Online systém pro nahrávání a správu objemných souborů

Bc. Petr Koliba

Diplomová práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Koliba**
Osobní číslo:	**A16543**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Online systém pro nahrávání a správu objemných souborů**
Téma anglicky:	**An Online System for Uploading and Managing Big Files**

Zásady pro vypracování:

1. **Proveďte výběr technologií řešení dle požadovaných funkcí systému**
2. **Navrhněte systém a jeho uživatelské rozhraní**
3. **Specifikujte hardwarovou konfiguraci pro běh systému dle specifikace zadavatele**
4. **Implementujte systém v testovacím prostředí**
5. **Proveďte testy a reálné ověření funkčnosti systému**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **SCHILDT, Herbert.Mistrovství - Java. Brno: Computer Press, 2014. Mistrovství. ISBN 978-80-251-4145-8.**
2. **BLOCH, Joshua.Effective Java. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, c2008. ISBN 978-0-321-35668-0.**
3. **FRÄNKEL, Nicolas. Learning Vaadin 7 master the full range of web development features powered by Vaadin-build rich Internet applications. 2nd ed. Birmingham, UK: Packt Pub, 2013. ISBN 9781782169772.**
4. **SCHWARTZ, Baron., Peter. ZAITSEV a Vadim. TKACHENKO.High performance MySQL. 3rd ed. Cambridge [Mass.]: O'Reilly, c2012. ISBN 978-1449314286.**
5. **SHELDON, Robert.SQL: začínáme programovat. Praha: Grada, 2005. Průvodce (Grada). ISBN 80-247-0999-6.**

Vedoucí diplomové práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **1. prosince 2017**

Termín odevzdání diplomové práce: **16. května 2018**

Ve Zlíně dne 11. prosince 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Lašek, Ph.D.
garant oboru

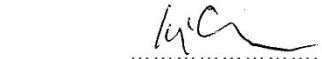
Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 14. 5. 2018



podpis diplomanta

ABSTRAKT

Cílem práce je navrhnout a implementovat aplikaci určenou ke stahování velkých souborů ze serveru na klienta a naopak. Verifikace správnosti souborů probíhá pomocí hashovacích funkcí. Uživatelé jsou rozděleni na dvě role: administrátor a běžný uživatel. Administrátorské rozhraní umožňuje správu souborů a uživatelů. Informace o odeslaných souborech jsou ukládány do databáze pro možnou pozdější kontrolu.

Klíčová slova:

Java, Vaadin, MySQL, JPA, velký soubor, hash, SHA-1, MD5

ABSTRACT

The goal of this thesis is to design and develop application to manage downloads of big files from server to client and vice versa. Verification of files is done trough hash functions. There are two roles for users: admin and common user. Administrator‘s view allows management of files and users. Informations about uploaded files are stored in database for possible future check.

Keywords:

Java, Vaadin, MySQL, JPA, big file, hash, SHA-1, MD5

Děkuji vedoucímu práce Ing. Davidu Malaníkovi, Ph.D. za rady a doporučení při psaní práce. Dále děkuji svým rodičům za velkou podporu a děkuji všem, kteří se mnou museli mít trpělivost v době, kdy práce vznikala.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tato diplomová práce se věnuje návrhu, vývoji a testům aplikace, která má sloužit jako softwarová podpora ve cvičeních předmětu Kybernetická bezpečnost. V tomto předmětu si studenti od vyučujícího stahují a odevzdávají zpět objemné soubory v řádech několika gigabytů. Předkládaná aplikace tuto činnost zefektivní a zjednoduší. Aplikace tak bude umožňovat lektorovi nabídnout studentům soubory, se kterými budou pracovat. Lektor bude mít možnost tyto soubory třídit a studenti stahovat, odevzdávat a zobrazovat již odevzdané soubory s časovým označením a hashem souboru. Autentifikace bude zařízena pomocí loginu a hesla, autorizace bude mít dva stupně: uživatelský a administrátorský.

V teoretické části práce se seznámíme s jednotlivými technologiemi, které budou při vývoji použity, a s jejich výběrem, a to programovacím jazykem, vývojovým prostředím, frameworky, a dále použitým aplikačním serverem a databází. Dalším krokem bude návrh uživatelského prostředí pro jednotlivé role a návrh hardwarového řešení, na kterém by byl zajištěn plynulý provoz aplikace. V praktické části budou popsány použité komponenty UI, které budou v projektu použity a konkrétní postup při vývoji UI s ukázkami kódu. Dále budou popsány vytvořené stránky a jejich funkčnosti. Poté bude následovat popis nasazení aplikace na testovací systém a testy vedoucí k potvrzení funkčnosti.

Cílem práce je návrh a implementace úložiště objemných souborů, které umožní nahrávání a stahování přes webové rozhraní, jejich správu včetně hromadných operací jako kopírování či přesun.

I. TEORETICKÁ ČÁST

1 VÝBĚR A POPIS POUŽITÝCH TECHNOLOGIÍ

1.1 Debian Linux

Na základě požadavku zadání práce, bude aplikace implementována na operačním systému Linux, konktrétně jeho distribuci Debian, který je vyvíjen pod licencí GNU. Celý název distribuce je Debian GNU/Linux. Projekt Debian založil v roce 1993 Ian Murdock (stabilní verze pak byla dostupná od roku 1996). Od počátku jej zamýšlel jako otevřenou distribuci, nicméně každá verze má svého project leadera, který se stará o koordinaci prací komunity [13]. Aktuální verze systému je 9. Tato distribuce obsahuje velký balík aplikací, vytvořených převážně komunitou. Z Debianu vychází populární distribuce Ubuntu a jeho variace nebo Knoppix, které lze stáhnou a používat zdarma. K dispozici jsou porty pro různé architektury, od běžných x86 32bit a 64bit, ARM architekturu až po PowerPC či zSystem mainframy.

1.2 Výběr programovacího jazyka

Pro práci na projektu připadaly v úvahu prakticky dvě možnosti: použití jazyku PHP nebo Java. Obě možnosti jsou bez problému implementovatelné pod Linuxem

1.2.1 PHP

Jazyk PHP je open source skriptovací jazyk, používaný pro vývoj webových aplikací, a může být zapouzdřen do HTML kódu. Jeho výhodu je, že na rozdíl od např. Java Scriptu běží na straně serveru a nikoliv klienta. Ten obdrží jen výsledky, ale není pro něj viditelný PHP kód samotný. Velkou výhodou tohoto jazyka je jeho jednoduchost [14].

Používá se především ve třech oblastech:

- Skriptování na straně serveru
- Skriptování v příkazovém řádku
- Psaní desktopových aplikací

PHP může být použito na všech hlavních operačních systémech, Windows a Linux systémy jsou samozřejmostí. Má podporu většiny webových serverů, zejména nejpoužívanějšího Apache nebo IIS od Microsoftu [15].

Důležitou součástí PHP je jeho spolupráce se všemi nejpoužívanějšími standardy databází.

1.2.2 Java

Jazyk Java vznikl v roce 1991 v Sun Microsystems. Původním záměrem bylo vytvořit multiplatformní programovací jazyk, který by byl používán ve spotřební elektronice. Nevýhodou jazyků C/C++ je, že musí být kompilovány pro konkrétní procesor, v případě Javy by tato nevýhoda odpadla. Záležitost přenositelnosti a bezpečnosti byla vyřešena bytekódem, kdy nedochází ke kompilaci přímo do spustitelného kódu, ale to vysoce optimalizovaného kódu, který je spouštěn v prostření JVM (Java Virtual Machine). Předpokladem pro spuštění kódu napsaného v Javě je tak existence JVM pro danou platformu. Spouštění programu ve virtuálním stroji též zlepšuje bezpečnost, protože virtuální stroj může kontrolovat aktivity programu, pokud by se o ně pokoušel mimo tento stroj. Daní za multiplatformnost je snížení výkonu, ke kterému dochází při zpracování bytekódu virtuálním strojem, nicméně tento rozdíl není markantně veliký v porovnání s kompilovanými jazyky [1].

1.2.2.1 Applety

Applet je program napsaný v Javě, který je určen ke spouštění ve webových prohlížečích podporujících Javu. Jejich hlavním účelem je přesunutí funkcionality ze strany serveru na klienta. Javovské applety výrazně rozšířily možnosti dynamických webových stránek. Jejich nevýhodou byla bezpečnost, protože si uživatel na svůj počítač stahoval program, který mohl obsahovat škodlivý kód. Ochrana uživatele je zabezpečena během programu uvnitř virtuálního stroje, nicméně ten mohl obsahovat bezpečnostní díry, které mohl malware využít [1].

1.2.2.2 Servlety

Servlet je javovský program, který je spouštěn na straně serveru a rozšiřuje tím tak jeho dynamické možnosti. Nyní tak bylo možné dosáhnout dynamičnosti jak na straně klienta pomocí appletů, tak na straně serveru pomocí servletů.

Servlety jsou opět přenositelné mezi platformami, jedinými podmínkami je podpora JVM a aplikační server s podporou kontejnerů pro servlety, jako například Apache Tomcat [1].

1.3 Výběr vývojového prostředí

1.3.1 Eclipse

Eclipse je vývojové prostředí, které začalo vyvíjet IBM koncem 90. let. V roce 2001 přešlo na open source licencování, čímž bylo podpořeno rozšíření a adaptace mezi vývojáři.

Vznikla nezisková organizace Eclipse, která převzala starost nad dalším vývojem produktu [16].

Eclipse nabízí robustní prostředí pro vývoj primárně v jazyku Java, ale v současné době podporuje i další jazyky pomocí systému pluginů: C, C++, C#, PHP, Python a mnohé další. Právě pluginy, které tvoří důležitou součást Eclipse, umožňují rozšiřitelnost o další obsah, ať už zpracování webu, práci s grafikou, videem, nebo uživatelským rozhraním.

1.3.2 NetBeans

NetBeans začal původně jako studentský projekt v roce 1996. Cílem bylo vytvořit integrované prostředí pro vývoj v jazyce Java, napsaným právě v Javě. Začala stoupat jeho obliba mezi vývojáři a rozšiřovala se nabídka pluginů, stejně jako v Eclipse. NetBeans bylo adaptováno firmou Sun Microsystems a na nějakou dobu bylo přejmenováno na *Forté for Java.* Poté byla přijata myšlenka přechodu na Open source licencování. Po akvizici Sunu firmou Oracle je používán jako oficiální nástroj pro vývoj v Javě a je dále efektivně rozvíjen [17].

1.3.3 IntelliJ IDEA

IntelliJ IDEA je na rozdíl od předcházejících dvou komerční prostředí primárně určené pro Javu. Vývoj probíhal firmou JetBrains od roku 2000. K dispozici je Community edition, která je zdarma, nicméně její použití se sebou nese řadu omezení. Ultimate edition je ryze komerční verze s podporou dalších jazyků, verzovacích systémů a frameworků.

Jde o mocný nástroj snažící se o maximalizaci produktivity vývojáře pomocí inteligentního napovídání pomocí analýzy kódu [18].

Pro vývoj zadané aplikace bude použito IDE Eclipse, které je zdarma, má podporu potřebných technologií jako podpora správy verzí, databáze MySQL a aplikačního serveru Apache Tomcat. Protředí NetBeans nabízí přibližně totéž, ale zde ve prospěch Eclipse rozhodla má předchozí zkušenost. IntelliJ IDEA je pravděpodobně nejkomplexnější vývojové prostředí na trhu, ale z výběru jej eliminuje jeho komerční povaha.

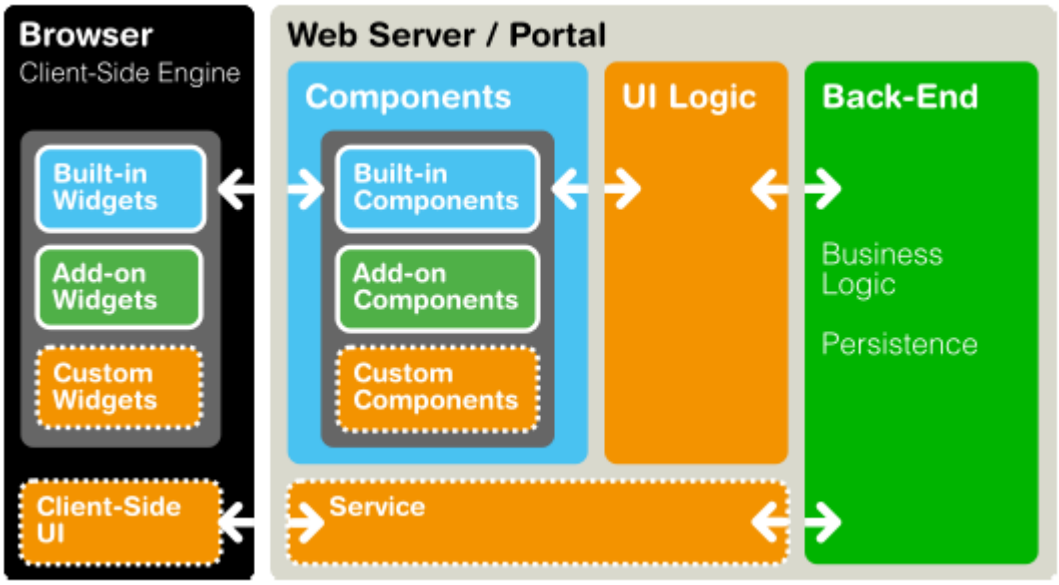
1.4 Použité frameworky

Pro vývoj webové aplikace je vhodné použít frameworky, který tento vývoj výrazně usnadní. Jejich cílem je automatizace často používaných činností, jako správy připojení, práce s databázemi a poskytnutí knihoven, tvorba uživatelského rozhraní a další.

Frameworky pro práci s webem zpravidla využívají model MVC (Model – View – Controller), kde je oddělen datový model od uživatelského rozhraní a business logiky. Jeho výhodou je zpřehlednění a modularita kódu a možnost jeho znovupoužití.

1.4.1 Vaadin

Vaadin Framework je určený pro vytváření a údržbu vysoce kvalitních webových uživatelských rozhraní. Podporuje dva programovací modely: serverový a klientský, kde serverový je mohutnější a umožňuje tvorbu UI, stejně jako na desktopové aplikaci, ale jednodušeji. Vaadin přistupuje k problematice tak, že odstraňuje potřebu znát HTML nebo JavaScript, které se starají o zobrazení stránek [8].



Obrázek 1: Architektura aplikace ve Vaadinu

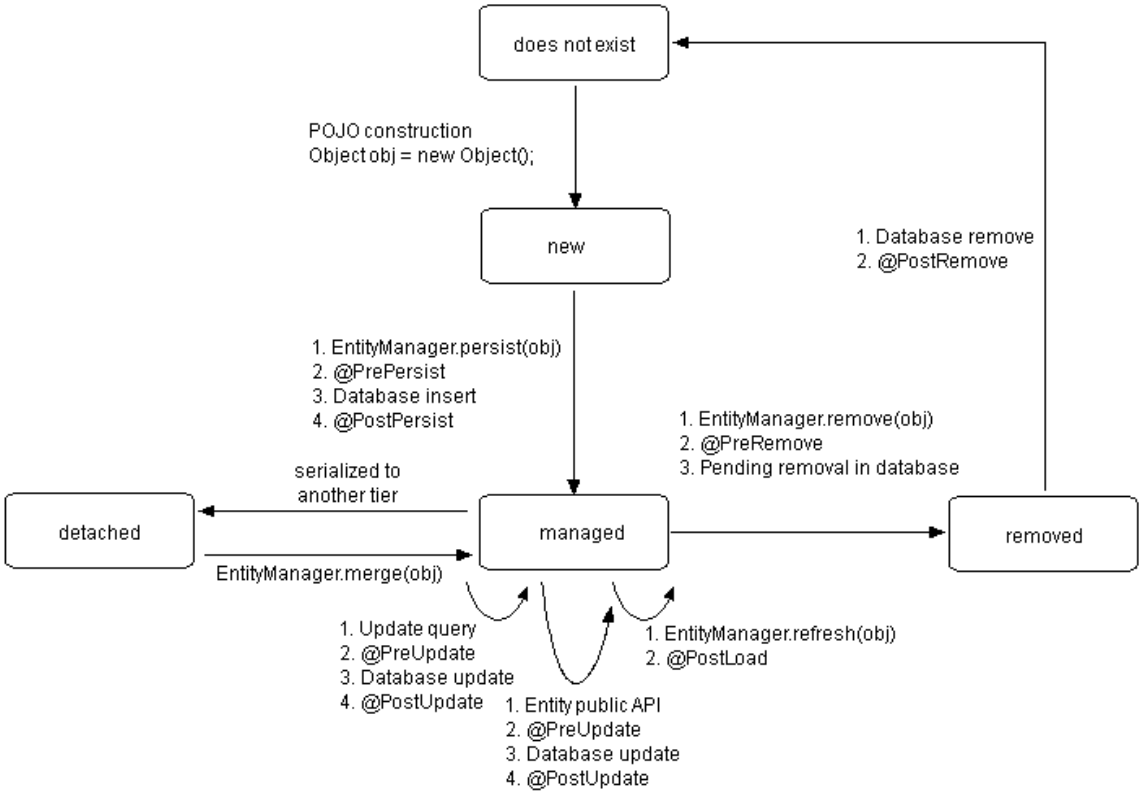
Serverový model se skládá ze serverového frameworku a klientského enginu, který běží v prohlížeči jako kód JavaScriptu, stará se o vykreslování uživatelského rozhraní a obstarává interakci uživatele se serverem. UI logika aplikace pak běží jako servlet na aplikačním serveru [8].

Klientský model se spouští jako JavaScript v prohlížeči, žádné další pluginy nejsou ke spuštění aplikací vytvořených pomocí Vaadinu potřeba. Využíván je Google Web Toolkit. Pro velké množství prohlížečů, které jej podporují vývojář nemá potřebu řešit problémy s kompatibilitou.

Vaadin není vázán na konkrétní IDE, ovšem nejlepší podporu má právě v Eclipse, což byl další z důvodů, proč zvolit toto vývojové prostředí. Jedinou nevýhodou je, že Vaadin Desinger, nástroj pro vizuální tvorbu UI je komerční, nicméně nevyužití této komponenty nebude vývoj aplikace výrazně komplikovat.

1.4.2 Java Persistence API

JPA je javovský framework, určený pro objektově-relační mapování při práci s databázemi. Při použití JPA je možno použít POJO třídu jako JPA entitu, která bude perzistentní k relační databázi a bude reprezentovat data v databázi. Entita používá anotaci *@Entity*, má alespoň jeden konstruktor, její atributy nejsou veřejné a ke každému atributu existuje getter a setter. Tato třída pak představuje jeden řádek v tabulce relační databáze [19].



Obrázek 2: Životní cyklus entity

1.4.3 Maven

Při vývoji byl také použit nástroj pro správu a automatizaci buildů Maven, který pracuje ve všech vývojových prostředích. Má vlastní repozitář pro většinu knihoven, kde jsou často dostupné i jejich zdrojové kódy. Výhodou je, že se o ně a jejich požadavky nemusíme starat, protože Maven zajistí splnění jejich závislostí. Nastavení projektu probíhá v souboru

pom.xml, kde jsou zásadní sekce <repositories> a <dependencies>. První z nich obsahuje informace o externích repozitářích, odkud je možné stahovat knihovny, druhá pak informace o vlastních knihovnách [20]. Záznamy vypadají následovně:

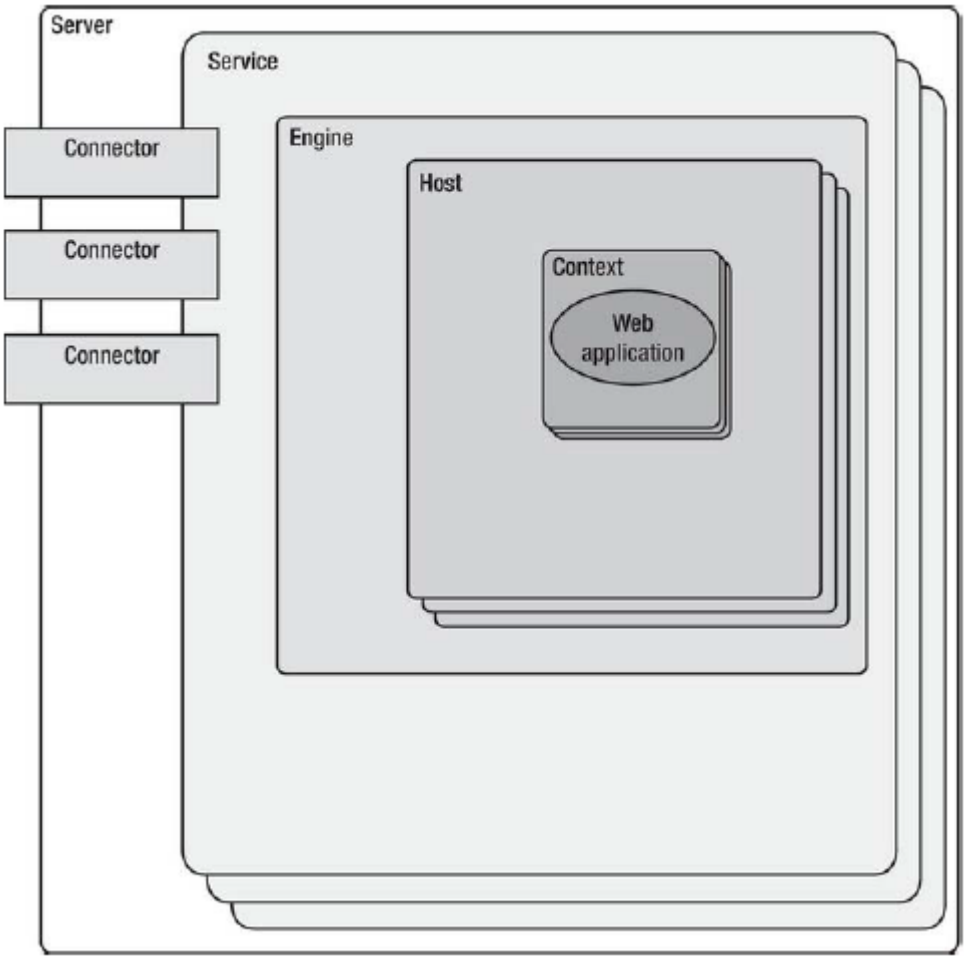
```
<repository>
       <id>vaadin-addons</id>
       <url>http://maven.vaadin.com/vaadin-addons</url>
</repository>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.5.1</version>
</dependency>
```

1.5 Apache Tomcat

Apache Tomcat je na Javě založený aplikační server, který umožňuje běh webových aplikací, obsahující servlety a Java Server Pages. Jde o open source projekt, který je udržován komunitou dobrovolníků. Má všechny nezbytné vlastnosti komerčních aplikačních serverů a v ničem za nimi nezaostává i přes to, že jde o komunitní produkt. Jeho součástí je Tomcat Manager, který obstarává správu webových aplikací a běží na Tomcat serveru z libovolného prohlížeče. Umožňuje start, ukončení, odstranění a další operace.

Hierarchicky nejvyšší jednotkou je instance Tomcatu, pouze jedna instance může běžet v jedné instanci JVM. Tímto se dosahuje toho, že pokud instance Tomcatu přestane pracovat, ostatní javovské aplikace budou fungovat i nadále. Instance Tomcatu se skládá z jednotlivých aplikačních kontejnerů, které mají jasně ustanovenou hierarchii [9].

Konkurenční servery jsou například WebSphere, GlassFish nebo Jetty.



Obrázek 3: Architektura Tomcatu [9]

1.6 MySQL

Jde o v současnosti nejoblíbenější open source systém pro správu databází, který je vyvíjen společností Oracle. Může běžet jak na pracovních stanicích, tak na serverech a vyžaduje minimální účast uživatele na provozu a údržbě. Původně byla vyvinuta pro manipulaci velkých databází a bývá úspěšně nasazena tam, kde je vyžadován vysoký výkon. Může běžet jako klient/server systém nebo jako zapouzdřená databáze.

MySQL má programové rozhraní pro mnoho programovacích jazyků jako Java, rodina jazyků C, PHP, Perl a další. Je schopna pojmout velké množství záznamů. Je znám případ, kdy databáze obsahovala 200 000 tabulek s 5 000 000 000 řádky [10]. Pro vyvíjenou aplikaci tedy představuje víc než solidní nástroj na uložení a manipulaci dat v databázi.

1.7 Shrnutí

Aplikaci vyvíjenou v rámci diplomové práce programuji v jazyce Java s použitím vývojového prostředí Eclipse. Pro tvorbu uživatelského rozhraní a zjednodušení kódu slouží Frameworky Vaadin ve verzi 7 a JPA pro práci s databázemi. Správu závislostí řeší Maven. Pro aplikační server jsem zvolil Apache Tomcat, systém pro správu databází MySQL. Ten, stejně jako databáze, na níž poběží aplikace, aplikační server a databáze, byli stanoveny zadáním diplomové práce.

2 NÁVRH SYSTÉMU A UŽIVATELSKÉHO ROZHRANÍ

Jedním z požadavků na moderní aplikaci je přehlednost a snadnost ovládání Proto volím framework Vaadin, který by měl zajistit příjemný vzhled uživatelského rozhraní a jeho snadnější tvorbu. Webové aplikace vyvíjené v Javě mají často nedostatky právě ve svém vzhledu, což je možno odstranit právě použitím vhodného frameworku.

2.1 Návrh aplikace

V diagramu jsou použiti dva aktoři: student a lektor. Student bude mít po nalogování do systému možnost uložit soubor na server (operace může trvat několik desítek minut vzhledem k velikosti souborů) a stáhnout si soubory zveřejněné lektorem. Každý soubor bude mít záznam v databázi a své atributy, mezi kterými je časové razítko, určující čas uložení, jméno studenta a vygenerovaný hash kód. Každý odevzdaný soubor se pojmenuje podle předem zadaných kritérií, aby byla možná identifikace autora a rovněž označen časovým razítkem. Student též bude mít možnost změnit si své přístupové údaje a možnost nechat si odeslat zapomenuté heslo na mail zadaný při registraci.

Lektor bude moci spravovat veškeré soubory, jak nahrané studenty, tak vlastní. Bude je moct třídit do složek odpovídajícím reálným adresářům na diskovém oddílu. Aplikace bude vlastně jakousi nadstavbou, která umožní lektorovi manipulaci se soubory přes webové rozhraní. V případě nefunkčnosti bude možné manipulovat se soubory „ručně“, ale bez dostupných informací z databáze. Implementované akce budou kopírování, přesunování a mazání souborů využívající javovské knihovny *Java.nio.file.**.

2.1.1 Rozdělení uživatelských rolí

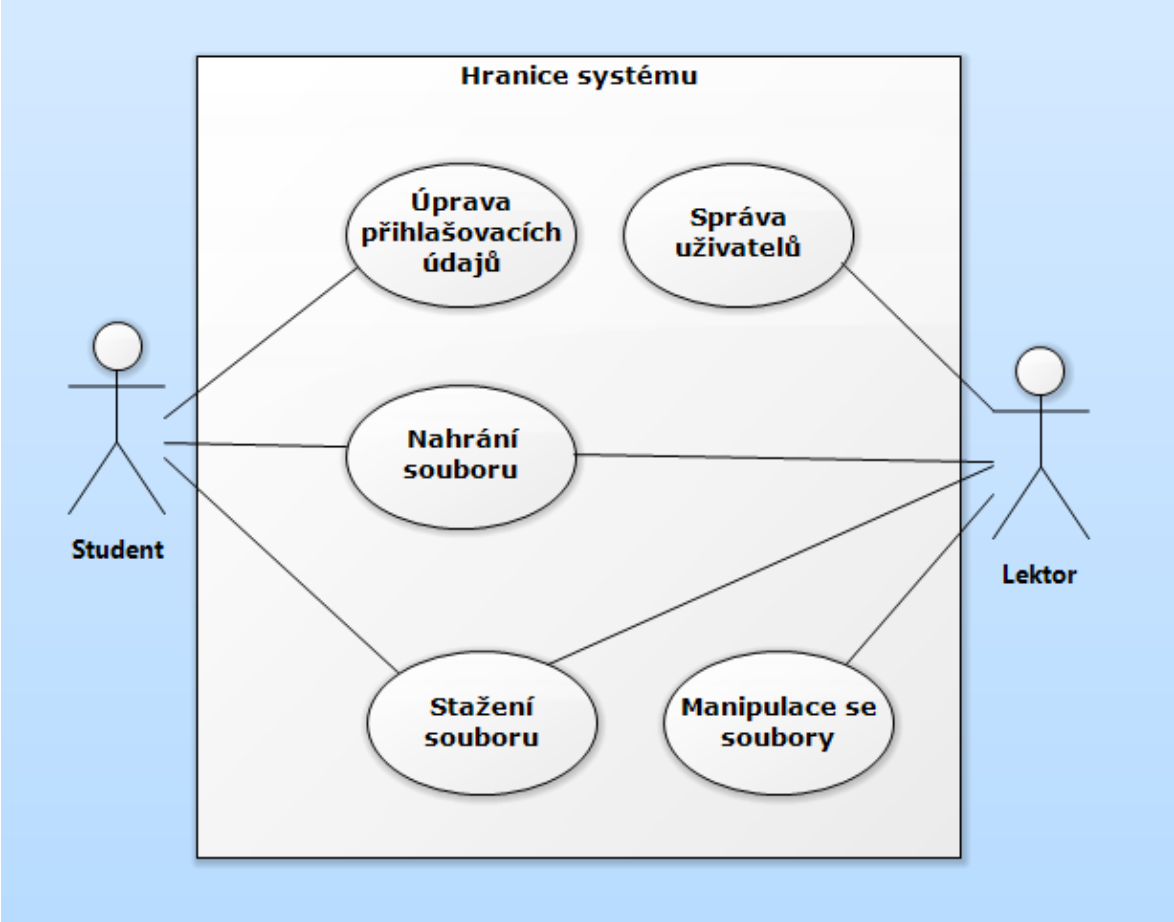
V aplikaci využijeme jen dvě úrovně: student/obyčejný uživatel a lektor/admin. Tyto role budou určeny záznamem v databázi u daného uživatele.

2.1.2 Use case diagram

Student bude mít k dispozici tři akce: nahrání, stažení souboru a úpravu přihlašovacích údajů.

Lektor bude mít tyto akce shodné a dvě akce navíc: manipulaci se soubory (zde je zahrnuto právě kopírování, přesouvání a mazání) a správu uživatelů.

UML diagram byl vytvořen v trial verzi programu Software Ideas Modeler, v. 11.45.

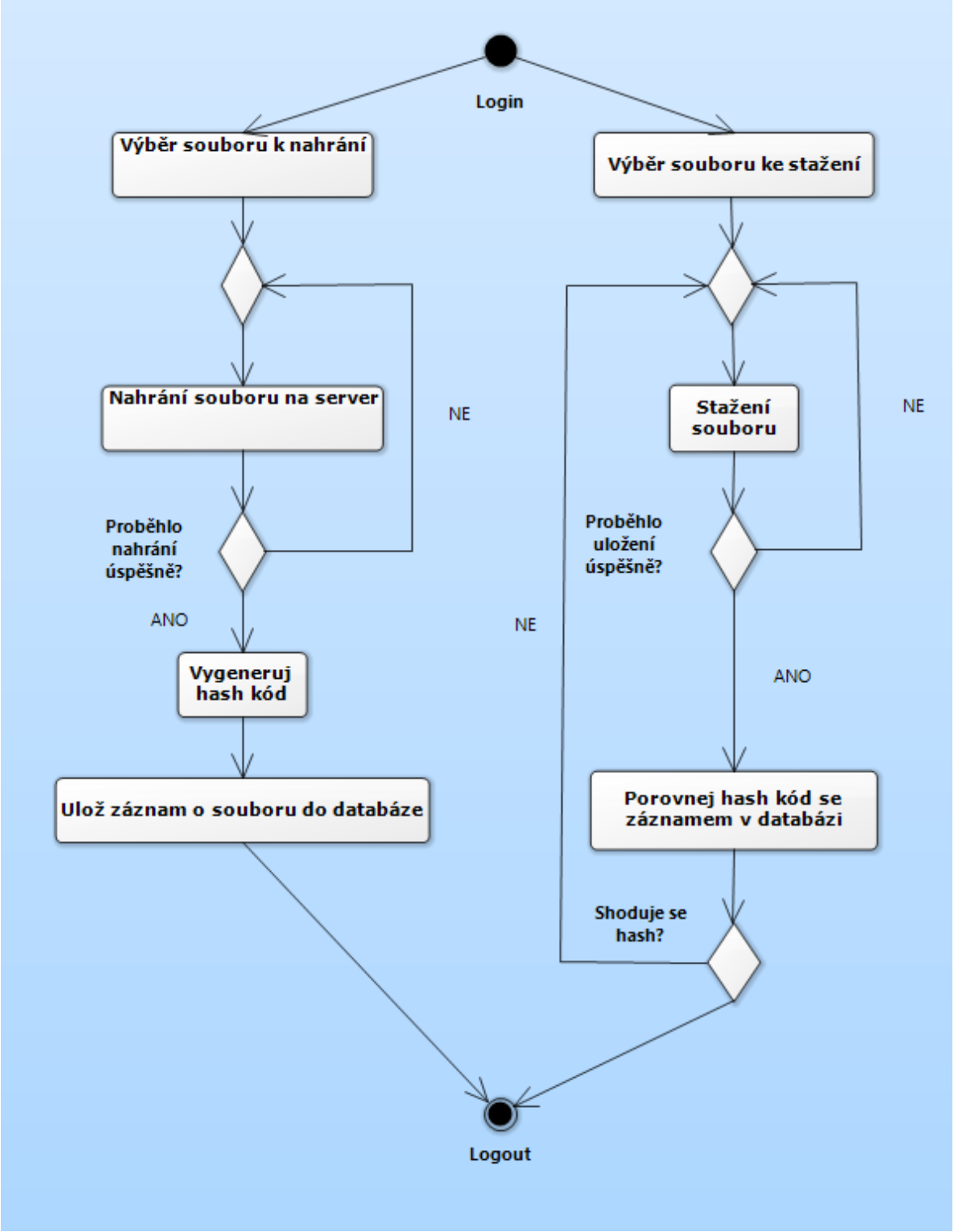


Obrázek 4: UML use case diagram

2.1.3 Diagram aktivit

Diagram aktivit zobrazuje dvě akce, které bude využívat student. Po nahrání souboru na server, které neskončí chybou se vygeneruje hash kód pro daný soubor a jeho další atributy spolu s cestou, kam byl soubor na disk uložen a tento záznam se uloží do databáze. Při stahování souborů ze serveru ke studentovi se bude ověřovat hash kód přijatého souboru se

záznamem v databázi, zda jde opravdu o nepoškozený soubor odpovídající originálu.



Obrázek 5: Diagram aktivit pro nahrání a uložení souboru.

2.1.4 Hashovací algoritmus

Hashovací funkce je algoritmus, který zpracovává prakticky neomezeně dlouhé řetězce tak, že výstupem je krátký kód předem dané délky. Principem funkčnosti je, že z cílového kódu nesmí být možné obnovit původní řetězec.

Použitý algoritmus MD5, který přišel na scénu v roce 1991 jako náhrada překonaného MD4 vytváří z libovolně velké vstupní množiny dat řetězec konstantní délky, který se používá mimo jiné pro kontrolu integrity dat. Vytváří se tak otisk malé velikosti, pomocí kterého můžeme určit, zda došlo v datech ke změně nebo nikoliv. Bohužel v dnešní době je algoritmus MD5 překonaný a jeho použití se nedoporučuje z důvodu bezpečnosti.

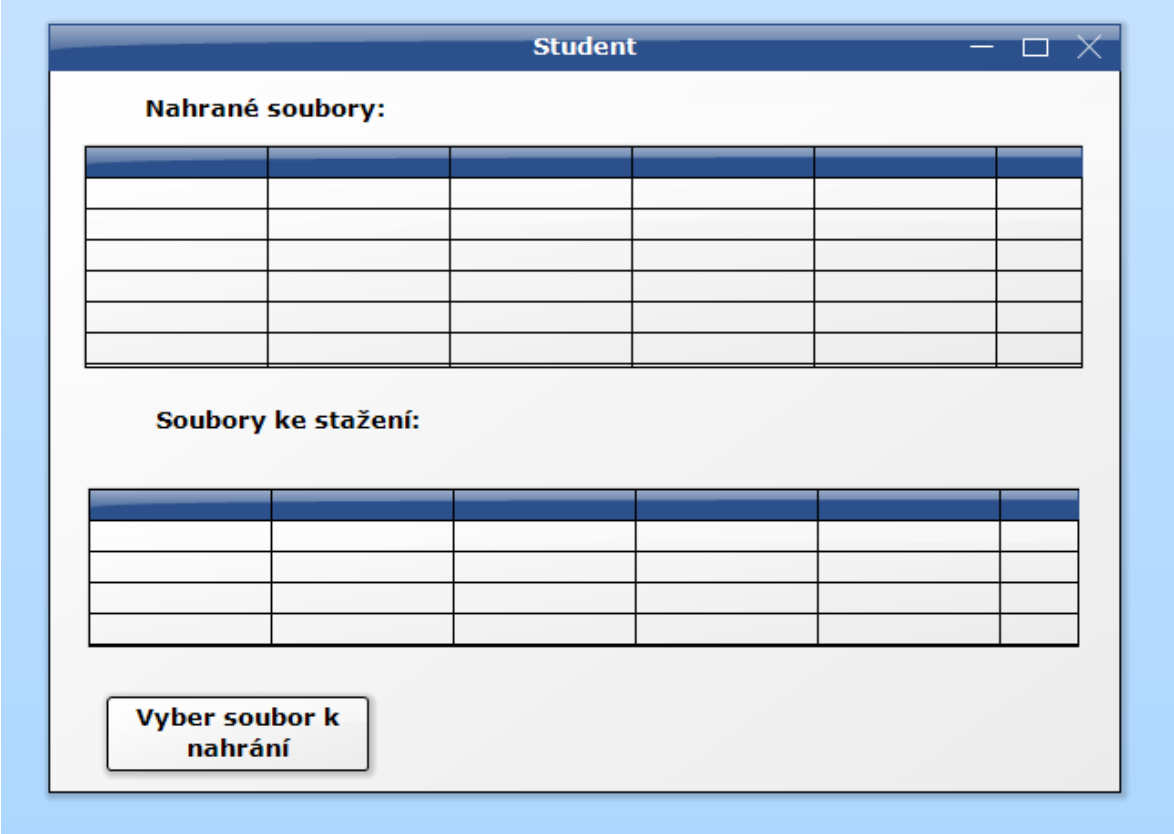
Dalším použitým algoritmem je SHA-1, který poskytuje na výstupu 40místný kód, byl uveden v roce 1993 americkou bezpečnostní agenturou NSA. Od roku 2005 je ovšem pokládán za zastaralý a doporučuje se použití bezpečnějších novějších verzí jako SHA-256 nebo SHA-512 [21].

2.2 Návrh uživatelského rozhraní

Úvodní obrazovka bude shodná pro obě role, až následující obrazovka po přihlášení bude odlišná.

2.2.1 Pohled studenta

Obrazovka pro uživatele se studentskou autorizací bude rozdělena na dvě části: jedna zobrazuje historii souborů, které student do systému nahrál v minulosti včetně jejich atributů, druhá část umožní zobrazit soubory aktuálně dostupné ke stažení. Dále bude dostupné tlačítko pro nahrání souboru.

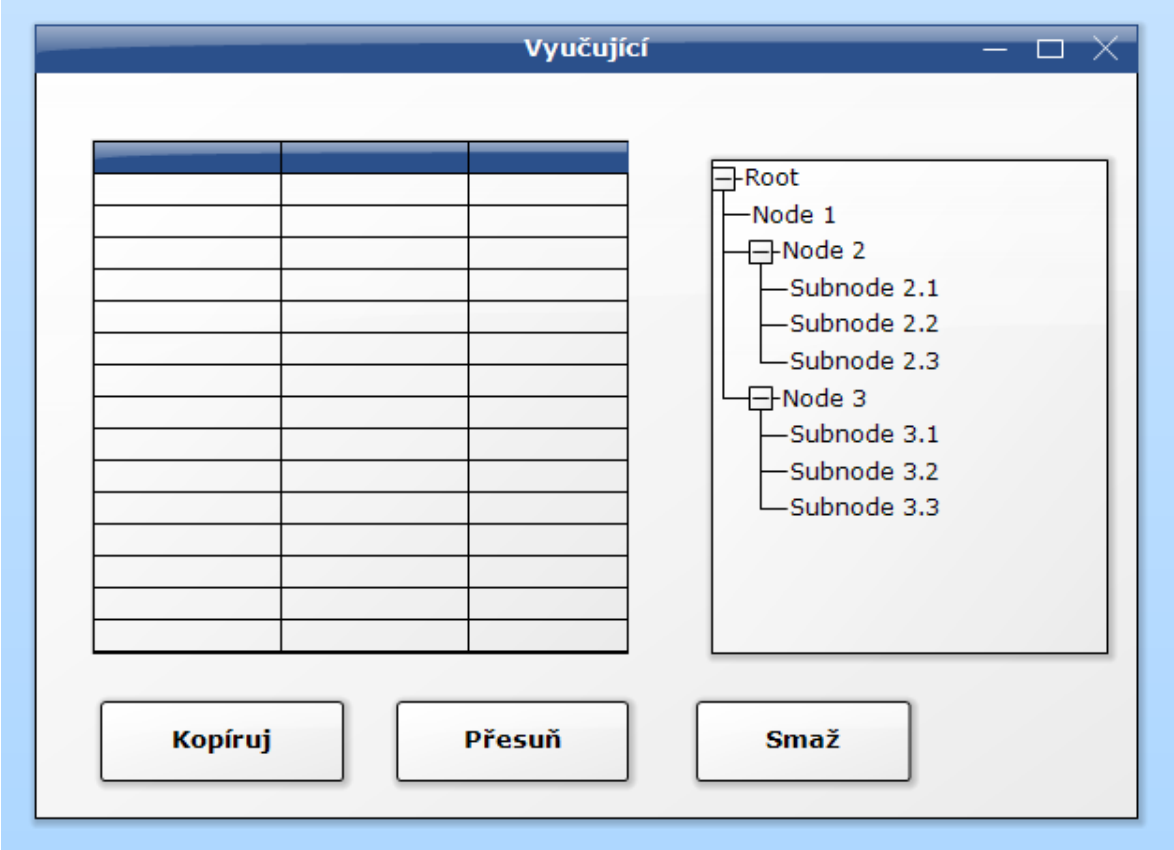


Obrázek 6: Pohled studenta

2.2.2 Pohled lektora

Pohled lektora bude podobná tomu studentskému: uvidí soubory, která studenti nahráli, včetně jmen studentů, času odevzdání a hash kódu. Další bude nabídka souborů ke stažení. Dále bude možnost přejít na obrazovku administrace uživatelů, kde bude možnost administrace uživatelských účtů.

Hlavním rozdílem oproti studentům bude možnost manipulace se soubory, která bude dostupná na další obrazovce. Zde bude možno hromadně kopírovat a přesunovat soubory do adresářů či mazat soubory. Jedna část obrazovky bude obsahovat nahrané soubory, druhá adresářovou strukturu. V případě přesunutí nebo zkopírování souboru bude aktualizován i záznam v databázi.



Obrázek 7: Pohled lektora na správu souborů

2.3 Hardwarové požadavky na server

Navrhované hardwarové požadavky budou vycházet z oficiálních požadavků softwaru, který bude na serveru využíván. Půjde jen o teoretické specifikace a doporučení, neboť na výsledný použitý hw nebudou mít vliv.

2.3.1 Procesor

Debian Linux a všechen další software podporuje architekturu x64 pro oba výrobce procesorů, Intel a AMD. Jako minimální požadavek je uvedeno Pentium 4 na 1GHz [22], což je v dnešní době již enormně zastaralé CPU. Po účely aplikačního serveru Tomcat s mnoha instancemi a databáze MySQL bude samozřejmě vhodné moderní desktopové CPU Intel i7 novější generace, tj. ideálně od páté řady. V současnosti je na trhu osmá generace, která se dodává ve verzích se 4 jádry a 8 vlákny nebo 6 jádry a 12 vlákny [23]. Ideální by bylo nasazení CPU Xeon, které je určeno právě pro servery. Nevýhodou je vysoká pořizovací cena. Dodávají se ve verzích E3, E5 a E7, od základní verze pro malé systémy po škálovatelné vysoce výkonné systémy. Nejvýkonnější řada E7 nabízí až 24 jader a 60 MB cache

[24]. Rozumným kompromisem by mohl být procesor řady E5 druhé generace se šesti či osmi jádry, jehož cena se pohybuje do 15 tisíc Kč.

2.3.2 Paměť

Z hlediska požadavků na paměť lze předpokládat výraznou náročnost. Systému samotnému stačí i 1 GB, ale hlavně Java SDK bude mít nároky mnohem vyšší. U MySQL je minimální požadavek 256 MB, který ovšem stačí maximálně na práci s malými databázemi v testovacím, nikoliv produkčním prostředí. Celkové požadavky na paměť MySQL lze špatně odhadnout, vždy záleží na velikosti databáze. Odhad hodnoty lze spočítat z parametrů databáze na stránce *mysqlcalculator.com*.

Požadavky na operační paměť Apache Tomcat jsou minimální, v řádu jednotek či desítek MB. Zde ovšem hrají roli paměťové nároky Javy SDK, která má jako minimální požadavek 1 GB [25]. Pro rozumný provoz jde samozřejmě o nedostatečné množství a doporučení bude mnohem vyšší.

Ač teoreticky by pro provoz systému mělo stačit 4 GB RAM, prakticky bude potřeba minimálně dvojnásobek, či ještě lépe čtyřnásobek této hodnoty. 16 GB by mělo zajistit slušnou odezvu a provozní rychlost systému bez nadbytečného swapování.

Celkově vzato, odhad paměťové náročnosti je nesnadný a skutečné nároky budou známy až po spuštění a otestování systému. Poté se může paměťová konfigurace eventuálně upravit.

2.3.3 Pevný disk

Instalace operačního systému a aplikací vyžaduje na pevném disku jenom nepatrné místo v řádech jednotek gigabytů. Hlavním problémem bude velikost úložného prostoru pro soubory, která budou nahrávány studenty. Jak již bylo zmíněno, jedná se o soubory velké velikosti, které budou přesahovat 10 GB.

Doporučuji využívat samostatný souborový systém, a nikoliv systém sdílený s operačním systémem, aby nedošlo k ohrožení chodu systému při případném vyčerpání volného místa na disku.

Rámcový odhad kapacity lze vymodelovat na příkladu 24 studentů (kapacita největší počítačové učebny na FAI), jako velikost souboru berme 20 GB. Tím se dostáváme na 480 GB i se započítanou rezervou. Do celkové velikosti je třeba připočítat archiv uložených souborů

stejného rozsahu a případné objemné soubory, které bude vyučující nabízet ke stažení. V tomto okamžiku jsme na hranici 1 TB, což už není zanedbatelná kapacita.

Hodnotu 1 TB využijeme při úvodní konfiguraci systému. Následně podle výsledků testování ji lze dle potřeb měnit.

2.3.4 Síťové připojení

Rychlost připojení PC v učebnách univerzity se pohybuje blízko 1 Gb/s, rychlost uplink připojení switche je 2 Gb/s. Servery jsou připojeny rychlostí 10 Gb/s. Při těchto rychlostech lze očekávat poměrně rychlé přesuny i gigabytových souborů.

2.4 Alternativní řešení

Nabízí se otázka, proč je třeba vytvářet nové řešení problému, když je jistě dostupný software, který dokáže svými funkcemi vyhovět. Největším problémem je pravděpodobně velikost souborů, se kterými se bude pracovat. Jednotky či desítky GB často představují problém. A pokud by systém uměl pracovat s tak velkými soubory, potom nebude zajisté umět práci s hashovacími funkcemi nebo správu souborů.

V současné době existuje na UTB zavedený Moodle, který ale zvládá v základním nastavení maximálně přílohy o velikosti 2 GB, což je nedostačující. Toto nastavení lze upravit na straně serveru, je ale potřeba počítat s tím, že Moodle používá celá univerzita, čemuž odpovídá jeho vytížení. [26]. Postrádá také další vyžadované funkcionality. Výhodou ovšem je, že každý student Moodle už používá a má do něj přístup.

Univerzita také využívá balík aplikací Office 365 včetně cloudového úložiště s kapacitou 1 TB, což je dostatečné množství, soubory by takto šly sdílet s lektorem. Výhodou je také dostupnost odkudkoliv, příjemná správa souborů a víceméně stoprocentní dostupnost služby. Nevýhoda je zřejmá: jde o internetovou službu, takže data se přenášejí mimo univerzitní síť s výrazně nižší rychlostí, pracovat se soubory s velikostí několika jednotek či desítek gigabytů tak není dostatečně rychlé.

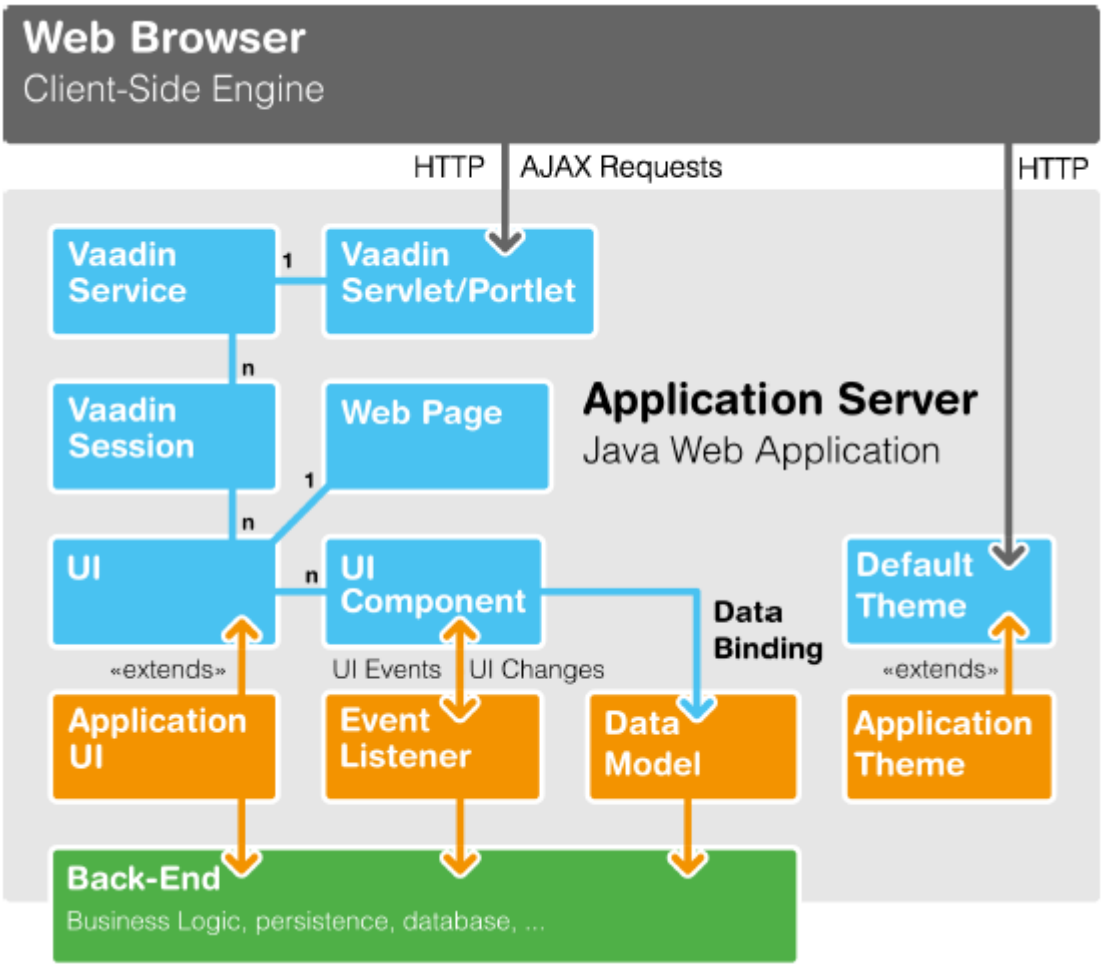
Alternativou je použití sady oddělených aplikací: vytvořit zvlášť hash souboru, odeslat jej na server (ideálně přes FTP), zde opět vytvořit hash a soubor přesunout tam, kde jej potřebujeme. Dále se musíme zabývat účty, oprávněními atd. To je samozřejmě zbytečně komplikované a zdlouhavé, vytvoření specializovaného nástroje je přesto ideální volbou.

II. PRAKTICKÁ ČÁST

3 TVORBA APLIKACE

3.1 Tvorba UI

Většina grafických prvků, které jsou v aplikaci použity, pochází z frameworku Vaadin. Aplikace vytvořené pomocí tohoto frameworku běží jako javovský servlet. Každá aplikace využívající Vaadin musí obsahovat jednu nebo více UI tříd, které dědí z abstraktní třídy *com.vaadin.ui.UI*, a implementovanou metodu *init()*. Vstup uživatele spravují *event listeners* [8].



Obrázek 8: Architektura server-side aplikace [8].

Základní elementy aplikace vytvořené ve Vaadinu jsou:

- UI – reprezentuje HTML fragment, ve kterém aplikace spouští webovou stránku, může jít o celou stránku nebo o její část.
- Page – UI je asociováno s objektem (Page) reprezentujícím webovou stránku, na které UI běží

- Vaadin session – představuje uživatelskou interakci s jedním či více UI, vzniká při spuštění prvního UI, zaniká expirací či explicitně příkazem ze serveru
- UI components – představují části aplikace, ze kterých je UI sestaveno, jsou uspořádány hierarchicky v layoutech. Patří jsem např. TextView, Label, Button a další
- Events, listeners – spravují události na základě uživatelské interakce v aplikaci
- Resources – spravují externí data aplikace jako obrázky, stažitelné dokumenty nebo externí odkazy
- Themes – separátně od aplikační logiky řeší vzhled aplikace a jsou definovány jako CSS nebo SCSS
- Data Binding – je využíván zejména při práci s databázemi, stará se o zobrazování dat v komponentách aplikace [8].

3.1.1 Layouty

Komponenty uživatelského rozhraní jsou seskupeny hierarchicky do layoutů, nejčastěji se používá vertical a horizontal layout. Mohou být navzájem nořeny do sebe a vytvářet tak strukturu komponent tvořené aplikace [8].

Příklad vytvoření Vertical Layoutu s Labelem:

```
VerticalLayout content = new VerticalLayout();
content.setSizeFull();
setContent(content);
content.addComponent(new Label("Hello!"));
```

3.1.2 Navigator

Třída Navigator umožňuje přechod mezi Views, které je třeba v prvním kroku zaregistrovat pomocí *addView()* a poté je možno na ně přejít přes *navigateTo()*. Views, které jsou spravovány Navigatorem, dostanou přiřazeny vlastní URI fragment, jenž může být použit pro operace vpřed/vzad v prohlížeči nebo k uložení stránky do oblíbených [8].

V konkrétní aplikaci je navigator použit následovně:

```
navigator = new Navigator(UI.getCurrent(), viewDisplay);
            navigator.addView("",  MainView.class);
            navigator.addView(STUDENTVIEW, SecondView.class);
            navigator.addView(SELFREGISTER, Registration.class);
```

Přechod ze stránky na stránku v kódu vypadá takto:

```
getUI().getNavigator().navigateTo(BoxUI.LOGIN);
```

3.1.3 Komponenty uživatelského rozhraní

Vaadin poskytuje množství komponent pro vytvoření uživatelsky přívětivé aplikace, umožňuje také tvorbu vlastních komponent. Tyto komponenty se umísťují hierarchicky do layoutů a dohromady tvoří UI aplikace. Každá komponenta má své jméno, popisný text a přiřazení do layoutu. Následuje popis komponent použitých v aplikaci.

3.1.3.1 Label

Tato komponenta obsahuje needitovatelný text a zpravidla se používá pro zobrazení krátkých zpráv. Text může být formátován jako HTML. Předávání textu do komponenty probíhá přímo v konstruktoru. Je to jedna z nejjednodušších komponent a využívá se poměrně často [8]. Ukázka kódu pro zobrazení krátké zprávy:

```
Label label = new Label("Select file for download");
layout.addComponent(label);
```

3.1.3.2 Notification

Notifikace slouží k zobrazení informací uživateli, převážně informuje o chybě nebo úspěšném dokončení operace. Defaultně se zobrazují uprostřed obrazovky a po jakékoliv interakci uživatele zmizí. Lze také nastavit možnost odstranění notifikace až po kliknutí uživatele přímo na ni.

```
Notification.show("Error","Something bad happened!",
                    Notification.Type.HUMANIZED_MESSAGE);
```

3.1.3.3 TextField

TextField je jednou z nejběžnějších komponent. Používá se pro textový uživatelský vstup, například ve formulářích. Hodnota pole se zpracovává metodou getValue(). Takto vypadá nejprimitivnější definice textového pole s popiskem „Name“:

```
TextField tf = new TextField("Insert username:");
tf.setValue("Stuff in the field");
```

Umožňuje implementovat řadu listenerů, které mohou ihned po zadání či změně textu provést nějakou akci, a není potřeba vytvářet další komponentu jako třeba Button, které by tuto akci spouštělo.

3.1.3.4 Link

Jde o komponentu, která umožňuje vytvářet hyperlink. Odkaz na cílovou lokaci je objekt ExternalResource:

```
Link link = new Link("Click Me!",
new ExternalResource("http://vaadin.com/"));
```

Kromě klasických textových odkazů lze nastavit k odkazu obrázek pomocí ThemeResource. Dále je možné specifikovat, zda se odkaz otevře v novém okně, nastavit velikost tohoto okna a další [8].

3.1.3.5 Button

Komponenta Button se používá ke spouštění nějaké akce, jako třeba zpracování vstupu z formuláře. Při stisknutí tlačítka se vytvoří ClickEvent(), který je zpracováván ClickListenerem [8]. Příklad využití komponenty „Button“ a clickEventListeneru z aplikace:

```
private Button loginButton() {
	Button button = new Button("Log In", new Button.ClickListener() {
		@Override
			public void buttonClick(ClickEvent event) {
				getUI().getNavigator().navigateTo(Login.STUDENTVIEW);
				}
			});
			return button;
		}

 private Button registerButton() {
	Button button = new Button("Self registration", new Button.ClickListener()
{		@Override
			public void buttonClick(ClickEvent event) {
				getUI().getNavigator().navigateTo(Login.SELFREGISTER);
				}
			});
			return button;
		}
```

3.1.3.6 Grid

Další využitou komponentou, ve které budou zobrazeny výpisy z databáze, jako historie uložených souborů nebo soubory k uploadu, je Grid. Data jsou uložena v řádcích a sloupcích, kde sloupce je možné řadit po kliknutí. Grid a tabulka jsou rozdílné komponenty. Hlavní rozdíl je, že Grid není kontejner, i když k němu mohou být přiřazena data. Dalším rozdílem

je přidávání řádků a rozdílná implementace listenerů [8]. Opět následuje příklad využití v aplikaci, konkrétně k výpisu souborů a adresářů:

```
private Grid gridUpload () {
              GetFileListInDirectory  fl = new GetFileListInDirectory();
              File [] list = fl.getFiles(download);
              Grid grid = new Grid("History of uploaded files");
              String date;
              grid.setSizeFull();
              grid.addColumn("Filename");
              grid.addColumn("Created");

              for (int i = 0; i < list.length; i++) {
                     TimeConverter tc = new TimeConverter();
                     date = tc.ConvertTime(list[i].lastModified());
                     grid.addRow(list[i].getName().toString(),date);

           }
              return grid;
       }
```

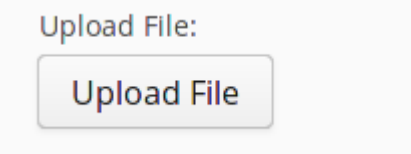
Výběr řádků z Gridu obstarává SelectionListener, který má dva módy: single selection a multi selection. Rozdíl mezi nimi je zřejmý z názvu. Zpracování události vrací *id* vybraného řádku, jak je ukázáno v následujícím kódu:

```
grid.addSelectionListener(selectionEvent -> {
       Object selected = ((SingleSelectionModel)
       grid.getSelectionModel()).getSelectedRow();
}
```

Opět umožňuje různá nastavení, jako velikost Gridu, sloupců, pořadí sloupců, jejich skrytí a další.

3.1.3.7 Upload

V této aplikaci jde o klíčovou komponentu, proto bude její použití popsáno podrobněji. Upload umožňuje uživateli nahrát soubory na server. Po kliknutí se zobrazí dialog pro výběr souborů a poté, podle zvoleného módu, začne upload souboru buďto ihned, nebo až po potvrzení kliknutím na tlačítko. Mezi těmito módy se přepíná *upload.setImmediate(true) nebo upload.setImmediate(false).*



Obrázek 9: Komponenta Upload v immediate módu

Upload vyžaduje metodu pro zpracování dat, která implementuje třídu Upload.Receiver. Komponenta zapisuje obdržená data jako proud do *java.io.OutputStream*, což dává značné možnosti při jeho zpracování. Po spuštění uploadu se aktivuje metoda *receiveUpload(),* která musí vracet OutputStream, jako vstupní parametry má jméno souboru MIME typ souboru podle toho, jak jej identifikuje prohlížeč. Probíhající upload může být monitorován přes *Upload.ProgressListener,* pokud chceme používat další komponentu pro grafické zobrazení průběhu uploadu ProgressBar, můžeme využít metodu *updateProgress(),* která má na vstupu počet přečtených bytů a celkovou délku souboru. Délka bývá často neznámá, takže je reportována jako hodnota -1.

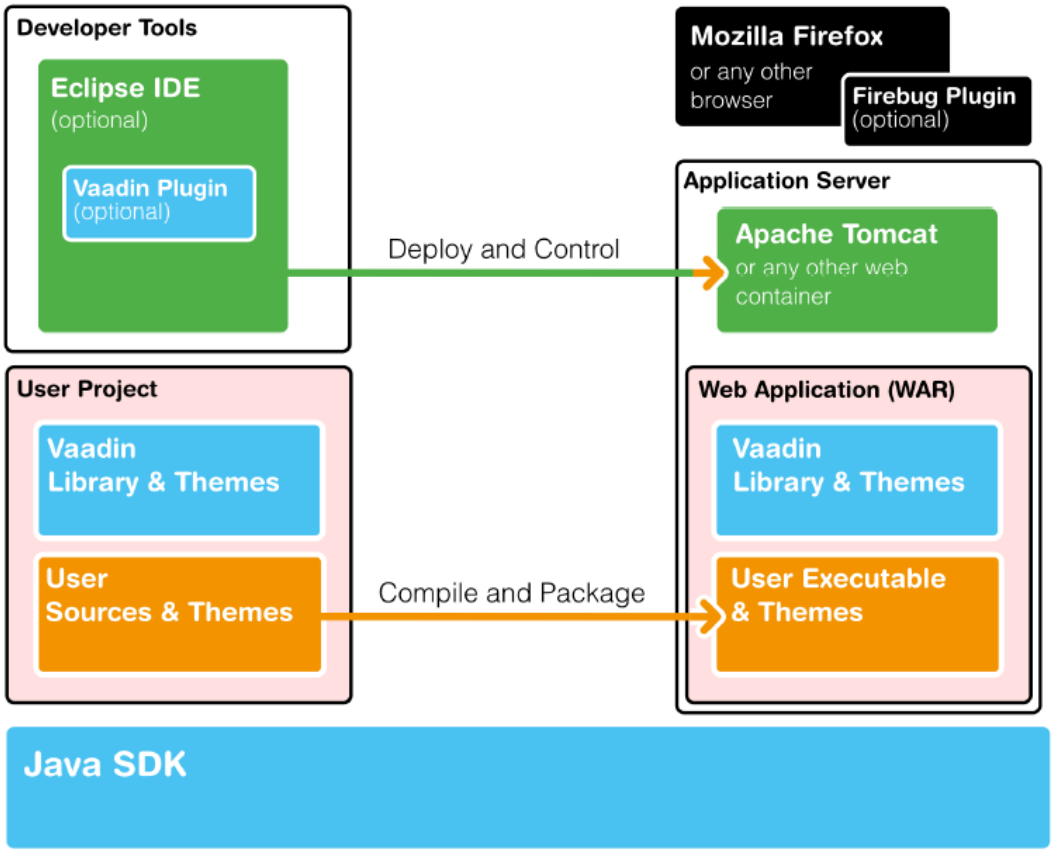
Po skončení uploadu komponenta spouští eventy *Upload.FinishedEvent*, další eventy *Upload.FailedEvent* nebo *Upload.SucceededEvent* se spouští v závislosti na tom, jak přenos dat dopadl [8].

V aplikaci je upload implementován takto:

```
private Upload upload () {
            upload.setReceiver(receiver);
            upload.setImmediate(true);
            upload.setButtonCaption("Upload File");
            upload.addSucceededListener(receiver);
            return upload;
}

    class FileUploader implements Receiver, SucceededListener {
            private static final long serialVersionUID = -9098063835825177961L;
            OutputStream outputFile = null;
          @Override
          public OutputStream receiveUpload(String strFilename, String strMIME-
Type)  {
              File file=null;
```

```
                tmr.start();
                try {
                    file = new File(uploaddir+"/"+strFilename);
                    Notification.show("Message", "Saving file: "+strFilename, Noti-
fication.Type.HUMANIZED_MESSAGE);
                    tempFile=file;
                    if(!file.exists()) {
                        file.createNewFile();
                    }
                    outputFile = new FileOutputStream(file);
                } catch (IOException e) {
                    //e.printStackTrace();
                     Notification.show("Error",   "Upload   failed!",   Notifica-
tion.Type.HUMANIZED_MESSAGE);
                }
                return outputFile;
    }
public void uploadSucceeded(SucceededEvent event) {
             CreateHash hsh = new CreateHash();
                    SaveFileInfo sv = new SaveFileInfo();
                    Date date = new Date();
                    String actualDate = date.toString();
                    try {
                           sv.addFile(tempFile.getName().toString(),   actualDate,
String.valueOf(tempFile.length()),  hsh.getMD5(tempFile),  hsh.getSHA1(tempFile),
fullname, out);
                           tmr.stop();
                           Notification.show("Message",   "Upload   completed   in
"+tmr.getLength()+" seconds.", Notification.Type.HUMANIZED_MESSAGE);
                           grid.refreshAllRows();
                    } catch (Exception e) {
                           Notification.show("Error",  "Something  bad  happened!",
Notification.Type.HUMANIZED_MESSAGE);
                           e.printStackTrace();
```

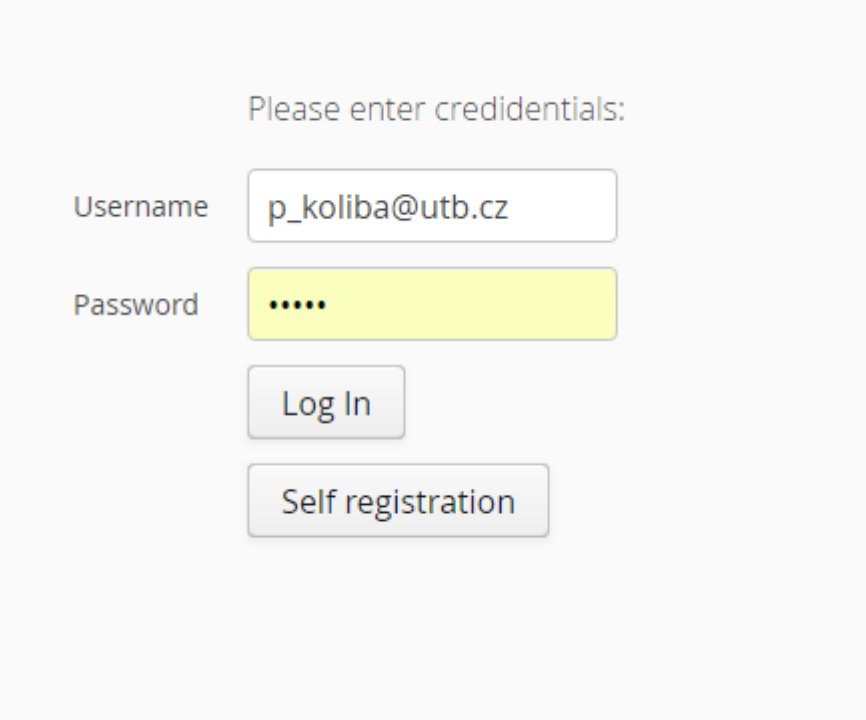
```
                    }

}
```



Obrázek 10: Proces vývoje aplikace ve Vaadinu

3.2 Úvodní obrazovka

Úvodní stránka vyžaduje zadání uživatelských údajů – a to uživatelského jména, ideálně ve formátu studentského emailu a hesla. Pokud student není na stránce zaregistrován, může provést autoregistraci kliknutím na dané tlačítko. Uživatelské jméno nevyžaduje konkrétní formát, nicméně by bylo vhodné, aby uživatelé používali jednotný formát uživatelského jména. Případná omezení je možné zapracovat do kódu aplikace. Pokud proběhne autentifikace a autorizace korektně, je uživatel přesunut na svou „hlavní“ stránku, kde může provádět obě operace, tj. upload a download souborů.



Obrázek 11: Snímek úvodní obrazovky

Třída Login.java, která zpracovává tyto požadavky, zároveň ukládá do vzniklé session parametry o uživateli a o nastaveních systému, které budou později využity. Tyto parametry jsou načteny z databáze:

- Uživatelské jméno, jméno a příjmení uživatele
- Úroveň oprávnění (0 = uživatel, 1 = admin)
- Nastavení serveru
- Adresáře pro upload a download

V kódu je předání parametrů aplikováno tímto způsobem:

```
VaadinService.getCurrentRequest().getWrappedSession().setAttribute("serve-
rip",fl.getValue());
```

Podobně pak probíhá předání ostatních parametrů.

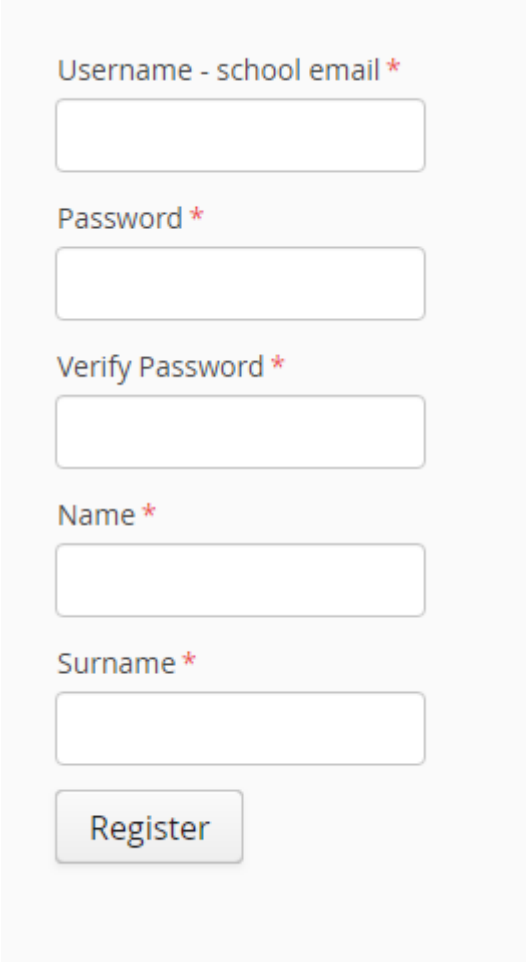
3.3 Obrazovka registrace uživatele

Na obrazovce registrace je možné vytvořit nového uživatele. Po zadání povinných údajů, kterými jsou uživatelské jméno (UTB email), heslo, potvrzení hesla, jméno a příjmení, se vytvoří nový uživatelský účet. Po kontrole se tyto data uloží do databáze spolu s aktuálním datem a úrovní oprávnění, což je v tomto případě „0“. Uživatele s administrátorským

oprávněním nelze tímto způsobem vytvořit. Po úspěšném uložení je uživatel přesměrován zpět na stránku s přihlašovacím formulářem.

Ve třídě je implementováno zabezpečení uživatelských hesel pomocí hashovací funkce SHA-512, která do databáze neukládala hesla v lidsky čitelném formátu, ale právě v tomto hashi. Pro tento účel byla použita knihovna *org.apache.commons.codec.digest.DigestUtils.sha512Hex*. Při logování do systému se tedy pokaždé generuje hash ze zadaného hesla a tento se porovnával s hashem v databázi.

```
if (pf.getValue().equals(pf2.getValue())) {
	passwd = pf.getValue();
	//Prevod na hash
	PasswordCheck pc = new PasswordCheck();
	if (pc.checkPwd(passwd) == true);
	CreateHash ch = new CreateHash();
	try {
	pwdhash = ch.getSHA512forString(passwd);
	}
	catch (Exception e) {
		e.printStackTrace();
		try {
			create.userAdd(name.getValue(), surname.getValue(), actualDate, pf.getValue(), tf.getValue(), null, "0");
		}catch (Exception e) {
			Notification.show("Error","User creation failed.", Notification.Type.HUMANIZED_MESSAGE);
		}
```



Obrázek 12: Snímek formuláře pro autoregistraci uživatele

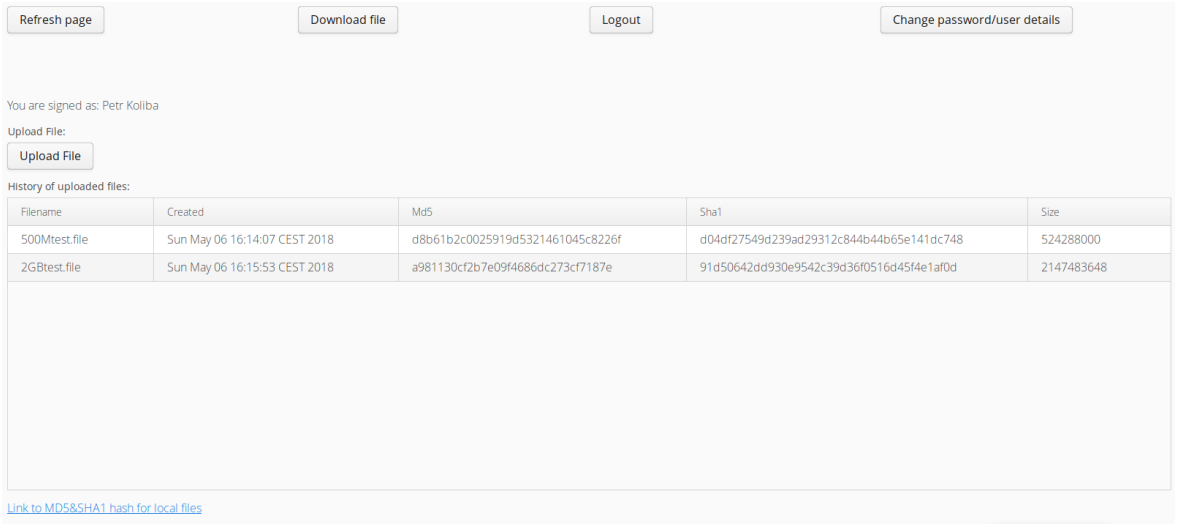
V prvních verzích byla implementována kontrola hesel pomoci regulárních výrazů, takže heslo muselo odpovídat určitým kritériím, jako délka či počet numerických znaků. Později bylo od těchto bezpečnostních opatření upuštěno, neboť se nezdají nezbytná. Důležité je rozdělení jednotlivých rolí na uživatele a administrátora. Následuje kód přidání uživatele za použití *EnityManager:*

```
EntityManager entitymanager = emfactory.createEntityManager( );
entitymanager.getTransaction( ).begin( );
User user = new User( );
        user.setFirstname(firstname);
        user.setLastname(lastname);
        user.setCreated(created);
        user.setUsername(username);
        user.setPasswd(password);
        user.setIsadmin(isadmin);
```

```
entitymanager.persist(user);
entitymanager.getTransaction( ).commit( );
entitymanager.close( );
emfactory.close( );
```

3.4 Hlavní stránka uživatele

Tato stránka, implementovaná třídou StudentView.java, obsahuje vyžadované funkcionality systému pro uživatelskou roli: možnost uploadu na server, přechod na stránku se soubory určenými ke stažení, možnost vytvoření SHA-1 a MD5 hashe z vybraných souborů a mřížka se zobrazením detailů o souborech, které uživatel odeslal na server, spolu s jejich hashem, velikostí a datem odeslání. Uživatel má tedy možnost si ověřit, zda upload proběhl skutečně korektně a hashe se shodují.



Obrázek 13: Hlavní stránka uživatele

Výpis detailů o souborech z databáze do mřížky je realizován následovně:

```
private Grid gridUpload () {
        Grid grid = new Grid ();
        grid.setCaption("History of uploaded files:");
             grid.addColumn("Filename");
             grid.addColumn("Created");
             grid.addColumn("MD5");
             grid.addColumn("SHA1");
             grid.addColumn("Size");
```

```
            grid.setHeightByRows(5);
            grid.setSizeFull();
           grid.getContainerDataSource().removeAllItems();
           ListUserFiles ufl = new ListUserFiles();
           List<database.File> list = (ufl.list(out));
           for (database.File fl : list)
               grid.addRow(fl.getFilename(), fl.getCreated(), fl.getMd5(), fl.ge-
tSha1(), fl.getSize());
            }
            return grid;
}
```

3.4.1 Upload souboru

V současnosti je v aplikaci implementováno nahrávání souborů přes komponentu Upload, která k přenosu používá protokol aplikační vrstvy HTTP. HTTP využívá spolehlivý transportní protokol TCP, který zajišťuje doručení každého paketu ve správném pořadí, odstraňuje duplicity a používá kontrolní součty [6].

Po úspěšném nahrání souboru na server se na straně serveru automaticky vygeneruje hash souboru a tyto informace se ukládají do databáze. Každý nahraný soubor má tak přiřazeno jméno uživatele, který jej nahrál, čas nahrání a hash. Bohužel, díky velikosti souborů a následnému generování hashe tato činnost vyžaduje značné množství času.

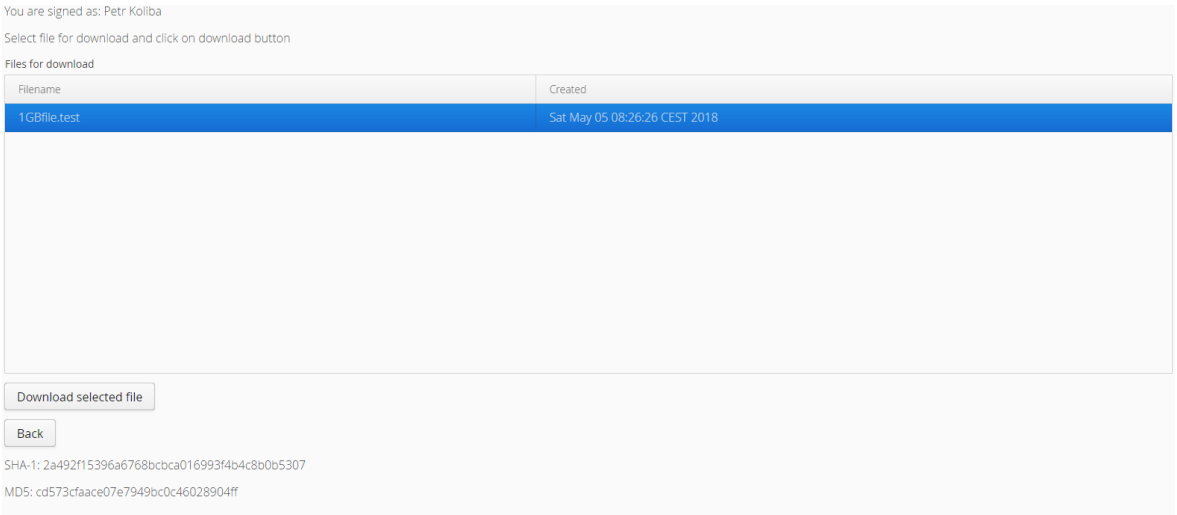
Dále je tu i možnost řešit přesuny souborů pomocí protokolu FTP. Tato možnost má nevýhodu v tom, že by vyžaduje klientskou část, takže by už nešlo o čistě serverovou aplikaci. Další komplikací by byla instalace FTP serveru, nutnost nového uživatele a nastavení práv. Nicméně tato možnost byla částečně implementována a v případě doplnění klientské části může být použita pro upload i download. Metody pro přenos souborů jsou v třídě FTPUploadDownload.java v balíku supportfiles.

3.4.2 Download souboru

Obrazovka obsahující soubory nabízené k downloadu je dostupná přes tlačítko z hlavní stránky uživatele. Po kliknutí na řádek se souborem se začne generovat hash souboru, po stisknutí tlačítka objeví se dialogové okno pro zahájení stahování.

```
downloadFilename = (event.getItem().getItemProperty("Filename").getVa-
lue().toString());
                res = new FileResource(new File(fullpath+"//"+downloadFilename));
                      FileDownloader fd = new FileDownloader(res);
                      fd.extend(downBtn);
```

V tomto kódu je ukázáno, jak probíhá download souboru. Využita je knihovna *com.vaadin.server.FileDownloader a com.vaadin.server.FileResource,* jméno souboru je získáno z mřížky. FileDownloader musí rozšiřovat nějaký aktivní prvek, v tomto případě tlačítko.



Obrázek 14: Obrazovka s výběrem souboru pro uložení.

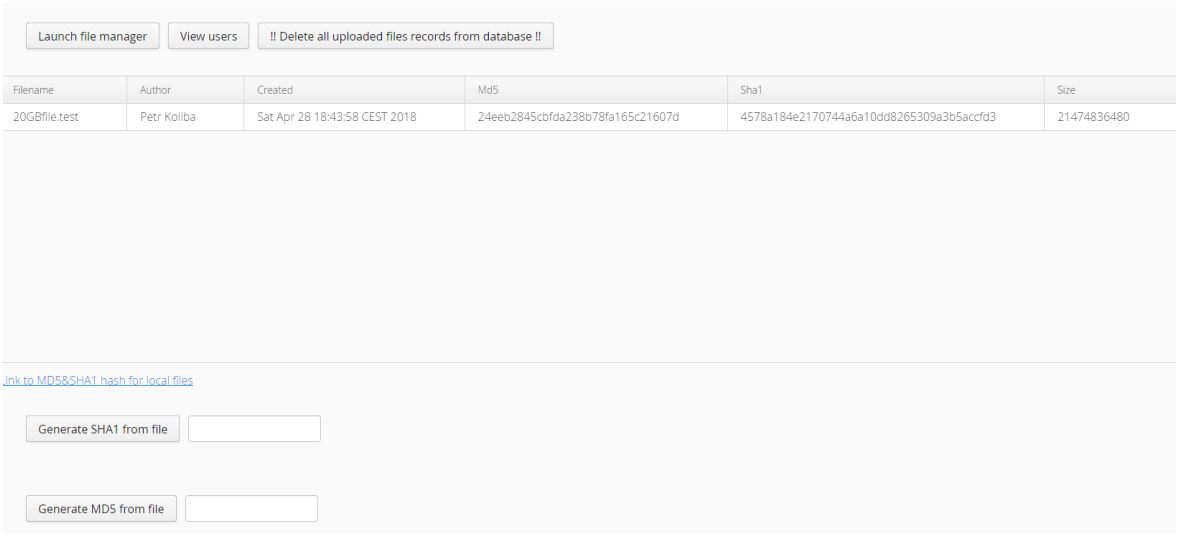
3.4.3 Generování hashe

Stránka uživatele obsahuje link na skript, který generuje hash lokálně, bez nutnosti odesílat data mimo klientský počítač. Uživatel si tak má možnost ověřit, zda se hashe shodují a soubor byl přenesen korektně. Na stránce se stahováním souborů se hash souboru generuje po kliknutí na soubor, může tak kdykoliv dojít k vygenerování hashe a k ověření korektnosti downloadu.

3.5 Hlavní stránka administrátora

Pokud má uživatel nastavenou v databázi hodnotu *isadmin="1"*, je po nalogování přesměrován na pohled administrátora. V něm je v mřížce přehled všech souborů nahranných uživateli. Soubory je možné řadit podle všech sloupců: jména uživatele, data, jména souboru a dalších. Dále, stejně jako v uživatelském pohledu je zde možnost generovat hash. Na rozdíl od neautorizovaných uživatelů, má vyučující k dispozici jednoduchý souborový manažer,

který umožňuje základní operace se soubory na serveru (kopírování, přesun a mazání) a dále výpis registrovaných uživatelů.



Obrázek 15: Hlavní obrazovka administrátora

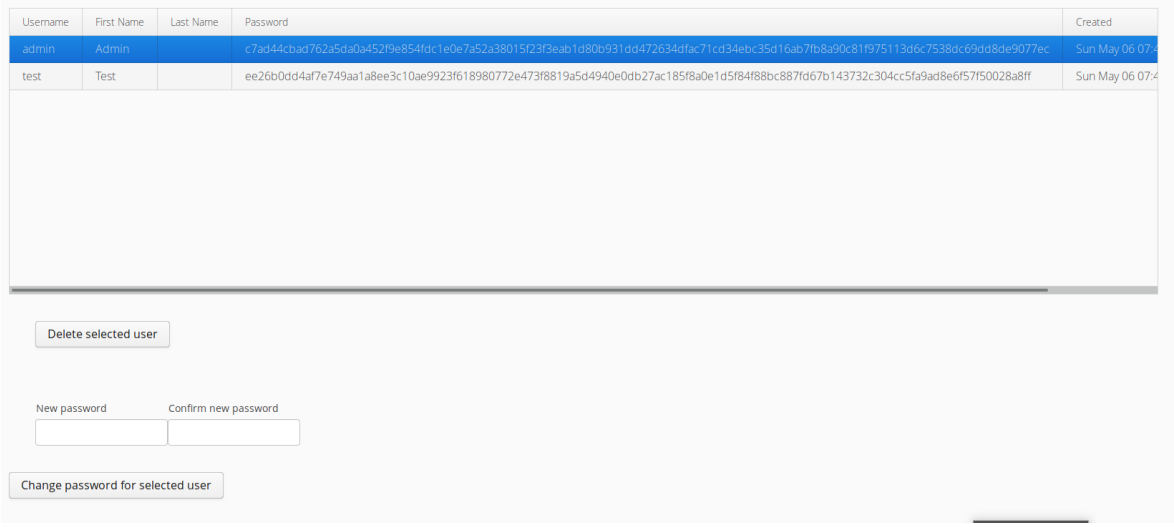
3.6 Souborový manažer

Dalším pořafavkem na aplikaci byla možnost správy souborů včetně hromadného kopírování a přesunů souborů. Při řešení tohoto problému se však objevily hranice možností frameworku Vaadin, které neumožnily implementaci tak, jak byla původně navržena. Implementované řešení sice zvládá definované požadavky, ale uživatelská přívětivost není ideální. Problémem je zobrazování datových struktur, a to konkrétně souborů a adresářů v komponentách Grid a TreeTable a práce s nimi. Dokumentace frameworku v těchto oblastech také není ideální a vzhledem k tomu, že Vaadin není masivně rozšířený framework, tak se získávaly informace o možných postupech složitě. Finálního řešení bylo nakonec dosaženo po velmi zdlouhavém a pracném zkoušení možností, které byly možné a zároveň UI nabízelo dostatečnou přehlednost a pohodlnost při ovládání.

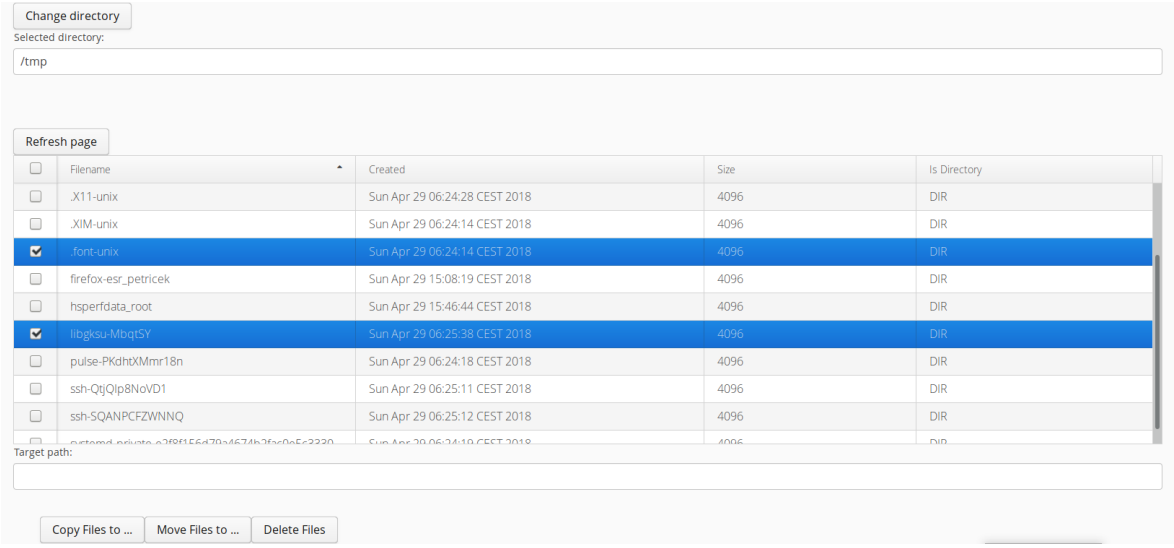
Další možností byla implementace knihovny třetí strany, která by tuto funkčnost řešila na vyšší úrovni, nicméně po experimentech byla tato možnost zavržena. Manažer umožňuje kopírování a přesun více souborů najednou.

3.7 Správa uživatelů

Na této obrazovce má administrátor přehled o registrovaných uživatelích a má možnost je odstranit nebo změnit jejich heslo.



Obrázek 16: Správa uživatelů



Obrázek 17: Souborový manažer se základními funkcemi

3.8 Vaadin Themes

Framework Vaadin odděluje vzhled uživatelského rozhraní od aplikační logiky, k čemuž používá *themes*. Ty mohou obsahovat prvky jazyka CSS, Sass, HTML a potřebnou grafiku. V projektu jsou tyto zdroje dostupné ve složce VAADIN/themes nebo v případě použití Mavenu src/main/webapp. Za pomocí *themes* lze měnit vzhled komponent tak, jak to vyžaduje uživatel a jeho projekt. Následuje příklad, který nastaví bílý text na černém pozadí:

```
.v-label {
        background: black;
        color: white;
        font-size: 24pt;
```

```
       line-height: 24pt;
       padding: 5px;
      }
```

Výhodou je, že dokumentace u každé komponenty uvádí, jak může být modifikována.

3.9 Databáze

Systém využívá tři tabulky: users, files a config. Jak již bylo zmíněno v teoretické části, použitá databáze je MySQL, pro snadnější práci s SQL dotazy byl použit nástroj MySQL Workbench.

3.9.1 Nastavení JPA

Pro správnou funkčnost připojení k databázi je potřeba dodat do Eclipse knihovnu pro připojení do MySQL Connector/J ve verzi nezávislé na operačním systému a nakonfigurovat soubor persistence.xml, který slouží jako zdroj informací pro IDE. Obsah souboru je následující:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLoca-
tion="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persis-
tence/persistence_2_1.xsd">
       <persistence-unit name="boxstorage">
              <mapping-file>META-INF/eclipselink-orm.xml</mapping-file>
              <class>database.User</class>
              <class>database.File</class>
              <class>database.Config</class>
              <properties>
         <property name="javax.persistence.jdbc.url" value="jdbc:mysql://lo-
calhost:3306/storage"/>
         <property name="javax.persistence.jdbc.user" value="test"/>
         <property name="javax.persistence.jdbc.password" value="test"/>
         <property name="javax.persistence.jdbc.driver" va-
lue="com.mysql.jdbc.Driver"/>
         <property name="eclipselink.logging.level" value="FINE"/>
         <property name="eclipselink.ddl-generation" value="create-tables"/>
```

```
      </properties>
       </persistence-unit>
</persistence>
```

Je vidět, že je zde nutné nastavit adresu a port serveru, kde MySQL běží a uživatelské jméno s heslem. Pro reálné použití je nutné použít složitější přihlašovací údaje.

3.9.2 Tabulka „users“

Zde jsou uloženy informace o uživatelích a jejich právech. Primární klíč je „id“, který se generuje automaticky s každým novým řádkem. Všechny hodnoty, kromě „isadmin“, které nabývá hodnot 0 nebo 1, jsou datového typu VARCHAR, vkládané hodnoty jsou ošetřeny na úrovni aplikace. Ač zde není nastavení NOT NULL, program neumožní vložení prázdných hodnot.

Sloupec „isadmin“ definuje, zda má uživatel administrátorská práva a tím pádem může zobrazit pohled administrátora. Hodnota „0“ je pro základní uživatelská práva, „1“ pro administrátorské.

3.9.3 Tabulka „files“

Tato tabulka obsahuje seznam souborů, které uživatelé odeslali na server. Obsahuje jména souborů, uživatele, čas odeslání a jejich hash hodnoty. Z tabulky se čte při zobrazení uživatelského pohledu, kde se zobrazí pouze soubory nahrané přihlášeným uživatelem. V administrátorském se pak zobrazují všechny soubory od všech uživatelů. Administrátor má také možnost tuto tabulku smazat, například po ukončené hodině nebo semestru.

3.9.4 Tabulka „config“

Aby bylo dosaženo přenositelnosti systému na jiný server, popřípadě jiný operační systém, bylo nutno eliminovat přihlašovací jména a cesty přímo v kódu a načítat je z databáze, kde mohou být jednodušeji měněny bez nutnosti editace kódu a následné rekompilace.

Obsahuje jen sloupce „name“ a „value“, kde se vkládají proměnné, které potřebujeme ukládat a které se mohou měnit v závislosti na operačním systému. Při implementaci a testování obsahoval tyto data:

```
download_directory /etc/storagebox/download
ftppassword        test
```

```
ftpusername        test

fulldownloadpath   /etc/storagebox/download

serverip           127.0.0.1

upload_directory   /etc/storagebox/upload
```

Před novým nasazením systému je potřeba tuto tabulku editovat, aby údaje v ní odpovídaly realitě. Jsou zde uloženy i údaje o FTP připojení, se kterým probíhaly experimenty v průběhu vývoje a bylo zvažováno jeho použití.

4 IMPLEMENTACE A TESTY

4.1 Konfigurace testovacího systému

Tato kapitola popisuje instalaci a konfiguraci nástrojů, které jsou nutné pro běh aplikace na cílovém systému. Lze ji chápat jako návod pro zprovoznění aplikace.

Testovací systém byl vytvořen na virtuálním stroji v aplikaci Oracle VirtualBox, verze 5.2.8, kde je nainstalovaná databáze MySQL a Apache Tomcat. K usnadnění práce s databází je používán nástroj MySQL Workbench.

Konfigurace fyzického stroje, na kterém při testování běžel virtualizovaný server je: AMD FX 8300 3,3GHz s 8 GB RAM. Operační systém je Windows 10 Pro. Virtuálnímu stroji byly přiřazeny 4 jádra a 3 GB operační paměti.

Jako klientský systém sloužil notebook o konfiguraci Intel i5 520M 2,4GHz, 6 GB RAM s nainstalovaným operačním systémem Lubuntu 18.04. Z téhož stroje pak probíhalo testování se systémem Windows 7.

Implementace a testování přinesla několik oprav kódu, protože aplikace byla vyvíjena pod operačním systémem Windows. I když se primárně počítalo s během aplikace pod Linuxem a byla snaha psát kód tak, aby byl multiplatformní, přesto se objevilo několik chyb, které bylo nutno opravit.

Další důležitý krok je vytvoření adresářů, se kterými bude systém pracovat. Je vyžadován jeden adresář pro upload a jeden pro download. Tyto adresáře musí být s plnou cestou uvedeny v také tabulce config. Pro testovací účely se využívaly adresáře /etc/storagebox/upload pro soubory, které budou nahrávané na systém a /etc/storagebox/download pro soubory, které budou k dispozici pro stažení. Je také nutné přiřadit adresářům správná přístupová práva pomocí příkazu *chmod.*

4.2 Instalace MySQL a příprava tabulek

První aplikací, která je nutná k běhu, je databáze MySQL. Je k dispozici ke stažení na stránce *dev.mysql.com/downloads/mysql/,* nebo je možné ji naistalovat příkazem apt:

```
sudo apt-get install mysql-server
```

Dalším krokem je přidání uživatele do MySQL a přiřazení oprávnění:

```
CREATE USER 'test'@'localhost' IDENTIFIED BY 'test';
```

```
GRANT CREATE ON *.* TO 'test'@'localhost';
```

Druhý řádek přidá uživateli *test* oprávnění ke všem databázím, pokud chceme přidat oprávnění jen ke všem tabulkám v určené databázi, použijeme:

```
GRANT CREATE ON 'storage'.* TO 'test'@'localhost';
```

Nebo přiřadíme všechny práva:

```
GRANT ALL ON *.* TO 'test'@'localhost';
```

Poté vytvoříme databázi:

```
CREATE DATABASE storage;
```

Každé další připojení vytvořeným uživatelem k databázi *storage* bude vypadat následovně:

```
mysql --host=localhost --user=test --password=test storage
```

Používané tabulky jsou vytvořeny takto:

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `passwd` varchar(255) DEFAULT NULL,
  `created` varchar(255) DEFAULT NULL,
  `lastlogin` varchar(255) DEFAULT NULL,
  `isadmin` char(1) DEFAULT NULL,
  `firstname` varchar(255) DEFAULT NULL,
  `lastname` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=1703 DEFAULT CHARSET=utf8;


CREATE TABLE `files` (
  `id` int(32) NOT NULL AUTO_INCREMENT,
  `filename` varchar(255) DEFAULT NULL,
  `created` varchar(255) DEFAULT NULL,
  `size` varchar(255) DEFAULT NULL,
  `md5` varchar(128) DEFAULT NULL,
  `sha1` varchar(128) DEFAULT NULL,
```

```
  `createdby` varchar(255) DEFAULT NULL,
  `username` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1952 DEFAULT CHARSET=utf8;


CREATE TABLE `config` (
  `name` varchar(255) NOT NULL,
  `value` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Tímto máme připravenu databázi, zbývá doplnit nastavení do tabulky „config“, odkud aplikace čte konfiguraci serveru. Detaily nastavení byly uvedeny dříve.

4.3 Instalace a příprava Apache Tomcat

Nejnovější verze jsou k dispozici na stránce *tomcat.apache.org/download-80.cgi.* Po rozbalení a nakopírování do cílové destinace je vhodné vytvořit uživatele, pod kterým se bude aplikace spouštět. Dále je nutné nachystat proměnnou prostředí CATALINA_HOME, aby směřovala na adresář s instalací Tomcatu. Pak by měla následovat kontrola, zda máme aktualizovanou verzi Javy a nastavenou proměnnou JAVA_HOME. Spuštění se provádí příkazem:

```
$CATALINA_HOME/bin/startup.sh
root@debian-dva:/usr/local/apache-tomcat-8.5.30/bin# ./startup.sh
Using CATALINA_BASE:   /usr/local/apache-tomcat-8.5.30
Using CATALINA_HOME:   /usr/local/apache-tomcat-8.5.30
Using CATALINA_TMPDIR: /usr/local/apache-tomcat-8.5.30/temp
Using JRE_HOME:        /usr/lib/jvm/java-8-openjdk-amd64/jre/
Using CLASSPATH:       /usr/local/apache-tomcat-8.5.30/bin/bootstrap.jar:/usr/local/apache-tomcat-8.5.30/bin/tomcat-juli.jar
Tomcat started.
```

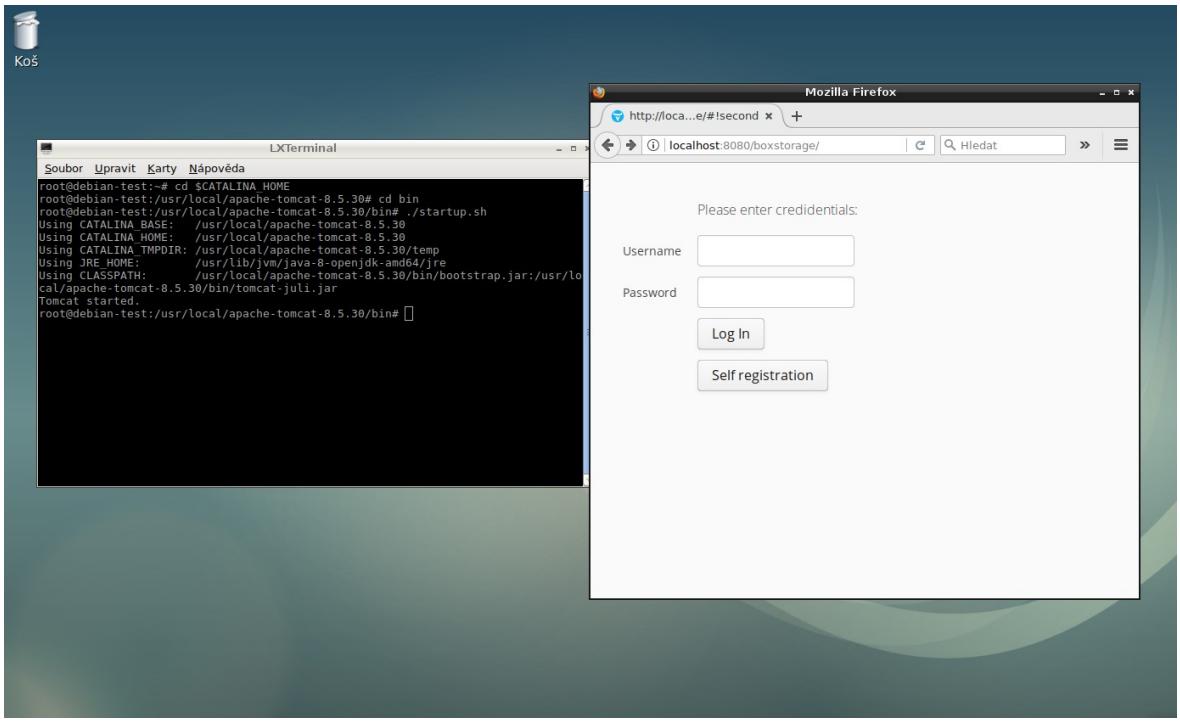
Nyní již jen zbývá exportovat projekt ve formátu WAR a nakopírovat jej do adresáře $CATALINA_HOME/webapps. Poté by měla být aplikace dostupná v prohlížeči pod adresou:

```
localhost:8080/boxstorage
```

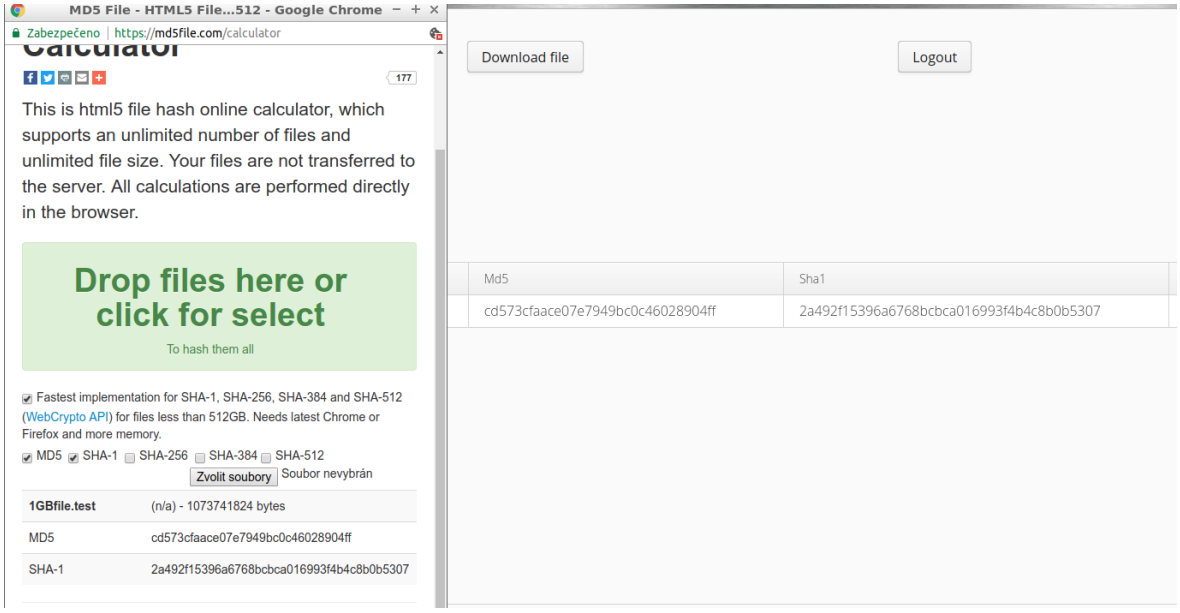
4.4 Testovací soubory

Pro testování uploadu a downloadu bylo potřeba vytvořit velké testovací soubory, byly zvoleny velikosti 1 GB, 5 GB, 10 GB a 20 GB. Ty byly střídavě nahrávány a stahovány ze serveru. Testovací soubory byly vytvořeny příkazem *fallocate*, např.

```
fallocate -l 20G 20GBtest.file
```



Obrázek 18: Start Tomcatu a úvodní stránka aplikace



Obrázek 19: Kontrola přenosu – hash odpovídá

Na server bylo přistupováno z operačních systémů Lubuntu, Windows 7 a Windows 10 za použití prohlížečů Chrome, Opera a Firefox.

Původně panovaly obavy, že přenos souborů přes protokol HTTP bude problematický nebo narazí na softwarové limity. Některé zdroje uváděly [27], že přenos souborů je omezen velikostí a je potřeba upravit nastavení aplikačního serveru. Toto se naštěstí při testech nepotvrdilo a přenos souboru o velikosti 20 GB proběhl bez problému.

4.5 Měření vytížení zdrojů serveru

Cílem testů bylo ověření funkčnosti aplikace a monitorovat vytížení CPU a operační paměti. Ověří se tak, zda byly správné odhady v úvodní kapitole, kde byly odhadovány nároky použitého software, a to zejména aplikačního serveru (Javy) a databáze.

V průběhu testování bylo k systému přistupováno z několika sessions a probíhal přenos dat. Sledovala se závislost na počtu stahovaných souborů a vytížení zdrojů.

První test je pouze se spuštěným aplikačním serverem, bez připojených uživatelů:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
704	root	20	0	436224	77172	37392	S	0,7	2,7	17:08.00	Xorg
6300	**root**	**20**	**0**	**43796**	**3720**	**3096**	**R**	**0,7**	**0,1**	**0:01.35**	**top**
901	petricek	20	0	379812	15772	13796	S	0,3	0,6	0:00.32	lxsession
963	petricek	20	0	115948	2176	1792	S	0,3	0,1	0:23.07	VBoxClient
1527	petricek	20	0	515664	23044	19544	S	0,3	0,8	0:18.66	clipit
1619	root	20	0	242084	24076	19068	S	0,3	0,8	0:01.35	x-terminal+
6226	root	20	0	0	0	0	S	0,3	0,0	0:00.23	kworker/0:0
6249	root	20	0	2735904	153852	17040	S	0,3	5,4	2:58.38	java

Obrázek 20: Spotřeba zdrojů bez připojených uživatelů (proces id 6249)

Proces id 6249 nespotřebovává téměř žádné zdroje, stejně tak jako databáze MySQL. Následující tabulka zobrazuje vytížení zdrojů při jednom probíhajícím uploadu ze sítě:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1826	root	20	0	2735532	160060	15592	S	14,6	5,6	1:38.27	java
3	root	20	0	0	0	0	S	1,3	0,0	0:07.94	ksoftirqd/0
659	root	20	0	437612	78272	37088	S	0,7	2,8	1:02.18	Xorg
1330	petricek	20	0	683960	38488	28068	S	0,7	1,4	0:01.01	pcmanfm
817	petricek	20	0	379812	16028	14056	S	0,3	0,6	0:00.11	lxsession
1658	petricek	20	0	515664	23128	19628	S	0,3	0,8	0:01.98	clipit
1770	root	20	0	240336	22992	18860	S	0,3	0,8	0:00.98	x-terminal+

Obrázek 21: Jeden probíhající upload (proces id 1826)

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1826	root	20	0	2736088	161296	15024	S	18,3	5,7	2:10.64	java
3	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**2,0**	**0,0**	**0:11.18**	**ksoftirqd/0**
659	root	20	0	437612	78272	37088	S	0,3	2,8	1:02.50	Xorg
858	petricek	20	0	117632	4320	3768	S	0,3	0,2	0:00.51	VBoxClient
1658	petricek	20	0	515664	23128	19628	S	0,3	0,8	0:02.23	clipit
1966	root	20	0	0	0	0	S	0,3	0,0	0:00.62	kworker/0:1

Obrázek 22: Dva probíhající uploady (proces id 1826)

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6249	root	20	0	2736996	166160	11700	S	23,3	5,8	3:19.03	java
3	root	20	0	0	0	0	S	1,0	0,0	0:02.03	ksoftirqd/0
704	root	20	0	436224	77172	37392	S	0,7	2,7	17:10.00	Xorg
7	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**0,3**	**0,0**	**0:01.07**	**rcu_sched**
83	root	20	0	0	0	0	S	0,3	0,0	0:02.64	kworker/u2+
1012	petricek	20	0	683960	38816	28352	S	0,3	1,4	0:01.25	pcmanfm
1527	petricek	20	0	515664	23044	19544	S	0,3	0,8	0:20.07	clipit
6300	**root**	**20**	**0**	**43796**	**3720**	**3096**	**R**	**0,3**	**0,1**	**0:04.37**	**top**
6427	root	20	0	0	0	0	S	0,3	0,0	0:00.76	kworker/0:1

Obrázek 23: Tři současné uploady (proces id 6249)

Podle přechozích zjištění se tedy dá říct, že s každým dalším aktivním připojením, které odesílá či přijímá soubory se mírně zvyšuje zátěž aplikačního serveru na testovacím systému o jednotky procent a obsazení operační paměti o desetiny procent. Vliv databáze na zatížení systému je zanedbatelný.

Soubor o velikosti 20 GB byl největším z testovaných, v testovacím prostření trvalo jeho odeslání po 100 Mb síti a následné vytvoření hashe na serveru 2331 vteřin, tj. 38 minut. Zde je úzké hrdlo v testovacím prostředí sítě, konkrétně v routeru, který zvládá přenos pouze 100 Mb/s, vliv hardwarové konfigurace systému je malý.

Byl také proveden test propustnosti sítě pomocí utility *iperf*, kterou můžeme testovat síť jak na jednosměrný, tak plně duplexní přenos. Získáme tak přehled, jak se připojení chová v zátěži a jaká je propustnost neovlivněná aplikacemi. V tomto případě byl zvolen testovací soubor o velikosti 10 GB a 20 GB. Z obrázku č. 24 je vidět, že rychlost při přenosu těchto souborů byla 93,5 Mb/s, doba přenosu při jednosměrné komunikaci 10 GB souboru byla 920 sekund (15,5 minuty) a 20 GB souboru 1838 sekund (30,6 minuty). Z předchozího odstavce vidíme rozdíl 8 minut, což je doba, kdy probíhalo hashování souboru a zároveň mohlo docházet zpomalení přenosu kvůli ostatním aplikacím, což tento test eliminoval.

```
root@debian-dva:~/eclipse# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.0.114 port 5001 connected with 192.168.0.112 port 52424
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec   112 MBytes  93.7 Mbits/sec
[  5] local 192.168.0.114 port 5001 connected with 192.168.0.112 port 52446
[  5]  0.0-919.8 sec  10.0 GBytes  93.4 Mbits/sec
[  4] local 192.168.0.114 port 5001 connected with 192.168.0.112 port 52502
[  4]  0.0-1838.1 sec  20.0 GBytes  93.5 Mbits/sec
```

Obrázek 24: Test propustnosti sítě

4.6 Možné rozšíření aplikace

Aplikace naskýtá celou řadu možností, jak by šla rozšířit její funkčnost nebo doplnit funkcionality, které jsou mimo možnosti této diplomové práce. Jednou z věcí, na které by bylo vhodné se v budoucnu zaměřit je souborový manažer. Ten sice splňuje požadavky na hromadné přesunování, kopírování a mazání souborů, ale práce není příliš komfortní. Tento problém lze obejít připojením na server a použitím aplikace třetí strany.

Jako další možnost vylepšení by mohlo být upravení vzhledu a využít naplno framework specializovaný právě na tvorbu uživatelského rozhraní. UI skýtá značné možnosti optimalizace pomocí jazyka CSS a je možné je v budoucnu upravit. Další alternativou je použití „themes“, kterými jde změnit celkový motiv vzhledu aplikace.

Zajímavou funkcionalitou by bylo zobrazení počtu připojených uživatelů, vytížení zdrojů a volné místo na disku či v souborových systémech.

Nové požadavky na úpravy pak zajisté vyplynou z případného nasazení v reálném provozu.

ZÁVĚR

V této práci byl představen proces tvorby aplikace, která má sloužit pro upload a download velkých souborů. Na začátku proběhlo představení možných nástrojů, které by mohly být k vývoji použity a pak blíže popsány ty, které použity byly. Další část práce popisovala návrh aplikace a jejího uživatelského rozhraní. Na základě použitých aplikací byly specifikovány hardwarové požadavky.

V praktické části byly popsány vybrané komponenty a jejich použití v praxi spolu s ukázkami kódu s jejich implementací. Jako programovací jazyk byla použita Java, vývojové prostření Eclipse s frameworky Vaadin a Java Persistence API. Dále byly představeny jednotlivé obrazovky aplikace a popsány jejich funkcionality. V poslední části bylo ukázáno nasazení aplikace na testovací server, instalace a konfigurace softwaru vyžadovaného k běhu aplikace a testy s různě velkými soubory. Při těchto testech byly monitorovány zdroje serveru, a to CPU a operační paměť. Byla také sledována závislost spotřeby zdrojů na počtu připojených uživatelů.

SEZNAM POUŽITÉ LITERATURY

[1] SCHILDT, Herbert. Mistrovství – Java. Brno: Computer Press, 2014. Mistrovství. ISBN 978-80-251-4145-8.

[2] BLOCH, Joshua. Effective Java. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, c2008. ISBN 978-0-321-35668-0.

[3] FRÄNKEL, Nicolas. Learning Vaadin 7 master the full range of web development features powered by Vaadin-build rich Internet applications. 2nd ed. Birmingham, UK: Packt Pub, 2013. ISBN 9781782169772.

[4] SCHWARTZ, Baron., Peter. ZAITSEV a Vadim. TKACHENKO. High performance MySQL. 3rd ed. Cambridge [Mass.]: O'Reilly, c2012. ISBN 978-1449314286.

[5] SHELDON, Robert. SQL: začínáme programovat. Praha: Grada, 2005. Průvodce (Grada). ISBN 80-247-0999-6.

[6] LASSER, Jon. Rozumíme UNIXu. Praha: Computer Press, 2002. Všechny cesty k informacím. ISBN 80-7226-706-x.

[7] EVANS, Benjamin J. Java in a Nutshell. Sixth edition. Beijing: O'Reilly & Associates, 2015. ISBN 978-1-449-37082-4.

[8] GRÖNROOS, Marko. *Book of Vaadin: Vaadin 7 Edition - 7th Revision* [online]. Vaadin, 2016 [cit. 2018-05-09]. Dostupné z: http://vaadin.com/download/book-of-vaadin/vaadin-7/pdf/book-of-vaadin.pdf

[9] VUKOTIC, Aleksa a James GOODWILL. Apache Tomcat 7. Berkeley, CA: Apress, 2011. ISBN 9781430237242.

[10] MySQL 5.7 Reference Manual: Including MySQL NDB Cluster 7.5 and NDB Cluster 7.6 [online]. Revision: 55858. 2018 [cit. 2018-02-03]. Dostupné z: https://downloads.mysql.com/docs

[11] VAJGL, Marek. Objektově orientované programování v Javě. 1. vyd. Ostrava, 2012

[12] DARWIN, Ian F. *Java cookbook*. Third edition. Sebastopol, CA: O'Reilly, 2014. ISBN 978-1-449-33704-9.

[13] Debian History. *Debian.org* [online]. 15.1.2016 [cit. 2018-02-02]. Dostupné z: https://wiki.debian.org/DebianHistory.

[14] What is PHP?. *MyPHP.net* [online]. [cit. 2018-02-02]. Dostupné z: http://cz1.php.net/manual/en/intro-whatis.php.

[15] What can PHP do?. *MyPHP.net* [online]. [cit. 2018-02-02]. Dostupné z: http://cz1.php.net/manual/en/intro-whatcando.php.

[16] CERNOSEK, Gary. A brief history of Eclipse. *IBM.com* [online]. 15.11.2005 [cit. 2018-02-05]. Dostupné z: https://www.ibm.com/developerworks/rational/library/nov05/cernosek/

[17] A Brief History of NetBeans. *NetBeans.org* [online]. [cit. 2018-02-04]. Dostupné z: https://netbeans.org/about/history.html.

[18] IntelliJ IDEA. *JetBrains.com* [online]. [cit. 2018-02-07]. Dostupné z: https://www.jetbrains.com/idea/.

[19] What is a JPA Entity?. *Oracle.com* [online]. [cit. 2018-02-12]. Dostupné z: https://docs.oracle.com/cd/E16439_01/doc.1013/e13981/undejbs003.htm.

[20] Maven. *FI MUNI WIKI* [online]. [cit. 2018-04-29]. Dostupné z: https://kore.fi.muni.cz/wiki/index.php/Maven.

[21] LORENZ, Róbert. *Bezpečnost: 5. Hašovací funkce, MD5, SHA-x, HMAC* [online]. 2011 [cit. 2018-04-23]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-BEZ/prednasky/bez5.pdf.

[22] Splnění minimálních hardwarových požadavků. Debian.org [online]. [cit. 2018-03-13]. Dostupné z: https://www.debian.org/releases/stable/amd64/ch03s04.html.cs.

[23] INTEL® CORE™ i7 PROCESSORS. Intel.com [online]. [cit. 2018-02-15]. Dostupné z: https://www.intel.com/content/www/us/en/products/processors/core/i7-processors.html?page=2.

[24] Intel® Xeon® Processor E7-8800/4800 v4 Product Families. Intel.com [online]. [cit. 2018-02-15]. Dostupné z: https://www.intel.com/content/www/us/en/processors/xeon/xeon-e7-8800-4800-v4-product-families-brief.html.

[25] Java Platform, Enterprise Edition 8 SDK - Release Notes: System Requirements. Oracle.com [online]. [cit. 2018-03-13]. Dostupné z:

http://www.oracle.com/technetwork/java/javaee/documentation/javaee7sdk-readme-1957703.html#System_Requirements.

[26] File upload size. *Moodle.org* [online]. [cit. 2018-04-23]. Dostupné z: https://docs.moodle.org/34/en/File_upload_size.

[27] You cannot download files that are 2 GB or larger. *Microsoft.com* [online]. [cit. 2018-04-29]. Dostupné z: https://support.microsoft.com/en-us/help/298618/you-can-not-download-files-that-are-2-gb-or-larger.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UI	User interface.
PHP	Hypertext preprocessor.
GPL	General Public License.
JVM	Java Virtual Machine.
JPA	Java Persistence API.
HTML	Hypertext Markup Language.
IIS	Internet Information Services.
IDE	Integrated Developmnent Enviroment.
MVC	Model-View-Controller.
CSS	Cascading Style Sheets.
TCP	Transmission Control Protocol.
FTP	File Transfer Protocol.
SQL	Structured Query Language.
UTB	Univerzita Tomáše Bati.
CPU	Central Processing Unit.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

Příloha č. 1: CD s textem diplomové práce a se zdrojovými kódy aplikace.