MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Správa experimentů pro knihovnu MESSIF

BAKALÁŘSKÁ PRÁCE

Zbyněk Nedoma

Brno, jaro 2011

Masarykova univerzita
Fakulta informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student(ka): Zbyněk Nedoma Datum: 3.11.2010

Program: Informatika

Obor: Počítačové systémy a zpracování dat

Vedoucí práce: RNDr. Vlastislav Dohnal, Ph.D.

Katedra (pracoviště): Katedra počítačových systémů a komunikací

Název práce: Správa experimentů pro knihovnu MESSIF

Zadání: Cílem bakalářské práce je navrhnout a implementovat informační systém pro definici a evidenci výsledků experimentů prováděných nad indexovými strukturami pro vyhledávání v multimediálních datech. Systém bude tvořen webovým rozhraním, které umožní definovat experiment, spustit jej a vyhodnotit jeho výsledek. Vyhodnocení výsledků znamená analýzu výstupu, výpočet souhrnných informací ze zachycených statistik a jejich prezentaci v grafické formě. Pro vývoj systému budou použity technologie JavaEE verze 6. Výstupem práce bude technická zpráva a implementovaný webový systém.

Základní literatura: Batko, Michal - Novák, David - Zezula, Pavel. MESSIF: Metric Similarity Search Implementation Framework. In *Digital Libraries: Research and Development*. Berlin, Heidelberg : Springer-Verlag, 2007. od s. 1-10, 10 s. ISBN 978-3-540-77087-9.

Souhlas se zadáním (podpis, datum) 3. 11. 2010

student(ka)	vedoucí bakalářské práce	vedoucí odpovědného pracoviště

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Zbyněk Nedoma

Vedoucí práce: RNDr. Vlastislav Dohnal, Ph.D.

Poděkování

Tímto bych chtěl poděkovat RNDr. Vlastislavu Dohnalovi, Ph.D. za jeho odborné vedení a čas, který mi věnoval.

Shrnutí

Cílem této bakalářské práce je navrhnout a implementovat systém, který eviduje stále rostoucí počet experimentů. Mělo by být možné navrhnout si vlastní experiment a spustit ho. Uživatel by měl být schopen vyhodnocovat statistiky, které experiment zachytával.

Klíčová slova

MESSIF, ActionBean, Java, webový systém, Stripes, experiment, nastavení experimentu, zpracování záznamu běhu, tvorba grafických výstupů

Obsah

1 Úvod

Na Fakultě informatiky byla vytvořena knihovna MESSIF, pomocí které lze vytvářet podobnostní vyhledávání. Tato problematika mě zaujala, proto jsem se rozhodl se jí dále zabývat. Po komunikaci s panem Vlastislavem Dohnalem mi byla nabídnuta možnost vypracovat webový systém, který by pracoval právě s knihovnou MESSIF.

V první kapitole se podíváme na dva způsoby vyhledávání v multimediálních datech. Řekneme si, který z těchto způsobů je v současnosti nejpoužívanější a jaké výhody proti němu má téměř nepoužívané podobnostní vyhledávání. Také si řekneme něco o knihovně MESSIF.

Analýza a návrh systému je tématem druhé kapitoly. Nastíníme si základní funkcionality, které by měl systém splňovat. Ukážeme si na ERD modelu, jak by měla vypadat databáze, nad kterou bude systém pracovat.

Výběr správných technologií, na kterých bude systém postaven a technologií, ve kterých bude realizován, je důležitým krokem při návrhu systému. Jelikož je jejich popis trochu delší, věnoval jsem jim celou třetí kapitolu. Jsou zde popsány jak vývojové prostředí NetBeans, databáze MySQL, tak také webový rámec Stripes a další technologie.

Uživatelský manuál nebo uživatelské rozhraní jsou důležitou součástí každého systému. Začínající uživatelé zpravidla nevědí, jak s ním pracovat. Proto je čtvrtá kapitola práce věnována popisu uživatelského rozhraní.

V závěru práce hodnotím, jestli jsem vyhověl zadání a zmiňuji se o možných rozšířeních pro tento systém.

V práci se předpokládá znalost knihovny MESSIF na úrovni předmětů "PV229 Multimedia Similarity Searching in Practice"a "PA128 Similarity Searching in Multimedia Data".

2 Vyhledávání v multimediálních datech

Počet videí, fotek nebo hudebních souborů na světě roste každý den neuvěřitelnou rychlostí. Nacházet nové možnosti a způsoby jejich prohledávání, v duchu požadavků a potřeb uživatelů, je tedy nanejvíš účelné smysluplné. Současné vyhledávání je z velké části založeno na porovnávání textových řetězců, zadaných uživatelem s popisy těchto dat. To má bohužel značnou nevýhodu: je totiž potřeba se spolehnout na to, že popis, zadaný uživatelem, odpovídá pravdě. Samozřejmě existují vyhledávače jako třeba Google, který u obrázků umožňuje vyhledávat podle barvy nebo Midom [1], který je schopen rozpoznat písničku podle vámi nahrané ukázky. Bohužel jsou zatím pouze na začátku a neposkytují plnou sílu podobnostního vyhledávání.

Podobnostní vyhledávání [2] má oproti současnému výhodu v tom, že se nemusí spoléhat na tyto popisné informace. Pracuje totiž s informacemi, které jsou získány přímo z dat. Celé to ve zkratce funguje tak, že se z obrázku vyextrahují určité vlastnosti, jako jsou třeba barevné spektrum, tvary, rozložení barev a další, ty jsou poté pomocí programu zpracovány a je jim určen bod ve více-dimenzionálním prostoru. Body, které jsou tomu našemu nejblíže, jsou poté považovány jako nejpodobnější.

Na Fakultě informatiky Masarykovy univerzity v Brně probíhá projekt s názvem MUFIN [3] [4] (Multi-Feature Indexing Network) pod vedením profesora Pavla Zezuly. Jeho cílem je poskytnout nové možnosti ve vyhledávání v multimediálních datech. Pro širokou veřejnost je přístupný zkušební vyhledávač, který pracuje s více jak 100 miliony obrázků převzatých z databáze Flickr [5]. Umožňuje jednak vyhledávání založené na textovém popisu, ale i na podobnosti.

Tento způsob vyhledávání se ale nemusí zastavit jen u obrázků, hudby a videa. Velkou budoucnost má i v dalších oblastech, jako jsou například medicína, kde lze tento systém použít pro hledání DNA sekvencí nebo v kriminalistice pro porovnávání otisků prstů.

2.1 Knihovna MESSIF

The Metric Similarity Search Implementation Framework zkráceně MESSIF [6] je knihovna naprogramovaná v jazyce Java. Snaží se poskytnout základní podporu a funkcionalitu pro podobnostní vyhledávání založené na metrickém prostoru. MESSIF obsahuje implementaci dvou základních modelů, těmi jsou vektor a řetězec. Architektura je zcela otevřená a nové moduly mohou být kdykoliv přímo integrovány do systému.

3 Analýza a návrh systému

Systém, který byl vytvořen na základě zadání této bakalářské práce, je nazvaný MESSIF Experiment Administration (zkráceně MEA). V této podkapitole si rozebereme jednotlivé požadavky na systém a navrhneme si jejich základní řešení. Dále zde bude uveden ERD diagram 1, který je důležitým prvkem při návrhu systému.

Umožnění definovat vlastní experiment 3.1.1, spustit ho, vyhodnotit jeho výsledek, provést výpočet souhrnných informací ze zachycených statistik 3.2 a prezentovat je v grafické formě, je seznam požadavků, které plynou ze zadání bakalářské práce. Můžeme je rozdělit do tří skupin.

- Vytvoření experimentu,
- Zpracování statistik,
- Generování grafů.

3.1 Vytvoření experimentu

3.1.1 Experiment

Každý experiment je v systému reprezentován jako speciální dávkový soubor 1 s koncovkou ".cf". Obsahem souboru jsou pravidla určující co, jak a kdy se má provést. Tento soubor se poté předá serveru a on ho vyhodnotí. Velkou výhodou tohoto způsobu vyhodnocování experimentů je, že k jeho vytvoření a spuštění nepotřebujeme žádné speciální nástroje. Pro vytvoření lze použít obyčejný Poznámkový blok. Aby mohl být experiment spuštěn je zapotřebí mít nainstalovanou Javu a samozřejmě staženou knihovnu MESSIF.

U každého experimentu je nutné definovat algoritmus, kterým budou data prohledána. Máme možnost si vybrat některý z již připravených, kterými jsou ParallelSequentialScan a SequentialScan. Samozřejmě existuje možnost napsat si vlastní algoritmus. Dále je nutné specifikovat soubor, obsahující vstupní data pro experiment a typ objektu, které tento soubor obsahuje. Nakonec je ještě dobré uvést operaci, která bude použita při nacházení nejbližších objektů. Opět máme na výběr z připravených operací, jako KNNQueryOperation, RangeQueryOperation a dalších. Samozřejmě, že si můžeme napsat i vlastní, stejně jako u algoritmů. U jednotlivých algoritmů a operací lze definovat statistiky, které budou zachytávány při vyhodnocování experimentu.

Experimenty nemusíme používat pouze k vyhledávání dat. Lze s nimi provádět i vkládání a mazání dat.

```
1.  actions = alg stream insert algorithmInfo k
2.
3.  alg = algorithmStart
4.  alg.param.1 = messif.algorithms.impl.SequentialScan
5.
6.  stream = objectStreamOpen
7.  stream.param.1 = data.txt
8.  stream.param.2 = messif.objects.impl.ObjectStringEditDist
9.  stream.param.3 = data
10.
11. insert = operationExecute
12. insert.param.1 = messif.operations.data.InsertOperation
13. insert.param.2 = data
14.
15. queryStream = objectStreamOpen
16. queryStream.param.1 = <queryfile:query.txt>
17. queryStream.param.2 = messif.objects.impl.ObjectStringEditDist
18. queryStream.param.3 = query
19.
20. k = queryStream query operationInfo operationAnswer
21. k.foreach = 2 4 6
22.
23. query = operationExecute
24. query.param.1 = messif.operations.query.KNNQueryOperation
25. query.param.2 = query
26. query.param.3 = <k>
27. query.param.4 = ORIGINAL_OBJECTS
```

Zdrojový kód 1: Ukázka dávkového souboru

Celý tento dávkový soubor se předá serveru a ten provede postupně jednotlivá pravidla tak, jak jsou napsána v prvním pravidle *actions*.

Dávkový soubor může obsahovat mnoho různých pravidel. My si však zde popíšeme jen některá z nich. Jména *alg*, *stream* a další jsou pouze ilustrativní, mohou být pojmenována podle potřeby.

- **action** - Tímto pravidlem musí každý dávkový soubor začínat. Před ním se mohou vyskytovat pouze komentáře,
- **alg** - Definuje algoritmus použitý v experimentu,
- **stream** - Definuje soubor, ze kterého budou načtena vstupní data. Také určuje, jaké objekty jsou v tomto souboru obsaženy,
- **query** - Definuje operaci, která bude použita při hledání nejbližších objektů.

3.1.2 Šablony pro experimenty

Aby byla ulehčena práce uživatelů, rozhodli jsme se do systému implementovat šablony 2. Ty by měly nahradit často se opakující kód v jednotlivých dávkových souborech. V podstatě by to mělo vypadat tak, že si uživatel navolí nějaké parametry a ty se mu nahrají do již připravené šablony. Jelikož si uživatel bude moct vytvořit svou vlastní šablonu, bylo by dobré stanovit nějaká pravidla pro jejich vytváření. O těchto pravidlech si povíme v kapitole 5.1.

```
1.  actions = alg streams inf knn
2.  actions.postponeUntil = <time>
3.
4.  <algorithm>
5.
6.  streams = query_objs
7.  query_objs = objectStreamOpen
8.  query_objs.param.1 = <ds:filename>
9.  query_objs.param.2 = <ds:class>
10.
11. query_objs_reset = objectStreamReset
12. query_objs_reset.param.1 = query_objs
```

Zdrojový kód 2: Ukázka části šablony

Bohužel při studiu bakalářské práce Repositář vyhledávacích struktur pana Bc. Petra Eliáše [7], na kterou má práce navazuje jsem narazil na problém: neřeší v ní totiž možnost existence více algoritmů se stejným jménem, ale různým konstruktorem. Proto bude nutné jeho tabulku upravit a přidat do ní nové sloupce. Jeden sloupec by měl ukládat právě zmiňovaný konstruktor a v dalším by měla být uložena pravidla pro dávkový soubor.

3.2 Zpracování statistik

Jak již bylo výše zmíněno, výstupní soubory z experimentu mohou obsahovat statistiky, které se uživatel rozhodl zachytávat. Proto by bylo vhodné umožnit v systému tento výstup nějakým způsobem zpracovat a vypsat.

Statistiky jsou informace, které je možno zachytávat v průběhu vyhodnocování experimentu. Mohou být různého typu, jako třeba čítače nebo měřiče času, za který proběhla určitá operace. Každá statistika je definovaná svým jménem a obsahuje nějakou zachycenou hodnotu nebo množinu hodnot. Tyto statistiky mohou být zachytávány globálně, tedy během celého běhu experimentu, nebo lokálně v průběhu určité operace. Knihovna MESSIF dovoluje zachytávat dva druhy statistik:

- **Základní statistiky** - Jejich výpis je ve tvaru **jménoStatistiky.podjméno: hodnota**. Mezi tyto statistiky patří například DistanceComputations nebo TimeOperation,
- **Hash statistiky** - Jejich výpis je trochu složitější než u normálních statistik. Vypadá takto **jménoStatistiky: {klíč=hodnota, ...}**, kde klíč je v našem příkladě 3 identifikace instance třídy LocalBucket ve tvaru **jméno:hashHodnota**. Parametr LocalBucket se může v každé množině objevit různě-krát. Jméno LocalBucket se také může změnit

- záleží na definici. Mezi tyto statistiky můžeme zařadit třeba BucketDistancecomputations nebo BucketRead.

```
1.  BucketDistanceComputations: {LocalBucket:0134=7,
     LocalBucket:0056=139,  LocalBucket:0101=19, ...}
2.  BucketRead: {LocalBucket:0134=1, LocalBucket:0070=1,
     LocalBucket:0155=1, LocalBucket:0131=1, ...}
3.  BucketRead.Optimistic: {LocalBucket:0162=1,
     LocalBucket:0165=1, LocalBucket:0119=1, ...}
4.  DistanceComputations: 2482
5.  DistanceComputations.Address: 38
6.  DistanceComputations.LowerBound: 0
7.  DistanceComputations.PivotChooser: 0
8.  DistanceComputations.Savings: 93170
9.  DistanceComputations.Split: 0
10. DistanceComputations.UpperBound: 0
```

Zdrojový kód 3: Ukázka části obsahu výstupního souboru experimentu

Zpracováním se rozumí aplikování základních agregačních funkcí, jako jsou suma, maximum, minimum, počet a průměr. Jako specialita bude do systému přidána možnost rozdělit statistiky na skupiny určité velikosti.

Vezměme si třeba ukázku výstupního souboru 3. Z ní v systému vytáhneme statistiku se jménem *DistanceComputations* a aplikujeme na ni funkce suma a maximum. Výstupem pak bude tabulka se sloupci sum s hodnotou 95690 a max s hodnotou 93170.

3.3 Generování grafů

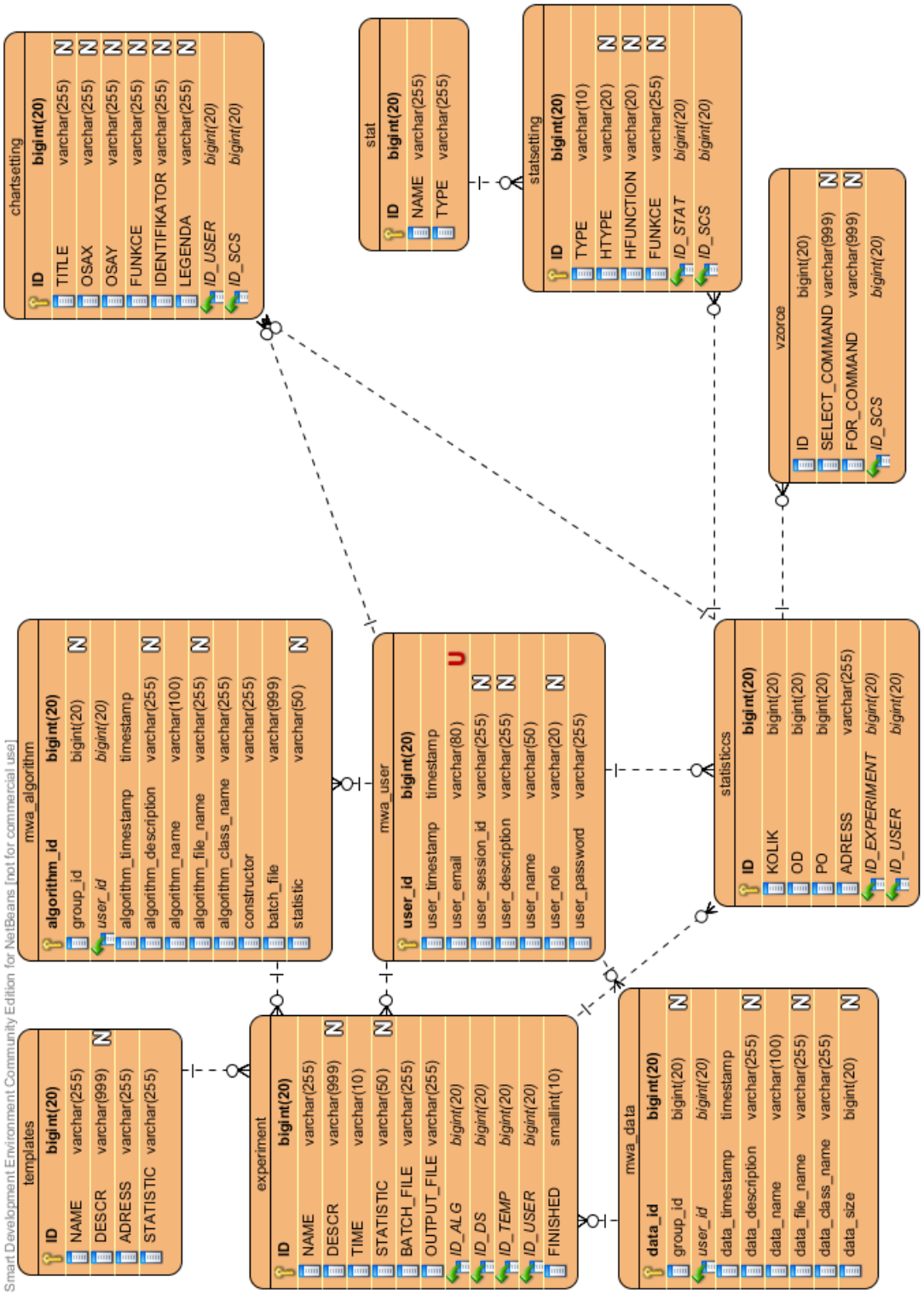
Tabulkové zobrazení výsledků může být pro někoho nepřehledné a také některé výsledky se lépe vyhodnocují, když jsou v podobě grafu. Z těchto důvodů budou do systému implementovány grafy. Pro práci s grafy bude použita knihovna JFreeChart. O ní se zmíním v kapitole 4.6. U každého z grafů by měl mít uživatel možnost definovat si jeho název a popisy jednotlivých os. Dále by pak měla být možnost přidat a odebrat jednotlivé křivky a nastavit jim určitý popisek.

3.4 Modely a diagramy

K vytvoření modelu a diagramu byl použit nástroj Visual Paradigma SDE for NetBeans [8] verze 6.1. Tento nástroj dovoluje vytvoření modelu přímo v prostředí NetBeans.

3.4.1 Entity Relationship Model

Entity Relationship Diagram (zkráceně ERD) [9] je model znázorňující strukturu dat databáze. Můžeme na něm vidět obsah jednotlivých tabulek a vztahy mezi nimi. ERD systému MEA 1 obsahuje celkem pět tabulek: Experiment, Stat, Statsetting, Chartsetting a Templates. Zbylé tři tabulky jsou převzaty z bakalářské práce Repositář vyhledávacích struktur [7]. Na tabulce *mwa_algorithm* jsou vidět změny, které jsou uvedeny v odstavci 3.1.2.



Obrázek 1: Entity Relationship Diagram

4 Použité nástroje a technologie

Celý webový systém byl vytvořen v jazyce Java pomocí programovacího nástroje NetBeans. Je postaven na databázi MySQL a aplikačním serveru Tomcat. O těchto nástrojích a dalších technologiích, které byly použity při vývoji tohoto systému si povíme v kapitole 4.

Důvodů pro výběr těchto nástrojů a technologií bylo více. Databázi MySQL jsem si zvolil hlavně kvůli návaznosti na bakalářskou práci Repo. Práce s knihovnou JFreeChart byla předepsána zadáním. Zbylé technologie se mi zdály vhodné pro použití v tomto systému.

4.1 NetBeans

Netbeans IDE (Integrated Development Environment) [10] je vývojové prostředí naprogramované v jazyce Java. Bylo vyvinuto v roce 1996 v České republice, tehdy ještě jako studentský projekt, pod názvem Xelfi [11]. Umožňuje programátorům psát, překládat, ladit a distribuovat aplikace. Jako základní programovací jazyk se považuje Java, ale podporuje prakticky jakýkoliv jiný programovací jazyk. Velikou výhodou je existence mnoha dodatečných modulů, pomocí kterých lze základní instalaci lehce rozšířit. NetBeans jsou volně šiřitelné, tudíž jejich používání není nijak omezeno. Lze ho použít na operačních systémech Windows, Linux, Mac OS X a Solaris. Verze NetBeans 6.9.1. přináší vylepšený Java editor a mnoho dalšího.

4.2 Databáze MySQL

MySQL [12] je světově nejoblíbenější a nejpoužívanější multiplatformní relační databázový systém typu DBMS (Systém řízení databáze). Důvodem oblíbenosti jsou vysoký výkon, vysoká spolehlivost a snadné použití. Je šířena s otevřeným kódem pod licencí GPL (Všeobecná veřejná licence). Komunikace s databází probíhá pomocí dotazovacího jazyka SQL (Strukturovaný dotazovací jazyk). Pro vzdálenou správu přes webové rozhraní se nejčastěji používá phpMyAdmin. Jedná se o lehce ovladatelnou webovou aplikaci naprogramovanou v PHP.

4.3 Aplikační server Apache Tomcat

Apache Tomcat [13] verze 6 je významný aplikační server. Je vyvíjen jako volně šiřitelný projekt založený na jazyce Java, Java Servletech a JSP (Java Server Pages) [14] a EJB (Enterprise JavaBeans) [15]. Oproti komerčním produktům, jako jsou "WebSphere Application Server"od IBM nebo "WebLogic Application Server"od Oracle, Tomcat vyniká jednoduchostí, transparentností a menší náročností na výpočetní výkon.

4.4 Stripes

Stripes [16] [17] je webový aplikační framework s otevřeným kódem, založený na modelu MVC (Model View Controller). Jeho cílem je být jednodušší než rámec Struts za použití Java technologií. Eliminuje velkou část konfigurace, která je tradičně spojována s rozvojem těchto webových aplikací.

4.4.1 ActionBeans

ActionBean je třída starající se o zobrazení a změnu dat vždy, když uživatel klikne na některý z odkazů. V podstatě se jedná o Java třídu implementující ActionBean interface 4, který vypadá takto:

```
1. interface ActionBean {
2.  public void setContext(ActionBeanContext context);
3.  public ActionBeanContext getContext();
4. }
```

Zdrojový kód 4: Ukázka interface ActionBean

Jak vidíme, interface ActionBean obsahuje pouze get a set metody pro ActionBeanContext, který obsahuje aktuální request a response objekty a další důležité informace o aktuálním požadavku.

```
1.  public class UserActionBean implements ActionBean {
2.
3.     private ActionBeanContext context;
4.     private String name;
5.
6.     /*
7.      * Následují get a set metody pro atributy name
8.      * a context
9.      */
10.
11.    public Resolution show() {
12.       return new ForwardResolution("/show.jsp");
13.    }
14. }
```

Zdrojový kód 5: Ukázka jednoduchého ActionBean

Tato ukázka 5 představuje jednoduchou implementaci ActionBean. Obsahuje základní get a set metody pro ActionBeanContext, které jsou předepsány interfacem. Dále obsahuje parametr *name* a jeho get a set metody. Nakonec je zde implementovaná metoda *show()* typu Resolution, která se stará o přesměrování na stránku show.jsp. Tato metoda se spustí, když někdo klikne na tlačítko, jehož parametr *name = "show"*.

Další kód, který je zde uveden, ukazuje způsob, jak se v jsp stránce může nastavit parametr *name* našeho, před chvilkou vytvořeného, UserActionBean. Při vytváření je nutné dodržet pár základních pravidel.

1. Pokud chceme, aby bylo něco zpracováno naším ActionBean, je nutné to uzavřít do parametru *s:form*,

2. Atribut *name* se vždy musí shodovat s jmény parametrů v našem ActionBean

Poté při kliknutí na tlačítko Show bude zavolána metoda *show()* z našeho UserActionBean a do parametru *name* se načte hodnota textového pole. Výhodou je, že nemusíme volat metodu *setName(String name)*, Stripes si ji totiž zavolá sám.

```
1. <s:form beanclass="UserActionBean">
2.     <s:text name="name"/>
3.     <s:submit name="show">Show</s:submit>
4. </s:form>
```

Zdrojový kód 6: Ukázka kódu pro nastavení parametru name

Pokud chceme parametr *name* zobrazit, stačí, když načteme náš ActionBean od parametru *actionBean* a poté na něj zavoláme *actionBean.name*. Stripes už si sám zavolá metodu *getName()* a zobrazí jeho hodnotu.

```
1. <s:useActionBean beanclass="UserActionBean" var="actionBean"/>
2. <c:out value="${actionBean.name}"/>
```

Zdrojový kód 7: Ukázka výpisu parametru name na stránce

4.5 Java Persistence API

Jedná se o framework [18] programovacího jazyka Java, který poskytuje objektově relační mapování, usnadňující vývojářům ukládání dat do databáze a jejich načtení. Využívá se v aplikacích používající Java SE nebo Java EE.

4.5.1 Entita

Entita 8 je objekt reprezentující data v databázi. Entita typicky reprezentuje tabulku v relační databázi; každá instance této entity koresponduje s jedním řádkem v této tabulce. Primární programovací artefakt entity je entitní třída, ačkoliv entity mohou využívat i pomocné třídy. Aby mohla být entitní třída persistována, musí splňovat tyto požadavky [19]:

- Třída musí být anotována pomocí javax.persistence.Entity anotace,
- Třída musí obsahovat public nebo protected bezparametrický konstruktor; může obsahovat i další konstruktor,
- Třída nesmí být deklarovaná jako final; totéž platí pro všechny její metody,
- Jestliže bude něco pracovat s instancí této třídy, například session bean, musí v tom případě entitní entitní třída implementovat interface Serializable,
- Entity mohou rozšiřovat jiné entitní a ne-entitní třídy, zato ne-entitní třídy mohou rozšiřovat pouze entitní třídy,
- Atributy musí být deklarovány jako private, protected nebo package-protected a lze k nim přistupovat pouze pomocí metod entitní třídy.

```
1.  @Entity
2.  @Table(name = "mwa_algorithm")
3.  @NamedQueries({
4.  @NamedQuery(name = "User.findByUserId",
5.              query = "SELECT u FROM User u
6.                       WHERE u.userId = :userId")
7.  public class User implements Serializable {
8.      private static final long serialVersionUID = 1L;
9.     @Id
10.     @GeneratedValue(strategy = GenerationType.AUTO)
11.     private Long id;
12.     private String name;
13.
14.     /*
15.      * Následují GET A SET metody pro jednotlivé parametry
16.      * a metody hashCode, equals a toString
17.      */
```

Zdrojový kód 8: Ukázka entitní třídy

4.5.2 Java Persistence Query Language

Velikou výhodou Java persistence je její nezávislost na typu databáze. Pouhou změnou konfiguračních dat přeneseme náš projekt třeba z databáze MySQL na PostgreSQL. Tato přenositelnost je možná díky jazyku Java Persistence Query Language (zkráceně JPQL). Tento jazyk je podobný jazyku SQL, i když má svá specifika. Hlavním rozdílem mezi těmito jazyky je, že SQL vykonává dotazy přímo nad tabulkami databáze, zatímco JPQL pracuje s objekty aplikace, entitami 4.5.1.

JPQL lze použít dvěma způsoby. Jednak můžeme napsat příkaz přímo ve zdrojovém kódu aplikace, čehož se využívá, že se příkaz nikde v kódu neopakuje.

```
1. Query q = em.createQuery("SELECT u FROM User u
2.                           WHERE u.id = :userId");
3. q.setParameter("userId", 1);
4. User user = (User) q.getSingleResult();
```

Zdrojový kód 9: Ukázka nepojmenovaného dotazu

Tento kód nám z tabulky User vytáhne uživatele, jehož id = 1. Můžeme si zde povšimnout i další odlišnosti od SQL: chceme-li vytáhnout všechny záznamy z databáze, nepoužijeme *SELECT* * jako v SQL, ale *SELECT x*, kde *x* je naše přejmenovaná tabulka. To v SQL nenajdeme.

Druhý způsob použití JPQL je pomocí anotace @NamedQuery, kterou přidáme do Entity. Touto anotací se vytvoří pojmenovaný příkaz přímo v entitě. V kódu pak používáme jen jméno tohoto příkazu. Je to určité zkrácení kódu a také máme všechny příkazy pohromadě.

Tento kód provede totéž jako příkaz předchozí, ovšem s tím rozdílem, že každá změna, provedená v pojmenovaném příkazu, se nám automaticky projeví na všechna jeho použití.

```
1. Query q = em.createNamedQuery("User.findByUserId");
2. q.setParameter("userId", 1);
3. User user = (User) q.getSingleResult();
```

Zdrojový kód 10: Ukázka pojmenovaného dotazu

4.6 JFreeChart

JFreeChart [20] je pomocný nástroj pro vykreslování různých typů grafů 2 3. Byl vytvořen v únoru roku 2000, mužem jménem David Gilbert a je distribuován jako open source pod licencí LGPL [21] (GNU Lesser General Public Licence).

```
1. final JFreeChart chart = ChartFactory.createXYLineChart(
2.                       "Titulek",
3.                       "Popis osy X",
4.                       "Popis osy Y",
5.                       dataset,
6.                       PlotOrientation.VERTICAL,
7.                       true, false, false);
```

Zdrojový kód 11: Ukázka části nastavení grafu

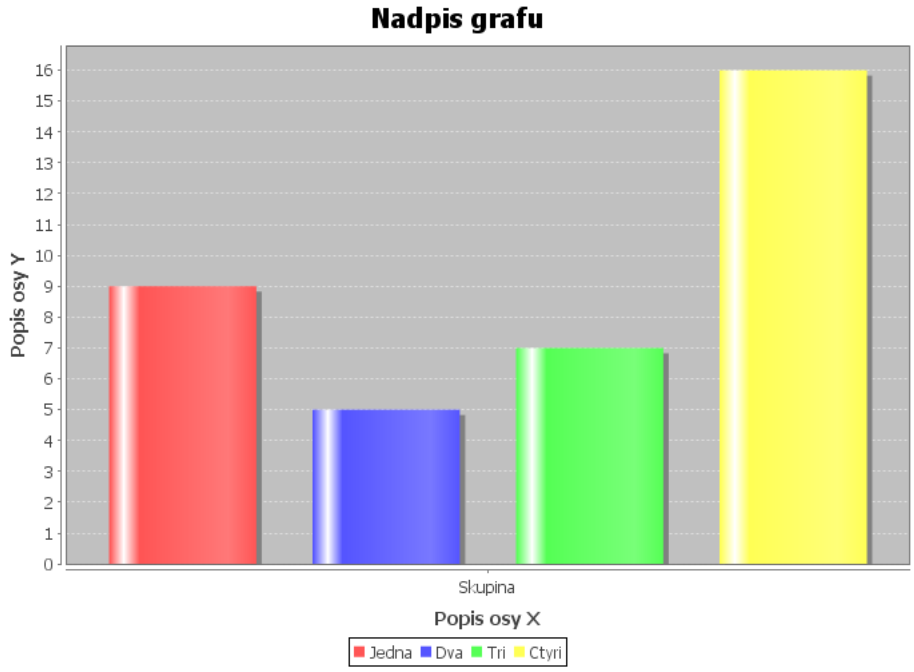
Jak je vidět na zdrojovém kódů 11 vytvoření grafu pomocí JFreeChart je naprosto jednoduché. Podobnými konstruktory lze vytvořit různé typy grafů. Stačí mu nastavit jen potřebný zdroj dat.

- 1. parametr - Titulek grafu,
- 2. parametr - Popis osy X,
- 3. parametr - Popis osy Y,
- 4. parametr - Zdroj dat s informacemi o tom, co vykreslit,
- 5. parametr - Nastavení, jestli se má graf vykreslit na výšku nebo na šířku,
- 6.-8. parametr - Nastavení, jestli chceme zobrazit legendu, tool tip a odkazy,

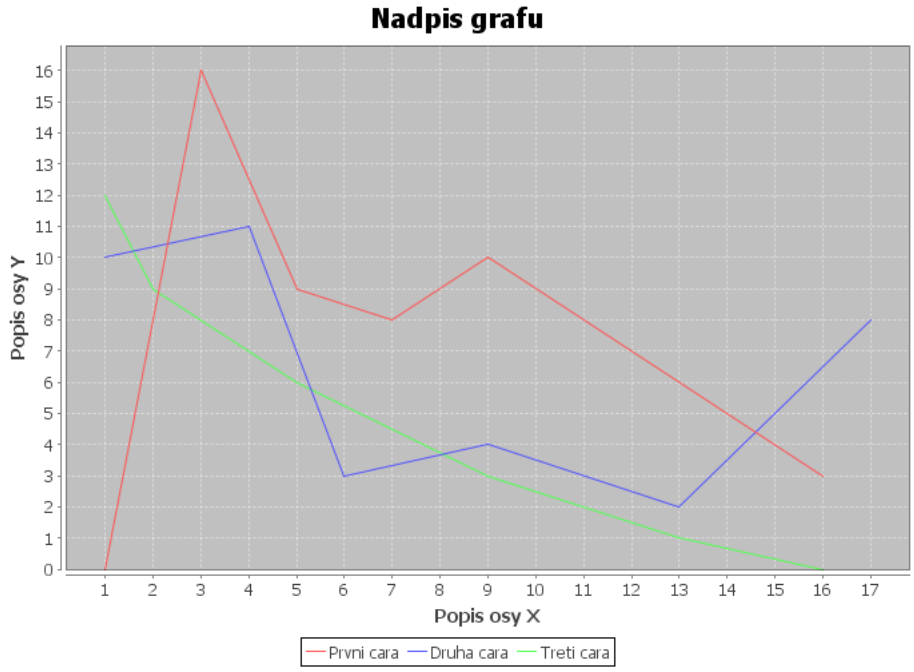
```
1. final XYSeries series = new XYSeries("Cara");
2.              series.add(13.0, 1.0);
3.              series.add(9.0, 3.0);
4.              series.add(16.0, 0.0);
5.
6. final XYSeriesCollection dataset = new XYSeriesCollection();
7.              dataset.addSeries(series);
```

Zdrojový kód 12: Ukázka jednoduchého zdroje dat

Zdrojový kód 12 je ukázka zdroje dat pro vytvoření spojnicového grafu s jednou křivkou a třemi vrcholy. Výsledný graf je poté možno vygenerovat třeba jako JPEG, PNG a SVG obrázek, nebo rovnou do PDF souboru [22].



Obrázek 2: Ukázka sloupcového grafu

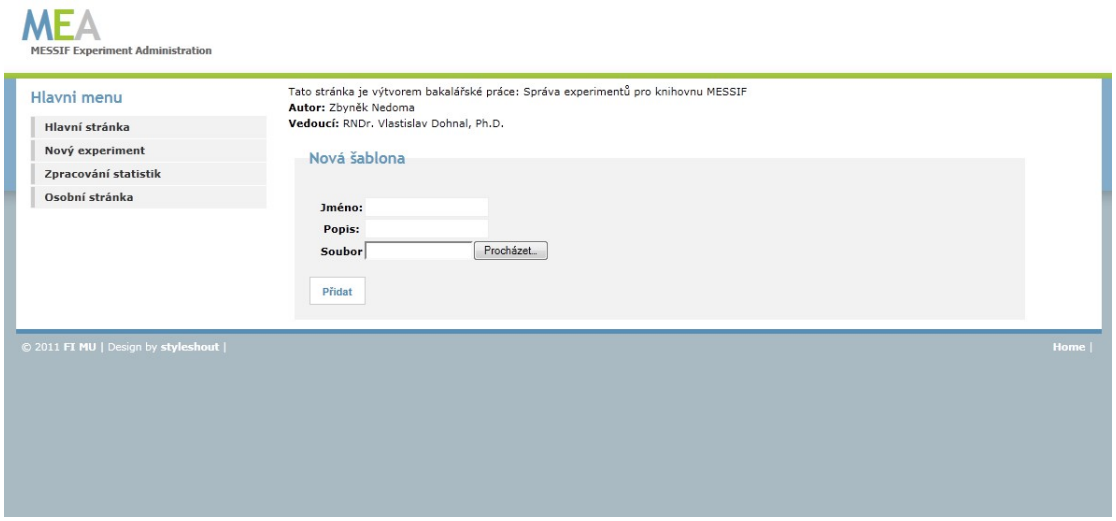


Obrázek 3: Ukázka spojnicového grafu

5 Uživatelské rozhraní

Každý určitě zná ten okamžik, když začíná pracovat s novým systémem. Bez konkrétní zkušenosti nevíme, co všechno v něm můžeme dělat nebo jak se k některým věcem dostaneme. Proto jsem se rozhodl, že tato kapitola bude věnována uživatelské příručce. Pokusím se na jednoduchých příkladech ukázat hlavní funkce systému a samozřejmě popsat i pravidla, která je nutné dodržovat. Systém obsahuje tři hlavní části:

- Vytvoření experimentu,
- Zpracování statistik,
- Generování grafů.



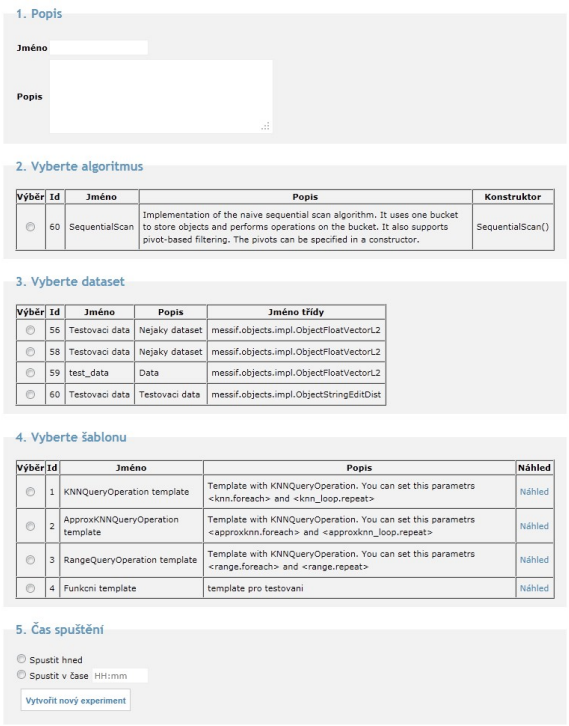
Obrázek 4: Uvodní stránka

5.1 Vytvoření experimentu

Jak už jsme si řekli v kapitole 3.1.1, experimenty jsou v systému reprezentovány jako dávkové soubory s příponou .cf. Aby byla ulehčena práce uživatelů, jsou zde vytvořeny šablony, zastupující statické části experimentu, do nichž si můžeme navolit algoritmus a datovou kolekci, které chceme použít, popřípadě i čas spuštění, pokud nechceme, aby byl experiment proveden okamžitě. V systému je pro začátek vytvořeno pár základních šablon. Pokud ovšem uživatel potřebuje nějakou vlastní, může ji do systému nahrát přes hlavní stránku. Musí přitom však dodržet několik zásadních pravidel.

1. Název šablony je povinný parametr, popis však uvádět nemusíme.
2. Soubor obsahující šablonu musí obsahovat povinné parametry.
   - **<algorithm>** - Za tento parametr bude nahrán námi zvolený algoritmus,
   - **<ds:filename>** - Jméno souboru se vstupnímy daty,
   - **<ds:class>** - Udává typ objektů, které se nachází v dtto,
   - **<algStat>** - Zde budou nahrány uživatelem zvolené statistiky, zachytávané v průběhu algoritmu,
   - **<opStat>** - Zde budou nahrány uživatelem zvolené statistiky pro operaci.

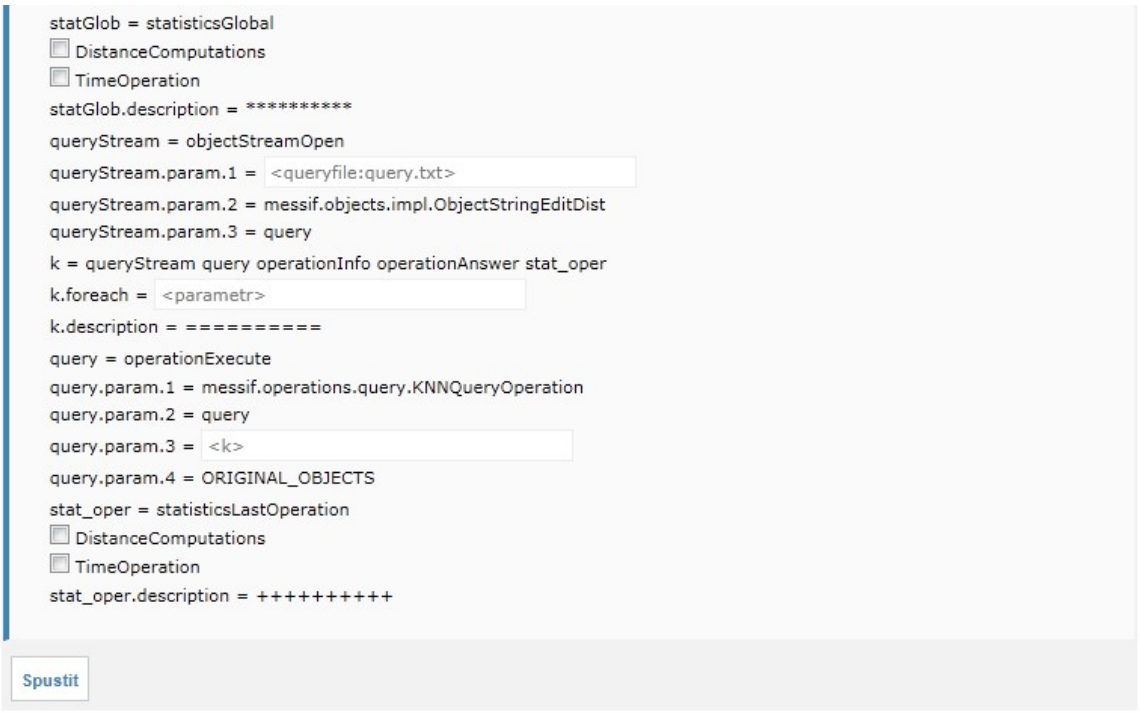
Příklad: Vytvoříme experiment s názvem Můj první Experiment a jeho popis necháme prázdný. Použijeme algoritmus SequentialScan s prázdným konstruktorem a jako zdroj dat vybereme Testovacích data. Základ našeho experimentu bude tvořit KNNQueryOperation template a čas spuštění nastavíme na 16:30.
Řešení: *Klikneme na záložku Nový experiment. Do kolonky název napíšeme Můj první experiment a popis necháme prázdný. Dále zaklikneme u algoritmu SequentialScan s prázdným konstruktorem, u zdroje dat Testovací data a jako template zvolíme KNNQueryOperation template. Jako poslední zaklikneme Spustit v a do vedlejší kolonky napíšeme 16:30. Toto celé odešleme pomocí tlačítka Vytvořit experiment*.



Obrázek 5: Navolení parametrů pro nový experiment

V předchozím příkladu jsme si vytvořili strukturu experimentu. Nyní mu ještě musíme nastavit statistiky, které má zachytávat a pokud je potřeba vyplníme potřebné parametry. Pokud tak učiníme a klikneme na tlačítko **Vytvořit nový dávkový soubor** je experiment odeslán na server s příkazem **java -cp MESSIF.jar messif.utility.Application batchFile.cf**.

Ve stejný okamžik jsme přesměrováni na stránku **Moje Experimenty**, kde můžeme experiment dále zpracovávat.

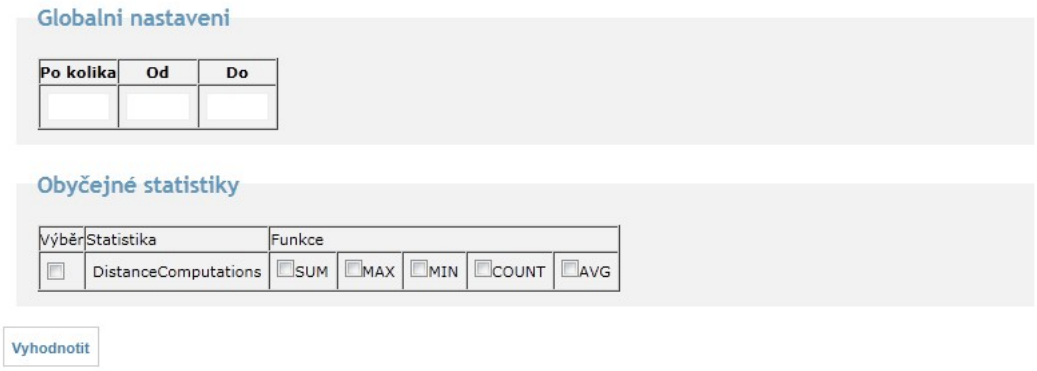


Obrázek 6: Kontrola batch souboru před vyhodnocením

5.2 Zpracování statistik

Pro zpracování statistik můžeme použít buď výstupní soubor námi vytvořeného experimentu, nebo nahrajeme do systému vlastní soubor. V obou případech se dostaneme na stránku, kde si můžeme zvolit, jaké statistiky chceme zpracovat. Vždy nám budou nabídnuty pouze ty statistiky, které náš soubor obsahuje.

Příklad: V předchozím příkladu jsme si vytvořili experiment. Dejme tomu, že jsme mu při kontrole nastavili, ať zachytává statistiky DistanceComputations. Nyní se je pokusíme zpracovat. Nastavte statistiku DistanceComputations tak, aby se její hodnoty rozdělily na skupiny po deseti. Poté z těchto skupin vyberte hodnoty 2-7 a na nich aplikujte funkce SUM, MAX a AVG.
Řešení: *Přejdeme do části Moje experimenty a u našeho experimentu klikneme na Zpracovat. Dostaneme se na stránku, kde uvidíme všechny statistiky, které náš soubor obsahuje. My zatrhneme DistanceComputation a dáme zpracovat. Poté jsme přesměrování na stránku Podrobnější nastavení. Do globálního nastavení napíšeme KOLIK=10, OD=2, PO=7 a u statistiky DistanceComputation zaškrtneme SUM, MAX a AVG.*

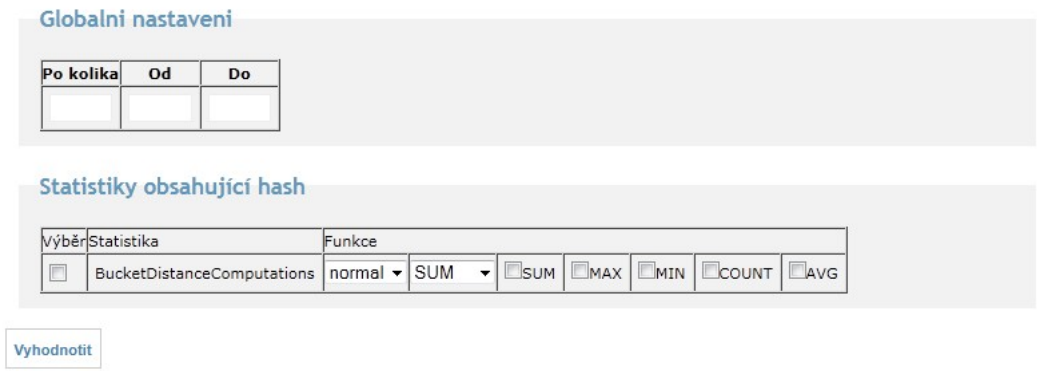


Obrázek 7: Podrobnější nastavení normálních statistik

Jak můžeme vidět na obrázku 8, hash statistiky mají o něco více nastavení než statistiky normální. Je to způsobeno tím, že hash statistiky jsou složeny z množin 3.2 a proto se nejdříve musí provézt jejich agregace. Poté už můžeme aplikovat stejný postup jako u normálních statistik.

Na začátku si musíme vybrat, na čem chceme agregační funkci aplikovat. Možnosti jsou **Množina** a **Hash**. Při výběru **Množina** se funkce aplikuje na každou jednotlivou množinu hash hodnot zvlášť. U možnosti **Hash** se množiny přeskupí podle hodnot jejich hashe a pak se na ně aplikuje agregační funkce.

Příklad: Statistiku BucketDistanceComputation nastavte tak, aby byla agregována na hash hodnotách funkcí maximum. Hodnoty rozdělte na skupiny po pěti a funkce MIN a COUNT nechte aplikovat na všech hodnotách každé skupiny.
Řešení: *Na začátku postupujeme stejně jako v příkladu na zpracování normálních statistik. Pouze místo DistanceComputation zvolíme BucketDistanceComputation. Poté do globálního nastavení napíšeme KOLIK=5, OD=1, DO=5. A nakonec u statistiky BucketDistanceComputation vybereme agregovat přes hash a jako funkci MAX. Jako další funkce zaškrtneme MIN a COUNT.*



Obrázek 8: Podrobnější nastavení hash statistik

Funkce jako Suma, Maximum, Minimum, Count a Average by byly samozřejmě trochu málo pro zpracování statistiky. Proto byly do systému implementovány vzorce, pomocí kterých můžeme provádět základní matematické operace mezi sloupci jednotlivých statistik.

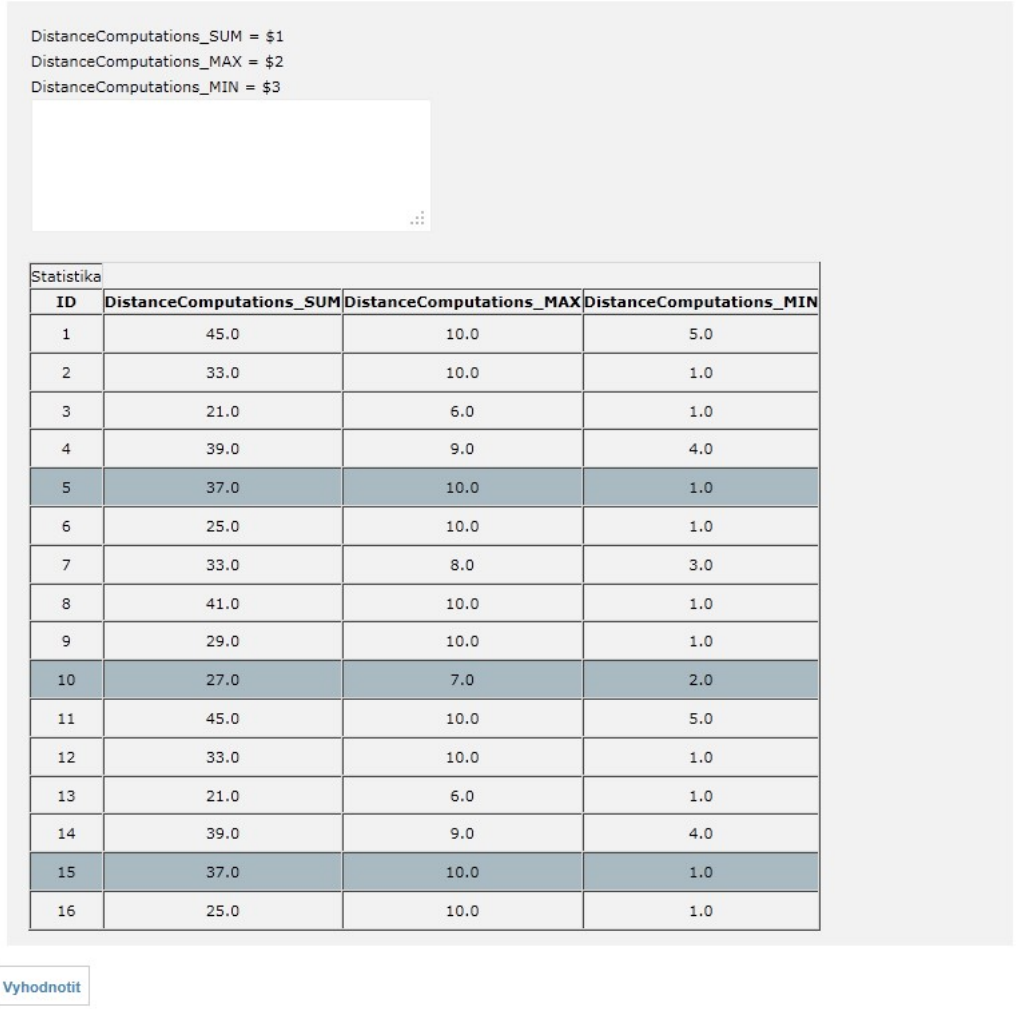
Tvar vzorce je: **názevNovéhoSloupce = vzorec**, kde vzorec může obsahovat obyčejné závorky, základní matematické operace +, -, *, / a samozřejmě jakékoliv reálné číslo. Sloupce jsou ve vzorci zastoupeny parametry $Z, kde Z je celé číslo od 1.

Pokud budeme chtít aplikovat na tabulku více než jeden vzorec, je nutné je oddělit pomocí středníku.

Příklady vzorců:

součin = ($1 * 100) – Vytvoří se nový sloupec, který bude obsahovat součin prvního sloupce s číslem 100.

součet = ($1 + $3); rozdíl = (58 - $3) – Vytvoří se dva nové sloupce. První bude obsahovat součet prvního a druhého sloupce. Druhý zase rozdíl čísla 58 a hodnoty třetího sloupce.

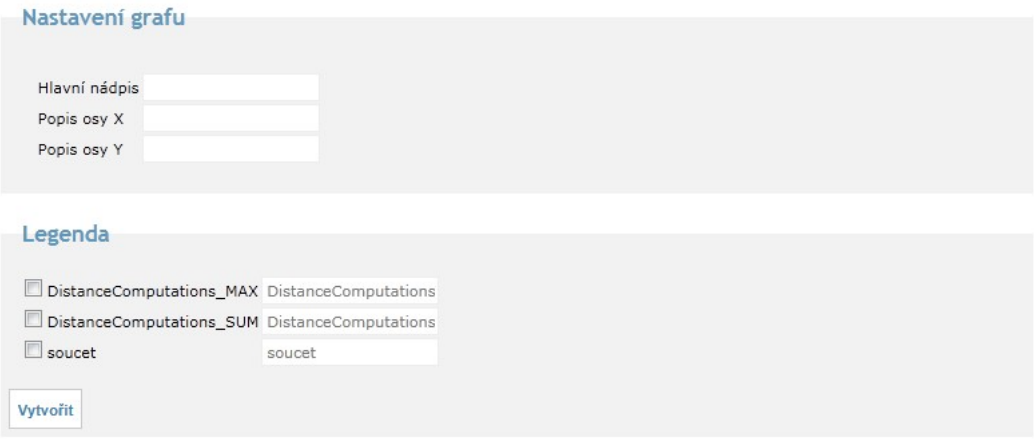


Obrázek 9: Stránka pro psaní vzorců

5.3 Generování grafů

Aby bylo možné generovat graf, je zapotřebí mít nejdříve zpracované nějaké statistiky. Pokud žádné statistiky zpracované nemáte, postupujte, prosím, podle návodu 5.2. V opačném případě přejděte na stránku **Moje experimenty**. Rozklikněte některý z vašich experimentů a u statistiky, kterou chcete zpracovat, klikněte na **Generovat graf**.

Jsme přesměrováni na stránku, kde si můžeme nastavit nadpis grafu, popisy jednotlivých os a vybrat si, jaké funkce chceme vykreslit. K jednotlivým funkcím jde také napsat nová legenda. Pokud některou z hodnot nevyplníme, nastaví se jim implicitní hodnoty.



Obrázek 10: Nastavení parametrů pro graf

Obrázek 11: Ukázka výsledného grafu s jednou křivkou

6 Nasazení systému

Před samotným nasazením systému na server je důležité zkontrolovat, zda je na stroji nainstalován databázový server MySQL (nejlépe verze 5 nebo vyšší) a aplikační server Tomcat verze 6. Bude nutné získat adresu připojení k databázi, její port a přihlašovací údaje. Poté vytvoříme databázi se jménem Experiments a vytvoříme v ní všechny naše tabulky. Jelikož jsme upravili tabulku mwa_algorithm v bakalářské práci Repositář vyhledávacích struktur [7], budeme ji muset v systému znovu vytvořit.

6.1 Konfigurace

Konfigurace je velice jednoduchá. Stačí jen nastavit údaje o připojení k databázi, adresy odkud se mají získávat a kam ukládat soubory.

- persistence.xml - tento soubor se nachází ve složce mea/scr/conf a nastavuje připojení k databázi. Nastavíme zde přihlašovací údaje (user a password) a URL adresu připojení (url), která obsahuje i číslo portu,
- configuration.properties - tento soubor nalezneme ve složce mea. V něm nastavíme adresy uložení dávkových souborů, souborů se statistikami a šablon. Je také důležité nastavit zde připojení k databázi pro uchování pomocných tabulek. Parametry User a Password budou stejné jako v persistence.xml, lišit se bude pouze URL adresa databáze.

6.2 Nasazení systému

Pokud máme dokončenou celou konfiguraci aplikace, je nutné ji zkompilovat. Spustíme webové administrační prostředí aplikačního serveru Tomcat. Zvolíme Manager, po přihlášení vybereme deploy. Dále už nám stačí jen vybrat soubor mea.war, který nalezneme ve složce mea/dist. Zkontrolujeme, zda připojení k databázi funguje správně a jsme hotovi.

7 Závěr

Cílem mé bakalářské práce bylo vytvořit webový systém, který by ulehčil a nějakým způsobem sjednotil vytváření experimentů pomocí knihovny MESSIF. Jelikož systém nemusí být pro začínající uživatele intuitivní, je v této práci popsáno uživatelské rozhraní 5.

Při zpětném pohledu musím říci, že pro naprogramování webového systému jeden člověk nestačí. Jedná se o velice náročnou práci, nejlépe určenou pro tým programátorů. Na začátku programování jsem toho o Stripes a Java Persistence API moc nevěděl. Nyní už je ovládám a zdají se mi velice snadné na použití. V kapitole Uživatelské rozhraní 5 jsem, v souladu se zadáním bakalářské práce, pomocí názorných příkladů ukázal hlavní funkce navrhovaného systému a upozornil na dodržování některých nezbytných pravidel. Dále jsem popsal souhrn funkcí pro zpracování statistiky s cílem vygenerovat graf.

Samozřejmě se nejedná o konečnou verzi systému - počítá se s jeho dalším vývojem. Hlavními funkcionalitami, o které by bylo vhodné systém rozšířit, jsou:

- webové rozhraní pro evidenci procesů,
- dávkové spouštění sady algoritmů,
- zlepšení ošetření přístupu uživatelů.

Seznam obrázků

Seznam zdrojových kódů

Literatura

[1] Midomi [online]. 2009 [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.midomi.com>.

[2] ZEZULA, Pavel; AMATO, Guiseppe; DOHNAL, Vlastislav; BATKO, Michal. Similarity Search : The Metric Space Approach. New York, NY 10013 USA : Springer, 2006. 220 s. ISBN 0-387-29146-6.

[3] MUFIN [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://mufin.fi.muni.cz>.

[4] Michal Batko, Vlastislav Dohnal, David Novak, Jan Sedmidubsky, "MUFIN: A Multi-feature Indexing Network,"sisap, pp.158-159, 2009 Second International Workshop on Similarity Search and Applications, 2009

[5] Flickr [online]. 2004 [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.flickr.com/>.

[6] Michal Batko, David Novák and Pavel Zezula. MESSIF: Metric Similarity Search Implementation Framework. In Digital Libraries: Research and Development, Lecture Notes in Computer Science, vol. 4877. Berlin, Heidelberg : Springer-Verlag, 2007, 10 pages, ISBN 978-3-540-77087-9.

[7] ELIÁŠ, Petr. Repositář vyhledávacích struktur. Brno, 2010. 40 s. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Dostupné z WWW: <http://is.muni.cz/th/255575/fi_b/>.

[8] NetBeans UML plugin [online]. 2010 [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.visual-paradigm.com/product/sde/nb>.

[9] Entity-relationship model. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 28. 12. 2010 [Citace: 03. 01. 2011]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Entity-relationship_model>.

[10] NetBeans [online]. 2010 [Citace: 02. 01. 2011]. NetBeans IDE 6.9.1 Release Information. Dostupné z WWW: <http://netbeans.org/community/releases/69/index.html>.

[11] A Brief History of NetBeans [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://netbeans.org/about/history.html>.

[12] MySQL [online]. 2010 [Citace: 03. 01. 2011]. Dostupné z WWW: <http://www.mysql.com>.

[13] Apache Tomcat [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://tomcat.apache.org/index.html>.

[14] Java Server Pages [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://java.sun.com/products/jsp/>.

[15] Enterprise JavaBeans [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.oracle.com/technetwork/java/index-jsp-140203.html>.

[16] DAOUD, Frederic. Stripes : ...and Java Web Development Is Fun Again. Dadlas, Texas : The Pragmatic Bookshelf, 2008. 381 s. ISBN 1-934356-21-2, 978-1-934356-21-0.

[17] Stripes framework [online]. 2010 [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.stripesframework.org/display/stripes/Home>.

[18] Java Persistence API [online]. 2007 [Citace: 02. 01. 2011]. Dostupné z WWW: <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>.

[19] The Java EE 5 Tutorial: Entities [online]. [Citace: 02. 01. 2011]. Dostupné z WWW: <http://download.oracle.com/javaee/5/tutorial/doc/bnbqa.html>.

[20] JFreeChart [online]. 2005 [Citace: 16. 05. 2011]. Dostupné z WWW: <http://www.jfree.org/jfreechart/>.

[21] GNU Lesser General Public License [online]. 2010. [Citace: 16. 05. 2011]. Dostupné z www: <http://www.gnu.org/licenses/lgpl.html>.

[22] GILBERT, David. The JFreeChart Class Library : Developer Guide. 1.0.9., 2008. Exporting Charts to Acrobat PDF, s. 805.

Příloha A

Obsah přiloženého CD

Součástí práce je disk, který obsahuje:

- Složku mea s celým projektem,
- Tuto práci ve formátu PDF,
- Tuto práci ve formátu LaTeX.