VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

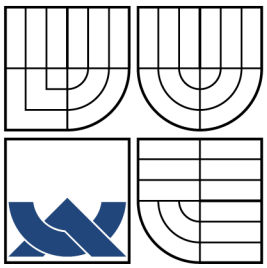
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

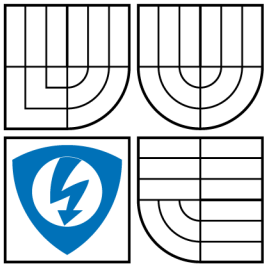
WEBOVÁ APLIKACE NA PLATFORMĚ JAVA PRO ODEVZDÁVÁNÍ DAT A JEJICH VYHODNOCENÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVEL ZELENKA
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE NA PLATFORMĚ JAVA PRO ODEVZDÁVÁNÍ DAT A JEJICH VYHODNOCENÍ

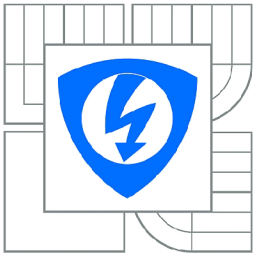
WEB APPLICATION ON JAVA PLATFORM FOR DATA SUBMISSION AND EVALUATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PAVEL ZELENKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM BURGET, Ph.D.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Pavel Zelenka ***ID:*** 143613
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Webová aplikace na platformě JAVA pro odevzdávání dat a jejich vyhodnocení

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s knihovnami na platformně JAVA a vzájemně srovnejte. Vytvořte jednoduchou aplikaci, kam se bude možné přihlásit a uploadovat data (např. soubor obrázků) a vyhodnotit je. Výsledky vyhodnocení nechte zobrazit pro všechny přihlášené, aby mohli srovnat své výsledky.

DOPORUČENÁ LITERATURA:

[1] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
[2] Milan Sonka, Vaclav Hlavac, and Roger Boyle. Image Processing: Analysis and Machine Vision. CL-Engineering, 2 edition, September 1998.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Radim Burget, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se týká především využití programovacího jazyka Java pro tvorbu webové aplikace pro pořádání soutěží s biomedicínskými daty. Je zaměřena na zhodnocení a výběr nástrojů pro vývoj a provoz takovéto aplikace. Součástí práce je databázové schéma, zdrojové kódy aplikace a základní programátorská dokumentace.

KLÍČOVÁ SLOVA

Java, webová aplikace, framework, Spring, Apache Tomcat, MVT, cron, databáze

ABSTRACT

This project mainly refers to use of the Java programming language for developing web application for organizing challenges focused on biomedical data. The project is focused on the comparison and selection of tools for developing and running application on server. The project includes a database schema, source code of application and basic development documentation.

KEYWORDS

Java, web application, framework, Spring, Apache Tomcat, MVT, cron, database

ZELENKA, Pavel *Webová aplikace na platformě JAVA pro odevzdávání dat a jejich vyhodnocení*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 51 s. Vedoucí práce byl Ing. Radim Burget, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Webová aplikace na platformě JAVA pro odevzdávání dat a jejich vyhodnocení“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrálního projektu panu Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Tato práce a hlavně webová Java aplikace je podpůrnou součástí projektu „Brain Tissue Analysis“ [1], který je vedený výzkumnou skupinou „Signal Processing Laboratory“. Skupina SPLab [1] je součástí Vysokého učení technického v Brně a věnuje se základnímu a aplikovanému výzkumu zejména v oblastech zpracování hlasu, obrazů, videa a textu.

V současné době je známo více než 600 onemocnění postihující nervový systém. Současná zobrazovací zařízení jsou schopna zobrazit tkáň. Bohužel vyhodnocení údajů je časově náročné a často nepřesné. Brain Tissue Analysis je soutěžně–výzkumný projekt zabývající se procesem měření objemu mozku ze snímků magnetické rezonance. Aby bylo možné změřit objem, je nejdříve nutné z těchto snímků extrahovat samotný obrázek mozku.

Cílem pro soutěžící je tedy navrhnout a implementovat algoritmus, který nejlépe provede zpracování snímků z magnetické rezonance pacienta. Zpracováním snímku se v tomto případě rozumí transformace na obrázek, ve kterém je oblast mozku obarvena bílou barvou a zbytek je černý, viz příloha A.

Vyvinutá webová Java aplikace bude sloužit pro automatické vyhodnocení přesnosti algoritmu, která se měří porovnáním zaslaných výsledků od soutěžícího s manuálně připravenými výsledky od odborníka. Manuálně připravené výsledky a výsledky zpracování algoritmem od soutěžícího vycházejí ze stejné testovací množiny dat. Cílem obou projektů je snaha o co největší automatizaci zpracování dat.

Hlavním přínosem práce je vytvoření aplikace jako centrálního bodu, kde mohou soutěžící i administrátoři vidět pokrok všech prací a porovnávat výsledky mezi sebou. Použitím automatických prostředků pro zpracování je generována i úspora lidské práce.

V každé nové aplikaci je snaha používat nejnovější technologie a postupy. Občas se bohužel stane, že se nová technologie dlouho neudrží pro své nedostatky nebo obtížnost a meze při používání. Při výběru bylo dbáno i na tuto problematiku. Právě Java si vybudovala silné postavení na trhu. Dnes je nainstalovaná na milionech zařízení a to jí zajistí kvalitní rozvoj a dlouhé trvání. Příkladem další zajímavé, oblíbené a úspěšně používané technologie, která bude mít také jistě dlouhou trvanlivost, je Ajax.

Pro zajištění vyšší kvality výsledné práce bude aplikace podrobena testování několika nezávislých osob. Cílem tohoto testování bude vyzkoušení funkcí dostupných koncovému uživateli a kontrola jejich výstupů a chování. Nalezené chyby, které vývojář při návrhu nebo implementaci přehlédl, budou opraveny a bude provedeno

[1] http://splab.cz

opětovné přetestování opravy.

Úvodní kapitola se snaží stručně porovnat programovací jazyk Java s konkurenčními jazyky určenými k tvorbě webových aplikací.

Druhá kapitola této práce se zabývá porovnáním dostupných technologií a postupů při práci a následným výběrem těch nejvhodnějších, které budou použity při implementaci. Při výběru je zohledněno několik faktorů a jsou popsány důvody výběru právě těch použitých.

Ve třetí kapitole je přistoupeno k samotnému vývoji. Je zde popsána struktura aplikace z pohledu programátora na úrovni kódu i z pohledu koncového uživatele, kterého zajímají dostupné funkce a možnosti aplikace. Z programátorského hlediska je zde také uvedeno využití těch nejzajímavějších postupů cílených k dosažení pohodlného a intuitivního používání aplikace koncovým uživatelem.

Poslední kapitola se věnuje závěrečným činnostem, které zahrnují kompletní přípravu serveru a samotné zveřejnění výsledné aplikace. V této kapitole je také popsáno testování nezávislými osobami, pro dosažení určité kvality.

2 VÝVOJ WEBOVÝCH APLIKACÍ

V této kapitole jsou stručně shrnuty možnosti tvorby webových aplikací z pohledu použitého programovacího jazyka.

2.1 Skriptovací jazyk PHP

Stále nejvíce používaným programovacím jazykem webových aplikací zůstává PHP. Tento jazyk má dobře propracovanou dokumentaci [2] a často se vyučuje již na středních školách. Firmy poskytující hosting PHP aplikací nejčastěji nabízejí variantu tzv. LAMP serveru. Jedná se o množinu obsahující operační systém Linux, Apache v roli webového aplikačního serveru, MySQL jako databázový systém a PHP jako programovací jazyk. Jelikož se jedná o svobodný software, pořizovací náklady jsou nulové.

Některé prvky této množiny lze libovolně nahrazovat podobným řešením dle přání zákazníka. Místo databáze MySQL lze využít například PostgreSQL. Místo programovacího jazyka PHP využít jiné skriptovací jazyky, jako jsou například Perl nebo Python.

Alternativou k serveru LAMP je WAMP server. Toto řešení využívá místo open source operačního systému Linux komerční systém Windows od společnosti Microsoft.

Nevýhodou PHP je, že se jedná o skriptovací jazyk. Nedochází k překladu do binární podoby, ale jednotlivé příkazy jsou interpretované řádek po řádku. Tímto je jazyk připraven o možnost optimalizace a je tedy i pomalejší oproti kampilovaným jazykům.

2.2 Jazyk Java

Programovací jazyk Java je jedním z nejpoužívanějších jazyků. V mnoha případech se jedná o primární jazyk pro výuku objektově orientovaného programování. Pro vývoj webových aplikací se Java používá méně často než zmíněné PHP. Její největší výhodou, která navíc zvyšuje bezpečnost, je silná typová kontrola. Znamená to, že do proměnné nadefonované v určitém datovém typu není možné přiřadit hodnotu jiného typu. PHP sice také má možnosti kontroly datových typů, ale jazyk toto přímo nevyžaduje.

2.3 Jazyk Ruby

Ruby je méně známý skriptovací jazyk používaný pro tvorbu webových aplikací. Jedná se o silně objektový jazyk (vše je objekt) a používá jednoduchou syntaxi. Jelikož je tento jazyk skriptovací, je jeho nevýhodou, stejně jako to bylo u PHP, nižší rychlost oproti kompilovaným řešením.

Asi nejznámějším termínem v oblasti programování v jazyce Ruby je framework pro tvorbu webových aplikací nazvaný **Ruby on Rails** [3]. Tento framework používají světově známé společnosti Apple nebo The New York Times.

Jazyk Ruby stále čeká na větší popularitu mezi programátory a provozovateli webových hostingů. Na českém trhu zatím existuje pouze jeden specializovaný provozovatel hostingu s podporou Ruby on Rails.

3 POUŽITÉ TECHNOLOGIE

Tato kapitola je zaměřena na dostupné technologie a postupy a na důvody výběru použitých pro tento projekt. Při výběru je potřeba zohlednit náklady, licenční podmínky, kvalitu a množství dokumentace, rozšířenost i osobní preferenci a dosavadní znalost jednotlivých technologií a nástrojů. Nutností je i ověření kompatibility s ostatními nástroji a prostředky pro vývoj.

3.1 Java EE

Pro vývoj webových aplikací na platformě Java se používá edice Java Enterprise Edition [11]. Základem Java EE je stadardní edice Java SE [1], která obsahuje knihovny základních funkcí programovacího jazyka Java a definuje virtuální stroj (JVM) [2].

JVM [12] je softwarový balík, který zajišťuje spuštění Java programů. Tento virtuální stroj je dostupný na více počítačových platformách, například na více operačních systémech nebo na různém hardware (stolní počítač, mobilní zařízení, server). Tímto je docíleno stavu, kdy je Java označována za multiplatformní jazyk.

Ze zdrojového kódu v programovacím jazyce Java se překladem stane tzv. bytecode. Bytecode je předán virtuálnímu stroji a ten má na starosti interpretaci jednotlivých funkcí a komunikaci s operačním systémem. Celý tento postup je popsaný na obrázku 3.1

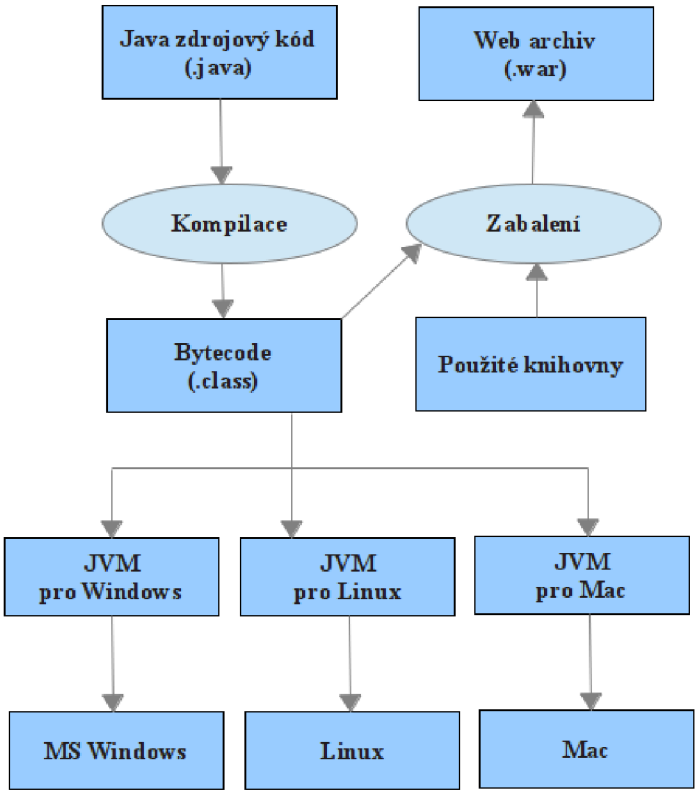
Další součástí edice Java EE jsou specifikace rozhraní pro vývoj webových aplikací. Například se jedná o Java Servlets, Java Server Pages [10], JavaBeans, ...

3.2 Vývojové prostředí

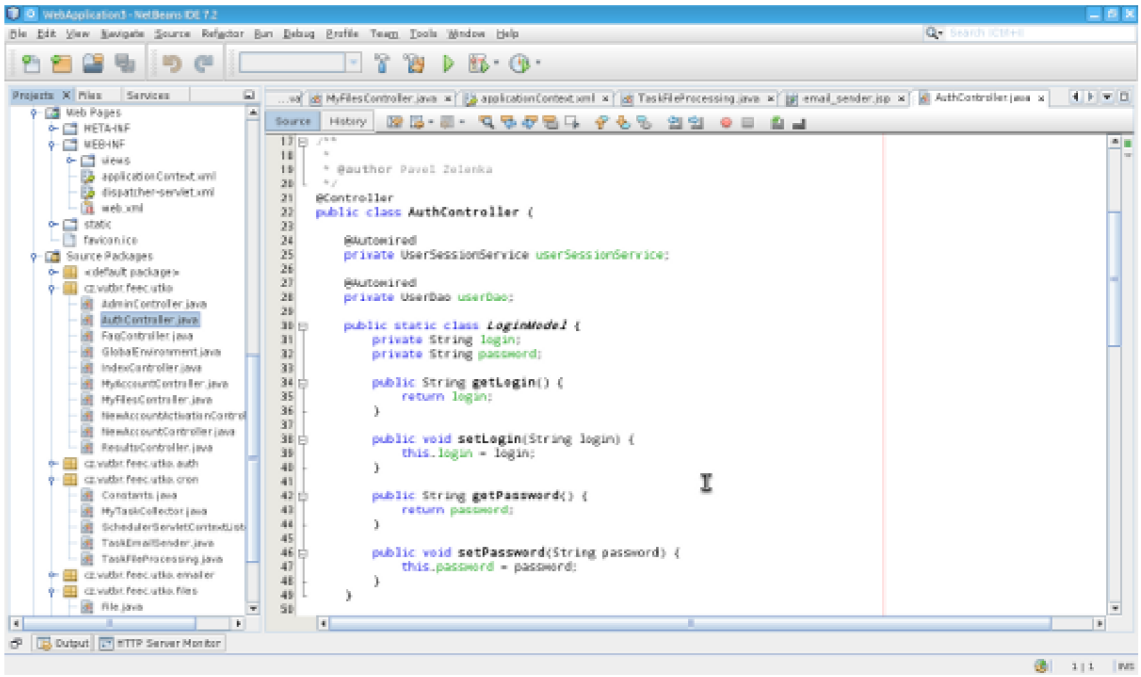
V dnešní době jsou programovací jazyky i vytvářené aplikace tak složité a rozsáhlé, že by bylo opravdu obtížné pracovat bez některého z dostupných vývojových prostředí, takzvaného IDE [3]. Tento software nám při psaní kódu kontroluje syntaxi i vztahy mezi jeho částmi a zároveň nám napovídá nebo poskytuje dokumentaci jazyka. Obsahuje také kompilátor, debugger pro snadné nalezení případných chyb a nástroje pro automatické testování a spouštění napsané aplikace. Pro programovací jazyk Java jsou nejpoužívanější Netbeans a Eclipse [13].

Pro dlouholeté zkušenosti s vývojem aplikací v různých jazycích v prostředí Netbeans je použit tento software i při vývoji této aplikace. Prostředí Netbeans je při vývoji zachyceno na obrázku 3.2

[1] Java Standard Edition
[2] Java Virtual Machine
[3] Integrated Development Environment



Obr. 3.1: Java virtuální stroj



Obr. 3.2: Vývojové prostředí Netbeans

3.3 Aplikační server

Aplikační server je vrstva mezi operačním systémem a webovou aplikací. V rámci tohoto projektu půjde o Java webovou aplikaci a Java webový (aplikační) server. Server poskytuje aplikaci možnosti, jak komunikovat s operačním systémem a jinými aplikacemi. Příkladem může být přístup k souborovému systému nebo spojení s databázovým serverem.

Prvním kandidátem je **JBossAS** [4] [5]. Jedná se o open source Java aplikační server od známé společnosti Red Hat. Pro jednodušší nastavení je k dispozici management konzole dostupná přes webové rozhraní. Tento server je využívaný hlavně pro velké komerční aplikace. Platící zákaníci mají v ceně i podporu.

Apache Tomcat [6] je také vyvíjen jako open source projekt a poskytován pod licencí Apache License verze 2. Je jednoduchý a nenáročný a dobře poslouží na menší projekty. Pro Tomcat je k dispozici základní management rozhraní (obrázek 3.3), kde může administrátor najít statistiky serveru a spravovat nahrané aplikace.

Dalším open source produktem v této kategorii je **GlassFish** [9]. Je podobný Tomcatu s tím rozdílem, že GlassFish má více možností nastavení a konfigurace přes webovou management konzoli. Je tedy i náročnější na zdroje.

Vedle open source projektů existují i některé komerční java aplikační servery. Pro tento malý nevýdělečný projekt zde dobře poslouží nenáročné řešení v podobě open source aplikačního serveru Apache Tomcat.

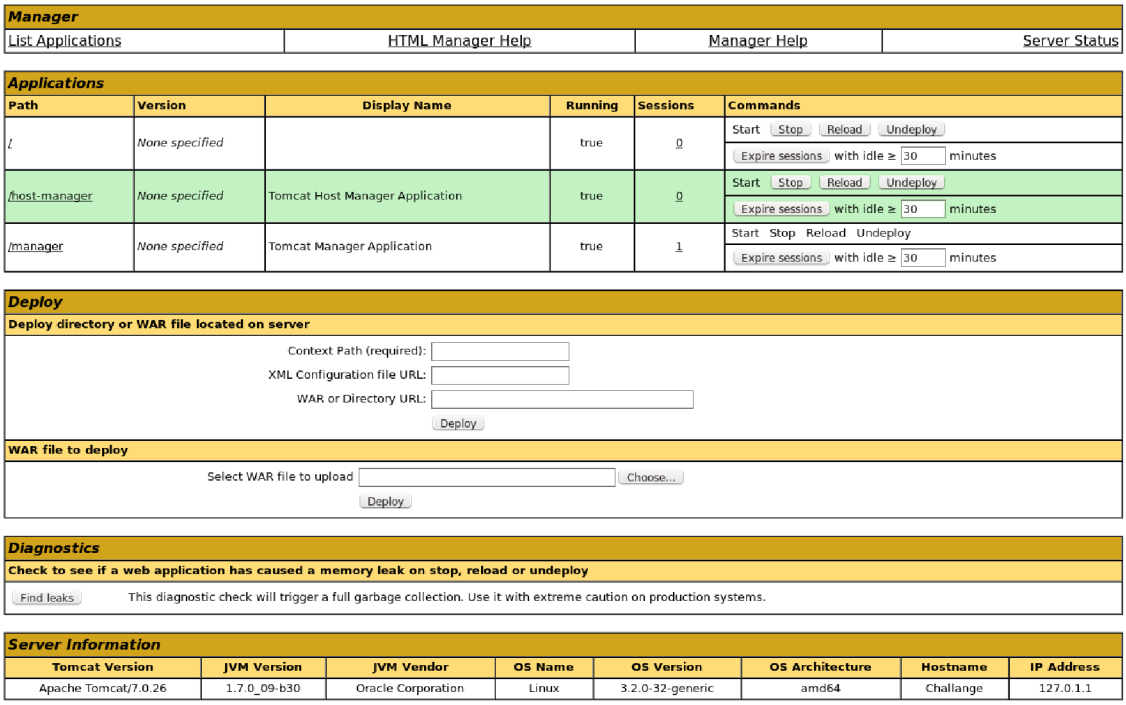
3.4 Framework

Framework je softwarové rozhraní mezi aplikací a aplikačním serverem, které řeší obecné problémy a programátor je nemusí znovu implementovat. Jedná se například o řešení přístupu k datům, zpracování uživatelského vstupu nebo výstup.

Aplikace psané pomocí frameworků jsou založeny na návrhovém vzoru MVC (Model-View-Controller).

- Práci s daty v databázi, v souborovém systému nebo webových formulářů má na starosti vrstva Model. Stará se o správnou interpretaci uložených dat do aplikace a naopak o manipulaci uložených dat příkazy aplikace.
- View (pohled) je prezentační vrstva a stará se o správné zobrazení výsledků k uživateli.
- Controller je řídící vrstvou aplikace. Spravuje všechny požadavky od uživatele, provádí příslušné akce a na výstup předává pohled a model s daty. Na výstup je předán pohled naplněný daty z modelu.

Spring framework [7] je známý Java framework používaný na malé i velké projekty. Má širokou komunitu a lehce se s ním dá vytvořit aplikace s dobře strukturova-



Obr. 3.3: Management rozhraní aplikačního serveru Apache Tomcat

ným a srozumitelným kódem. Příjemným řešením je pak takzvané řízení anotacemi nebo dependency injection.

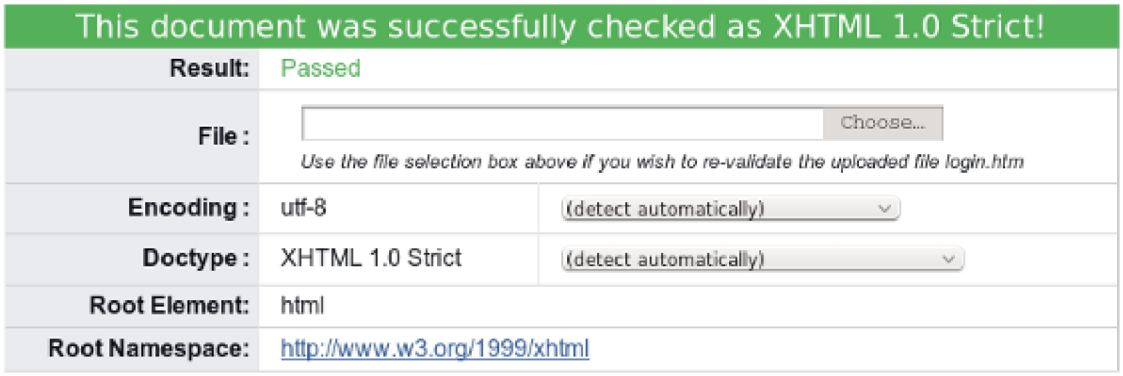
Struts [8] se velikostí řadí mezi menší frameworky. Je podobný Springu a lépe řešené má například zpracování formulářů. Statisticky se používá méně a v některých případech řeší problém složitěji než konkurence.

Pro široké rozšíření, dobrou dokumentaci a množství diskusních skupin a tutoriálů byl pro tuto práci vybrán framework Spring.

3.5 Databáze

Jako databázové uložiště byla použita všeobecně známá MySQL [17]. Jedná se o open source produkt firmy Oracle. Tato databáze je prověřená dlouholetým používáním ve firemním i domácím prostředí. Pro její dobře zpracované manuálové stránky, rozšířenost a z toho plynoucí podporu komunity byla použita i pro tuto práci.

Alternativami databázového systému mohl být například open source systém PostgreSQL, vyvíjený primárně pro platformu Linux. MySQL je přece jenom nejrozšířenější a její správ je známá široké komunitě vývojářů a správců.



Obr. 3.4: Výsledek úspěšné validace webové stránky

3.6 Webové standardy

Při tvorbě webových aplikací je také nutné pamatovat na webové standardy. Jsou to vlastně pravidla a doporučení, kterými se řídí psaní kódu používaného ve webových aplikacích. Dodržením standardů dosáhneme takzvaných validních stránek podle zvoleného schématu. Dále zajistíme bezpečné a bezchybné fungování ve většíně prohlížečů.

Webové standardy vydává mezinárodné konsorcium W3C (World Wide Web Consortium) [18]. V této aplikaci je použité doporučení XHTML verze 1.0 se schématem Strict označené jako `XHTML 1.0 Strict` [4] . Výsledek úspěšného procesu validace webové stránky je znázorněn na obrázku 3.4

3.7 Cron4j

V tomto projetku je nutné provádět časově náročné výpočty zaslaných dat. Jednou z možností je provádět tyto výpočty hned po nahrání uživatelských dat (na základě uživatelova podnětu). Uživatel by tedy musel čekat na dokončení výpočtu, aby mohl dále aplikaci používat. Mnohem lepší možností je jakékoli náročné výpočty provádět mimo uživatelký prostor. K tomu nám slouží takzvaný cron. Tento proces, běžící na pozadí, spouští v předem nastavených intervalech zadané akce v samostatném vlákně.

Každý běžně používaný operační systém má corn implementován v základu. Pro zachování platformní nezávislostí se v tomto projektu použije Javová implementace cronu Cron4j [19]. Nastavuje se a provádí stejné akce, jako systémový cron, ale běží na Javovém virtuálním stroji (JVM). Cron tedy lze přenést spolu s aplikací včetně všech nastavení.

[4] http://www.w3.org/2010/04/xhtml10-strict.html

3.8 Analýza a návrh

Na začátku tvorby každé větší aplikace je potřeba sesbírat požadavky od zákazníka. Jedná se o slovní popis možností aplikace pro koncového uživatele, popis chování v určitých případech a popis nebo základní nákres vizuální podoby prostředí.

Druhým krokem je analýza těchto požadavků a samotný návrh aplikace [15]. Výstupem a podkladem pro další práci je Use Case diagram a seznam událostí systému. U složitějších prací bývají k dispozici další podklady. Příkladem může být kontextový diagram, diagram datových toků (DFD), entitně relační diagram (ERD) [16] a datový slovník.

3.8.1 Kontextový diagram

Kontextový diagram znázorňuje aplikaci jako celek, její rozhraní, uživatele a systémy využívající toto rozhaní. Vztahy mezi uživateli a aplikací reprezentují uživatelské příkazy a interakce aplikace. Tento diagram nejobecněji popisuje možnosti práce s aplikací.

3.8.2 Diagram datových toků

DFD také znázorňuje aplikaci, ale na rozdíl od kontextového diagramu je detailnější. Diagram se skládá z procesů, toků, datových uložišť a terminátorů.

Proces je výpočetní část, která transformuje vstupní data na výstupní. Přesun dat mezi procesy, uložišti a terminátory znázorňuje datový tok. Datové uložiště představuje například databáze jako zdroj dat. Terminátory jsou uživatelé nebo jiné komunikující systémy, kteří stojí vně systému.

U rozsáhlejších systémů jsou tyto diagramy víceúrovňové. Každá další úroveň představuje detailnější popis části předchozí úrovně.

3.8.3 Entitně relační diagram

ERD je grafické znázornění datových entit a vztahů mezi nimi. Entita reprezentuje určitý typ objektu a jeho vlastnosti. Nejčastěji se jedná o databázovou tabulku a její sloupce. Vztahy mezi entitami se nazývají relace. Nad databázovými tabulkami definujeme čtyři druhy relací:

- 1:1 – jeden záznam v první tabulce je svázán s jedním záznamem ve druhé tabulce
- 1:n – jeden záznam v první tabulce je svázán s několika záznamy ve druhé tabulce

- n:1 – několik záznamů v první tabulce je svázáno s jedním záznamem ve druhé tabulce
- n:m – několik záznamů v první tabulce je svázáno s několika záznamy ve druhé tabulce (tento vztah na implementaci příliš složitý, proto se nahrazuje dvěma vztahy n:1 a 1:m a přidává se jedna pomocná vazební entita)

3.9 Autentizace

Aby mohl uživatel využívat možnosti systému, musí prokázat svou identitu. Existuje několik možností ověřování identity [14].

3.9.1 Znalost tajemství

Uživatel prokazuje svoji identitu na základě znalosti tajemství. Toto tajemství zná také systém a je schopen ho porovnávat s tajemstvím zadaným od uživatele. Nejčastější implementací tajemství je heslo nebo PIN. Jedná se o řetězec složený z velkých a malých písmen abecedy, číslic a ze speciálních tisknutelných znaků. PIN bývá složený pouze z číslic. Za bezpečná hesla jsou považovány řetězce delší než 8 znaků a složené ze znaků z různých kategorií. Na jednu stranu je nutné volit netriviální hesla, odolná proti slovníkovým útokům a útokům hrubou silou a na druhou stranu je nutné volit zapamatovatelná hesla. Pokud si člověk není schopen heslo zapamatovat, poznamená si ho a tím zvyšuje pravděpodobnost zneužití jeho identity. Oběť se o zcizení hesla ani nemusí dovědět. Útočník se může vydávat za někoho jiného a tím ohrožovat systém i data v něm uložená.

3.9.2 Vlastnictví tokenu

Token je předmět, který oprávněná osoba vlastní a autentizace probíhá ověřením tohoto vlastnictví a vlastností předmětu. K ověření je potřeba speciální zařízení, které je schopné z předmětu vyčíst potřebné vlastnosti nebo uložená data. Token by neměl být zkopírovatelný nebo by měl být obtížně zkopírovatelný. Příkladem tokenu je karta s magnetickým proužkem, karta s čipem nebo USB token. Případnou ztrátu je oběť schopna zjistit a učinit opatření zamezující zneužití.

3.9.3 Biometrika

Biometrika je založena na fyziologických vlastnostech lidského těla. Těmito vlastnostmi bývají nejčastěji otisky prstů, vzorek oční duhovky nebo sítnice. Autentizace probíhá změřením konkrétní vlastnosti a porovnáním se vzorkem uloženým při prvotním zakládání do systému. U biometrik se nekontroluje 100% přesnost vzorků,

ale je stanovena určitá tolerance odchylky. Není totiž pravděpodobné, že by dvě měření byla identická.

3.9.4 Výběr metody

V případě tohoto malého projektu se využije nejčastějšího a nejjednoduššího způsobu autentizace, a to ověřování pomocí znalosti tajemství – přihlašovacího jména a hesla. Standardem bývá, že hesla nejsou v systému uložena v otevřené podobě, ale je z nich vytvořen a uložen otisk, takzvaný hash. Hash se vytváří pomocí jednosměrné funkce a je obtížné až nemožné zpětně z hashe zjistit původní heslo.

4 TVORBA APLIKACE

4.1 Podklady pro práci

Pro zahájení práce je nutné nejdříve znát požadavky od zadavatele (zákazníka). Nejčastěji jsou vyjádřeny prostým textem, který obecně popisuje požadované vlastnosti a chování budoucí aplikace. Požadavky je nutné převzít, zpracovat a případně si ujasnit nedostatky. Následuje vytvoření diagramů chování aplikace a diagramu datového uložiště. Tyto podklady byly popsány v kapitole 3.8.

4.1.1 Zadání zákazníka

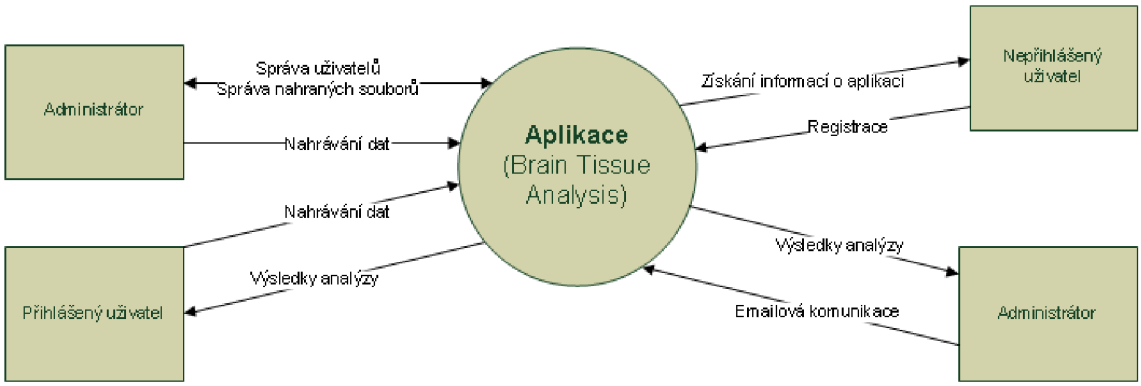
Cílem je vytvoření portálu pro pořádání vědeckých soutěží. Zde by měl být administrátor, který může dělat vše + vytvářet/mazat účty. Jednotliví uživatelé potom mohou posílat 1) své výsledky, 2) k nim dát nadpis, 3) popisek, 4) případné chybové hlášky, které se objeví při ohodnocení.

Poté, co někdo odevzdá soubor, by se měl zpracovat a zobrazit výsledek na webu, tak aby výsledky viděli všichni. Výpočet může trvat i několik minut (desítek minut), proto by se nemělo počítat přímo při odeslání.

Uživatelé mohou vidět a stahovat své odevzdané soubory + vidět své výsledky + výsledky ostatních. Uživatelé nesmějí vidět odevzdané soubory ostatních uživatelů.

4.1.2 Diagramy

Nejobecnějším diagramem je **kontextový diagram**. Tento diagram je znázorněn na obrázku 4.1 a je na něm rozpoznatelná aplikace jako celek a tři role uživatelů, včetně jejich možností práce s aplikací.

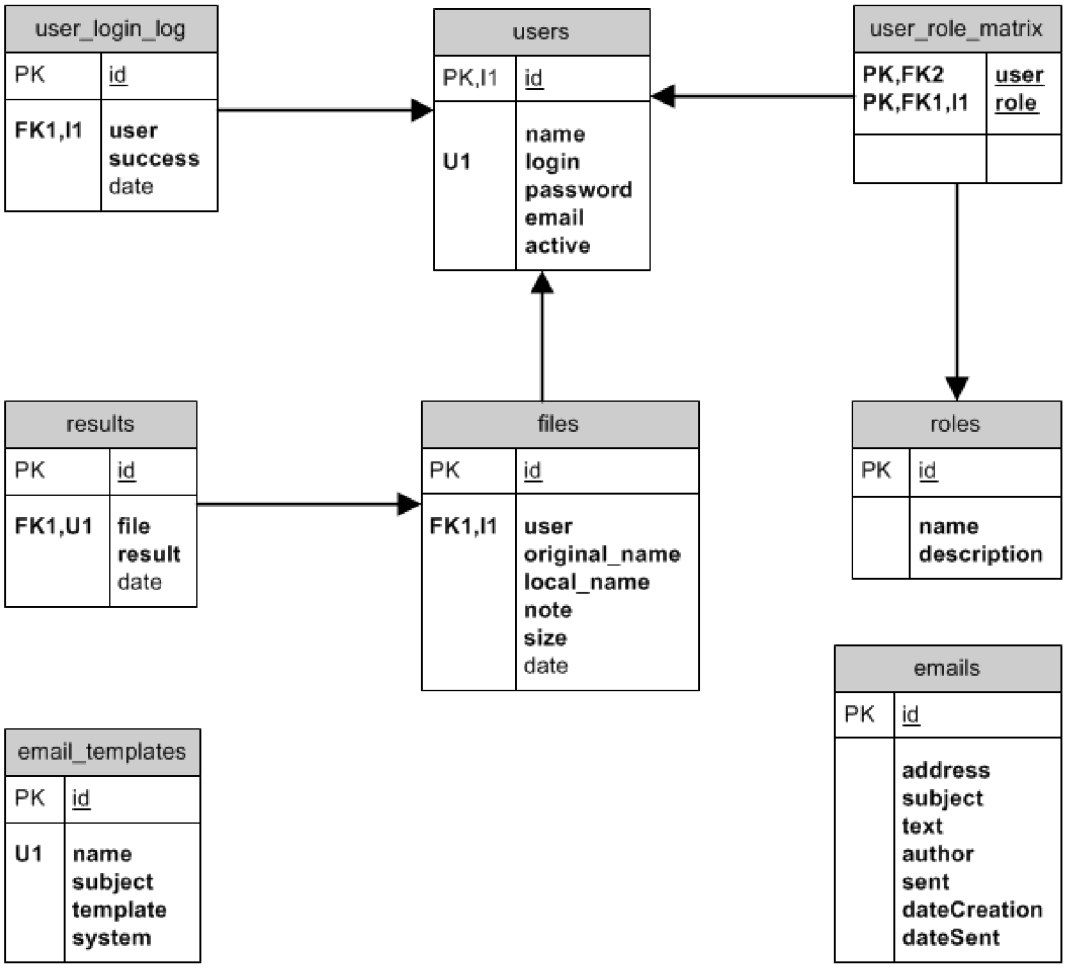


Obr. 4.1: Kontextový diagram aplikace

Detailnější verzí kontextového diagramu je diagram datových toků. Ten již nezobrazuje aplikaci jako celek, ale sleduje základní skladbu aplikace a vztahy mezi uživateli.

Entitně relační diagram v této aplikaci představuje navržení databázových tabulek. Databáze aplikace obsahuje 8 tabulek, z toho jedna reprezentuje vazební entitu pro relaci n:m. Základní entitně relační diagram je znázorněn na obrázku 4.2. Rozšířený diagram a detailní popis tabulek je v příloze C této práce [21].

Tabulky jsou typu InnoDB a využíva se u nich vlastnosti Foreign Key Constraints [17]. Jedná se o vlastnost propojení tabulek a nastavení vhodných závislostí. Pokud například dojde ke smazání záznamu o souboru v tabulce files, automaticky se také smaže záznam o výsledku zpracování z tabulky results. Tato vlastnost umožňuje přesunout část logiky z aplikace na databázi a vyhnout se mnohonásobně delšímu kódu a i případným chybám. Jedná se o takzvané zajištění integrity dat.



Obr. 4.2: Entitně relační diagram aplikace

4.2 Role a oprávnění

V aplikaci se budou uživatelé dělit do následujících tří skupin podle získaných rolí:

- Nepřihlášený uživatel
- Přihlášený uživatel
- Administrátor

Nepřihlášený uživatel bude mít možnost se seznámit s projektem, případně si založit účet a dále pak využívat aplikaci jako přihlášený uživatel.

Přihlášený uživatel bude moci nahrávat data ve formě souborů, mazat a stahovat svoje nahrané soubory, zobrazovat výsledky zpracování nahraných dat všech uživatelů a bude mít možnost si změnit heslo.

Administrátor bude mít nejvíce možností. Jedná se vlastně o uživatele s rolí admin. Bude spravovat uživatelské účty (zakládat, měnit údaje a mazat), nastavovat hesla a spravovat veškerá nahraná data. Dále bude mít možnost komunikovat se zaregistrovanými uživateli přes emailové zprávy a u tvorby těchto zpráv využívat šablony.

Samotná autentizace bude probíhat dle následujícího scénáře:

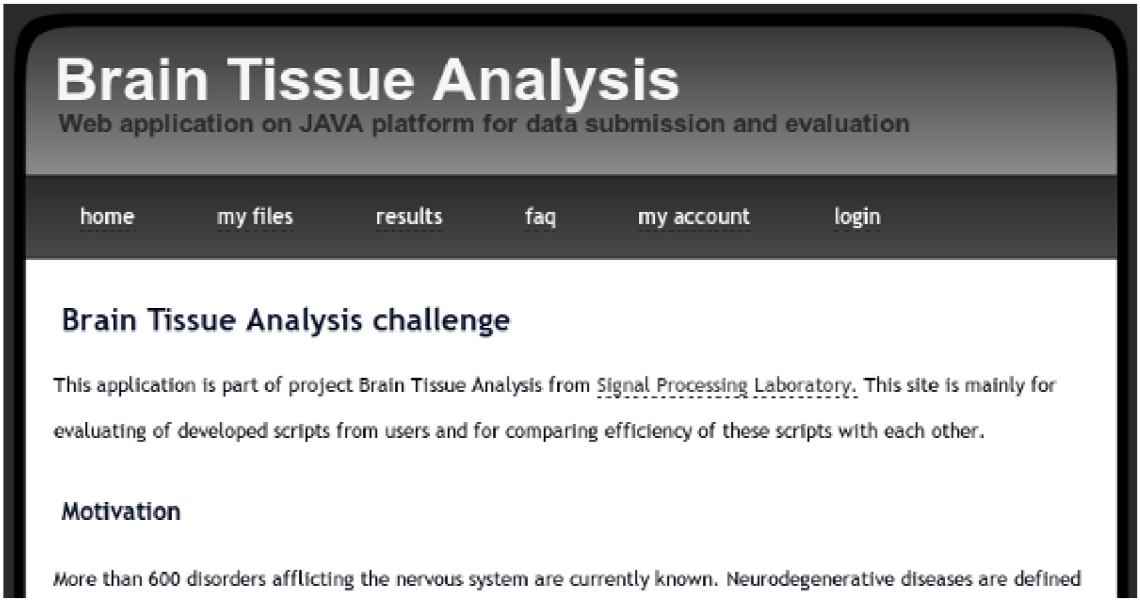
- Získání přihlašovacího jména a hesla od uživatele, který vyplní a odešle formulář
- Zahashování hesla
- Hledání v databázi odpovídající dvojici jmého a hash
- Pokud není nalezen žádný záznam, znamená to nenalezení uživatele nebo neodpovídající heslo a dochází k přesměrování na prázdný přihlašovací formulář a je mu zobrazena odpovídající chybová hláška
- V případě, že je uživatel nalezen, ale ještě nemá aktivován účet, je o této situaci informován chybovou hláškou a k přihlášení nedojde
- V případě, že je uživatel nalezen a účet má aktivní, načtou se jeho další informace včetně rolí, uloží se do uživatelského sezení (session) a dojde k přesměrování na stránku s nahranými soubory

4.3 Struktura uživatelského prostředí

Aplikace je rozdělena na několik stránek, které jsou dostupné z menu umístěného pod hlavičkou. Dále existují stránky, které v menu viditelné přímo nejsou, ale vede k nim cesta přes odkazy na stránkách nebo v zasílaných emailech.

4.3.1 Úvodní stránka

Úvodní stránka obsahuje pouze statický text, který popisuje projekt „Brain Tissue Analysis“ a tuto aplikaci a snaží se návštěvníkovi nabídnout dostatek informací. Obsahuje také odkazy na stránku FAQ a na stránky „Signal Processing Laboratory“. Návrat na úvodní stránku je možný v menu pod odkazem `home`. Úvodní stránka s hlavičkou a menu je zachycena na obrázku 4.3.



Obr. 4.3: Úvodní stránka

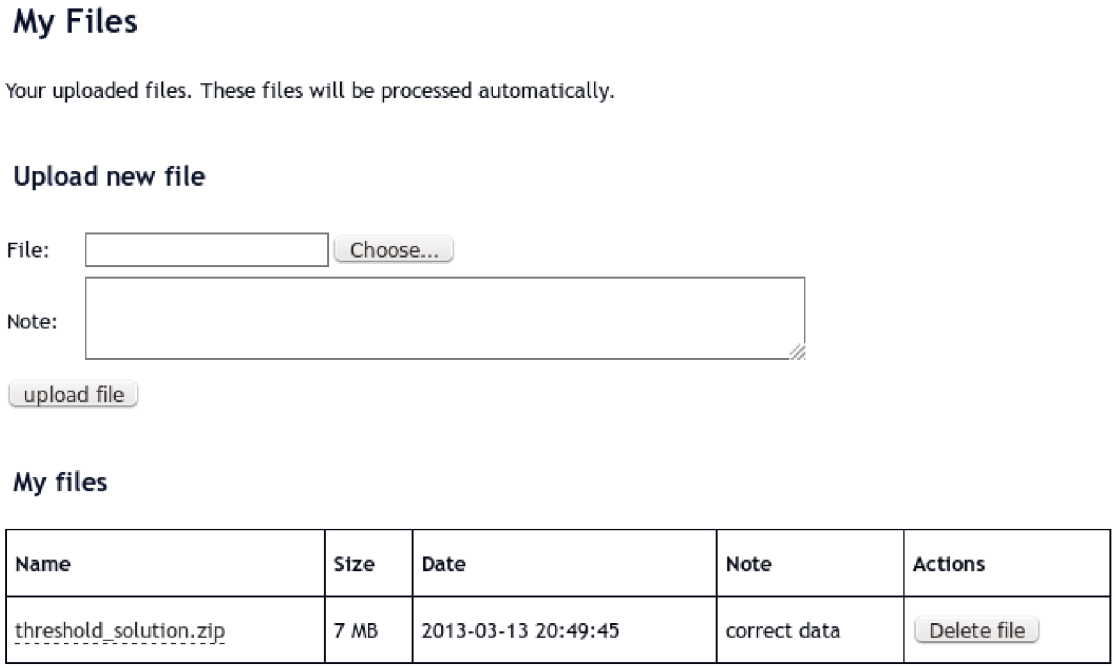
4.3.2 Stránka správy souborů

Tato stánka je dostupná pod odkazem „my files“ v menu, je určena pro správu uživatelských souborů a vyžaduje přihlášení. Pokud se k ní snaží přistoupit nepřihlášený uživatel, je přesměrován na přihlašovací stránku.

Přihlášení uživatelé mají možnost nahrávat soubory pro následné vyhodnocení. Mají k dispozici jednoduchý formulář, kde je mimo nahrávaného souboru možné přidat i krátký popis. Nahrané soubory jsou zobrazeny v tabulce umístěné pod formulářem. Soubor je možné zpětně stáhnout nebo smazat. V tabulce je dále k dispozici informace o velikosti souboru a o datu nahrání. Na obrázku 4.4 je zachycen formulář pro nahrání souboru a záznam jednoho nahraného souboru.

Administrátor má navíc k dispozici tabulku se soubory všech uživatelů systému. Má také možnost provádět stejné akce jako vlastník souboru, tedy stahovat i mazat.

Nahrávání uživatelských souborů se podrobněji věnuje kapitola 4.5.



Obr. 4.4: Stránka správy souborů

4.3.3 Stránka výsledků zpracování

Stránka pod odkazem „results“ je úzce spjata se stránkou správy souborů. Je také dostupná pouze přihlášenému uživateli a je také chráněna před přístupem nepřihlášeného uživatele. Obsah se věnuje výsledkům zpracování nahraných souborů od všech uživatelů systému. Tyto výsledky jsou zobrazeny v přehledné tabulce včetně názvu zpracovaného soubru, číselného výsledku a data zpracování. V případě, že uživatel nebo administrátor smažou již zpracovaný soubor, je i výsledek odstraněn z tohoto přehledu.

Zpracování nahraných souborů se podrobněji věnuje kapitola 4.7

4.3.4 Stránka FAQ

Termín FAQ je anglická zkratka z výrazu „Frequently Asked Questions“ a používá se jako všeobecně známá i na českých serverech. Stránka FAQ, podobně jako úvodní stránka, obsahuje pouze statický text. Návštěvníkovi poskytuje odpovědi na často pokládané otázky.

Sekce FAQ se stala běžnou součástí moderních webových prezentací. Snahou je hlavně odfiltrovat často pokládané dotazy na správce webu.

4.3.5 Stránka uživatelkého účtu

Pod odkazem „my account“ nalezne přihlášený uživatel informace o svém účtu (jméno a emailovou adresu), které použil při registraci. Dále má na tomto místě možnost změny přihlašovacího hesla. Pro změnu hesla je nutné vyplnit a odeslat formulář se všemi povinnými vstupy. Formulář požaduje vyplnění správného původního hesla a nového hesla.

V tomto formuláři se využívá několika standardních technik pro zvýšení bezpečnosti. První je skrývání psaného hesla určená proti odpozorování z okolí. Při psaní se místo stisknutých znaků píšou hvězdičky (*) nebo kuličky (•) v závislosti na použitém webovém prohlížeči.

Druhou technikou je nutnost dvojího zadání nového hesla. Cílem je předejít špatné změně hesla způsobené překlepem. Poslední používanou technikou je požadavek na minimální délku hesla, zde nastavený na 8 znaků. Krátká hesla jsou s dnešními výpočetními možnostmi snadno prolomitelná a minimální požadovaná délka hesla tomuto alespoň částečně zabraňuje.

4.3.6 Stránka administrátorských nástrojů

Přihlášený uživatel s rolí admin má na této stránce k dispozici nástroje pro správu uživatelů a nástroj pro rozesílání emailových zpráv. Těmto nástrojům se věnují samostatné kapitoly 4.8 a 4.9.

Záložka v menu pro tuto stránku je viditelná až po přihlášení uživatele a rozpoznání jeho role admin. Při pokusu přistoupit na tuto stránku přihlášeným uživatelem bez práva admin nebo bez přihlášení je vrácena standardní chybová stránka 404 (stránka nenalezena).

4.3.7 Stránky pro přihlášení a odhlášení

Jak už název napovídá, tyto stárnky se starají o proces přihlášení a odhlášení uživatele. Nepřihlášený uživatel vidí v menu stránku login a naopak přihlášený vidí stránku logout.

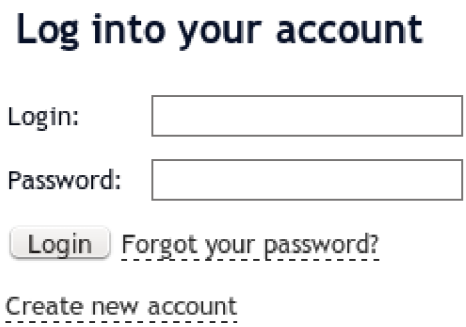
Na přihlašovací stránce se nachází standardní přihlašovací formulář s textovými poli pro jméno a heslo. Při neúspěšném přihlášení se uživateli zobrazí chybové hlášení a k přihlášení nedojde. Při úspěšném ověření přihlašovacích údajů dojde k založení uživatelského sezení (uživatel je přihlášen) a k přesměrování na stránku správy souborů. Na této stránce se dále nachází odkaz na registraci nového uživatelského účtu. Registrací se zabývá samostatná kapitola 4.4.

Důležitější součástí je ovšem odkaz „Forgot your password?“, který navede uživatele na řešení problému se zapomenutým heslem. Odkaz vede na stránku FAQ,

na které je mimo jiné popsán postup řešení při zapomenutí nebo ztrátě přihlašovacích údajů. Uživatelé zapomínají přihlašovací jména a hesla neustále a tento odkaz se stal nepsaným standardem u přihlašovacích formulářů.

Přihlašovací formulář se všemi položkami a souvisejícími odkazy je zachycen na obrázku 4.5.

Proces odhlášení pouze smaže uživatelské sezení a přesměruje uživatele na přihlašovací stránku.



Obr. 4.5: Stránka pro přihlášení uživatele

4.3.8 Adresy jednotlivých stránek

Jednotlivé stránky viditelné z menu, jejich adresy a popis jsou uvedené v tabulce 4.1. Menu přihlášeného uživatele se jménem „administrator“ a s právy admin je zobrazeno na obrázku 4.6

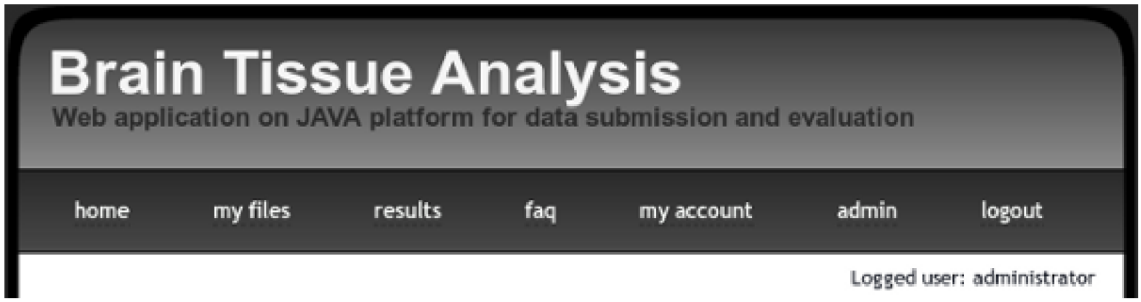
Tab. 4.1: Stránky aplikace a jejich adresy viditelné z menu

Stránka	**Adresa**	**Popis**
home	/	úvodní stránka s popisem aplikace
my files	/myfiles	správa souborů
results	/results	zobrazení výsledků výpočtů nahraných souborů
faq	/faq	nejčastější otázky a odpovědi
my account	/myaccount	údaje o přihlášeném uživateli
admin	/admin	administrátorské rozhaní
login	/login	formulář pro přihlášení uživatele
logout	/logout	akce odhlášení viditelná pouze po přihlášení

Stránky, které přímo nejsou dostupné z menu, jsou uvedené v tabulce 4.2.

Tab. 4.2: Stránky aplikace a jejich adresy bez odkazu v menu

Stránka	Adresa	Popis
new account	/newaccount	formulář pro registraci nového uživatele
account activation	/newaccountactivation	aktivace nového uživatelského účtu (adresa zasílaná emailem)



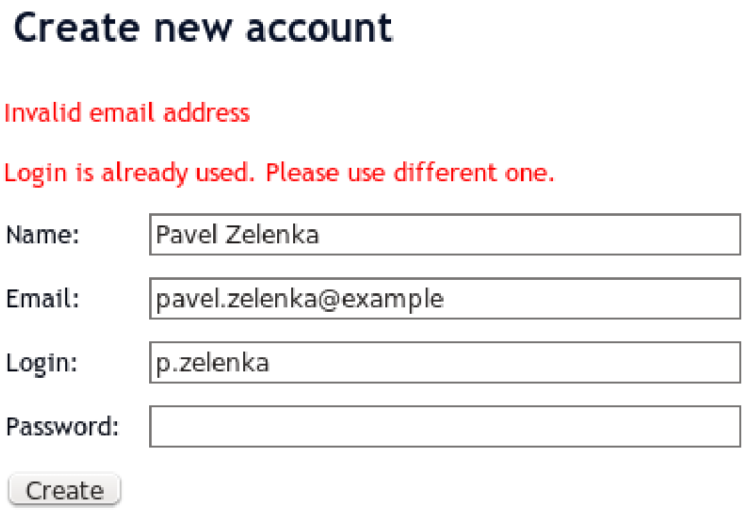
Obr. 4.6: Hlavička a menu aplikace

4.4 Registrace uživatelského účtu

Nový účet si může uživatel založit sám bez nutnosti zásahu administrátora. Na stráce pro přihlášení se nachází odkaz vedoucí na adresou `/newaccount`, kde se nachází samotný registrační formulář. Pro založení účtu je nutné vyplnit povinná textová pole pro jméno, emailovou adresu, přihlašovací jméno (login) a heslo. Po odeslání formuláře jsou data zkontrolována, zda mají povinnou minimální délku, nepřekračují maximální povolenou délku nebo neobsahují nepovolené znaky. Dále se kontroluje formát a unikátnost emailové adresy. To znamená kontrolu, zda již uživatel se stejnou emailovou adresou není založený. Unikátnost se také kontroluje u pole login.

Po správném vyplnění a odeslání formuláře je založen zatím neaktivní účet a uživateli je odeslán aktivační email. Uživateli zatím není umožněno se přihlásit do aplikace.

Odeslaný registrační formulář s chybovými hlášeními o špatně zadané emailové adrese a již existujícím účtu s přihlašovacím jménem „p.zelenka“ je vyobrazen na obrázku 4.7. Z obrázku je také zřejmé, že vyplněné heslo se do odeslaného formuláře nepředvyplnilo. Jedná se o standardní chování webových formulářů zvyšující bezpečnost.



Obr. 4.7: Formulář pro registraci nového účtu

4.5 Aktivace uživatelského účtu

Aktivační email, který uživatel obdržel po registraci, obsahuje odkaz vedoucí do aplikace na stránku pro aktivaci účtů. Odkaz již obsahuje parametry pro rozeznání uživatele (userlogin) a kontrolní řetězec (hash). Příkladem může být odkaz uživatele jnovak[1].

Proces aktivace účtu spočívá ve výpočtu hashe pro rozpoznaného uživatele a jeho porovnání s předaným hashem. V případě shody se v databázi nastaví příznak active.

Úspěšná aktivace dovolí uživateli přihlášení a využívání všech dostupných funkcí. Tento mechanismus ověřování se běžně používá u většiny webových služeb a má zabránit alespoň částečnému zneužití aplikace.

4.6 Nahrávání souborů

Na stránce správy souborů je přihlášený uživatel schopen nahrát soubor s daty. Je zde k dispozici jednoduchý formulář se vstupem pro soubor a vstupem pro volitelný popis k souboru. V případě úspěšného uložení souboru na serveru je do databáze přidán záznam o této akci, který obsahuje referenci na uživatele, originální a dočasný název souboru, velikost souboru a datum nahrání.

Pro zajištění větší bezpečnosti není doporučené ukládat na serveru nahrávané soubory s původním názvem. Uživatel by mohl vhodným pojmenováním souboru

[1] `/newaccountactivation?userlogin=jnovak&hash=fd0822522969a095df72e78ee5717201`

využít vlastností webového serveru a souborového systému a přepsat některý z konfiguračních souborů. Proto je generován nový náhodný řetězec, který slouží k pojmenování nahraného souboru na serveru. Dalším zabezpečením je limit velikosti nahrávaného souboru nastavený na 20MB.

Všechny tyto uživatelské soubory jsou ukládány do adresáře definovaného v konfiguraci. Na linuxových systémech se „často měnící“ soubory ukládají do adresáře `/var`. Na cílovém serveru bylo zvoleno umístění `/var/brainTissueAnalysis/data/`.

Obr. 4.8: Formulář pro nahrání souboru v sekci správy souborů

4.7 Vyhodnocování dat

Automatický skript spouštěný cronem v pravidelných intervalech kontroluje záznamy v databázi a hledá doposud nezpracované uživatelské soubory. V případě nalezení některého takového záznamu se spouští skript `evaluator.jar`, který porovná nahraná data s referenčními statickými daty. Výsledkem zpracování je procentuální shoda s těmito statickými daty. Tato hodnota se ukládá zpět do databáze spolu s příznakem dokončeného zpracování.

Před samotným spuštěním zpracujícího skriptu je nutné připravit potřebná data. Nahrané uživatelské soubory jsou ve formátu archivu zip. Tento archiv je nejdříve nutné rozbalit do dočasného adresáře a ten je následně možné předat ke zpracování jako parametr skriptu. Rozbalení je prováděno pomocí standardní Java knihovny `java.util.zip.ZipInputStream`, která zpřístupní obsah archivu a jednotlivé soubory a adresáře jsou kopírovány hierarchicky do nového umístění. Po dokončení zpracování je tento adresář i s obsahem smazán. Pro rozbalená data bylo zvoleno umístění `/var/brainTissueAnalysis/temp/`.

Celý příkaz, spouštěný cronem, by mohl být složen následovně:
příkaz `java -jar /var/brainTissueAnalysis/app/evaluator.jar`
první parametr `/var/brainTissueAnalysis/temp/upl_6849640157678678301`
druhý parametr `/var/brainTissueAnalysis/refData/`

Existuje určitá pravděpodobnost, že se zpracování jednoho balíku dat nestihne do doby následného spuštění skriptu cronem (instance). Proto byl implementován takzvaný zámek ve formě souboru, který se kontroluje při spuštění nové instance. V případě uzamčení (právě probíhajícího zpracování) se nedovolí druhé spuštění zpracování a instance končí.

Další dodatečnou ochranou při zpracování a použití zámků je nastavení maximální doby zamčení. Pokud by například došlo k chybě a ukončení probíhající instance, zámek by zůstal ve stavu zamčeno až do zásahu správce a během této doby by se nespustilo žádné další zpracování dat. Při kontrole existence zámku se tedy kontroluje i doba jeho vytvoření a pokud je delší než nastavené 2 hodiny, je automaticky smazán a je umožněno další spuštění zpracování.

4.8 Administrace uživatelů

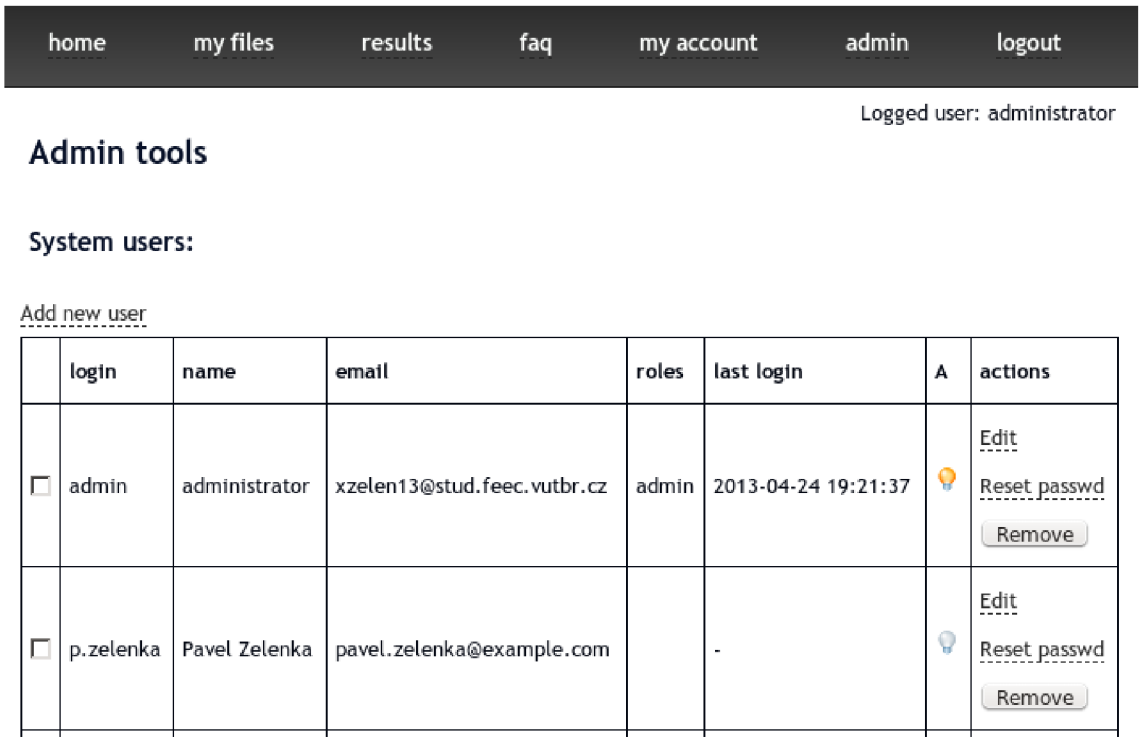
Přihlášený uživatel s rolí admin má oprávnění přistoupit na stránku s administrátorskými nástroji a tyto nástroje využívat. Jedná se hlavně o správu uživatelů. V přehledné tabulce jsou vidět založení uživatelé, jejich jména, emaily, role a datum posledního přihlášení. Ke každému uživateli jsou také k dispozici akce úpravy údajů, nastavení hesla bez znalosti předchozího a smazání uživatele.

Zaškrtávací políčka v prvním sloupci jsou určena k vybrání uživatelů, kterým bude pomocí dalšího nástroje odeslán email. Emailové komunikaci je věnována následující kapitola. Indikátor žárovky v předposlením sloupci poskytuje informaci, zda byl uživatelský učet aktivován nebo ne.

Pod odkazem „add new account“ se skrývá stránka s formulářem pro vytvoření nového uživatele. Při zakládání nebo úpravě uživatelského účtu je potřeba vyplnit všechny povinné údaje včetně validní emailové adresy a popřípadě přiřadit role. Navíc lze uživatelský účet nastavit rovnou jako aktivovaný. V případě odeslání formuláře s nevalidními daty se vypíše odpovídající chybové hlášení o původu chyby a k uložení dat nedojde. Formulář s provedenými změnami, včetně chybně zadaných údajů, bude k dispozici pro opravu a opětovné odeslání.

Pokud uživatel zapomene přihlašovací údaje, měl by využít standardní cesty popsané na stránce FAQ, tj. kontaktování administrátora pro vygenerování nového hesla. Administrátor má k dispozici příslušný nástroj, který umožňuje nastavení nového hesla bez znalosti předchozího. Pomocí jednoduchého formuláře je nastaveno nové heslo a například pomocí nástroje emailové komunikace je možné toto zaslat uživateli.

Možnosti administrátora při správě uživatelských účtů představuje obrázek 4.9



Obr. 4.9: Administrační nástroje – správa uživatelů

4.9 Emailová komunikace

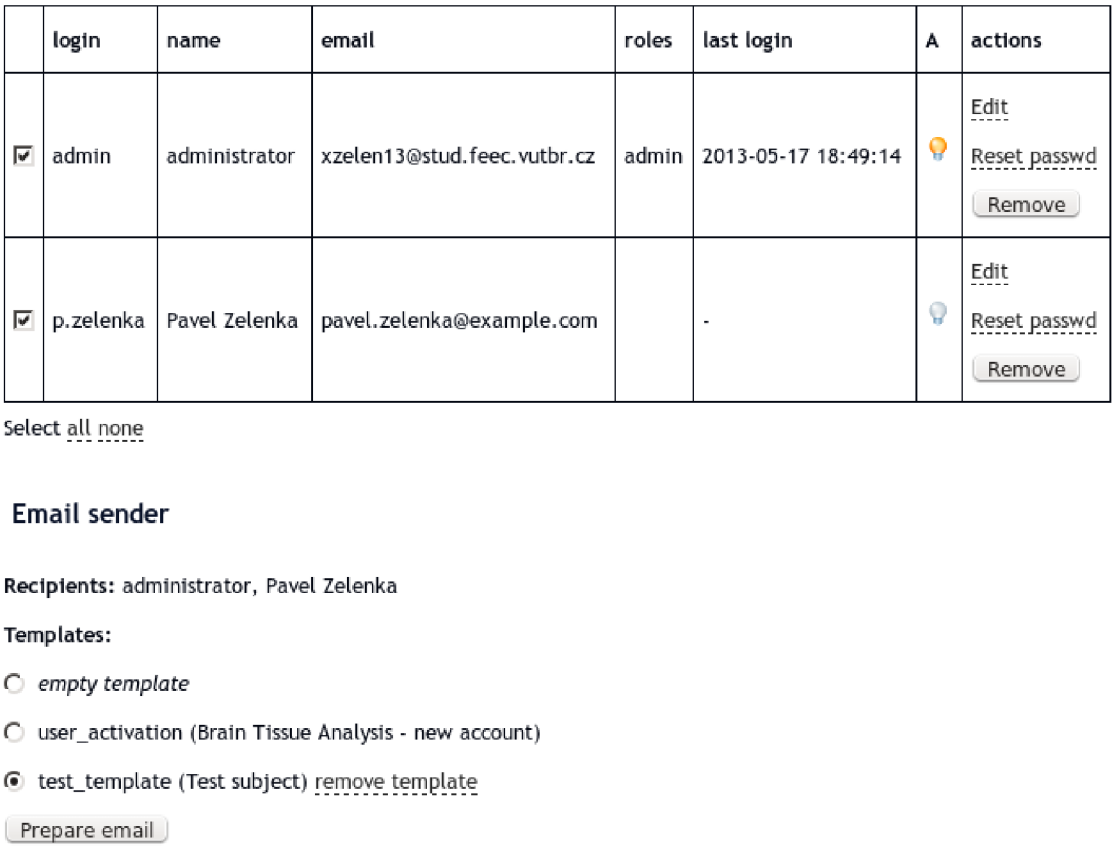
Pro komunikaci s uživateli byla implementována podpora odesílání emailů. O zasílání emailů již bylo řečeno v kapitole 4.5.

Prvním krokem pro odeslání emailu je výběr adresátů. Administrátor zaškrtnutím políčka vedle záznamu uživatele přidá tohoto uživatele k adresátům připravovaného emailu. Odškrtnutí políčka uživatele z adresátů odebere. Navíc je možné použít tlačítka „Select all“ a „Select none“ umístěné pod tabulkou uživatelů, které najednou vyberou všechny uživatele nebo žádného.

Tento první krok přípravy odeslání emailu je zachycen na obrázku 4.10.

Druhým krokem je výběr šablony (templatu). Je možné vybrat prázdnou nebo předpřipravenou šablonu. Template je předpřipravený text emailu, včetně předmětu, který může obsahovat proměnné. Při konečném zpracování zprávy se proměnné nahradí za odpovídající údaje. Takto lze nahrazovat uživatelské jméno, login nebo emailovou adresu. Příkladem použití může být úvodní oslovení na prvním řádku, které by se zapsalo následovně: `Hello %userName%`. V samotné zprávě bude tedy text nahrazovaný skutečnými jmény uživatelů.

Vytvořenou zprávu lze uložit jako nový template a použít v budoucnu pro konstrukci nového emailu. Konstrukce textu emailové zprávy je zachycena na obrázku 4.11. Z obrázku je také možno vypozorovat, že předmět a text připraveného emailu



Obr. 4.10: Administrační nástroje – emailová komunikace

bude uložen jako nová šablona s názvem „monthly_update“.

Odesílání emailové zprávy je časově náročný proces, a proto je lepší využít cron stejně jako při vyhodnocování dat popsané v kapitole 4.7. Připravené emailové zprávy jsou ukládány společně s adresátem a předmětem do databáze, odkud se následně předávají ke zpracování (odeslání). Na rozdíl od procesu vyhodnocování dat, kdy se zpracovávají soubory po jednom, se zpracování (rozeslání) emailů provádí hromadně. Zpracují se všechny čekající emaily najednou v běhu jedné instance. Po úspěšném odeslání emailu se záznam v databázi označí za zpracovaný a nadále zůstává uložený.

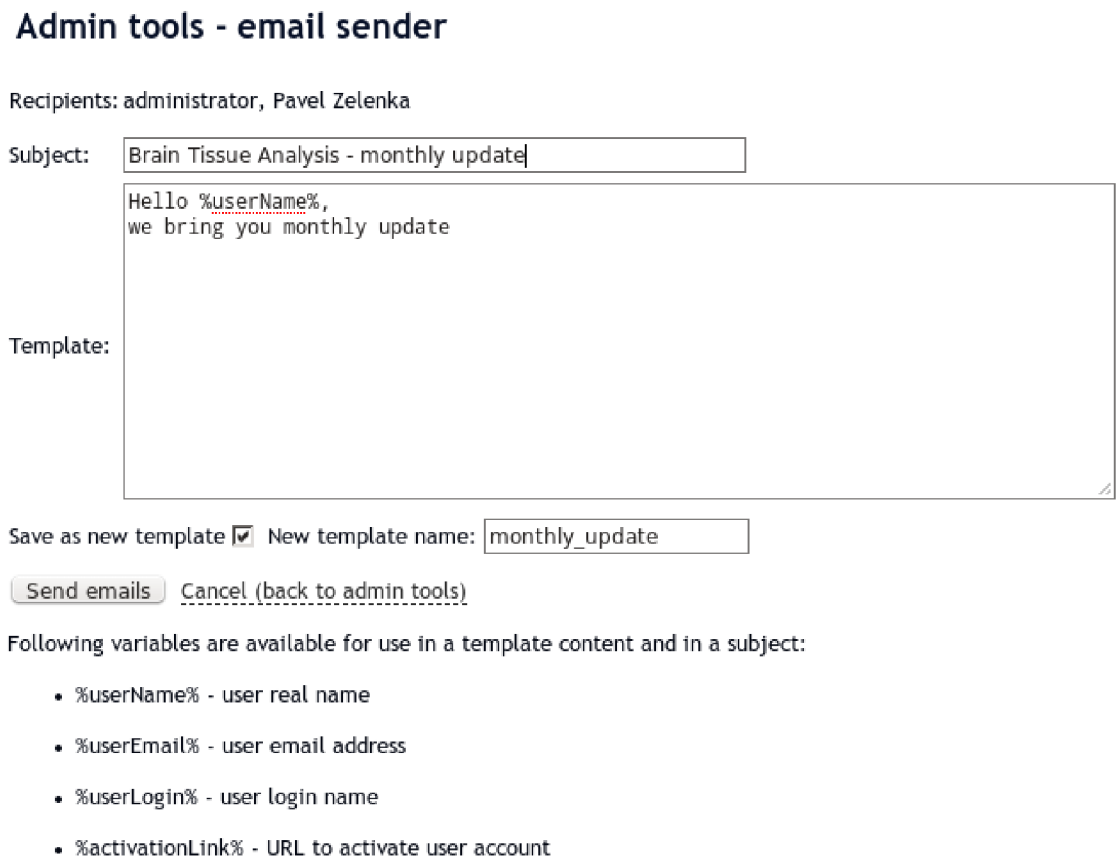
Vytvořené šablony je možné také smazat. Slouží k tomu odkaz „remove template“ zobrazený vedle názvu šablony. Tento odkaz se nezobrazuje u prázdné šablony a u šablon označených v databázi jako systémové. Systémová šablona je taková, kterou přímo používá zdrojový kód. Příkladem může být šablona aktivačního emailu, zasílaného po registraci nového účtu.

Pro samotný proces mazání je použita technologie AJAX[2]. Pomocí JavaScriptu se na pozadí odešle HTML požadavek na smazání šablony. Odpověď na tento po-

[2]Asynchronous JavaScript and XML

žadavek obsahuje informaci o úspěchu nebo neúspěchu procesu mazání. Místo znovunačtení stránky a vypsání nového seznamu šablon se pouze skryje řádek se smazanou šablonou. Opět se k tomu využije JavaScript.

Stránky se použitím této technologie stávájí interaktivnější a „šetrnější“ k datovým přenosům. Nepřenáší se tedy celá stárnka, ale pouze její část nebo informace potřebná k její úpravě.



Obr. 4.11: Administrační nástroje – příprava emailové zprávy

4.10 Grafická úprava

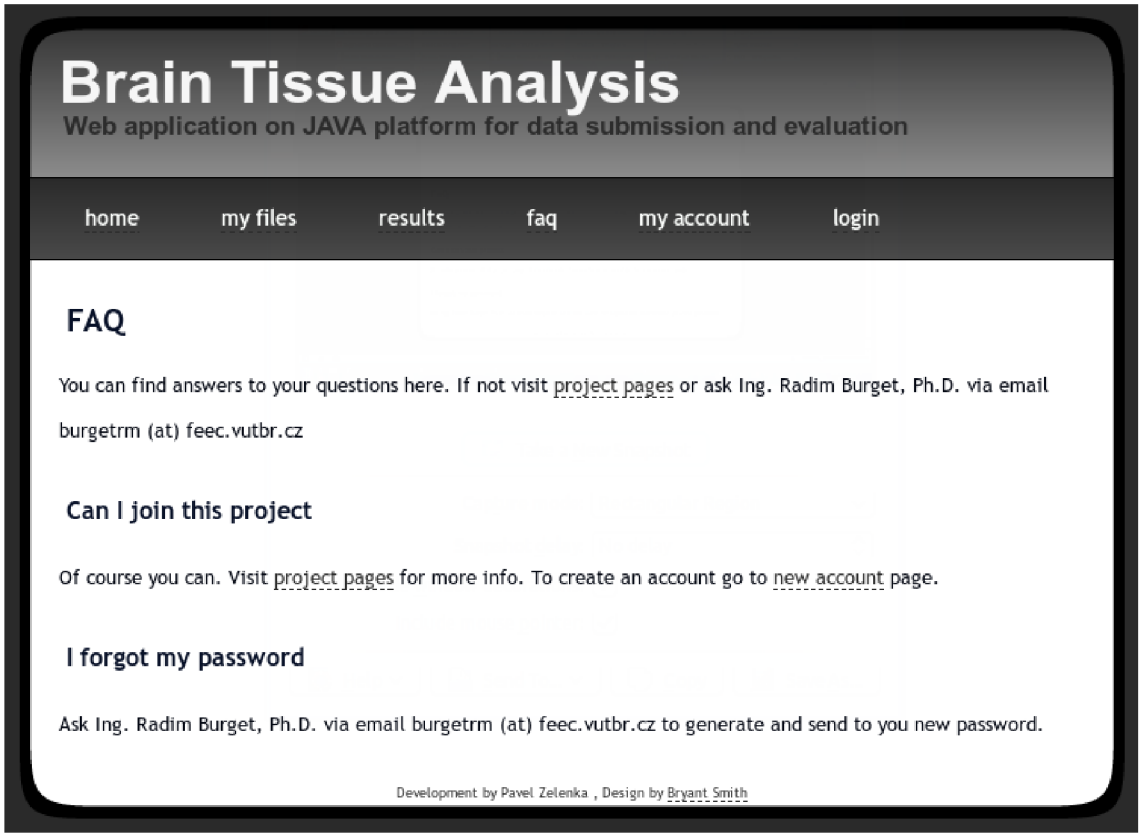
S aplikací budou pracovat vědci po celém světě, a proto byl zvolen jeden z nejrozšířenějších jazyků, angličtina. Popisné texty jsou formulované jednoduše, takže i pro neanglicky mluvící osoby s použitím automatického překladače je text srozumitelný.

Jako grafická úprava byl zvolen návrh „A Farewell to Color“ od designerské skupiny Bryant Smith [20] (obrázek 4.12). Tato skupina dala k dispozici zdarma několik šablon pro jednoduché a rychlé vytvoření webové prezentace. Jsou určeny

především jednotlivcům nebo menším podnikům. Autoři pouze chtějí zanechat svůj kontakt a odkaz v patičce stránky.

Všechny stránky dodržují standardní zvyky umístění prvků (formuláře, tabulky, odkazy, stavová hlášení) a hierarchii nadpisů.

V nástroji pro správu uživatelů, konkrétně v seznamu uživatelů, byl použit mechanismus přímého vložení obrázku do stránky. Bylo použito takzvaného base64 způsobu zakódování obrázku. Konkrétně se jedná o obrázky žárovek, které signalizují aktivovaného uživatele – rozsvícená žárovka nebo doposud neaktivovaného uživatele – zhasnutá žárovka. Jedná se vlastně o obrázek převedený na text a vložený přímo do kódu stránky. Webová stránka je běžně složena ze samotného textu (zdrojový kód) a přidružených souborů (obrázky, styly, skripty), které si webový prohlížeč stahuje zvlášť. Cílem tohoto snažení je zrychlit načítání stránky. Je lepší načíst o pár kilobytů textu víc, než se několikrát dotazovat na server pro menší kousky dat (typicky malé obrázky).



Obr. 4.12: Design stránky s odkazem na autora v patičce

4.11 Struktura kódu

Pořádek je dobré dodržovat nejen v uživatelském prostředí, ale také v samotném zdrojovém kódu. V kódu jsou Javové třídy v samostatných souborech. Třídy jsou rozdělené do logických balíčků, které fyzicky tvoří hierarchii adresářů. Název balíčku je tvořen adresářovou cestou, kde se místo lomítek zapisuje tečka. Na první úrovni je balíček pojmenovaný `cz.vutbr.feec.utko`. Tento balíček obsahuje základní třídy, takzvané controllery, které obshluhují jednotlivé stránky. Na dalších úrovních jsou balíčky a v nich třídy, které se věnují vždy konkrétnímu tématu. Jsou zde například balíčky sdružující práci se soubory, práci s výsledky zpracování nebo proces přihlášení a uživatelský účet. Vedle balíčků a tříd existují ještě takzvané pohledy (views). Tyto JSP[3] soubory se starají o vzhled a uspořádání prvků na stránce. Balíčky, kontrollery a pohledy jsou sepsané v příloze D.

4.12 Řízení anotacemi

Javové třídy ve webových aplikacích se dělí do určitých kategorií, podle toho jaké plní role. Nejdůležitější kategorií tříd jsou takzvané kontrolery, které se starají o vyřízení událostí a požadavků od uživatele. Standardně obsahuje název této třídy klíčové slovo `Controller`.

Další kategorií jsou třídy starající se o přístup k datům. Nejčastěji se jedná o přístup a komunikaci s databází. Těmto třídám se říká repository a v jejich názvu bývá klíčové slovo `Dao`.

Poslední důležitou kategorií, když nepočítáme standardní třídy představující objekty, jsou třídy modelů. Tyto třídy uchovávají data pro naplnění formulářů nebo data odeslaná uživatelem pomocí formulářů. Opět se v názvu třídy používá klíčová slovo `Model`.

Anotace představují jednu z možností konfigurace tříd a jejich metod. Na rozdíl od konfiguračních souborů se anotace zapisují přímo k definici tříd a metod. Tento způsob zápisu konfigurace je přehledný u malých projektů.

Jednotlivé metody ve třídách představující kontrolery je nutné namapovat na konkrétní akce uživatele. Těmito akcemi jsou nejčastěji dotaz na stránku nebo odeslání formuláře. Z následujícího příkladu je zřejmé, že třída `MyFilesController` uvozena anotací `@Controller` představuje kontroler a všechny její metody jsou namapované na adresu `/myfiles`. Prvotní přístup na stránku `my files` je zpracováván metodou `myFilesIndex`. Akci nahrání souboru zpracovává metoda `uploadFile` namapovaná

[3] Java Server Pages

na adresu `/myfiles/upload`. Dále je možné si všimnout, že této metodě je předáván parametr `UploadModel`. Jedná se o třídu uchovávající data z odeslaného formuláře.

```
@Controller
@RequestMapping( "/myfiles" )
public class MyFilesController {
...
@RequestMapping( value = "", method = RequestMethod.GET )
public String myFilesIndex( Model model ) {
...
@RequestMapping( value = "/upload",
                 method = RequestMethod.POST )
public String uploadFile( UploadModel uploadModel ) {
...
```

Velice užitečnou anotací je Autowired. Slouží k automatickému přiřazení instance odpovídajícího datového typu. Pokud je například nutné, aby metody třídy pracovaly se záznamy uživatelů v databázi, jednoduše se toto zajistí použitím anotace `@Autowired` nad proměnnou typu UserDao.

```
@Autowired
private UserDao userDao;
```

5 VYDÁNÍ APLIKACE DO PRODUKCE

Tato kapitola se zabývá přípravou produkčního prostředí a nasazení vytvořené aplikace na toto prostředí.

5.1 Server

Pro potřeby zveřejnění této práce byl správci na Fakultě elektrotechniky a komunikačních technologií VUT připraven virtuální server se systémem `Ubuntu Server 12.10 (32 bit verze)`, samozřejmě bez grafického prostředí. Na serveru byly založeny základní uživatelské účty, přiřazena veřejná IP adresa a spuštěn SSH server pro vzdálený přístup.

Pro možnost přístupu veřejnosti mimo síť VUT byla také na firewallu přidána vyjímka pro tento server.

5.2 Příprava programového vybavení

Operační systém Ubuntu patří k nejrozšířenějším linuxovým distribucím. Systém Ubuntu, včetně jeho derivací, disponuje balíčkovacím systémem APT a rozsáhlým repozitářem obsahujícím tisíce softwarových balíčků.

Instalaci balíčků obstarává konzolový nástroj **aptitude**. Tento program při instalaci balíčku kontroluje jeho závislosti a nabízí uživateli instalaci těchto závislých balíčků.

Při přípravě byl nejdříve instalován základní balíček Javy **openjdk-7-jre** včetně jeho závislostí. Druhým v pořadí byla příprava Java webového aplikačního serveru Tomcat, která byla provedena instalací balíčku **tomcat7** a samozřejmě s instalací závislostí.

Konfigurace serveru Tomcat probíhala podle manuálu na oficiálních stránkách a jiných návodů. Konfigurační soubory se nachází v adresáři **/etc/tomcat7**. Konkrétně se jedná o hlavní konfigurační soubor **server.xml**.

Následovala instalace databázového serveru **MySQL** pomocí balíčku **mysql-server**. Při instalaci bylo pouze nutné nastavit administrátorské heslo k databázi. Další nastavení probíhalo již pomocí konzolového rozhraní k tomu určenému a pomocí SQL dotazů. Takto byl nakonfigurován nový účet pro připojení aplikace k databázi. Prvotní naplnění nově vytvořené databáze bylo provedeno exportem struktury vývojové databáze včetně všech tabulek a následným importem na serveru.

5.3 Zveřejnění aplikace

Nahrání aplikace na server předcházelo zkompilování zdrojových kódů a přibalení potřebných knihoven. Výsledný WAR soubor byl nakopírován na server a umístěn do adresáře **/var/lib/tomcat7/webapps/**, kde ho Tomcat detekuje a postará se o jeho publikaci.

Aplikace je dostupná na veřejné webové adrese `http://147.229.149.9/`.

5.4 Testování

Aplikace jako celek byla po vydání testována dvěma nezávislými osobami. Byla testována funkčnost a chování jednotlivých nástrojů pro koncového uživatele. Testování probíhalo v několika běžně používaných prohlížečích, kterými jsou zejména Google Chrome, Mozilla Firefox a Opera. Bohužel byly mezi prohlížeči nalezeny rozdíly ve vykreslování formulářových prvků. Vzhled těchto prvků byl úpravou v jejich nastavení sjednocen. Rozdíly chování v různých prohlížečích byly také nalezeny ve zpracování JavaScriptu, kdy vždy některý z prohlížečů neakceptoval zapsané funkce. Oprava obnášela nalezení jiných obdobných funkcí, které jsou dostupné a funkční ve všech prohlížečích.

Důležitější částí testování bylo ověření bezpečnosti jednotlivých nástrojů, zejména možnost neoprávněného přístupu a využívání funkcí uživatelem bez patřičných práv. Bylo ověřováno, aby nepřihlášený uživatel nemohl využívat funkce přihlášeného, a aby přihlášený uživatel bez administrátorských oprávnění nemohl využívat a přistupovat k nástrojům určeným administrátorovi.

Všechny nalezené chyby byly reportovány a po jejich opravě a vydání došlo k opětovnému přetestování.

6 ZÁVĚR

V této práci byla vytvořena aplikace se stabilním zázemím jazyka Java pro odevzdávání a vyhodnocování soutěžních výstupů. Pro všechny uživatele je k dispozici příjemné a intuitivní prostředí. Nahraná data se automaticky zpracovávají a výsledky jsou přehledně viditelné na jedné stránce.

Hlavním přínosem práce je vytvoření aplikace jako centrálního bodu, kde mohou soutěžící i administrátoři vidět pokrok všech prací a porovnávat výsledky mezi sebou.

Aplikace je nyní plnohodnotným prostředím pro pořádání soutěže pro vývoj algoritmu zpracovávající biomedicínská data, který může v budoucnu dopomoci i k záchraně lidského života. Zdrojové kódy jsou psané podle návrhu MVC, což umožňuje bezproblémové rozšíření nebo úpravu pro většinu vývojárů. Přínosem je také vytvořená programátorská dokumentace a diagramy, které se většinou vytvářejí pouze pro větší aplikace.

Použitím moderních technoligí bylo dosaženo určité úrovně bezpečnosti a například použitím Ajaxu se zvýšila interaktivita dostupných nástrojů.

Po zveřejnění byla otestována správná funkčnost a chování celku dvěma nezávislými osobami. Testována byla také bezpečnost a implementovaná omezení přístupu pro jednotlivé kategorie uživatelů.

Do budoucna může být aplikace upravena i pro jiné soutěže nebo rozšířena tak, aby se stala univerzálnějším prostředím pro pořádání několika soutěží současně. Je zde také prostor pro doimplementaci dalších možností pro uživatele i nástrojů pro administrátory a možnosti pro další automatizaci (statistiky, zálohování).

LITERATURA

[1] Burget Radim. *Challenge 3 – Brain Tissue Analysis* [online]. c2011, [cit. 2012-11-17]. URL: <http://splab.cz/ch3>

[2] The PHP Group. *PHP Manual* [online]. c2013, [cit. 2013-04-13]. URL: <http://cz2.php.net/manual/en/index.php>

[3] Ruby on Rails. *RubyOnRails.cz* [online]. c2013, [cit. 2013-04-13]. URL: <http://rubyonrails.cz/>

[4] Red Hat. *Jboss Application Server 7* [online]. c2012, [cit. 2012-11-17]. URL: <http://www.jboss.org/jbossas>

[5] Večeřa, Martin. *Jboss: Aplikační server* [online]. 2008-02-18, [cit. 2012-11-17]. URL: <http://www.root.cz/clanky/jboss-aplikacni-server/>

[6] The Apache Software Foundation. *Apache Tomcat* [online]. c2012, [cit. 2012-11-17]. URL: <http://tomcat.apache.org/>

[7] SpringSource. *SpringSource.org* [online]. c2012, [cit. 2012-11-17]. URL: <http://www.springsource.org/>

[8] Zapletal, Lukáš. *Apache Struts 1.1* [online]. 2003-05-06, [cit. 2012-11-17]. URL: <http://www.root.cz/clanky/apache-struts-1-1/>

[9] java.net. *GlassFish Product Documentation* [online]. c2012, [cit. 2012-11-17]. URL: <http://glassfish.java.net/docs/index.html>

[10] Bollinger Gary a Bharathi Natarajan. *JSP Java Server Pages: podrobný průvodce začínajícího tvůrce*. Praha : Grada, 2003. ISBN 80-247-0340-8. s. 418.

[11] Oracle. *Java EE Reference* [online]. c2012, [cit. 2012-11-17]. URL: <http://www.oracle.com/technetwork/java/javaee/documentation/index.html>

[12] Patel Viral. *Java Virtual Machine, An inside story!!* [online]. c2008, [cit. 2013-05-10]. URL: <http://viralpatel.net/blogs/java-virtual-machine-an-inside-story/>

[13] Skoumal Vladislav. *Nástroje pro podporu vývoje na platformě Java* [online]. 2010-01-23, [cit. 2012-11-21]. URL: <http://www.skoumal.net/cs/nastroje-pro-podporu-vyvoje-na-platforme-java>

[14] Matyáš Václav, Krhovják Jan. *Autentizace a identifikace uživatelů* [online]. 2011-11-14, [cit. 2012-11-21]. URL: <http://www.ics.muni.cz/bulletin/articles/560.html>

[15] Řepa Václav. *Analýza a návrh informačních systémů*. Praha : Ekopress, 1999. ISBN 80-86119-13-0. s. 403.

[16] Melichar Jan. *Datové modelování podruhé – Databázový svět* [online]. 2002-04-12, [cit. 2012-12-01]. URL: <http://www.dbsvet.cz/view.php?cisloclanku=2002041201>

[17] Oracle. *MySQL 5.5 Reference Manual* [online]. c2012, [cit. 2012-11-17]. URL: <http://dev.mysql.com/doc/refman/5.5/en/index.html>

[18] Pemberton Steven. *W3C - World Wide Web Consortium* [online]. c2012, [cit. 2013-05-10]. URL: <www.w3c.org/>

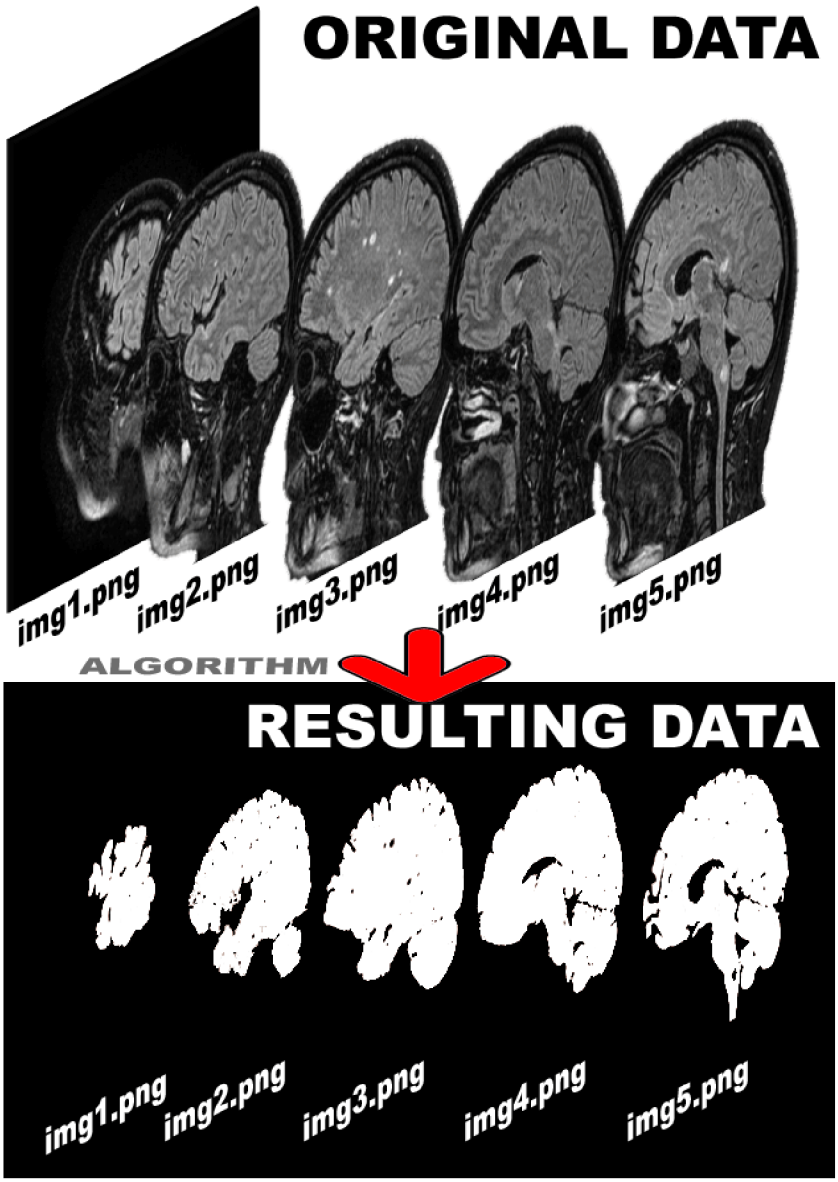
[19] Sauron Software. *Cron4j - a pure Java cron-like scheduler* [online]. c2012, [cit. 2012-11-17]. URL: <www.sauronsoftware.it/projects/cron4j/>

[20] Smith Briant. *CSS Templates and XHTML Template for free* [online]. c2012, [cit. 2012-12-01]. URL: <http://bryantsmith.com/template/#afarewelltocolor>

[21] MySQL. *MySQL Workbench 5.2: Visual Database Design* [online]. c2012, [cit. 2012-12-05]. URL: <http://www.mysql.com/products/workbench/design/>

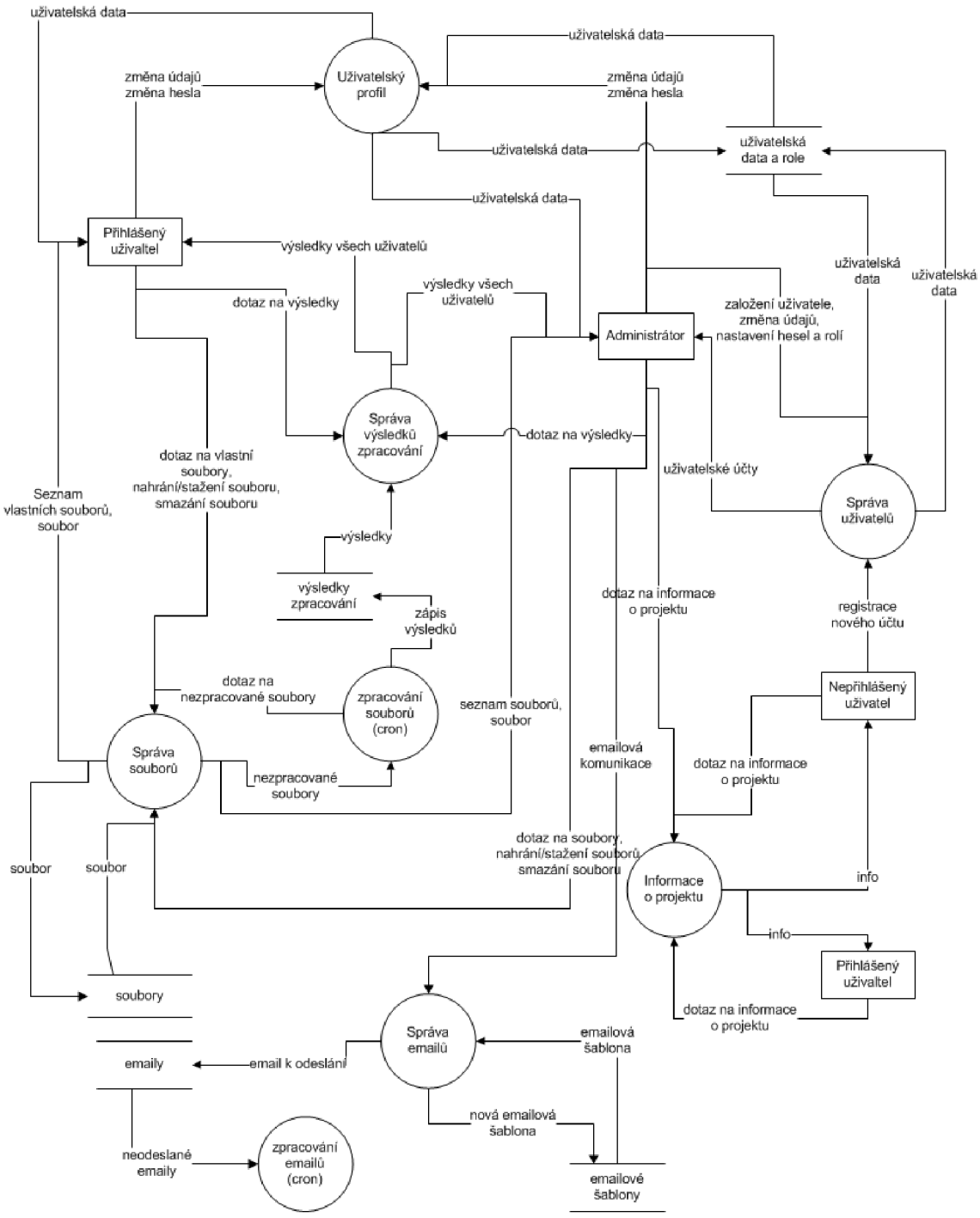
SEZNAM PŘÍLOH

A ZNÁZORNĚNÍ TRANSFORMACE DAT



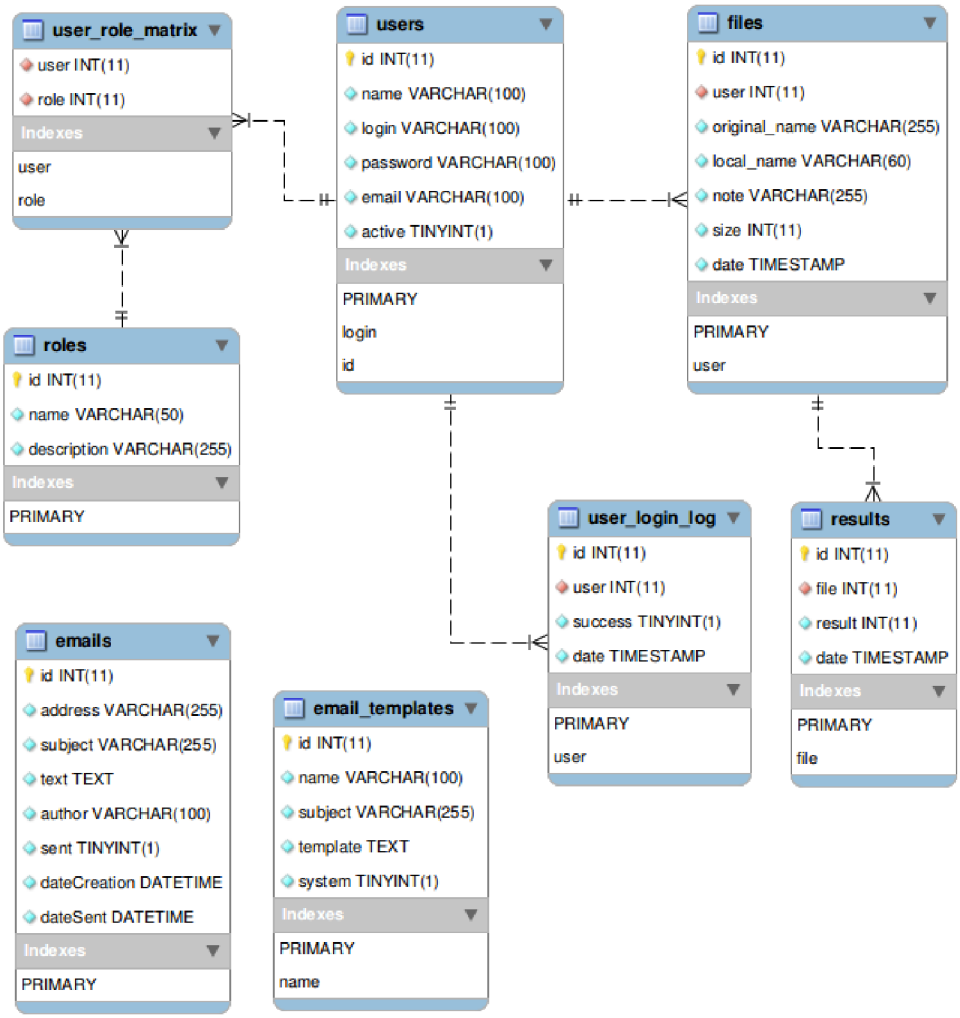
Obr. A.1: Vzorová transformace dat. Převzato a použito se svolením autora ze stránek SPLab projektu.

B DIAGRAM DATOVÝCH TOKŮ



Obr. B.1: Diagram datových toků

C POPIS DATABÁZOVÝCH TABULEK



Obr. C.1: Entitně relační diagram včetně datových typů a indexů

Všechny tabulky (mimo vazební user_role_matrix) obsahují číselný primární index, přes který jsou záznamy jednoznačně identifikované.

Tabulka **users** obsahuje záznamy o uživatelích aplikace. Skládá se z textových polí name, login a email a pole password obsahující hash hesla a soli. Navíc je v číselném poli active je uložený příznak o aktivování uživatele.

Tabulka **roles** obsahuje textová pole name a description a je propojena s tabulkou users přes vazební tabulku **user_role_matrix**. Takto je vyřešena vazba m:n mezi uživateli a rolemi, kdy jeden uživatel může mít více rolí a jedna role může být přiřazena více uživatelům.

Tabulka **user_login_log** uchovává pokusy o přihlášení a skládá se z pole user (vazba na záznam uživatele), pole pro datum a čas a pole success obsahující identifikaci úspěšného nebo neúspěšného přihlášení.

Tabulka **files** uchovává záznamy o nahraných souborech. Obsahuje položky user (vazba na záznam uživatele), original_name (originální název nahrávaného souboru), local_name (název souboru na serveru), note (volitelná poznámka), size (velikost souboru v bytech) a date (datum a čas nahrání).

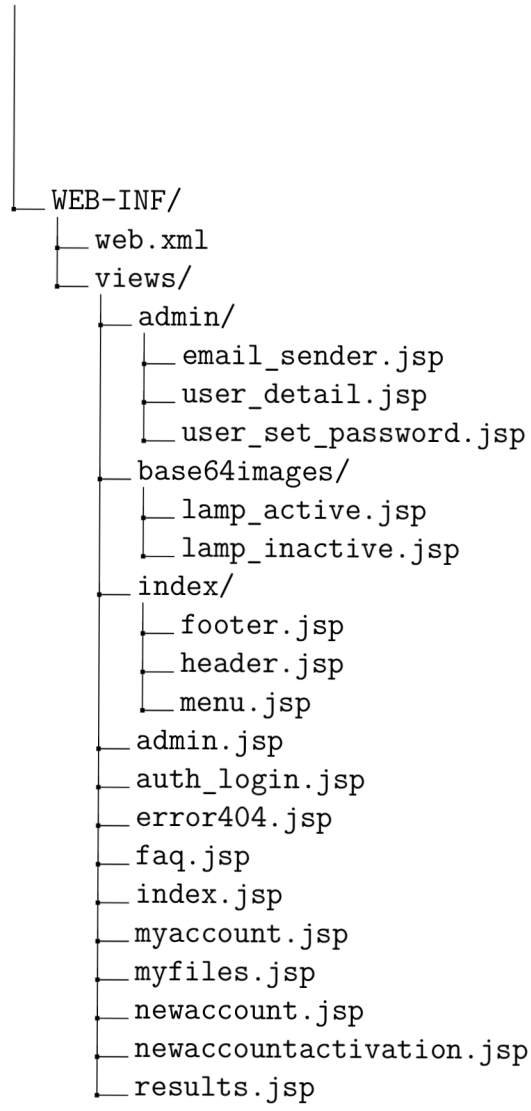
Tabulka **results** je napojena na tabulku files přes cizí klíč file a uchovává záznamy o zpracování souboru. Mimo vazby na soubor obsahuje dále číselný výsledek v poli result a datum zpracování.

V tabulce **emails** jsou uloženy emailové zprávy odesílané z aplikace. Pro jednotlivé záznamy jsou uchovávány adresa příjemce (address), předmět zprávy (subject), obsah zprávy (text), jméno odesilatele (author), příznak úspěšného odeslání (sent) a datum vytvoření a odeslání (dateCreation, dateSent).

Poslední tabulkou je **email_templates**, která uchovává vytvořené emailové šablony. Uchovává se název šablony (name), předmět zprávy (subject), text zprávy (template) a příznak systémové šablony (system).

D STRUKTURA ZDROJOVÝCH KÓDŮ

```
projectDirectory/
├── src/java/cz/vutbr/feec/utko/
│   ├── auth/
│   │   ├── Role.java
│   │   ├── User.java
│   │   ├── UserDao.java
│   │   ├── UserModel.java
│   │   └── UserSessionService.java
│   ├── cron/
│   │   ├── Constants.java
│   │   ├── MyTaskController.java
│   │   ├── SchedulerServletContextListener.java
│   │   ├── TaskEmailSender.java
│   │   └── TaskFileProcessing.java
│   ├── emailer/
│   │   ├── EmailMsg.java
│   │   ├── EmailSenderModel.java
│   │   ├── EmailTpl.java
│   │   ├── Emailer.java
│   │   └── EmailerDao.java
│   ├── files/
│   │   ├── FileDao.java
│   │   └── File.java
│   ├── filter/
│   │   └── UserLoggedFilter.java
│   ├── results/
│   │   ├── Result.java
│   │   └── ResultDao.java
│   ├── AdminController.java
│   ├── AuthController.java
│   ├── FaqController.java
│   ├── GlobalEnvironment.java
│   ├── IndexController.java
│   ├── MyAccountController.java
│   ├── MyFilesController.java
│   ├── NewAccountActivationController.java
│   ├── NewAccountController.java
│   └── ResultsController.java
└── web/
    ├── static/
    │   ├── css/
    │   │   └── style.css
    │   └── img/
    │       └── <soubory pro design - obrázky>
```


WEB-INF/
web.xml
views/
admin/
email_sender.jsp
user_detail.jsp
user_set_password.jsp
base64images/
lamp_active.jsp
lamp_inactive.jsp
index/
footer.jsp
header.jsp
menu.jsp
admin.jsp
auth_login.jsp
error404.jsp
faq.jsp
index.jsp
myaccount.jsp
myfiles.jsp
newaccount.jsp
newaccountactivation.jsp
results.jsp