VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMÚ

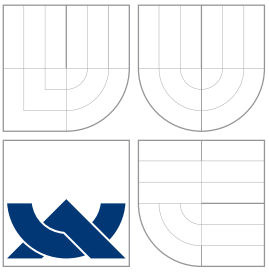
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM MONITOROVÁNÍ STAVU PLÁNOVACÍCH ÚLOH

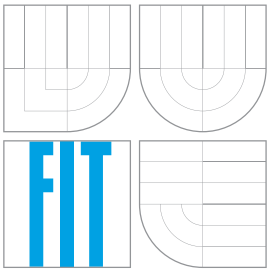
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN MAGA
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMÚ

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 Brnì, 2013

Systém monitorování stavu plánovacích úloh

Prohlášení

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

.....
Martin Maga
16. dubna 2014

Poděkování

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

Obsah

Kapitola 1

Úvod

Kapitola 2

Java Enterprise edition 6

V posledných rokoch prevláda tendencia tvorby komplexných informačných systémov, ktoré spracovávajú veľké množstvo dát. Preto sa zvyšuje tlak na vývojárov na tvorbu prostriedkov, ktoré dokáže takéto systémy ľahko a rýchlo vytvárať. Jedným z takýchto protriedkov je platforma Java Enterprise Edittion 6(Java EE 6). Java EE 6 predstavuje platformu určenú na vývoj webových a podnikových aplikácií.[6]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 vrsta 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 EE 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 Comunnity process(JCP).

Java EE 6 bola uvoľnená v decembri 2009 a poskytuje ľahké používanie a kompletnú sadu nástrojov na tvorbu podnikových aplikácií. V súčastnosti vyšla ďalšia verzia, ktorá priniesla ďalšie novinky, ktoré môžte dohľadať v online dokumentácií. V ďalších kapitolách si rozoberieme aplikaný model jazyka, ktoré je veľmi dôležitý pre pochopenie princípu činnosti aplikácií vyvinutých touto platformou. V ďalšej kapitole rozobereme aplikačný model platformy Java EE 6.

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.

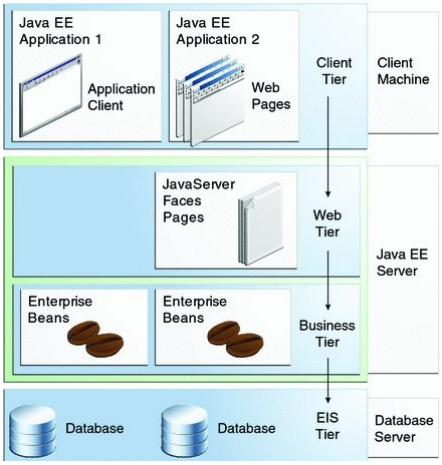
Java EE definuje spôsob implementácie služieb, ktoré sú škálovateľné a prístupné.

2.2 Distribuovaná viacstupňová aplikácia

Java EE definuje aplikácie, ktoré sú viacvrstvové(multitier). Aplikačná logika je rozdelená medzi komponenty podľa ich funkcie.[2] Jednotlivé komponenty sa následne rôzne inštalujú v závislosti, do ktorého stupňa patria. Jednotlivé stupňe sa skladajú z rôznych komponent:

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

Napriek tomúto rozloženiu býva aplikácia rozložená len do 3 stupňov: Java EE server, klient, databáze. Typicky beží medzi klientskom a databázou častou viac-vláknový Java EE server, ktorý býva označovaný skratkou EIS. Viacstupňovérozloženie môžete názorne vidieť na obrázku č. 2.1. Java EE aplikácia beží na klientskej stanici, býva obykle re-



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

prezentovaná tenkým klientom(webovým prehliadačom), nazývaným „thin client“, alebo hrubým klientom, do ktoré je čiastočne vložená logika aplikácia. V strednej časti obrázku sa nachádza Java EE server, na ktorom môžu bežať rôzne technológie v závislosti od požiadavky výslednej aplikácie a možností daného servera. 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 interagujú na požiadavky klienta a na druhej strane komunikujú s databázovým systémom. Tieto komponenty predstavujú logiku aplikáciu. Posledná časť predstavuje databázový systém, ktorý obsahuju 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 programátori povinnosti túto časť dodatočne implementovať.

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

2.4 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[6].
- Enterprise JavaBeans (EJB) komponenty bežia na Java EE serveri[6].

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

2.5 Java EE klienti

Java EE client je typicky 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 bussiness logiku. Všetky tieto činnosti vykonavá 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 značkovacích jazykov. Typicky býva vytváraný technológiou Swing[1] alebo Abstract Window Toolkit(AWT)[2], občas sa vyskytuje aj prikázový riadok ako rozhranie.

[1] http://cs.wikipedia.org/wiki/Swing_(Java)
[2] http://en.wikipedia.org/wiki/Abstract_Window_Toolkit

2.6 The JavaBeans Component Architecture

Server a klient tiež zahŕňa komponenty založené na JavaBeans component architecture(JavaBeans components), ktoré správujú 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.

2.7 Webové komponenty

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

2.7.1 Java Persistence API

Java Persistence API(JPA) je špecifikácia jazyku Java, ktorý poskytuje objektovo relačné mapovanie. Java Persistence API využíva pre mapovanie entity, ktoré reprezentujú dáta v databázi. Typicky teda reprezentuju tabuľku databáze a jej každá inštacia riadok tabuľky. Entitná trieda má teda atribúty, ktoré priamo odpovedajú názvu stĺpcov v databázovej schéme. To značne uľahčuje a zprehľadňuje prácu s databázou. Pre prístup k databáze sa používa trieda EntityManager, ktorá spracováva transakcie a zabezpečuje aby boli spĺňali podmienky ACID. Java Persistence API rovnako definuje vlastný jazyk Java Persistence Query Language, čo je jazyk podobný jazyku SQL, ktorý využíva trieda EntityManager pri svojej práci. Výhodou je, že tento jazyk je nezávislý na zvolenej databázovej technológií a má objektové vlastnosti, teda pri dotazoch nepoužívame konkrétne názvy tabuliek ale názvy entitný tried a jej vlastností.

Medzi konkrétne implementácie JPA patrí: Hibernate, Oracle TopLink, OpenJPA.

2.7.2 Enterprise Bean

Enterprise Bean(WB) sú Java EE komponenty, ktoré implementujú Enterprise JavaBeans(EJB) technológie. Enterprise bean beží v kontajneri EJB. EB je server-side komponentov, ktorá zapuzdruje enterprise logiku aplikácie. Obchodná logika je kód, ktorý spĺňa účel použitia. Z niekoľkých dôvodov, EB zjednodušuhe vývoj rozsiahlych, distribuovaných aplikácií. Po prvé, pretože kontajner EJB poskytuje služby na úrovni systému a

vývojár sa môže sústrediť na riešenie obchodných problémov. Kontajner EJB, je zodpovedný za služby na úrovni systému, ako je riadenie transakcií a autorizácie zabezpečenia.

EB sa delia na 2 kategórie:

- Message-driven - Vykoná úloha pre klienta; voliteľne môže implementovať webové služby
- Session - Pôsobí ako poslucháča pre určitý typ správ, ako je API Java Message Service

2.8 JavaServer Faces

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

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

V ďalšej časti sa zameriame na JBoss a OptaPlanner.

2.9 Web Service

Web Service sú klientské a serverové aplikácie, ktoré komunikujú prostredníctvom HTTP protokolu vymenieňaním XML správ. Tieto aplikácie poskystujú interoperabilitu medzi rôznymi platformami naprieč počítačou sieťou. Web Service umožňuje komunikáciu medzi rôznymi aplikáciami, ktoré bežia na rôznych platformách, napr. Java aplikácie založené na Windowse komunikujú s Net. aplikácia bežiaci na Linuxe. Webo services sa delia na 2 kategórie "big"web services a "RESTFul"web services.

2.9.1 "Big"Web Services

V Jave EE 6 existuje API, ktoré sa nazýva JAX-WS, ktoré umožňuje vytvorenie práve tohto typu web servici.[6] "Big"web service využíva XML správy, spolu so SOAP a XML jazykom, ktorý definuje architekrúru a formát správ. Tento typ Web Service obsahuje definíciu pre Web Service vo formáte WSDL, ktorý je čitatelný aj počítačom. Formát SOAP správ a definíciu jazyka WSDL rozhrania môže znížiť zložitosť vývoja aplikácií, webových služieb.

2.9.2 RESTful Web Service

V Java EE 6 existuje pre druhý typ web service API, ktoré sa nazýva JAX - RS. Tento typ web serrvice je vhodný pre základné , ad hoc integračné scenáre. REST webové služby,

často lepšie integrované s HTTP ako službami založenými na SOAP je , nevyžadujú XML správ alebo definície WSDL služby - API.

2.9.3 Iné technológie

Maven

Maven je založený na centrálnej konceptu životného cyklu zostavenie. Čo to znamená, že proces budovania a distribúciu určitého artefaktu(projektu) je jasne definovaná. Depency management je jedným z rysov Maven-u, ktorý je najlepšie známy pre užívateľa a je jednou z oblastí, kde Maven vyniká. Maven sa používa hlavne pre multimodulové aplikácie pre zachovanie vysokého stupňa kontroly a stability.

Životný cyklus pozostáva z fáz. Každá z týchto zostavenie životný cyklus je definovaný iným zoznamu zostavenie fáz, pričom fáza zostavenie predstavuje fázu životného cyklu. Maven môže generovať dokumentáciu projektu a vytvoriť stavbu. Maven má niekoľko správ, ktoré sa môžu pridať na webové stránky a zobraziť aktuálny stav projektu. Tieto správy majú formu pluginov, rovnako ako tie používané na zostavenie projektu.

Internacionalizácia v Maven je veľmi jednoduché.[4]

Arquillian

Arquillian je integračne a funkčne testovacia platforma, ktorá môže byť použitá pre testovanie middleware Java. Jej hlavným cieľom je urobiť integráčné(a funkčné) testy tak jednoduché písať unit testy, čo umožňuje vývojárom riadenie behu v rámci testu. Arquillian podporuje integráciu s Java EE kontajnermi, ako je JBoss AS a GlassFish a servlet kontajnerov, ako sú Tomcat a Jetty, a podporuje vykonávanie testov v cloudových službách. Podpora kontajnerov umožňuje vývojárom zamerať sa na rad technologických platforiem, vrátane Java EE 5 a 6, servlet prostredie, OSGI, Embedded EJB a samostatné CDI.[1]

Kapitola 3

JBoss Aplication Server

JBoss, čo je vlastne skratka pre JavaBeans Open Source Applicatom Server, v súčastní nazývaný WildFly je aplikačný server, ktorý je založený na platforme Java a Java Enterprise Edition.[5] Aplikačný server tvorí vrstvu medzi aplikáciami a operačným systémom, pričom rovnako poskytuje aplikácia často využívané funkcie, napr. spracovanie transakcií, výmená správ, ... Aplikačné servery podobne ako JBoss Aplication Server(JBoss AS) je open source. JBoss je Java EE platforma pre vývoj a nasadzovanie Java aplikáciu, webových aplikácií, služieb a portálov.

JBoss AS je napísaný v Jave, preto existuje možnosť ho používať naprieč rôznym platformám. Tento server slúži na vývoj a nasadzovanie podnikových aplikácií, webových aplikácií, služieb a portálov. Tento server je licensovaný pod GNU Lesser General Public License(GNU PL).

3.1 História JBoos-u

Všetko naštartoval v roku 1999 Marc Fleury. Pre podporu vývoja middleware InterBohemia sebe 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:[11]

- 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 vydané 12. júla 2011, len šesť mesiacov po poslednej hlavnej verzie , JBoss AS 6. JBoss AS 7 podporuje rovnaké 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. , 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 .

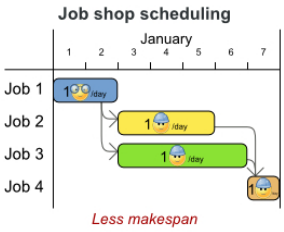
V ďalšej časti sa zameriame na OptaPlanner, čo je vlastne systém, pre ktorý je rozhranie navrhované.

Kapitola 4

OptaPlanner

OptaPlanner je odľahčený open source software a ďalšie prokračovanie frameworku JBoss Drools, ktorý optimalizuje plánovacie problémy.[9] Tieto plánovacie problémy môžu byť nasledujúceho charakteru:

- Plánovanie agendy: plánovanie schôdzok, vymenovanie, práca údržby
- Plánovanie vzdelávania: plánovanie lekcie, kurzov



Obrázek 4.1: Job, Shop scheluding, prevzaté z http://www.optaplanner.org/

Obrázok č. 4.1 zobrazuje typické použitie OptaPlanner-u. Môžme vidieť v nasledujúcom obrázku vystupú 4 osoby, ktoré vykonávajú nejakú činnosť. Ich činnosť je špecifická a silne závisí od práce predchádzajúcich. Optaplanner sa snaží ich činnosti maximálne optimalizovať a jednotlivé činnosti zvoliť v následnosti tak, aby výsledná práca bola spravená za najkratší možný čas vzhľadom na činnosť, ktorá sa optimalizuje.

4.0.1 NP-úplný problém

Každý plánovací problém je NP-úplný problém.[7] NP-úplné problémy sú nedeterministicky polynomiálne problémny, ktoré nie sú riešitelné v dostupnom čase, pretože sa nepodarilo nájsť deterministický algoritmus. Príkladom NP úloh môžme považovať: problém obchodné cestujúceho, ...

Riešenia poskytnutým týmto frameworkom, ktorý využíva pokročilé optimalizačné algoritmy, sú dosiahnuteľné v reálnom čase. Dosiahnutie v reálnom čase znamená nájdenie 1 alebo viacerých riešení, alebo nenájdenie žiadneho riešenia vzhľadom na poskytnutý čas a optimalizačné algoritmy, ktoré sú implementované.

Každý plánovací problém je definovaný na základe obmedzení, ktoré musia minimálne spĺnať: [3]

- Negatívne "hard"obmedzenie, ktoré nesmie byť porušené
- Negatívne "soft"obmedzenie, ktoré by nemali byť porušené pokiaľ sa dá tomu vyhnúť.

Niektoré problémy môžu obsahovať aj pozitívne podmienky alebo odmeny, ktoré by mali byť splnené pokiaľ je možné ich splniť.

Tieto podmienky definujú skóre plánovacieho problému. Tieto podmienky môžu byť zapísané v Jave alebo v Drools pravidlách, ktoré značne zjednodušujú kód.

Vytvorenie je pomocou pravidiel môže robiť oveľa jednoduchšie spájať mnoho pravidiel s mnohými akciami. Tieto pravidlá bývajú typicky definované pomocou XML súboru.

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

4.0.2 Výsledky plánovacieho problému

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.

4.0.3 Ukážka XML configuračného súboru

V nasledujúcom obrázku by som rád ukázal príklad XML configuračného súboru pre OptaPlanner.

Listing 4.1: Test

```
<?xml version="1.0" encoding="UTF-8"?>
<solver>
  <!--<environmentMode>FAST_ASSERT</environmentMode>-->
  <!-- Domain model configuration -->
  <solutionClass>org.optaplanner.examples.cloudbalancing.domain.
      CloudBalance</solutionClass>
  <planningEntityClass>org.optaplanner.examples.cloudbalancing.domain.
      CloudProcess</planningEntityClass>
  <!-- Score configuration -->
  <scoreDirectorFactory>
    <scoreDefinitionType>HARD_SOFT</scoreDefinitionType>
    <simpleScoreCalculatorClass>org.optaplanner.examples.cloudbalancing
        .solver.score.CloudBalancingSimpleScoreCalculator</
        simpleScoreCalculatorClass>
    <!--<scoreDrl>/org/optaplanner/examples/cloudbalancing/solver/
        cloudBalancingScoreRules.drl</scoreDrl>-->
  </scoreDirectorFactory>
  <!-- Optimization algorithms configuration -->
  <termination>
    <maximumSecondsSpend>120</maximumSecondsSpend>
  </termination>
  <constructionHeuristic>
    <constructionHeuristicType>FIRST_FIT_DECREASING</
        constructionHeuristicType>
    <!--forager-->
      <pickEarlyType>FIRST_NON_DETERIORATING_SCORE</pickEarlyType>
    <!--/forager-->
  </constructionHeuristic>
  <localSearch>
    <acceptor>
      <entityTabuSize>7</entityTabuSize>
    </acceptor>
    <forager>
      <acceptedCountLimit>1000</acceptedCountLimit>
    </forager>
  </localSearch>
</solver>
```

Konfigurácie solveru pozostáva z 3 častí:

- Domail model configuration: Musíme Optaplanneru uviesť hlávnú triedu.
- Configurácia skóre, ktorá hovorí Optaplanneru ako ma optimalizovať premenné. Pokiaľ používame hard a soft obmedzenia, použijeme "HardSoftScore". Musíme tiež uviesť ako vypočítať také skóre, v závislosti na našich požiadavkách. Ďalej sa, sme musíme pozrieť do 2 alternatívy pre výpočet skóre: pomocou jednoduchej implementácie Java, alebo pomocou Drols DRL.Optaplanner bude hľadať riešenie s najvyšším skóre. Budeme používať HardSoftScore, čo znamená, plánovač bude hľadať riešenie s žiadnymi tvrdými obmedzeniami členenie (spĺňajú požiadavky na hardvér) a pokiaľ možno čo najmenej mäkkých obmedzenia členenie (minimalizovať náklady na údržbu).
- Konfigurácia optimalizačných algoritmov

4.0.4 Optimalizačné algoritmy

V nasledujúcej časti textu si ukážeme optimalizačné algoritmy, ktoré používa optaplanner.

- First FIT - To je veľmi jednoduché greedy algoritmus aproximácie. Algoritmus spracováva položky v ľubovoľnom poradí. Pre každú položku, pokúsi sa umiestniť na položku v prvej priehradke, ktorý sa môže ubytovať položku. Ak nie je nájdený žiadna priehradka, otvára novú priehradku a kladie položku v rámci nového zásobníka.
- Firt FIT Decreasing
- Firt FIT Decreasing + heurestic local search
  - Hill Climbing - hill climbing je matematická optimalizácia technika, ktorá patrí do rodiny miestneho vyhľadávania. Jedná sa o iteratívny algoritmus, ktorý začína s ľubovoľným riešenie problému, potom sa pokúsi nájsť lepšie riešenie tým, že postupne mení jeden prvok riešenia. Ak zmena vytvára lepšie riešenie zmeny sa opakujú až žiadne ďalšie zlepšenie nie je možno nájsť.[8]
  - Tabu Search - Tabu search používa miestny alebo susedcký postup vyhľadávanie, tak že iteratívne presuvá z jedného možného riešenia x k lepšiemu riešeniu x v susedstve x, kým sa niektoré kritérium zastavenia splní. Miestne postupy vyhľadávania sa často uviaznu v zle bodovaných oblastiach. V snahe vyhnúť sa týmto nástrahám a preskúmať oblasti hľadaného miesta, ktoré by mali zostať bez prehliadky inými miestnymi postupmi vyhľadávania, tabu search starostlivo skúma okolí každého toku ako hľadanie postupuje. Riešenie prijatí do novej štvrti sú určené pomocou pamäťových štruktúr.

– Simulated Annealing - Simulované žíhanie je všeobecný algoritmus pre globálne optimalizačné problémy lokalizovať dobré priblíženie k globálnej optimálnej danej funkcie vo veľkom vyhľadávacieho priestoru. To sa často používa pri hľadaní priestorov diskrétnych (napr. všetky výlety, ktoré navštevujú danú množinu miest). U niektorých problémov, môže simulované žíhanie byť účinnejšie ako vyčerpávajúci zoznam - za predpokladu, že cieľom je iba nájsť prijateľne dobré riešenie v stanovenú dobu, skôr ako tým najlepším možným riešením .

V ďalšej kapitole by som rád uviedol problematiku užívateľského rozhrania.

Kapitola 5

Grafické užívateľské rozhranie

V tejto kapitole sa zameráme na problematiku užívateľského rozhrania, ktoré vlastne je reprezentované prostredíctvom technológie Java Server Faces v kombinácií frameworkov Rich Faces a Twitter Bootstrap-u. Toto rozhranie bude umožňovať nahrávať pravidlá, zobrazovať výsledky, spúšťať, pozastovať a zobrazovať detaily úloh. Keďže táto výsledná aplikácia by mala byť použíteľná aj na mobilnom telefóne bol vybratý štýlovací framework Twitter Bootstrap, ktorý značne uľahčenie tvorbu takéhoto rozhrania.

5.1 Twitter Bootstrap

Twitter Bootstrapje veľmi jednoduchý a voľne dostupný súbor nástrojov pre vytváranie moderného webu a webových aplikácií.[10] Ponúka podporu najrôznejších webových technológií HTML , CSS , JavaScript 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 .

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

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

5.2 Rich Faces

Rich faces predstavuje open-source Ajax knižnicu, ktorá predstavuje rozšírenie pre JavaServer Faces. Umožňuje integráciu schopností ajaxu do enterprise aplikácií. RichFaces obohacuje framework Ajax4jsf v dvoch dôležitých ohľadoch. Po prvé, sa rozširuje množstvo vizuálnych pripravených komponent. Po druhé, plne implementuje funkciu skinnability rámca Ajax4jsf vrátane veľkého množstva preddefinovaných vzhľadov. Pomocou skinnability, že je oveľa ľahšie riadiť vzhľad aplikácie.

5.3 Rozbor aplikácie

Výsledná aplikácia bude zobrazovať priebežné výsledky výpočtu frameworku optaplanner. Tento framework bude pre túto prácu optimalizovaný pre úlohu N Dám, ktorú bude schopný riešiť. Bude sa deliť na aplikáciu, ktorá predstavuje užívateľské rozhranie vytvorené prostredníctvom technológie Java Server Faces v kombinácií s Rich Faces nastylované prostredníctvom frameworku Twitter Boostrap, ktoré zároveň zabezpečuje prenositeľnosť rozhrania na mobilné telefóny. Užívateľské umožňuje zobrazovanie a spravovanie úloh, organizácií, do ktorých náležia jednotliví užívatelia a užívateľov. Každá časť systému bude sprístupnená podľa príslušnosti užívatela k užívateľskej role. Z tohto rozhrania bude môcť užívateľ pokiaľ mu to užívateľská rola dovoľuej pridať úlohu, ktorú môžme následne spustiť alebo pozastaviť. Spustenie prebehia zavolaním služby web service, ktorá obsahuje potrebné prostriedky na spustenie úlohy. Web service následne zavolá enterprise bean, v ktorej prebieha spracovanie danej úlohy. Výsledok výpočtu sa priebežne ukladá do databáze. Výsledku sa následne priebežne zobrazuje v užívateľskom rozhraní. Užívatelia majú prístup k obmedzenému počtu úloh, zároveň môžu vykonávať obmedzené akcie a to nasledovne:

- Administrátor - má prístup ku všetkým úlohám v systéme, úlohy môže editovať vytvárať, mazať, publikovať a odpublikovať , môže vytvárať, mazať a editovať užívateľov, rovnaké môžnosti má aj s organizáciami
- Plánovač - má prístup k úlohám v rámci svojej organizácie, môže vytvárať, editovať, mazať úlohy, publikovať a odpublikovať
- Čitateľ - úlohy môže len zobrať v rámci svojej organizácie, publikovať , odpublikovať

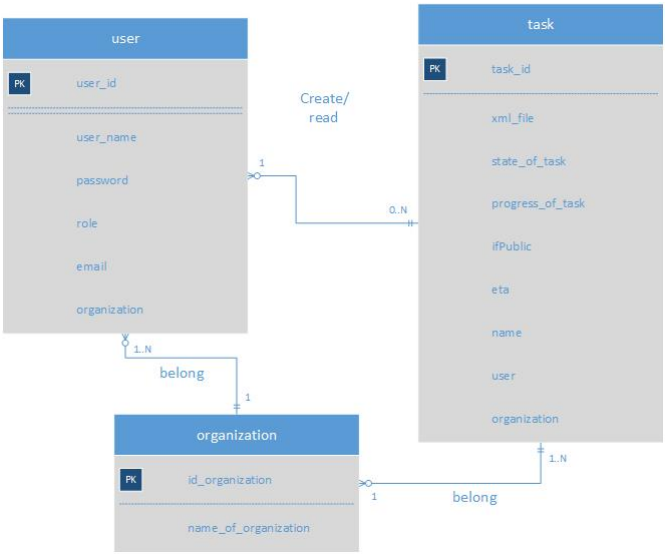
Výsledná aplikácia bude nasadená na aplikačný server JBoss.

5.3.1 Databázová technológia

Aplikácia potrebuje pre spracovanie úloh, organizácií a užívateľov databázu. Pre potreby bakalárskej práce bol vybratý relačný open-source databázový model MySQL. MySQL databázová technológia je veľmi vhodná pre malé a stredne veľke aplikácie, čo tá naša je, rovnako poskytuje dobrý výkon pri vykonávaní transakcií, umožňuje vytvárať procedúry, databázové triggere a jej inštalácia je pomerne jednoduchá a nezaberá veľa diskové priestoru, rovnako je MySQL multiplaformová, keďže je možné ju nasadiť na systémy s operačným systémov windows, linux, max os. Medzi nevýhody tejto technológie patrí neefektívna práca s databázovými transakciami, neefektívne ukladanie veľkého množstva dát.

5.3.2 Návrh modelu databáze

Na nasledujúcom obrázku je ukázaný ER diagram, ktorý bol použitý pre dtabázu:



Obrázek 5.1: ER diagram

Tento obrázok zobrazuje jednotlivé entity, ktoré sú potrebné na uloženie v databáze, každá z nich ma určité položky. ER diagrama sa skladá z 3 entít: user - entita, ktorá reprezentuje užívateľ, task - entita, ktorá reprezentuje úlohu a organization - entita, ktorá reprezentuje organizáciu. Výsledný návrh odpovedá skutočnosti, že každý užívateľ musí byť súčašťou organizácia, rovnako môže mať vytvorené 0 až N úloh. Taktiež pre zjednodušenie je každa úloha priradená priamo organizácií pre zlepšenie rýchlosti získania výsledkou a zjednodušenia ich nájdenia. Každá entita obsahuje primárny kľúč(jedná sa o silné entitné množiny), ktorý je odvodený od názvu a začína predponou „id_“ a pokračuje názvom entity. Poďme sa pozrieť bližšie na jednotlivé entity. Entitná množina organization obsahuje 2 položky jednou z nich je primárny klúč a ďalšou názov organizácia podľa, ktorej sú zaraďovaný jednotlivý užívatelia. Ďalej prejdime k entitnej množine user. Táto entita má rovnako primárny kľúč. Ďalej obsahuje položku pre užívateľské meno(user_name), heslo(password), email, užívateľskú rolu(role) a cudzí kľúčč organization, ktorý obsahuje na organizáciu. Nakoniec prejdime k entitnej množine task. Táto entitná množina obsahuje primárny kľúč, ďalej obsahuje xml súbor, ktorý reprezentuje danú úlohu(v našom prípade N dám), stav úloh(state_of_task, ktorý reprezentuje rôzne stavy úlohy), ktorý si podrobnejšie rozobereme. Úloha sa môže nachádzať v jednom z nasledujúcich stavov:

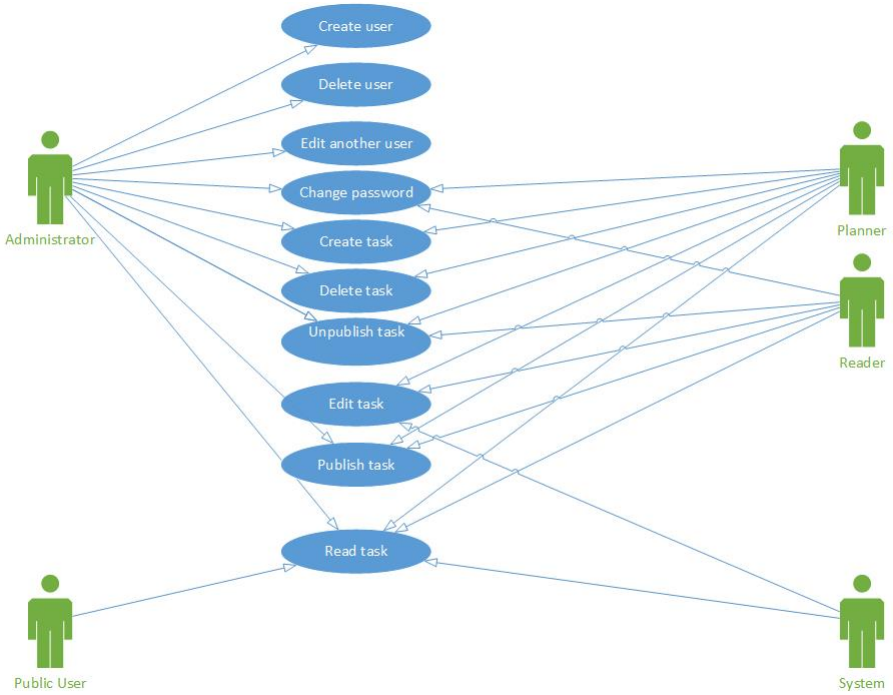
- NEW - úloha bola vytvorená
- MODIFIED - xml súbor bol modifikovaný
- WAITING - úloha čaká na spracovanie
- IN_PROGRESS - práve prebieha výpočet

- PAUSED - úloha je pozastavená
- COMPLETE - úloha je dokončená

Entitná množina task ďalej obsahuje položku, ktorá percentuálne hodnotí stav výpočtu úlohy(progress_of_task), čas do skončenia výpočtu úlohy(eta), nastavenie úlohy na privátnu alebo verejnú(ifPublic), názov úlohy(name) a cudzie kľúce user, ktorý odkazuje na užívateľa, ktorým bola úloha vytvorená a organization, ktorá odkazuje na organizáciu užívateľa, ktorým bola vytvorená. V ďalšej kapitole sa pozrieme na use case diagram.

5.3.3 Diagram užitia

Nasledujúci obrázok ukazuje príprady užitia systému: Na nasledujúcom obrázku č.5.2



Obrázek 5.2: UseCase diagram

môžme identifikovať 4 užívateľov systém. Public user je verejný úžívateľ, ktorý môže čítať úlohy, ktoré sú nastavené ako verejné(public). Systém môže načítať úlohy z databáze pre potreby výpočtu, z ktorými následne pracuje, a potom výsledky ukladá do databáze, teda edituje úlohy(tasky). Čitateľ(Reader) môže úlohy čítať, publikovať a odpublikovať. Plánovač môže úlohy čítať, vytvárať v databáze, mazať, editovať už vytvorené úlohy v rámci svojej organizácie,publikovať a odpublikovať úlohy a meniť vlastné heslo. Administrátor môže vytvárať úlohy, mazať úlohy , editovať úlohy, publikovať a odpublikovať úlohy. Rovnako si môže meniť heslo, vytvárať,mazať editovať organizácie a užívateľov.

5.4 Návrh rozhrania

Výsledné rozhranie kladie dôraz na jednoduchosť a prehľadnosť zobrazených úloh. Z tôhto dôvodu boli implementované mechanizmy vyhľadávanie úloh, organizácií a užívateľov. Rovnako možnosti lexikografického triedenia. Po prihlásení do systému Jednotlivé môžnosti sú následne zakompotované do záložiek, v ktorých je sprístupné príslušná funkčnosť. Výsledné rozhranie je prenositeľné aj na mobilné zaradenie, čo je spôsobené použitím frameworku Twitter Boostrap.

5.5 Implementácie

Aplikácie bola rozložená do viacerých tried podľa zodpovednosti daných komponent. Jednotlivé

5.6 Testovanie

5.7 Vyhodnotenie aplikácie

Kapitola 6

Záver

Literatura

[1] Ament, J. D.: *Arquillian Testing Guide*. Packt Publishing, 2013, iSBN 978-1782160700.

[2] Arun Gupta: *Java EE 6 Pocet Guide*. O'REILLY, 2012, iSBN 978-1-449-33668-4.

[3] Bali, M.: *Drools JBoss Rules 5.X Developer's Guide*. Packt Publishing, 2013, iSBN 978-1782161264.

[4] Company, S.: *Maven: The Definitive Guide*. O'Reilly Media, 2008, iSBN 978-0596517335.

[5] Davis, T. M. S.: *JBoss at Work: A Practical Guide*. O'Reilly Media, 2006, iSBN 978-596-00734-8.

[6] 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].

[7] Johnson, M. R. G. D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. Macmillan Higher Education, 1979, iSBN 978-0716710455.

[8] Michiels, J. K. E. A. W.: *Theoretical Aspects of Local Search*. Springer, 2007, iSBN 978-3540358534.

[9] Thilo, M. O. J. T. C. R. J.: Twitter Bootstrap [online]. `hhttp://getbootstrap.com/`, 2013-12-16 [cit. 2013-12-27].

[10] Thilo, M. O. J. T. C. R. J.: Twitter Bootstrap [online]. `hhttp://getbootstrap.com/`, 2013-12-16 [cit. 2013-12-27].

[11] WWW stránky: JBoss. `https://docs.jboss.org/author/display/AS71/Documentation`, 2013-12-16 [cit. 2013-12-27].