve funkci `procNewJobsSerial($inputs, $opers)`, která pro každý obrázek přiřadí všechny operace. Přitom se musí hlídat správné návaznosti vstupů a výstupů jednotlivých operací.

Druhá možnost provádění je provádění všech operací nad jedním vstupem. Tento algoritmus je vytvořen ve funkci `procNewJobsParallel($inputs, $opers)`. Tato funkce naopak

pro každou operaci přiřadí všechny obrázky. Z hlediska implementace se jedná o jednodušší řešení, protože se nemusí hlídat jednotlivé návaznosti úkolů.

Zasílání zpráv

Zasílání zpráv daemonům je realizováno pomocí třídy `DaemonHandler`, která zpracovává příchozí POST požadavky a volá požadované funkce, které pak vypíší kód odpovědi jako HTML výstup. Třída se tedy stará o správnou komunikaci s jednotlivými daemony a o obsluhu jejich požadavků.

Registrace daemona Při požadavku na registraci daemona je zavolána funkce `registerDaemon($config)`, která otevře konfigurační XML soubor daemona pomocí PHP třídy `SimpleXML`. Pro zapsání daemona do databáze je potřeba vytvořit několik objektů, které jsou vzájemně propojeny tak, že předchozí vždy obsahuje seznam následujících objektů.

```
Daemon -> (array) Program -> (array) Operation -> (array) Param
```

Všechny objekty jsou tedy přístupné z jednoho rodičovského objektu `Daemon`. Pokud jsou všechny správně vytvořeny a načteny, tak se zavolá funkce `insertIntoDB()` objektu `Daemon`, která rekurzivně uloží veškeré potřebné informace do databáze.

Aktualizace daemona Při příchozím požadavku na aktualizaci daemona se volá metoda `updateDaemon()`. Vytvoří se dva rodičovské objekty - jeden z databáze a druhý z konfiguračního XML souboru daemona. Po vytvoření těchto objektů je zavolána funkce `update()`. Dojde k porovnání a aktualizaci údajů o daemonovi, jeho programech a operacích. Největším problémem aktualizace bylo zajistit, aby případné změny programů a operací nezpůsobily problémy u již vytvořených úkolů. Proto se u každé operace kontroluje, zda došlo k nějaké zásadní změně (změna názvu nebo popisu není brána jako zásadní). Pokud ano, tak úkoly, nad kterými měla být tato operace provedena, jsou prohlášeny za neplatné. Provedení této kontroly je zajištěno pomocí funkce `checkChanges()` objektu `Operation`.

Smazání daemona Při požadavku na smazání daemona z databáze dojde k zavolání funkce `deleteDaemon()`. Ta získá objekt `Daemon` a poté je zavolána funkce `delete()` objektu `Daemon`. Tato funkce nejdříve upraví úkoly, na kterých měly být provedeny operace poskytované tímto daemonem tak, že změní jejich stav na *Úkol nelze provést*. Poté smaže veškeré informace o daemonovi z databáze a odešle daemonovi odpověď s potvrzením o jeho smazání.

Správa přihlášených uživatelů

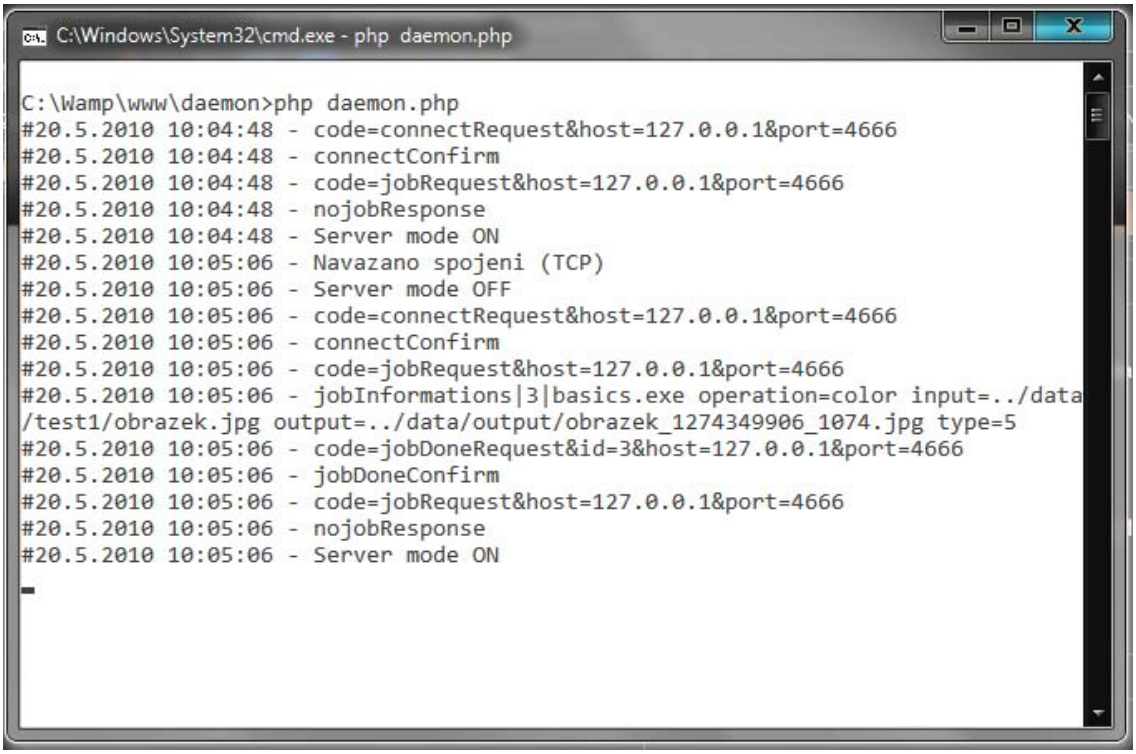
Správu přihlášených uživatelů mají na starosti třídy `User` a `Session` pomocí proměnných Session prohlížeče. Každá stránka má stupeň zabezpečení (proměnná `$page['level']`) a každý uživatel má stupeň oprávnění. Přístup na stránku má poté uživatel pokud je jeho stupeň oprávnění stejný nebo vyšší než stupeň zabezpečení stránky. Při každém načtení stránky se kontroluje oprávnění uživatele zobrazit tuto stránku. Pokud uživatel nemá dostatečné oprávnění, tak je přesměrován na stránku, pro kterou oprávnění má.

5.3 Daemon

Ukázkový daemon v této diplomové práci je implementován pomocí jazyka PHP a je reprezentován třídou `Daemon`, která obsahuje provázané metody pro celý a nepřerušený chod daemona. Jako první se po spuštění daemona zavolá funkce `callHandler()`, která zašle požadavek na připojení řídícímu rozhraní.

Zasílání zpráv

Zasílání zpráv probíhá za pomoci POST požadavku protokolu HTTP. Pro tvorbu těchto požadavků slouží metoda `postData($host, $path, $data)`, která datům přidá správnou HTTP hlavičku, odešle požadavek na daný server a zavolá funkci `getContent($msg)`. Tato funkce odstraní HTTP hlavičku z odpovědi a vrátí zprávu zaslanou řídícím rozhraním. Tato zpráva je pak předána metodě `msgSwitch($msg)`, která podle obsahu zprávy spustí další navazující funkci. V rámci této funkce se také volá metoda `msgPrint($msg)`, která slouží k vypsání obsahu zprávy. Ukázku výpisu zpráv daemona lze vidět na obrázku 5.7.



Obrázek 5.7: Ukázka výpisu daemona (obrázek byl invertován)

Zpracování úkolu

Při přijetí odpovědi od řídícího rozhraní, která obsahuje informace o úkolu, se spustí provádění funkce `procJob($msg)`. Tato funkce rozdělí přijatou zprávu na tři části - kód zprávy `jobInformations`, identifikační číslo úkolu `id` a příkaz pro spuštění operace `exec_cmd`. Pokud jsou všechny tři části zprávy v pořádku, tak dojde k provedení zadané operace nad daty a řídícímu rozhraní se odešle zpráva o ukončení úkolu. Pokud některá z částí v pořádku

není, tak běh daemona skončí chybou a vypsáním přijaté zprávy. K této situaci může dojít například při chybě v kódu řídícího rozhraní, které odešle nesmyslnou odpověď, což znamená, že je nefunkční a nemá tedy smysl pokračovat v další komunikaci s ním. Provedení operace probíhá zavoláním PHP funkce `exec($exec_cmd)` a následným čekáním na její ukončení.

Pasivní režim

Jesliže pro daemona není žádný úkol ke zpracování, tak ten se přepne do pasivního režimu, kdy naslouchá na svém portu a čeká na příchozí TCP spojení. Toto čekání je implementováno pomocí BSD socketů, které zajistí, že se daemon chová jako TCP server. Daemon čeká na příchozí spojení od řídícího rozhraní, díky kterému získá informaci o novém úkolu a přepne se zpět do aktivního režimu. Toto zajišťuje funkce `server($host, $port)`.

Aktualizace a smazání daemona

Daemon se mimo normálního spuštění bez parametrů dá také spustit dalšími dvěma způsoby. První možností je spustění programu s parametry `-update [host] [port]`, díky čemuž dojde jako první k zavolání funkce `updateDaemon()`, která řídícímu rozhraní odešle žádost o aktualizaci údajů daemona. Po provedení aktualizace pokračuje daemon nadále ve své činnosti. Parametry `[host]` a `[port]` zastupují původní adresu a port daemona. Tyto údaje je nutné uvést, protože v konfiguračním souboru již může být například uveden jiný port, než na kterém dosud daemon běžel. Druhou možností je použít parametr `-delete`, čímž se jako první spustí funkce `deleteDaemon()`, která se postará o zaslání požadavku na odstranění daemona z databáze řídícího rozhraní. Po smazání daemona dojde k ukončení provádění skriptu daemona.

5.4 Programy pro zpracování obrazu

Všechny programy pro zpracování obrazu, použité v rámci této diplomové práce jako příkladové programy, jsou implementovány v jazyce C pomocí knihovny OpenCV verze 2.1. Jednotlivé implementované operace jsou rozděleny do tří programů, a to hlavně z důvodu demonstrace následného zápisu konfiguračního XML souboru daemona, protože každý z programů má jiný formát spuštění.

Program *basics*

Program *basics* obsahuje dvě jednodušší operace naimplementované za použití funkcí knihovny OpenCV. Mezi tyto operace patří změna velikosti obrázku, která využívá funkci `cvResize`, a změna barevného modelu obrázku z RGB na jiný za pomoci metody `cvCvtColor`.

Vstup a XML konfigurace Vstup tohoto programu je vytvořen tak, že mezi přepínačem a hodnotami jednotlivých parametrů není prázdná mezera. Parametr pro volbu operace se definuje jako `operation=[operace]`, vstupní soubor pomocí přepínače `input=[soubor]`, výstupní soubor jako `output=[soubor]`. Ostatní parametry jsou definovány pomocí přepínače, který má tvar `[název]=[hodnota]`. Na pořadí jednotlivých parametrů nezáleží.

Ukázka XML konfigurace pro jednu z operací tohoto program je následující (bez atributů `name` a `description`):

```
...
<program exec="basics.exe" path="../data/">
  <operations>
    <operation switch="operation=resize">
      <param switch="input=" type="input" />
      <param switch="output=" type="output" />
      <param switch="width=" type="slider" default="800" from="480"
        to="1600" step="1" />
      <param switch="height=" type="slider" default="600" from="480"
        to="1600" step="1" />
    </operation>
    ...
  </operations>
</program>
...
```

Program *edge*

Program, který obsahuje implementaci hranových detektorů, které hledají hrany pomocí první derivace jasové funkce. Mezi detektory v tomto programu patří Sobelův, Robertsův, Prewittové, Robinsonův, Kirschův a Frei-Chenův hranový detektor. Algoritmus pro všechny detektory je stejný a mění se akorát hodnoty matic `matrixX` a `matrixY`.

```
//konvoluce
for(y = 0; y < height; y++) { //cyklus vysky obrazku
 for(x = 0; x < width; x++) { //cyklus sirky obrazku
  // vypocet pixelu pro danou pozici
  outX = outY = out = 0;
  for(j = 0; j<3; j++) { //cyklus vysky matice
    for(i = 0; i<3; i++) { //cyklus sirky matice
     //testovani hranic obrazku
     if((x-1+i) < 0 || (y-1+j) < 0 || (x-1+i) > width-1 || (y-1+j) > height-1)
      value = 0;
     else value = data[(y-1+j)*step+(x-1+i)];

     //vlastni vypocet
     outX += value * matrixX[i][j];
     outY += value * matrixY[i][j];
     out = (abs(outX) + abs(outY))/2;
    }
   }
   if(out > 255) out = 255;
   else if(out < 0) out = 0;

   out_data[y*step+x] = out; //zapis do vystupu
  }
 }
```

Vstup a XML konfigurace Formát vstupu tohoto programu je nastaven tak, že obsahuje pouze jednu operaci, takže nepotřebuje přepínač pro nastavení této operace. Název vstupního souboru se definuje přepínačem `-input [soubor]` a výstupní soubor se definuje přepínačem `-output [soubor]`. K určení typu hranového detektoru slouží přepínač `-type [číslo]`. U jednotlivých parametrů operace nezáleží na pořadí.

Ukázka XML konfigurace pro tento program je následující (bez atributů `name` a `description`):

```
...
<program exec="edge.exe" path="../data/">
  <operations>
    <operation switch="null">
      <param switch="-input " type="input" />
      <param switch="-output " type="output" />
      <param switch="-type " type="select" default="1"
        array="1=Sobel#2=Roberts#3=Prewwit#4=Robinson#5=Kirsch#6=Frei-Chen" />
    </operation>
  </operations>
</program>
...
```

Program *edge2*

Tento program obsahuje funkce pro hranovou detekci pomocí Sobelova a Cannyho hranového detektoru a také Houghovu transformaci. Pro jednotlivé operace byly použity funkce z knihovny OpenCV. Sobelův hranový detektor používá funkci `cvSobel`, Cannyho hranový detektor funkci `cvCanny` a Houghova transformace je realizována za pomoci funkce `cvHoughLines2`.

Vstup a XML konfigurace Formát vstupu programu *edge2* je vytvořen bez přepínačů pro jednotlivé parametry. Vstup má pouze přepínač pro zvolení operace, a to pro každou operaci unikátní ve tvaru `-[operace]`. Ostatní parametry operace jsou již uvedeny jako samostatné hodnoty. Jednotlivé parametry mají předem určené pořadí zadávání. Všechny atributy kromě atributů pro vstupní a výstupní soubor jsou typu `slider`.

Ukázka XML konfigurace pro jednu z operací tohoto programu je následující (bez atributů `description` a některých atributů `name`):

```
...
<program exec="edge2.exe" path="../data/">
  <operations>
    <operation name="HoughCV" switch="textcolorblue-hough">
      <param switch="null" type="input" />
      <param switch="null" type="output" />
      <param name="Rho" switch="null" type="slider" default="1" from="1"
        to="5" step="0.1" />
      <param name="Theta" switch="null" type="slider" default="180" from="0"
        to="360" step="10" />
      <param name="Práh" switch="null" type="slider" default="100" from="10"
        to="200" step="10" />
```

```
    </operation>
    ...
  </operations>
</program>
...
```

5.5 Nasazení systému

Vývoj a testování systému probíhal v rámci dvou prostředí. První přostředí je lokální síť, kde byly všechny prvky systému umístěny na lokálním serveru. Druhé prostředí je síť internetu, kdy došlo k rozdělení jednotlivých částí systému na dva servery.

Nasazení v rámci lokálního serveru

Hlavní vývoj systému probíhal na serveru Apache verze 2.2.11, který byl nainstalován lokálně. Na tomto serveru běží PHP verze 5.3.0 a MySQL verze 5.1.36.

Všechny části systému byly umístěny v různých složkách, které však byly vzájemně dostupné v rámci serveru Apache.

Nasazení v rámci internetu

Řídící rozhraní bylo umístěno na server dostupný na internové adrese `http://dp.offlajn.net`. Na serveru běží PHP ve verzi 5.2.12 a databáze MySQL ve verzi 5.1.41-log.

Daemon byl umístěn na školní server `merlin.fit.vutbr.cz`, na kterém je nainstalován systém CentOS 5.4 64bit Linux a PHP ve verzi 5.1.6. Na tomto serveru je také datové úložiště, které je z řídícího rozhraní dostupné přes adresu `http://www.stud.fit.vutbr.cz/~xberan25/data/`.

V případě nasazení systému v rámci internetu bylo potřeba udělat některé změny v kódu oproti původní implementaci. Tyto změny souvisí zejména s nutností simulace společného datového úložiště tak, aby bylo dostupné jak z řídícího rozhraní, tak z daemona.

Načtení konfiguračního souboru Pro načtení konfiguračního souboru bylo potřeba nejdříve tento soubor získat z datového úložiště na serveru `merlin` a uložit jako dočasný soubor na serveru `offlajn.net`. Ke stažení tohoto souboru proto byla využita PHP knihovna `cURL`. A pro uložení souboru na disk bylo využito PHP funkcí pro manipulaci se soubory. Poté už bylo možné s konfiguračním XML souborem pracovat normálně a údaje o daemonovi zapsat do databáze.

Načtení seznamu obrázků Pro načtení seznamu dostupných obrázků byl na serveru `merlin` vytvořen SOAP server pracující jako jednoduchá webová služba poskytující funkci `getImageList()`. V kódu řídícího rozhraní pak byl vytvořen SOAP klient, který volá tuto funkci. K vytvoření této služby byla využita PHP knihovna `SOAP`.

5.6 Experimenty

Po nasazení systému byly prováděny různé experimenty, které měly odhalit chyby systému a otestovat jeho výkonnost. Tyto experimenty probíhali v rámci nasazení na lokálním serveru, protože školní server `merlin` neposkytuje dostatečný výkon pro jejich provádění.

Výkonnost systému

Výkonnost systému byla testována pomocí tří operací nad různým počtem zadaných úkolů[5]. Pro toto testování byly vybrány operace *Detekce hran*, *Převod barevného modelu* a *SobelCV*. Výběr operací proběhl tak, aby každá byla součástí jiného programu. Celkově proběhly dvě sady experimentů. První sada pracovala s úkoly, které na sebe navazovali, a tak bylo potřeba předávat si výstupní data mezi sebou. V druhé sadě experimentů se úkoly prováděly samostatně.

Nejdříve se testoval systém, kde byl připojen jeden daemon, který obsluhoval všechny programy. Poté byla obsluha programů rozdělena mezi dva daemony tak, aby první a třetí operaci prováděl jeden daemon a prostřední operaci prováděl druhý daemon. Nakonec byl přidán třetí daemon a každý program byl obsluhován samostatným daemonem.

Tabulka jednotlivých časů při zpracování úkolů zřetězeně:

Počet obrázků	**100**	**200**	**300**	**400**	**500**
Počet úkolů	**300**	**600**	**900**	**1200**	**1500**
1 daemon	1:32	3:06	5:10	7:44	8:28
2 daemoni	1:23	2:57	4:37	6:06	7:00
3 daemoni	1:20	2:37	3:51	5:24	6:39

Tabulka jednotlivých časů při zpracování úkolů samostatně:

Počet obrázků	**100**	**200**	**300**	**400**	**500**
Počet úkolů	**300**	**600**	**900**	**1200**	**1500**
1 daemon	1:26	2:51	5:05	7:25	8:14
2 daemoni	1:11	2:25	3:32	4:51	5:54
3 daemoni	0:57	1:58	2:43	4:14	4:56

Z výsledků jde vidět zrychlení provádění operací pokud je zapojeno do systému více daemonů. Rozdíl v časech mezi prováděním zřetězených operací a samostatných operací je největší u systému se třemi daemony, kdy je tento rozdíl způsoben nutností čekat na výsledky předchozí operace. Naopak u systému s jedním daemonem není rozdíl téměř žádný, protože všechny operace provádí pouze jeden daemon.

Testování prohlížečů

Uživatelské rozhraní bylo testováno v různých prohlížečích. Zkoumalo se především, zda je rozhraní dobře navrženo a zda funguje pod různými prohlížeči stejně. Prohlížeče, které byly testovány jsou:

[5] Obrázky pro provedení experimentů byly převzaty z http://www.cs.washington.edu/research/imagedatabase/

- *Google Chrome 4.1*
- *Internet Explorer 8.0*
- *Mozilla Firefox 3.6.3*
- *Opera 10.53*
- *Safari 4.0.5*

Jediný problém byl u prohlížeče *Internet Explorer 8.0*, kde správně nefunguje plugin `jsTree`. Ve všech ostatních prohlížečích je funkčnost dynamických prvků stejná a webové rozhraní se liší pouze drobnými rozdíly v zobrazení některých stránek (např. jiné odřádkování v dialogu pro výběr parametrů operace).

Kapitola 6

Uživatelské testování

Následující kapitola se zabývá uživatelským testováním implementovaného systému. Testovat se bude zejména webové rozhraní, a to z pohledu uživatelské jednoduchosti a přístupnosti. Celé testování spočívá v tom, že vybraní uživatelé vypracují předpřipravené úkoly, týkající se práce s webovým rozhraním. Po splnění každého z testových úkolů zodpoví uživatelé několik dotazů, pro poskytnutí zpětné vazby.

6.1 Navržené testy

Testy byly navrženy tak, aby postupně zvyšovaly náročnost na využití funkcí aplikace a aby otestovaly všechny ovládací prvky, které jsou ve webovém rozhraní implementovány.

Test č. 1 - Základní práce s webovým rozhraním

První test má za úkol otestovat základní funkčnost webového rozhraní. Uživatel je testovnán na schopnost ovládání rozhraní bez předešlé znalosti tohoto rozhraní. Test se skládá z následujících kroků:

1. Zaregistrujte se jako nový uživatel a přihlaste se do systému.
2. Vytvořte nový úkol, který změní rozlišení obrázku `obrazek.jpg` ze složky `test1` tak, aby jeho rozměry byly poloviční.
3. Zobrazte detail právě vytvořeného úkolu a zjistěte, jak dlouho se prováděl tento úkol.

Po provedení zadaných kroků uživatelé zodpoví následující otázky:

- Který krok Vám činil největší potíže?
- U kolika kroků jste potřeboval poradit?
- Jak těžké pro Vás bylo zorientovat se v navrženém prostředí? (1 (nejtěžší) - 10 (nejlehčí))
- Jak dlouho se prováděl Vámi zadaný úkol?

Test č. 2 - Pokročilá práce s vytvářením úkolů

Druhý test se zaměřuje zejména na stránku s tvorbou úkolů a testuje pokročilou funkčnost této stránky. Testování je zaměřeno na ovládací prvky umožnující výběr jednotlivých operací a obrázků. Test se skládá z následujících kroků:

1. Přepněte se na stránku s tvorbou úkolů a vyberte všechny obrázky ze složky `test2`.
2. Vyberte operaci *Převod barevného modelu* a nastavte libovolný barevný model.
3. Jako další vyberte operaci *Změna velikosti* a zadejte libovolné rozměry.
4. Jako poslední vyberte operaci *CannyCV* a zvolte libovolné parametry.
5. Prohoďte operace *Převod barevného modelu* a *Změna velikosti*.
6. Upravte operaci *Změna velikosti* a nastavte rozměry na 1024x768.
7. Nastavte provádění operací *Za sebou* a vytvořte zadané úkoly.
8. Vyčkejte na provedení všech úkolů a přepněte se na stránku s tvorbou úkolů. Vyberte všechny obrázky ze složky `output`. (Po tomto kroku bude pozastaven skript daemona)
9. Vyberte 3 libovolné operace.
10. Odstraňte 1 libovolnou operaci.
11. Nastavte provádění operací *Samostatně* a vytvořte zadané úkoly.

Po provedení zadaných kroků uživatelé zodpoví následující otázky:

- Který krok Vám činil největší potíže?
- U kolika kroků jste potřeboval poradit?
- Změnil byste ovládací prvek pro volbu operací? (Pokud ano, tak proč a jak?)
- Změnil byste ovládací prvek pro volbu obrázků? (Pokud ano, tak proč a jak?)

Test č. 3 - Práce se seznamem úkolů a detaily úkolů

Tento test se zaměřuje na práci se seznamem úkolů a na získávání informací o zadaných úkolech a jejich provádění. Test se zaměřuje na ovládání tabulky se seznamem prvků, její filtrování a na ovládání seznamu souvisejících úkolů a navigaci mezi souvisejícími úkoly. Test se skládá z následujících kroků:

1. Přepněte se na stránku se seznamem úkolů.
2. Za pomoci filtru zobrazte všechny úkoly, nad kterými byla provedena jedna z Vašich operací.
3. Za pomoci filtru zobrazte všechny úkoly, nad kterými byla provedena jedna z Vašich operací a jejich stav je *Čeká na daemona*.
4. Seřaďte úkoly podle jména Výstupu vzestupně (A-Z).

5. Vyberte úkol, který je třetí v pořadí a zobrazte jeho detail.
6. Za pomoci seznamu Souvisejících úkolů zjistěte kolik úkolů bylo provedeno nad stejným vstupním obrázkem.

Po provedení zadaných kroků uživatelé zodpoví následující otázky:

- Který krok Vám činil největší potíže?
- U kolika kroků jste potřeboval poradit?
- Změnil byste ovládací prvek pro seznam úkolů? (Pokud ano, tak proč a jak?)
- Změnil byste ovládací prvek pro orientaci mezi souvisejícími úkoly? (Pokud ano, tak proč a jak?)
- Kolik úkolů bylo provedeno nad stejným vstupním obrázkem?

Test č. 4 - Práce se seznamem uživatelů

Poslední test je zaměřen na vyzkoušení ostatních funkcí systému, které se týkají zejména správy uživatelů. Test se skládá z následujících kroků:

1. Najeďte na stránku umožňující změnu osobních údajů a změnte si své heslo.
2. Odhlašte se ze systému.
3. Přihlašte se do systému jako uživatel `admin` s heslem `heslo`.
4. Zobrazte seznam uživatelů a vyberte uživatele s nejmenším počtem úkolů.
5. Nastavte tomuto uživateli práva administrátora.
6. Zobrazte seznam uživatelů a smažte Vámi vytvořeného uživatele.
7. Odhlašte se ze systému.

Po provedení zadaných kroků uživatelé zodpoví následující otázky:

- Který krok Vám činil největší potíže?
- U kolika kroků jste potřeboval poradit?

Závěrečný dotazník

Po absolvování všech testů vyplní uživatelé ještě závěrečný dotazník, který se týká jejich názoru na webové rozhraní a práci s ním:

- Jakou známkou byste ohodnotil ovládání rozhraní? (1 (nejhorší) - 10 (nejlepší))
- Jakou známkou byste ohodnotil vzhled rozhraní? (1 (nejhorší) - 10 (nejlepší))
- Jak těžké pro Vás bylo pochopit práci s jednotlivými ovládacími prvky? (1 (nejtěžší) - 10 (nejlehčí))

- Bylo rozložení jednotlivých ovládacích prvků vyhovující?
- Co se Vám na rozhraní líbilo nejvíce?
- Co se Vám na rozhraní líbilo nejméně?
- Co Vám v rozhraní chybí?
- Využíváte nějaký program pro zpracování obrazu, který se ovládá pomocí příkazové řádky.
- Využíváte nějaký program, který se ovládá pomocí příkazové řádky.
- Bylo by pro Vás toho rozhraní využitelné, když byste si mohl přidat vlastní programy?

6.2 Průběh testování

Testování probíhalo na systému nasazeném v rámci lokálního serveru, a to hlavně z důvodu lepšího výkonu, než při nasazení v internetu. Dále také kvůli předpřipraveným datům k jednotlivým testům, aby podmínky pro jednotlivé testy byly pro všechny uživatele stejné. Testování rozhraní probíhalo v internetovém prohlížeči *Mozilla Firefox 3.6.3*.

Výběr uživatelů

Výběr uživatelů na testování webového rozhraní neprobíhal náhodnou formou. Autor vybíral uživatele tak, aby byl zastoupen co největší vzorek uživatelů s různým stupněm počítačové gramotnosti. Díky tomu mohlo být získáno více různých pohledů a názorů na aplikaci, a to hlavně z hlediska uživatelské přístupnosti.

Co se hodnotí

U každého uživatele jsou měřeny časy, za které se jim podaří splnit jednotlivé testy. Čas se začíná počítat od prvního vstoupení na stránku až po dokončení posledního bodu testu. Dále se zaznamenávají odpovědi na jednotlivé otázky z dotazníků a také se zaznamenávají dotazy uživatelů na práci s prostředím během provádění testů.

Co bylo uživatelům vysvětleno

Uživatelé před začátkem testování obdrželi seznam úkolů a byla jim poskytnuta základní informace o účelu webového rozhraní:

"*Webové rozhraní, které budete testovat, je součástí systému, který provádí zadané operace nad obrazovými daty. Toto rozhraní umožnuje definovat parametry těchto operací a vybírat data, nad kterými se mají provést.*"

Práce s rozhraním uživatelům nebyla vysvětlena hlavně z důvodu, aby se zjistilo, jak moc je ovládání tohoto rozhraní pochopitelné a intuitivní. Během jednotlivých testů měli uživatelé možnost ptát se autora v případě, že nevěděli jak postupovat dále.

6.3 Vyhodnocení testů

Z naměřených výsledků byl u jednotlivých testů spočítán průměrný čas provádění, který byl porovnán s předpokládaným časem. Předpokádaný čas byl určen jako čas, za který test provedl autor + 2 minuty.

Textové odpovědi a připomínky uživatelů byly zpracovány ručne a byly z nich vybrány nejčastější a nejzajímavější odpovědi.

Test č. 1

- Předpokládaný čas: **2:45**
- Průměrný naměřený čas: **3:16**

V tomto úkolu dělalo uživatelům největší problém se narychlo zorientovat v navrženém prostředí a chtěli si ho nejdříve pořádně prohlédnout. S prvním krokem (registrace a přihlášení) němel nikdo z uživatelů problém. U druhého kroku (vytvoření jednoduchého úkolu) nevěděli někteří uživatelé, kde začít se samotným vytvářením úkolu. Dále pak uživatelům nejčastěji činilo potíže najít způsob přidávání operace. Nejdříve na operaci klikali, teprve poté zkoušeli jiné možnosti. Pokud operaci chtěli přenést, tak někteří nevěděli kam ji mají umístit. Výběr obrázku uživatelům problém nečinil.

Otázka, kdy měli uživatelé zhodnotit, jak těžké pro ně bylo se v navrženém prostředí zorientovat, získala průměrné hodnocení 5,5 bodů. Hodnocení je způsobeno zejména kvůli problému při výběru operace, kdy jim hned od počátku nebylo jasné, jak operaci vybrat.

Test č. 2

- Předpokládaný čas: **3:15**
- Průměrný naměřený čas: **4:06**

U druhého testu již uživatelé s vytvářením úkolu problém neměli. Řazení operací pod sebe jim přislo logické a výběr obrázku byl také v pořádku. Editace a mazání operací uživatelům vyhovovalo.

U otázky, zda by uživatelé změnili ovládací prvek pro výběr operací, se jich nejvíce shodlo na odpovědi, že by místo funkce *Drag&Drop* a přesouvání úkolů mezi seznamy, uvítali možnost přidávat operace kliknutím. Přesouvání vybraných operací mezi sebou by již nechali tak, jak je.

Ovládací prvek pro výběr obrázků by většina uživatelů neměnila. U uživatelů, kteří by měnili, se jednalo pouze drobnosti (např. změnit ikonku pro otevření složky na +).

Test č. 3

- Předpokládaný čas: **2:30**
- Průměrný naměřený čas: **2:58**

Třetí test uživatelům nedělal žádné větší potíže. Pouze některým bylo potřeba nejdříve vysvětlit, co přesně znamená a k čemu slouží seznam souvisejících úkolů.

Tabulku se seznamem úkolů by uživatelé neměnili, pouze někteří navrhli, že by bylo dobré ukazovat více informací (např. i náhledy vstupu a výstupu) již po najetí nad řádek s úkolem.

Seznam souvisejících úkolů by žádný z uživatelů neměnil, pouze někteří by opět uvítali více informací po najetí myši nad vybraný úkol.

Test č. 4

- Předpokládaný čas: **2:30**
- Průměrný naměřený čas: **1:07**

S tímto testem neměl nikdo z uživatelů problémy a všichni se shodli na tom, že správa uživatelů je vyřešena dobrým způsobem.

Dotazník

V závěrečném dotazníku hodnotili uživatelé ovládaní rozhraní průměrnou známkou 8,25. Vzhled rozhraní pak hodnotili průměrnou známkou 8,75. Obtížnost pochopení práce s jednotlivými prvky ohodnotili uživatelé průměrnou známkou 4,5. Rozdíl prvního a třetího hodnocení ukazuje, že po vysvětlení práce s rozhraním již bylo ovládání aplikace bez problémů. Uživatelé byli po absolvování testů schopni s rozhraním pracovat bez větších obtíží. Všichni uživatelé se shodli na tom, že umístění jednotlivých ovládacích prvků bylo vyhovující a neměnili by jej.

Nejčastěji se uživatelům líbilo webové rozhraní jako celek a pak některé jeho jednotlivé funkce (např. náhledy obrázků u vytváření úkolu, dynamické menu na úvodní stránce nebo dialog pro výběr parametrů operací).

Uživatelům se naopak nejméně líbila volba operací za pomoci přetahování mezi dvěma seznamy. Většina testovaných uživatelů by uvítala možnost přidávat operace pomocí nějakého tlačítka, nebo by alespoň zvýraznili plochu, kam se mají operace umístit.

Uživatelé se shodli na tom, že jim žádný výrazný prvek v rozhraní nechybí. Jednomu z uživatelů by se líbila možnost použít program i pro jiný druh dat než obrazová data.

Poslední tři otázky měly dvě různé skupiny odpovědí. První skupina uživatelů (běžní uživatelé počítače) žádné podobné programy nepoužívá. Druhá skupina (studenti informatiky a programátoři) používá buď programy zpracující obrazová data, a nebo programy, které je možné ovládat za pomoci příkazové řádky. První skupina by pro rozhraní využití nenašla. Druhá skupina by si práci s rozhraním po menších úpravách dokázala představit a někteří dokonce projevili zájem o zdrojové kódy aplikace.

Vyhodnocení

Z provedených testů vyplývá většinová spokojenost uživatelů s ovládáním a vzhledem testovaného rozhraní. Intuitivnost při ovládání je o něco horší, ale po vysvětlení již nebylo ovládání jednotlivých prvků problém. Tento fakt by se dal vylepšit umístěním nápovědy k jednotlivým prvkům přímo do rozhraní, nebo vytvořením manuálu k ovládání rozhraní.

Během prvního testování narazil testovaný uživatel na několik chyb v kódu, které musely být během testu opraveny a test musel být anulován a proveden znovu. Jednalo se o chyby související zejména se zabezpečením registračního formuláře proti nepovoleným znakům a chybně nastavené oprávnění stránky s editací uživatele.

Kapitola 7

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat systém s webovým rozhraním, který umožní uživateli definovat operace nad obrazem, měnit parametry operací, provede zadané operace a bude průběžně zobrazovat výsledky zadaných úkolů uživateli. První část práce spočívala v nastudování teoretických podkladů pro návrh tohoto systému. Tyto podklady jsou v kapitole 2 a týkají se tří oblastí informačních technologií - distribuovaných systému, zpracování obrazu a webových technologií.

Hlavní část práce se věnuje samotnému návrhu a implementaci systému. Návrh je nejdříve zaměřen na obecný návrh systému (kapitola 3), kde je uvedeno několik možných propojení jednotlivých prvků systému. Po navrhnutí optimálního řešení je v kapitole 4 popsána podrobná specifikace tohoto finálního návrhu. Kapitola 5 se zabývá popisem implementace jednotlivých prvků systému a v závěru této kapitoly jsou uvedeny dva případy nasazení systému a experimenty zkoumající výkon systému. Pro ukázku zpracování obrazu byly využity zejména programy provádějící detekci hran v obraze.

V poslední části práce (kapitola 6) jsou pak provedeny testy webového rozhraní za pomoci uživatelů. V těchto testech byla zjišťována zejména jednoduchost, pochopitelnost a uživatelská přístupnost webového rozhraní.

Celý systém byl navržen a implementován zcela samostatně. Volba jednotlivých částí systému a jejich propojení jsou založeny zejména na zkušenostech autora práce a na nastudovaných informacích. Je možné, že existuje podobný systém založený na stejných nebo podobných principech. Při návrhu a implementaci však žádný takový systém nebyl zkopírován nebo napodoben.

Možných pokračování práce do budoucna je několik. Je možné systém rozšířit o podporu více typů dat a tím z něj udělat univerzální systém pro obsluhu programů, které se běžně ovládají pouze za pomoci příkazové řádky. Systém dovoluje libovolnou změnu uživatelského rozhraní, a tak lze přidat lepší podporu pro vytváření a nastavování potřebných parametrů. Další možností je rozšířit systém o podporu daemonů, kteří dovolují paralelní provádění operací, a zvýšit tak výkonnost celého systému.

Hlavním nedostatkem práce je málo experimentů s možným využitím systému v rámci sítě. Experimenty se systémem probíhaly zejména v rámci lokálního serveru. V rámci internetu byl využit pouze server, který neposkytl dostatečný výkon pro potřebné otestování systému. Autor také neměl možnost vyzkoušet práci se systémem, který by disponoval větším množstím daemonů umístěných na více různých serverech internetu.

Přínos práce je v samotném návrhu systému. Systém by po drobných úpravách bylo možné nasadit například i pro vědecké účely nebo komerční využití. Uživatelé by měli možnost využívat programů a zejména výpočetního výkonu, ke kterým jinak nemusí mít přístup.

Literatura

[1] Borek Bernard: Rich Internet Applications v roce 2008 [online]. http://interval.cz/clanky/rich-internet-applications-v-roce-2008/, 2008-04-25 [cit. 2010-05-05].

[2] Darcy DiNucci: Fragmented Future [online]. http://www.cdinucci.com/Darcy2/articles/Print/Printarticle7.html, 2009 [cit. 2010-05-06].

[3] Jan Ambrož: Web 2.0: bublina, nebo nový směr webu [online]. http://www.lupa.cz/clanky/web-2-0-bublina-nebo-novy-smer-webu/, 2007-04-27 [cit. 2010-05-06].

[4] Jan Beránek: Metody detekce a reprezentace hran v obraze [online]. http://www.fit.vutbr.cz/study/DP/all.php?id=3045, 2007-05-14 [cit. 2010-05-09].

[5] Lacko, L.: *Ajax - Hotová řešení.* Computer Press, 2008, iSBN 978-80-251-2108-5.

[6] M. Španěl; V. Beran: Obrazové segmentační techniky - Přehled existujících metod [online]. http://www.fit.vutbr.cz/~spanel/segmentace, 2006-01-19 [cit. 2010-05-09].

[7] Martin Cvrček: Metody detekce a reprezentace hran v obraze [online]. http://www.onlio.com/clanky/web_3.0.html, 2008-10-08 [cit. 2010-05-11].

[8] Roman Pichlík: Rich Internet Application [online]. http://interval.cz/clanky/rich-internet-application/, 2005-06-14 [cit. 2010-05-05].

[9] Václavek, P.: *JavaScript - Hotová řešení.* Computer Press, 2003, iSBN 80-7226-854-6.

[10] WWW stránky: Web Services Architecture [online]. http://www.w3.org/TR/ws-arch/, 2004-02-11 [cit. 2010-05-05].

[11] WWW stránky: Houghova transformace [online]. http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm, 2008-10-24 [cit. 2010-05-05].

[12] WWW stránky: Multiple document interface - Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/wiki/Multiple_document_interface, 2010-03-04 [cit. 2010-05-05].

[13] WWW stránky: Service-oriented architecture - Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/wiki/Service-oriented_architecture, 2010-04-28 [cit. 2010-05-05].

[14] WWW stránky: Web service - Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/wiki/Web_service, 2010-04-28 [cit. 2010-05-05].

[15] WWW stránky: Cloud computing - Wikipedie, otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/Cloud_computing, 2010-05-04 [cit. 2010-05-06].

[16] WWW stránky: jQuery: The Write Less, Do More, JavaScript Library [online]. http://jquery.com/, [cit. 2009-12-30].

[17] WWW stránky: jQuery UI [online]. http://jqueryui.com/, [cit. 2009-12-30].

[18] WWW stránky: Welcome - OpenCV Wiki [online]. http://opencv.willowgarage.com/wiki/, [cit. 2010-01-05].

[19] WWW stránky: Cloud computing - Abakowiki [online]. http://cloud.abakowiki.cz/xwiki/bin/view/Main/WebHome?language=cs, [cit. 2010-05-06].

[20] WWW stránky: Cloud computing [online]. http://www.cloudcomputing.cz/, [cit. 2010-05-06].

[21] WWW stránky: Adobe Flex [online]. http://www.adobe.com/products/flex/, [cit. 2010-05-13].

[22] WWW stránky: JavaFX [online]. http://javafx.com/, [cit. 2010-05-13].

[23] WWW stránky: Microsoft Silverlight [online]. http://www.silverlight.net/, [cit. 2010-05-13].

Příloha A

CD se zdrojovými soubory a textem práce

Obsah jednotlivých adresářů:

- `daemon` - Zdrojové kódy a konfigurační XML soubor daemona.
- `databaze` - SQL skript pro inicializaci databáze.
- `experimenty` - Sady souborů a damonů využité při provádění experimentů.
- `nasazeni_localhost` - Systém nasazený na lokálním serveru.
- `nasazeni_internet` - Systém nasazený na internetu.
- `plakat` - Plakát prezentující práci.
- `programy` - Zdrojové kódy programů pro zpracování obrazu.
- `ridici_rozhrani` - Zdrojové kódy řídícího rozhraní.
- `testy` - Testy, dotazníky a testovaci sada pro uživatelské testování webového rozhraní.
- `text_prace` - Text diplomové práce.

Příloha B

Návod ke zprovoznění systému

1. Vytvořte novou databázi a naimportujte do ní SQL skript z adresáře \`databaze`\.
2. Na webový server zkopírujte zdrojové kódy řídícího rozhraní ze složky \`ridici_rozhrani`\.
3. V souboru \`core`\`constants.php` nastavte údaje pro připojení k databázi a cestu k datovému úložišti.
4. Na výkonný server nahrajte programy pro zpracování obrazu ze složky \`programy`\.
5. Na výkonný server nahrajte daemona ze složky \`daemon`\ a konfigurační soubor daemona nahrajte do složky `xml` na datovém úložišti.
6. Ve zdrojovém kódu daemona nastavte správné údaje o adresách serverů, port a název konfiguračního souboru.
7. V konfiguračním souboru nastavte správné údaje o adrese výkonného serveru, port a cesty k jednotlivým programům pro zpracování obrazu.

Příloha C

Ukázka konfiguračního XML souboru daemona

```
<?xml version="1.0" encoding="utf-8"?>
<daemon>
 <name>Daemon</name>
 <host>127.0.0.1</host>
 <port>4666</port>
 <description>Daemon na lokálním serveru</description>
 <programs>
  <program name="Edge 2" exec="edge2.exe" description="Pokročilé metody hranové
      detekce a transformace" path="../data/">
   <operations>
    <operation name="HoughCV" switch="-hough" description="Houghova transformace
        - OpenCV funkce">
     <param name="Vstup" switch="null" description="Vstupní soubor" type="input" />
     <param name="Výstup" switch="null" description="Výstupní soubor" type="output" />
     <param name="Rho" switch="null" description="Vzdálenost" type="slider"
         default="1" from="1" to="5" step="0.1" />
     <param name="Theta" switch="null" description="Úhel" type="slider"
         default="180" from="0" to="360" step="10" />
     <param name="Práh" switch="null" description="Práh pro transformaci. Přímka
         je vrácena pokud je hodnota v akumulátoru větší než práh." type="slider"
         default="100" from="10" to="200" step="10" />
    </operation>
    <operation name="SobelCV" switch="-sobel" description="Sobelův hranový detektor
        - OpenCV funkce">
     <param name="Vstup" switch="null" description="Vstupní soubor" type="input" />
     <param name="Výstup" switch="null" description="Výstupní soubor" type="output" />
     <param name="Derivace podle" switch="null" description="Derivace podle x nebo
         podle y" type="select" default="0" array="0=Podle x#1=Podle y" />
     <param name="Velikost matice" switch="null" description="Velikost matice pro
         provedení hranové detekce" type="select" default="3" array="3=3#5=5#7=7" />
    </operation>
    <operation name="CannyCV" switch="-canny" description="Cannyho hranový detektor
        - OpenCV funkce">
     <param name="Vstup" switch="null" description="Vstupní soubor" type="input" />
     <param name="Výstup" switch="null" description="Výstupní soubor" type="output" />
     <param name="První práh" switch="null" description="Spodní práh pro detekci"
         type="slider" default="50" from="0" to="100" step="1" />
     <param name="Druhý práh" switch="null" description="Horní práh pro detekci"
         type="slider" default="200" from="101" to="300" step="1" />
     <param name="Velikost matice" switch="null" description="Velikost matice pro
```

```
          provedení hranové detekce" type="select" default="3" array="3=3#5=5#7=7" />
    </operation>
   </operations>
  </program>
  <program name="Basics" exec="basics.exe" description="Základní operace"
      path="../data/">
   <operations>
    <operation name="Změna velikosti" switch="operation=resize " description="Změna
        velikosti obrazu">
     <param name="Vstup" switch="input=" description="Vstupní soubor" type="input" />
     <param name="Výsledek" switch="output=" description="Výsledný soubor"
          type="output" />
     <param name="Šířka" switch="width=" description="Šířka obrázku" type="slider"
          default="800" from="16" to="1600" step="1" />
     <param name="Výška" switch="height=" description="Výška obrázku" type="slider"
          default="600" from="16" to="1600" step="1" />
    </operation>
    <operation name="Převod barevného modelu" switch="operation=color "
          description="Převod barevného modelu z RGB na jiný">
     <param name="Vstup" switch="input=" description="Vstupní soubor" type="input" />
     <param name="Výsledek" switch="output=" description="Výsledný soubor"
          type="output" />
     <param name="Barevný model" switch="type=" description="Výsledný barevný model"
          type="select" default="0" array="0=Stupně šedi#1=HSV#2=CIE XYZ#3=YCrCb
          #4=HLS#5=CIE L*a*b*#6=CIE L*u*v*" />
    </operation>
   </operations>
  </program>
  <program name="Detekce hran" exec="edge.exe" description="Hranové detektory
      využívající první derivaci jasové funkce" path="../data/">
   <operations>
    <operation name="Detekce hran" switch="null" description="Detekce hran">
     <param name="Vstup" switch="-input " description="Vstupní soubor" type="input" />
     <param name="Výsledek" switch="-output " description="Výsledný soubor"
          type="output" />
     <param name="Typ detektoru" switch="-type " description="Typ hranového
          detektoru" type="select" default="1" array="1=Sobel#2=Roberts#3=Prewwit
          #4=Robinson#5=Kirsch#6=Frei-Chen" />
    </operation>
   </operations>
  </program>
 </programs>
</daemon>
```