

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM MONITOROVÁNÍ STAVU PLÁNOVACÍCH ÚLOH

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

MARTIN MAGA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM MONITOROVÁNÍ STAVU PLÁNOVACÍCH ÚLOH

LANNING TASK MONITORING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MAGA

VEDOUcí PRÁCE
SUPERVISOR

Ing. ZDĚNEK LETKO, Ph.D.

BRNO 2013

Abstrakt

.

Abstract

Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Java EE 6, Java, SOA, Java Beans, Java Server Faces, Monitorovanie, Twitter, Bootstrap, Systém .

Keywords

Java EE 6, Java, SOA, Java Beans, Java Server Faces, Monitoring, Twitter, Bootstrap, System.

Citace

Martin Maga: Systém monitorování stavu plánovacích úloh, bakalářská práce, Brno, FIT VUT v Brně, 2013

Systém monitorování stavu plánovacích úloh

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Zděnka Letka

.....

Martin Maga
5. března 2014

Poděkování

Veľmi rád by som poďakoval za vedenie mojej bakalárskej práce pánovi Zďenkovi Letkovi a pánovi Martinovi Večeřovi, ktorý mi poskytl rady a podali pomocnú ruku vždy, keď som narazil na problém.

© Martin Maga, 2013.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	2
2	Java Enterprise edition 6	3
2.1	Aplikačný model	3
2.2	Distribovaná viacstupňová aplikácia	4
2.3	Bezpečnosť	5
3	Java EE komponenty	6
3.1	Java EE klienti	6
3.2	The JavaBeans Component Architecture	7
3.3	Webové komponenty	7
3.3.1	Web Services	7
3.4	JavaServer Faces	7
3.5	Enterprise JavaBeans	8
4	JBoss	9
4.1	História JBoos-u	9
4.2	OptaPlanner	10
4.2.1	Princíp plánovania	10
5	Grafické užívateľské rozhranie	11
5.1	Twitter Bootstrap	11
5.1.1	História	11
5.1.2	Výhody	11
5.1.3	Komponenty	11
5.2	Návrh rozhrania	12
6	Záver	15

Kapitola 1

Úvod

Kapitola 2

Java Enterprise edition 6

V posledných rokoch prevláda tendencia tvorby komplexných systémov, ktoré spracovávajú veľké množstvá dát.

Java Enterprise Edition 6 (Java EE 6) predstavuje platformu určenú na vývoj webových a podnikových aplikácií.^[2] Tieto aplikácie sú viacvrstvové z dôvodu lepšej prenositeľnosti, nasaditeľnosti a modifikovateľnosti. Frontend, predstavujúci užívateľské rozhranie a logiku na jeho ovládanie, pozostáva z webových frameworkov, stredná vrstva poskytuje bezpečnosť a transakcie. Najnižšia vrstva poskytuje pripojenie k databázam. Java EE 6 je vhodná pre implementáciu podnikovej logiky, ktorá dokáže byť riadená alebo interagovať s inými podnikovými aplikáciami. Java Enterprise Edition 6 je platformou, ktorá poskytuje širokú škálu aplikačných programových rozhraní (API), ktoré zjednodušujú, zkracujú a znižujú komplexnosť výslednej aplikácie. Jej vývoj neustále napreduje a je spravovaný Java Community process (JCP).

Java EE 6 bola uvoľnená v decembri 2009 a poskytuje ľahké používanie a kompletnú sadu nástrojov na tvorbu podnikových aplikácií. V súčasnosti vyšla ďalšia verzia, ktorá priniesla ďalšie novinky, ktoré môžete dohľadať v online dokumentácii. V ďalších kapitolách si rozoberieme aplikačný model jazyka, ktoré je veľmi dôležitý pre pochopenie princípu činnosti aplikácií vyvinutých touto platformou. Následne sa zameriame na JBoss a OptaPlanner, pre ktoré je určené užívateľské rozhranie.

2.1 Aplikačný model

Java EE je určená pre implementáciu služieb pre zákazníkov, zamestnancov, dodávateľov kohokoľvek, kto prispieva do podniku. Takéto aplikácie sú komplexné a môže byť k nim pristupované z rozličných zdrojov. Pre lepšie zvládanie sú sústredené do stupňov.

Java EE definuje spôsob implementácie služieb, ktoré sú škálovateľné, prístupné a spravovateľné podnikovou aplikáciou. Tento model implementuje viacstupňový model do nasledujúcich častí:

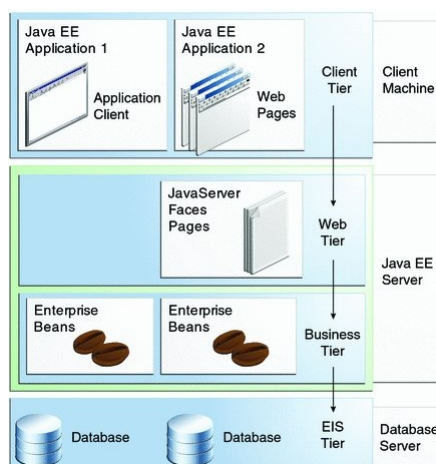
- Podniková a prezentačná logika poskytovaná Java EE platformou
- Štandardná systémová platforma

2.2 Distribuovaná viacstupňová aplikácia

Java EE používa aplikácie, ktoré sú viacvrstvové(multitier). Aplikčná logika je rozdelená medzi komponenty podľa funkcie.[1] Jednotlivé komponenty sa následne rôzne inštalujú v závislosti, do ktorého stupňa patria. Jednotlivé stupne sa skladajú z rôznych komponent:

- Klientská komponenta, ktorá beží na klientskom počítači
- Webová komponenta, ktorá beží na Java EE serveri
- Podniková komponenta, ktorá beží na Java EE serveri
- Enterprise Information System software bežiaci na EIS serveri

Napriek tomuto rozloženiu býva aplikácia rozložená len do 3 stupňov: Java EE server, klient, databázy na pozadí. Typicky beží medzi klientskom a databázou častou viac-vláknový Java EE server, ktorý býva označovaný skratkou EIS. Viacstupňovérozloženie môžete ná-zorne vidieť na obrázku č. 2.1. Java EE aplikácia beží na klientskej stanici, býva obvykle



Obrázek 2.1: Model Java EE, prevzaté z <http://docs.oracle.com/javaee/6/tutorial/doc/>

reprezentovaná tenkým klientom(webovým prehliadačom), nazývaným „thin client“, alebo hrubým klientom, do ktoré je vložená logika aplikácia. V strednej časti obrázku sa nachádza Java EE server, na ktorom môžu bežať rôzne technológie v závislosti od požiadavky výslednej aplikácie. Java EE server môže by reprezentovaný rôznymi dostupnými technológiami, či už sa jedná o open-source riešenia (JBoss, Tomcat, GlassFish) alebo komerčné riešenia (IBM WebSphere, BEA WebLogic), ten obsahuje rôzne komponenty, ktoré so sebou rôzne komunikujú a interaugujú na požiadavky klienta a na druhej strane komunikujú s databázovým systémom. Tieto komponenty predstavujú logiku aplikáciu. Posledná časť predstavuje databázový systém, ktorý obsahujú dáta, ktoré klient požaduje pri svojom požiadavku, tento server sa nazýva "EIS"

2.3 Bezpečnosť

Java EE vytvára prostriedky, ktoré sú prenositeľné a tak je možné systémy nasadzovať naprieč rôznymi platformami. Poskytuje mechanizmus prístupovej kontroly pravidiel, ktoré sú interpretované, keď je aplikácia nasadzovaná na server. Java implementuje mechanizmus prihlasovania štandardne, čím odpadá programátorovi povinnosť túto časť dodatočne implementovať.

V ďalšej časti sa zameriame na podrobné vysvetlenie jednotlivých Java EE komponent, pretože ich pochopenie bude nevyhnuté.

Kapitola 3

Java EE komponenty

Java EE pozostáva z komponent, čo je vlastne sada sebestačného softwaru, ktorý je zhromaždený do Java EE aplikácií a komunikujú s inými komponentami. Java EE definuje nasledujúce komponenty:

- Aplikačný klienti a aplety bežiaci na klientovi
- Java Servlet, JavaServer Faces, and JavaServer Pages (JSP) komponenty bežia na Java EE serveri[2].
- Enterprise JavaBeans (EJB) komponenty bežia na Java EE serveri[2].

V nasledujúcej časti sa zameriame na vysvetlenie jednotlivých pojmov, pretože sú dôležité z hľadiska pochopiteľnosti výslednej aplikácie.

3.1 Java EE klienti

Java EE client je typický webový klient alebo aplikačný klient.

Typický webový klient pozostáva z dvoch častí:

- Dynamické webové stránky pozostávajúce z rôzneho značkovacieho jazyka(HTML, XML), ktoré sú generované webovými komponentami
- Webovým prehliadačom, ktorý zobrazuje stránky

Typický webový klient sa nazýva "thin" klient, pretože nedotazuje nad databázou, alebo implementuje zložitú business logiku. Všetky tieto činnosti vykonáva Java EE server. Klient len posiela požiadavky Java EE serveru a ten vykonáva dotazovanie a predávanie výsledkov.

Aplikační klienti bežia na klientských počítačoch, kde poskytujú bohatšie užívateľské rozhranie ako webový rozhrania značkových jazykov. Typicky býva vytváraný technológiou Swing¹ alebo Abstract Window Toolkit(AWT)², občas sa vyskytuje aj príkazový riadok ako rozhranie.

¹[http://cs.wikipedia.org/wiki/Swing_\(Java\)](http://cs.wikipedia.org/wiki/Swing_(Java))

²http://en.wikipedia.org/wiki/Abstract_Window_Toolkit

3.2 The JavaBeans Component Architecture

Server a klient tiež zahŕňa komponenty založené na JavaBeans component architecture (JavaBeans components)³, ktoré správajú toky dát medzi:

- Aplikačným klientom, alebo apletom a komponentami bežiacimi na Java EE serveri
- Serverovými komponentami a databázami

Komunikácia medzi klientom s podnikovým stupňom, ktorý sa nachádza na Java EE serveri, v prípade, že klient beží v prehliadači.

3.3 Webové komponenty

Java EE webové komponenty sú súčasťou servletov alebo webových stránok, ktoré sú vytvorené JavaServer Faces technológiu⁴ (JSF) and JavaServer pages (JSP)⁵ technológiou. Servlety sú javovské triedy, ktoré dynamicky spracovávajú požiadavky a tvoria odpovede. JSP stránky sú textové dokumenty, ktoré pracujú ako servlety ale umožňujú prirodzenejší prístup k vytváraniu statického obsahu. JavaServer Faces technológia stavia na servlety a JSP technológii a poskytuje užívateľské rozhranie komponentný framework pre webové aplikácie.

3.3.1 Web Services

Webové služby sú klientske a serverové aplikácie, ktoré komunikujú cez Hypertext Transfer Protocol (HTTP)⁶. Webové služby poskytujú štandardné prostriedky, ktoré sú interoperabilné medzi softvérovými aplikáciami bežiacimi na rôznych platformách. Webové služby sa vyznačujú veľkou interoperabilitou a rozšíriteľnosťou, rovnako ako ich strojovým spracovaním, vďaka použitiu XML. Webové služby môžu byť kombinované vo voľne spojených spôsoboch, ako dosiahnuť zložité operácie. Programy poskytujúce jednoduché služby môžu na seba vzájomne a poskytovať sofistikované služby.

3.4 JavaServer Faces

JavaServer Faces zodpovedá serverovej strane frameworku pre užívateľské rozhrania vychádzajúce z Javy. JavaServer Faces (JSF) vytvára aplikácie na základe MVC - Model-View Controller⁷. Ďalej treba spomenúť, že pomáha previazať užívateľské údaje so serverom, rovnako ako aj komponenty, ktoré sú znova použiteľné.

Aplikácia, ktorá je vytvára týmto frameworkom pozostáva z webových stránok, grafických komponent, sadou komponent naviazané na serverovú časť. Môže obsahovať rôzne deskriptory a konfiguračné súbory, ktoré nám pomáhajú pri nasadzovaní aplikácie. Základom JavaServer Faces je Facelets, čo je vlastne výzorovo deklaračný jazyk pre JSF. Facelets stránky používajú XHTML 1 a CSS⁸.

³<http://docs.oracle.com/javaee/1.4/tutorial/doc/JSPIntro8.html>

⁴http://cs.wikipedia.org/wiki/JavaServer_Faces

⁵http://cs.wikipedia.org/wiki/JavaServer_Pages

⁶http://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol

⁷<http://en.wikipedia.org/wiki/Model-view-controller>

⁸http://en.wikipedia.org/wiki/Cascading_Style_Sheets

3.5 Enterprise JavaBeans

Enterprise JavaBeans(EJB) je používaný na vývoj a nasadzovanie komponentne založených aplikácií, ktoré sú škálovateľné a zabezpečené. Typicky obsahuje podnikovú logiku, ktorá pracuje s podnikovou databázou. Informácie o transakčných a bezpečnostných atribútoch býva typicky uložená v metadátach alebo v samostatnom XML súbore. Inštancia sa typicky spravuje za behu kontajnerom. Tento kontajner je prístupný klientovi. JavaBeans poskytuje všetky postriedky, ktorý súvisia s transakciami stavovým spravovaním, takže ponúkajú možnosť sa sústrediť na podnikovú logiku.

V ďalšej časti sa zameriame na JBoss a OptaPlanner.

Kapitola 4

JBoss

JBoss, čo je vlastne skratka pre JavaBeans Open Source Application Server, v súčasnosti nazývaný WildFly je aplikačný server, ktorý je založený na platforme Java a Java Enterprise Edition.[4] Keďže je tento server napísaný v Jave je možnosť ho používať naprieč rôznymi platformami. Tento server slúži na vývoj a nasadzovanie podnikových aplikácií, webových aplikácií, služieb a portálov. Tento server je licenovaný pod GNU Lesser General Public License(GNU PL).

4.1 História JBoss-u

Všetko naštartoval v roku 1999 Marc Fleury. Pre podporu vývoja middleware InterBohemia sa rozhodol implementovať jeden zo štandardov J2EE , EJB kontajner. Tým sa zrodil prvý projekt - EJBoss, ktorý sa neskôr premenoval na JBoss. TO niekoľko rokov neskôr sa stal prvým certifikovaným J2EE open source aplikačným serverom. Ďalej by som rád ukázal na vývoj rôznych verzií JBossu:

- JBoss AS 4.0 , Java EE aplikačný server 1.4 , je vybavený vloženým Apache Tomcat 5.5 servlet kontajnerom. JBoss môže bežať na mnohých operačných systémoch , vrátane mnohých POSIX platformách (ako GNU/Linux , FreeBSD a Mac OS X) , Microsoft Windows a ďalšie,.
- JBoss AS 5.1 , povolený v roku 2009 , pracuje ako Java EE 5 aplikačný server. Je to menšia aktualizácia hlavnej verzie JBoss AS 5.0, ktorý bol vo vývoji po dobu najmenej troch rokov a bol postavený na vrchole novej JBoss microcontainer.
- JBoss AS 6.0 , bol neoficiálne implementáciou Java EE 6 , vydané 28. decembra 2010 .
- JBoss AS 7 , bola vydaná 12. júla 2011 , len šesť mesiacov po poslednej hlavnej verzii , JBoss AS 6. JBoss AS 7 podporuje rovnakú špecifikáciu Java EE ako posledná verzia, a to Java EE 6. Java EE profil je iba čiastočne implementovaný v JBoss AS 7. Hlavné zmeny viditeľné pre užívateľa sú : oveľa menšiu veľkosť (menej než polovica z JBoss AS 6) a násobné zníženie v čase spustenia.
- JBoss AS 7.1 , aktuálna stabilná verzia bola vydaná vo februári 2012 . Zostávajúce časti EE špecifikácie boli realizované , a táto verzia bola certifikovaná pre EE plnom profile.

- WildFly 8 je priamym pokračovaním na JBoss AS projektu .

V ďalšej časti sa zameriame na OptaPlanner¹, čo je vlastne systém, pre ktorý je rozhranie navrhované.

4.2 OptaPlanner

OptaPlanner je open source software a ďalšie pokračovanie frameworku JBoss Drools. Optaplanner je framework, kde si môžete vytvoriť pravidlá, ktoré určujú, kedy by mali byť vykonané špecifické akcie. To by mohlo byť vykonané v kóde za použitia podmienok. Vytvorenie je pomocou pravidiel môže robiť oveľa jednoduchšie spájať mnoho pravidiel s mnohými akciami. Tieto pravidlá bývajú typicky definované pomocou XML súboru.

OptaPlanner pomáha programátorovi riešiť obmedzenie problémov spokojnosti efektívne. Pod kapotou sa kombinuje optimalizačné heuristiky na výpočet skóre.

Všetky prípady použitia sú pravdepodobne NP - úplné. Laicky povedané, to znamená : Neexistuje žiadne optimálne riešenie problému v primeranej lehote.

Brute force algoritmus bude trvať príliš dlho . Rýchly algoritmus nie je zďaleka optimálny. Pri použití pokročilých optimalizačných algoritmov je možné nájsť dobré riešenie v primeranom čase po týchto problémoch.

4.2.1 Princíp plánovania

Obvykle , problém plánovanie má aspoň 2 úrovne obmedzenia :

- Negatívne obmedzenia
- Pozitívne obmedzenia

Tieto obmedzenia definujú výpočetné skóre problému plánovania. Každé riešenie problému plánovanie môže byť odstupňovaná so skóre.

Plánovanie problému má niekoľko riešení . Existuje niekoľko kategórií riešení:

Možným riešením je nejaké riešenie, či je alebo nie je ľubovoľný počet obmedzení. Problémy plánovanie mávajú neuveriteľne veľké množstvo možných riešení. Mnoho z týchto riešení sú bezcenné . Uskutočniteľným riešením je riešenie, ktoré neporušuje žiadne (negatívne) tvrdé obmedzenia. Niekedy nie sú realizovateľné riešenie. Každý uskutočniteľné riešenie je možné riešenie. Optimálnym riešením je riešenie s najvyšším počtom bodov . Problémy plánovanie mávajú jedno alebo niekoľko optimálnych riešení. K dispozícii je vždy aspoň 1 optimálnym riešením, a to aj v prípade , že neexistujú žiadne uskutočniteľné riešenie, a optimálne riešenie nie je možné . Najlepším riešením je nájsť riešenie s najvyšším skóre zistené implementáciou v danom čase.

OptaPlanner podporuje niekoľko optimalizačných algoritmov ako efektívne prehrýzť týmto neuveriteľne veľkým množstvom možných riešení. V závislosti na prípade použitia, niektoré optimalizačné algoritmy dosahujú lepšie výsledky ako ostatné, ale to je nemožné povedať dopredu. Pri plánovaní , je ľahké prepnúť algoritmus optimalizácie, zmenou konfigurácie Solver na niekoľkých riadkov XML alebo kódu.

V ďalšej kapitole by som rád uviedol problematiku užívateľského rozhrania.

¹<http://www.optaplanner.org/>

Kapitola 5

Grafické užívateľské rozhranie

V tejto kapitole sa zameráme na problematiku užívateľského rozhrania, ktoré vlastne je reprezentované Web services. Toto rozhranie bude umožňovať nahrávať pravidlá, zobrazovať výsledky, spúšťať, pozastávať a zobrazovať detaily úloh. Keďže táto výsledná aplikácia by mala byť použiteľná aj na mobilnom telefóne bol vybraný štýlovací framework Twitter Bootstrap, ktorý značne uľahčuje tvorbu takéhoto rozhrania.

5.1 Twitter Bootstrap

Twitter Bootstrap je veľmi jednoduchý a voľne dostupný súbor nástrojov pre vytváranie moderného webu a webových aplikácií.[3] Ponúka podporu najrôznejších webových technológií HTML , CSS , JavaScript¹ a mnoho prvkov , ktoré je možné ľahko implementovať do svojej stránky. Pre použitie Twitter Bootstrap sú nutné základné znalosti HTML a CSS. Interaktívne prvky ako sú tlačidlá, boxy , menu a ďalšie kompletne nastavené a graficky spracované elementy je možné vložiť iba pomocou HTML a CSS .

5.1.1 História

Mark Otto a Jacob Thornton z Twitteru vyvinuli Twitter Bootstrap ako framework pre podporu konzistencie interných nástrojov. Pred vyvinutím Twitter bootstrap , boli v spoločnosti používané rôzne knižnice pre vývoj rozhrania, čo viedlo k nejednotnosti zdrojových kódov a vysokým nákladom na údržbu.

5.1.2 Výhody

Výhodou tohto súboru nástrojov je jednoduché spracovanie akéhokoľvek užívateľského rozhrania vo webovej aplikácii a nerozhoduje , či to je napríklad užívateľské rozhranie v administrácii back-endových alebo front-endových aplikácií.

5.1.3 Komponenty

Dohromady poskytujú komponenty a JavaScript pluginy nasledujúce elementy užívateľského prostredia:

- Button groups - Skupiny tlačidiel

¹<http://en.wikipedia.org/wiki/Javascript>

- Button dropdowns - Vysúvacia tlačidlá
- Navigational tabs, pills, and lists - Záložky, pilulky a zoznamy pre navigáciu
- Navbar - navigačné položky
- Labels - štítky
- Badges - "odznáčky"
- Page headers and hero unit - hlavičky stránky a "hero unit"
- Thumbnails - náhľady
- Alerts - výstrahy
- progress bars
- Modals
- Dropdowns - vysúvacie menu
- Tooltips
- Popovers
- Accordion
- Carousel - posuvný slider
- Typeahead

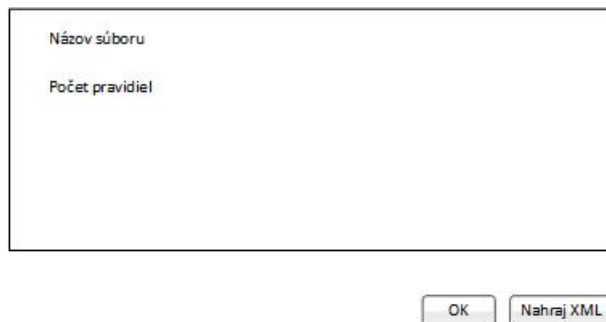
Podrobné vysvetlenie jednotlivých komponent nájdete na nasledujúcej adrese <http://getbootstrap.com/>, rovnako aj s príkladmi použitia. V nasledujúcej časti prejdeme na samotný návrh užívateľského rozhrania.

5.2 Návrh rozhrania

Výsledné rozhranie kladie dôraz na jednoduchosť a prehľadnosť zobrazených úloh. Rozhranie je rozdelené na 2 časti. Prvá časť, obrázok č. 5.1 popisuje spôsob nahratia xml súboru zo súborového systému. Užívateľ stlačí tlačidlo na nahranie xml, kde sa následne zobrazí obsah súborového systému a užívateľ zvolí príslušný xml súbor. Následne sa vyplnia údaje o názve súboru a počtu pravidiel v xml. Následne užívateľ potvrdí výber tlačidlom "OK" a prejde na hlavné okno, ktoré priebežne zobrazuje stav vykonávania úlohy a informácie o úlohe.

Ďalej by som rád ukázal na obrázku č. 5.2 spôsob zobrazovania úloh spolu s jeho stavom. Toho rozhranie sa dá rozdeliť na 2 časti: Prvá časť je reprezentovaná tlačidlami, ktoré reprezentujú spôsob na :

- Pridávanie nových úloh, ktoré po stlačení zobrazí okno, ktoré je reprezentované obr. č. 5.1
- Editovanie skončenej úlohy
- Zrušenie vybranej úlohy



Obrázek 5.1: Ukážka nahratia súboru

- Spravovanie užívateľov(platí pre rolu Administrátor)
- Pozostavenie bežiacej úlohy

Ďalšia časť zobrazuje úlohy. Každá úloha je reprezentovaná rôznymi informáciami, ktoré sú rozdelené do stĺpcov. Každý záznam je prezentovaný položkou ID, ktorá reprezentuje jedinečný identifikátor v rámci systému spracovania, názvu úlohy, ktorá bola vyextrahovaná z xml súboru, statusu úlohy, ktoré je reprezentovaný 3 položkami:

- Dokončená úloha - finished
- Running - bežiaca úloha
- Paused - pozastavená úloha

, ukazateľom spracovania úloh, ktorá ukazuje aktuálne spracovania úlohy a položkou ETA, ktorá ukazuje odhadovaný čas do konca spracovania úlohy. Daná tabuľka obsahuje viacero položiek, ktoré rôzne zobrazujú stavy úloh. Úlohy v stave "finished" je možné otvoriť a zobraziť detaily spracovania, zmeniť parametre a spustiť nový beh úlohy, pričom pôvodná úloha zostane zachovaná. Pridávanie úloh možno realizovať aj za behu iných úloh, rovnako pozastavené úlohy možno znova spustiť z inými detailami.

<div> <div>Pridať novú úlohu</div> <div>Pozastaviť úlohu</div> <div>Spravovať užívateľov</div> <div>Editovať úlohu</div> <div>Zrušiť úlohu</div> </div>				
ID	Name	Status	Progress %	ETA
			<div></div>	

Obrázek 5.2: Ukážka zobrazenie stavu úloh

Kapitola 6

Záver

Táto práca je zameraná na priblíženie technológie Java EE 6, JBoss, OptaPlanner a výsledného rozhrania. Bežnému užívateľovi sú krok po kroku vysvetlované všetky používané metódy. Všetky tieto technológie sú voľne dostupné na internete, preto užívateľ má možnosť sa naučené veci vyskúšať si aj svojpomocne. Výsledný návrh rozhrania kladie dôraz na jednoduchosť, prehľadnosť a širokú podporu. Všetky tieto nové technológie mi pomohli zlepšiť rozhľad v IT technológiách a ukázali mne aj iným princíp podnikových aplikácií, ktoré sa dnes vyvíjajú a používajú.

Literatura

- [1] Arun Gupta: *Java EE 6 Pocket Guide*. O'REILLY, 2012, iISBN 978-1-449-33668-4.
- [2] Jendrock, E.; Cervera-Navarro, R.; Evans, I.; aj.: The Java EE 6 Tutorial [online]. <http://docs.oracle.com/javaee/6/tutorial/doc/>, 2013-11-03 [cit. 2013-10-25].
- [3] Thilo, M. O. J. T. C. R. J.: Twitter Bootstrap [online]. <http://getbootstrap.com/>, 2013-12-16 [cit. 2013-12-27].
- [4] WWW stránky: JBoss. <http://www.jboss.org/>, 2013-12-16 [cit. 2013-12-27].