

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta stavební

Katedra mapování a kartografie



**Webový systém pro projekt vyrovnání sítí
GNU Gama**

Web system for project GNU Gama

Diplomová práce

Studijní program: Geodézie a kartografie

Studijní obor: Geoinformatika

Vedoucí práce: Ing. Jan Pytel, Ph.D.

Bc. Jan Synek

květen 2013



ZADÁNÍ DIPLOMOVÉ PRÁCE

studijní program: Geodézie a Kartografie
studijní obor: Geoinformatika
akademický rok: 2012/2013

Jméno a příjmení diplomanta: Bc. Jan Synek
Zadávající katedra: Katedra mapování a kartografie
Vedoucí diplomové práce: Ing. Jan Pytel, Ph.D.
Název diplomové práce: Webový systém pro projekt vyrovnání sítí GNU Gama
Název diplomové práce v anglickém jazyce: Web system for project GNU Gama

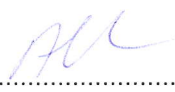
Rámcový obsah diplomové práce: Cílem práce je návrh a implementace webového systému pro projekt vyrovnávání sítí GNU Gama. Součástí implementace je návrh datových struktur, jednotkové testování. Systém bude založen na aplikačním frameworku Spring.

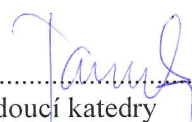
Datum zadání diplomové práce: 18.2.2013 Termín odevzdání: 17.5.2013
(vyplňte poslední den výuky přísl. semestru)

Diplomovou práci lze zapsat, kromě oboru A, v letním i zimním semestru.


Pokud student neodevzdal diplomovou práci v určeném termínu, tuto skutečnost předem písemně zdůvodnil a omluva byla děkanem uznána, stanoví děkan studentovi náhradní termín odevzdání diplomové práce. Pokud se však student řádně neomluvil nebo omluva nebyla děkanem uznána, může si student zapsat diplomovou práci podruhé. Studentovi, který při opakovaném zápisu diplomovou práci neodevzdal v určeném termínu a tuto skutečnost řádně neomluvil nebo omluva nebyla děkanem uznána, se ukončuje studium podle § 56 zákona o VŠ č. 111/1998 (SZŘ ČVUT čl 21, odst. 4).

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.


vedoucí diplomové práce


vedoucí katedry

Zadání diplomové práce převzal dne: 21.2.2013


diplomant

Formulář nutno vyhotovit ve 3 výtiscích – 1x katedra, 1x diplomant, 1x studijní odd. (zašle katedra)

Nejpozději do konce 2. týdne výuky v semestru odešle katedra 1 kopii zadání DP na studijní oddělení a provede zápis údajů týkajících se DP do databáze KOS.

DP zadává katedra nejpozději 1. týden semestru, v němž má student DP zapsanou.

(Směrnice děkana pro realizaci stud. programů a SZZ na FSv ČVUT čl. 5, odst. 7)

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

V Praze dne

.....
Bc. Jan Synek

Poděkování

V první řadě bych rád poděkoval Ing. Janu Pytlovi, Ph.D. za příkladné vedení diplomové práce. Jeho poznatky a rozsáhlé zkušenosti mi byly velkým přínosem zejména při formování tohoto projektu. Dále bych rád poděkoval tvůrci projektu GNU Gama Prof. Ing. Aleši Čepkovi, CSc., bez kterého by tato práce nemohla vzniknout.

Hlavní poděkování však patří mé rodině za jejich neustálou trpělivost a podporu během celého dosavadního studia.

Abstrakt

Diplomová práce se věnuje návrhu a implementaci webového systému pro projekt vyrovnání geodetických sítí GNU Gama. Podstatou tohoto textu je popis vytvořené webové aplikace napsané v jazyce Java s pomocí aplikačního frameworku Spring.

Práce obsahuje technologie, na kterých je aplikace založena, následuje popis architektury. Součástí práce je uživatelský průvodce stěžejními kroky aplikace. Důraz je kladen na poskytnutí převážně obecných informací o technologiích použitých v implementovaném systému.

Klíčová slova

GNU Gama, vyrovnání, aplikace, Web, Java, Spring, databáze.

Abstract

Diploma thesis deals with the design and implementation of web system for project GNU Gama dedicated to adjustment of geodetic networks. The essence of this text is a description of the created web application based on the Java language and the Spring application framework.

Thesis includes technologies on which the application is based, followed by a description of the application architecture. A part of this thesis is also a user guide through key steps of application. Particular emphasis is placed on providing mostly general information about the technologies used in the implemented system.

Keywords

GNU Gama, adjustment, application, Web, Java, Spring, database.

Obsah

Seznam tabulek	v
Seznam obrázků	vi
Seznam ukázek	vii
Seznam zkratk	viii
Úvod	1
1 GNU Gama	2
1.1 Historie	2
1.2 Program gama-local	3
1.2.1 Instalace	4
1.2.2 Vstupní data	5
1.2.3 Použití programu	6
1.2.4 Výsledek vyrovnání	7
2 Použité technologie	8
2.1 Programovací jazyky	8
2.1.1 Jazyk Java	8
2.1.2 HTML	11
2.1.3 CSS	12
2.1.4 JavaScript	13
2.2 Knihovny	14
2.2.1 Spring Framework	14
2.2.2 Spring Security	18
2.2.3 Apache Velocity	19

2.2.4	JUnit	20
2.2.5	jQuery	20
2.2.6	Ostatní	21
2.3	Programové vybavení	22
2.3.1	PostgreSQL	22
2.3.2	Apache Tomcat	24
2.3.3	Ostatní	25
3	Architektura aplikace	26
3.1	Úvod do architektury	26
3.2	Perzistentní vrstva	28
3.2.1	Databáze	28
3.2.2	Spring JDBC	30
3.3	Aplikační vrstva	31
3.3.1	Parsování GNU Gama XML	33
3.3.2	Spouštění externích procesů	34
3.3.3	Automaticky spouštěné procedury	35
3.3.4	Emailová služba	35
3.4	Prezentační vrstva	36
3.4.1	Spring MVC	36
3.4.2	Apache Velocity	38
3.4.3	Uživatelské prostředí	38
3.4.4	Zabezpečení	40
3.4.5	Lokalizace	41
3.4.6	Validace	42
3.5	Testování	44
3.5.1	JUnit	44
3.5.2	Selenium	46
3.6	Produkční prostředí	47
3.6.1	Konfigurace software	48
4	Popis aplikace	50
4.1	Registrace nového uživatele	52
4.2	Založení nového vyrovnání	55
4.3	Správa výpočtů	58

4.4	Export výsledků	60
4.5	Smazání uživatelského účtu	60
	Závěr	63
	Použité zdroje	64
A	Databázové schéma	I
B	Konfigurace aplikace	III
C	Schéma beanů	IV

Seznam tabulek

2.1	Limity PostgreSQL DBMS [8]	22
3.1	Výhody použití Spring JDBC	30
3.2	Dostupné anotace frameworku JUnit 4.x	45
A.1	Souhrnný přehled objektů v databázi	I

Seznam obrázků

1.1	Datová struktura observací GNU Gama [2]	4
2.1	Přehled modulů Spring Framework [4]	15
2.2	Infrastruktura Spring MVC [4]	17
3.1	Třívrstvá architektura webové aplikace	27
3.2	Schéma produkčního prostředí	47
4.1	Úvodní stránka aplikace WebGama	51
4.2	Registrační formulář aplikace WebGama	52
4.3	Chybný vstup registračního formuláře aplikace WebGama	53
4.4	Přihlašovací formulář aplikace WebGama	53
4.5	Hlavní stránka aplikace WebGama	54
4.6	Možné způsoby zadání nového vyrovnání	55
4.7	Zadání definice lokální sítě	56
4.8	Zadání měřených bodů lokální sítě	56
4.9	Zadání parametrů lokální sítě	57
4.10	Zadání clusterů lokální sítě	58
4.11	Správce výpočtů WebGama	59
4.12	Sdílení výpočtů WebGama	59
4.13	Výběr výpočtů připravených k exportu	60
4.14	Export výsledku výpočtu	61
4.15	Smazání uživatelského účtu	62

Seznam ukázek

1.1	GNU Gama XML vstupní dávka	5
2.1	Velocity Template syntaxe	19
3.1	Příklad užití typu <code>JdbcTemplate</code>	30
3.2	Příklad XML bean konfiguračního souboru	32
3.3	Příklad POST metody <code>Controlleru</code>	36
3.4	Příklad dvou lokalizačních bundlů	41
3.5	Implementace <code>Spring Validator</code>	42
3.6	Jednotkové testování pomocí <code>JUnit</code>	45
B.1	Konfigurační soubor aplikace <code>WebGama</code>	III

Seznam zkratek

ACID	A tomicity, C onsistency, I solation, D urability
API	A pplication p rogramming i nterface
AJAX	A synchronous J avaScript A nd X ML
AJP	A pache J Serv P rotocol
CAS	C entral A uthentication S ervice
CRUD	C reate, R ead, U ppdate, D elele
CSS	C ascading S tyle S heets
ČVUT	České vysoké učení technické
DAO	D ata A ccess O bject
DBCP	D atabase C onnection P ool
DBMS	D atabase M anagement S ystem
DML	D ata m anipulation language
DNS	D omain N ame S ystem
DOM	D ocument O bject M odel
EJB	E nterprise J ava B ean
DTD	D ocument T ype D efinition
GNU	G NU's N ot U nix
GSO	G ram-Schmidt O rthogonalization
HTML	H yper T ext M arkup L anguage
HTTP	H ypertext T ransfer P rotocol
IDE	I ntegrated d evelopment e nvironment
IOC	I nversion of C ontrol
IP	I nternet P rotocol
JAXB	J ava A rchitecture for X ML B inding
JDK	J ava D evelopment K it
JNDI	J ava N aming and D irectory I nterface

JPA	J ava P ersistence A PI
JSON	J ava S cript O bject N otation
JSP	J ava S erver P ages
JSR	J ava S pecification R equest
JVM	J ava V irtual M achine
LDAP	L ightweight D irectory A ccess P rotocol
MB	M egabyte
MS	M icrosoft
MVC	M odel- V iew- C ontroller
OS	O perační systém
PHP	P HP: H ypertext P reprocessor
QUEL	Q uery L anguage
REST	R epresentational S tate T ransfer
SAX	S imple A PI for X ML
SIM	S ubscriber I dentify M odule
SLF4J	S imple L ogging F acade for J ava
SMTP	S imple M ail T ransfer P rotocol
SQL	S tructured Q uery L anguage
SSL	S ecure S ockets L ayer
StAX	S treaming A PI for X ML
SVD	S ingular V alue D ecomposition
SVG	S calable V ector G raphics
TERENA	T rans- E uropean R esearch and E ducation N etworking A ssociation
TCP	T ransmission C ontrol P rotocol
TDD	T est- d riven d evelopment
UI	U ser I nterface
URI	U niform r esource i dentifier
URL	U niform r esource l ocator
VTL	V elocity T emplate L anguage
XHTML	E xtensible H yper T ext M arkup L anguage
XML	E xtensible M arkup L anguage

Úvod

Cílem této diplomové práce je navrhnout a implementovat webový systém pro vyrovnání lokálních geodetických sítí pomocí projektu GNU Gama. Projekt GNU Gama je vynikající projekt, který však postrádá grafické uživatelské rozhraní. Program (`gama-local`) se ovládá z příkazové řádky a data jsou zadávána pomocí XML. V době všudypřítomných webových aplikací a cloud řešení je tato práce a aplikace určena zejména uživatelům, kteří se nechtějí zabývat překážkami při zadávání svých měření. Dosud bylo vytvořeno několik více či méně úspěšných grafických uživatelských rozhraní, ale proč nezajít ještě o kousek dál a nenabídnout uživateli vyrovnání místních geodetických sítí jen výměnou za jednoduchou registraci bez instalace a složitého vytváření XML dávek nebo databázových souborů? Navíc systém nabízí uživateli správu jeho výpočtů uchovaných v databázi a možnou pozdější interaktivní editaci či export výsledků do několika formátů.

Práce je rozdělena do několika kapitol. V první kapitole je představen projekt GNU Gama, jeho přednosti, nástroje a ovládání z příkazové řádky. Tato část rovněž obsahuje ukázkou vstupu a možnosti výstupů.

Druhá kapitola ve zkratce představí technologie a knihovny použité ve webové aplikaci nazvané *WebGama*. Následná kapitola popisuje samotnou architekturu aplikace. Spojuje výše popsané technologie s jednotlivými vrstvami aplikace.

Poslední kapitola provede uživatele od registrace až po vyrovnání jednoduché sítě. Jednotlivé úkony budou demonstrovány na *screenshotech* aplikace.

Kapitola 1

GNU Gama

Projekt GNU Gama se věnuje vyrovnání geodetických sítí. Byl uvolněn pod licencí *GNU General Public License*. Projekt je psán v jazyce C++ a jeho součástí je knihovna geodetických tříd a funkcí `GaMaLib`. Dále využívá třídu `matvec` pro práci s maticemi a externí XML *parser* `expat` napsaný Jamesem Clarkem. Je určen pro práci s daty získanými tradičním měřením (podzemní měření nebo měření s vysokou přesností apod.), kde například není možné využít GPS měření. Z naměřených dat provede vyrovnání sítě a doplní výsledek statistickými analýzami.[2]

Součástí projektu jsou dva řádkové programy:

- `gama-g3` – vyrovnání v globálním souřadnicovém systému, dosud není plně implementováno,
- `gama-local` – vyrovnání v místním kartézském souřadnicovém systému.

Program `gama-local` bude podrobněji představen v sekci 1.2, protože je použit jako výpočetní základ webové aplikace *WebGama* a jeho zevrubná znalost je nutná pro snazší pochopení dalších částí textu.

Další informace lze najít na domovské stránce projektu:

<http://www.gnu.org/software/gama/>

1.1 Historie

Historie projektu sahá do roku 1998, kdy byly vytvořeny prof. Ing. Alešem Čepkem, CSc. první verze na katedře mapování a kartografie fakulty stavební ČVUT jako demonstrace

vyučovaného objektového programování. Celý projekt byl inspirován systémem Geodet/PC navrženým Ing. Františkem Charamzou implementovaným v jazyce Fortran. Původní návrh počítal s distribucí projektu pouze v rámci fakulty. Český jazyk byl použit jak v dokumentaci, tak i při implementaci software. Pokud se však měl projekt dostat i za hranice, musely být provedeny některé úpravy, například překlad vstupní XML dávky do anglických ekvivalentů. Projekt GNU Gama byl veřejně prezentován nejprve na semináři *FIG Working Week 2000* v Praze a poté v roce 2001 na *FIG Workshop and Seminar* v Helsinkách.

Postupem času byla GNU Gama rozšiřována o nové funkce a do projektu se zapojovali dnes již bývalí studenti. Za zmínku stojí například inženýři Jiří Veselý, Petr Doubrava nebo Jan Pytel. Seznam všech přispěvatelů je uveden v dokumentaci projektu.[2]

1.2 Program gama-local

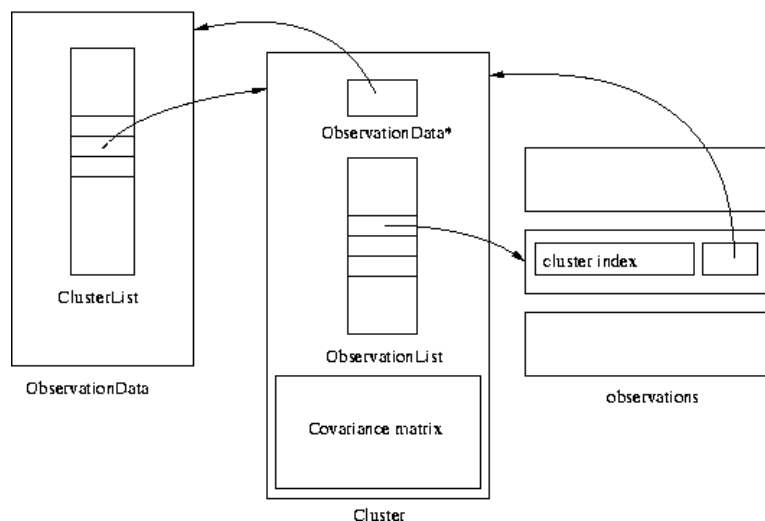
Program `gama-local` je program pro příkazovou řádku, který je určen pro vyrovnání geodetických sítí v místním souřadnicovém systému. V době psaní této práce (duben 2013) byla aktuální stabilní verze 1.13f. V této verzi je podporováno jedenáct jazyků pro výstup a čtyři různé výpočetní algoritmy:

- algoritmus singulárního rozkladu (SVD),
- Gram-Schmidtova ortogonalizace (GSO),
- Choleskyho rozklad (Cholesky),
- Choleskyho rozklad řídké matice normálních rovnic (Envelope).

Celá koncepce projektu stojí na tzv. *clusterech*, což jsou skupiny korelovaných měření se známou kovarianční maticí. Samozřejmě lze vkládat i nekorelovaná měření. Datové struktury byly navrženy tak, aby bylo možné snadno přidávat nové typy observací pouhým odvozením báze třídy `Observation` a definováním několika virtuálních funkcí. Na obrázku 1.1 je schématicky naznačena interní datová struktura observací GNU Gama.

V současné době projekt podporuje následujících 8 typů měření:

- směry,
- vodorovné délky,
- vodorovné úhly,



Obrázek 1.1: Datová struktura observací GNU Gama [2]

- šikmé délky,
- zenitové úhly,
- převýšení,
- měřené souřadnice (např. souřadnice s danou kovarianční maticí),
- 3D vektory (měřené souřadnicové rozdíly).[2]

1.2.1 Instalace

Projekt GNU Gama je vyvíjen a testován na operačním systému GNU/Linux, je však otestován i na platformě MS Windows pro určité druhy překladačů (například MS Visual C++ Compiler). Nejprve je nutné získat zdrojové kódy projektu. Aktuální verzi zdrojového kódu aplikace lze získat z repozitáře projektu v Gitu.¹ Jedním ze způsobů, jak získat zdrojové kódy, je příkaz:

```
$ git clone git://git.sv.gnu.org/gama.git
```

Instalace se skládá z několika jednoduchých příkazů:

¹Git je systém pro správu verzí, více informací na domovské stránce: <http://git-scm.com/>

```
$ cd gama
$ ./autogen.sh
$ ./configure
$ make
```

Poté v adresáři bin najdeme mimo jiné zkompileovaný program `gama-local`.^[2]

1.2.2 Vstupní data

Pro práci s programem `gama-local` je nejprve nutné vytvořit vstupní data. Program na vstupu podporuje dva různé formáty vstupních dat. Historicky prvním způsobem bylo použití XML, které bylo vybráno díky jeho snadno pochopitelné syntaxi, možnosti snadného *parsování* a přípravy v prostém textovém souboru a relativně snadné definici struktury dat a její následné validaci. Příklad vstupní XML dávky je v ukázce 1.1.

Ukázka 1.1: GNU Gama XML vstupní dávka

```
<?xml version="1.0" ?>
<!DOCTYPE gama-local SYSTEM "gama-local.dtd">
<gama-local version="2.0">
  <network>
    <parameters sigma-apr="1" sigma-act="apriori" />
    <points-observations angle-stdev="15" distance-stdev="20">

      <point id="1" x="1118103.84" y="668559.14" adj="XY" />
      <point id="2" x="1117697.19" y="667132.98" adj="XY" />
      <point id="3" x="1119159.92" y="667054.59" adj="XY" />
      <point id="4" x="1119260.13" y="667932.57" adj="xy" />

    <obs from="1">
      <distance to="3" val="1838.258" />
      <distance to="2" val="1483.050" />
      <angle bs="2" fs="3" val="56.6464" />
    </obs>

    <obs from="2">
      <distance to="3" val="1464.841" />
      <angle bs="4" fs="1" val="52.2111" />
      <angle bs="3" fs="4" val="33.5127" />
      <angle bs="3" fs="1" val="85.7233" />
    </obs>
  </network>
</gama-local>
```

```
<obs from="3">
  <distance to="4" val="883.684" />
  <angle bs="1" fs="2" val="57.6310" />
  <angle bs="4" fs="1" val="46.1976" />
  <angle bs="4" fs="2" val="103.8266" />
</obs>

<obs from="4">
  <distance to="2" val="1755.639" />
  <angle bs="2" fs="3" val="62.6593" />
</obs>

</points-observations>
</network>
</gama-local>
```

Strukturu dokumentu a jeho značek lze nalézt v DTD souboru `gama-local.dtd`. Struktura přibližně odpovídá datové struktuře GNU Gama a je navržena tak, aby byla pro uživatele co nejlépe pochopitelná. XML dávku lze vytvořit v libovolném textovém editoru nebo je možné upravit sadu ukázek, kterou lze stáhnout ze stránek projektu pouhým příkazem `git clone`:

```
$ git clone git://git.sv.gnu.org/gama/examples.git
```

Další způsob zadávání naměřených dat do GNU Gama programu `gama-local` je pomocí databázového souboru. Jedná se o soubor databáze SQLite 3. Strukturu databáze lze najít ve schématu `gama-local-schema.sql`.

1.2.3 Použití programu

Jak již bylo zmíněno, `gama-local` je řádkovým programem, proto je nutné mít základní znalosti spouštění příkazů z příkazové řádky dané platformy. Pro vypsaní ovládacích prvků a možností programu je možné zobrazit nápovědu spuštěním programu bez parametrů. Na OS GNU/Linux vypadá spuštění následovně:

```
$ ./gama-local input.xml --algorithm gso --language cz --xml test.xml
```

V předešlé ukázce byl spuštěn program `gama-local` umístěný v aktuálním adresáři společně se vstupní dávkou `input.xml`. Byl vybrán výpočetní algoritmus, jazyk výsledku a výstupní formát výsledku (XML soubor). V následující podsekcí jsou stručně popsány

možné výstupní formáty. Dále je také možné definovat kódování či úhlové jednotky. Všechny parametry je možné nalézt v dokumentaci projektu.[2]

1.2.4 Výsledek vyrovnání

Po úspěšném vyrovnání místní sítě je v současné verzi možné výsledek exportovat do čtyř formátů:

- textový formát – vhodný pro rychlý náhled na výsledek vyrovnání,
- XML – vhodný pro další zpracování,
- HTML – vhodný pro prezentaci výsledku v naformátované podobě,
- SVG – vektorový obrázek s vyznačenou konfigurací bodů a jejich elips chyb.

Mnohé uživatele může výše popsané ovládání odradit od použití tohoto programu k vyrovnání svých měření. Pro ně je určena aplikace vytvořená v rámci této diplomové práce. V následujících kapitolách budou popsány technologie a architektura navrhované a posléze implementované aplikace pro snadnější vyrovnání lokálních geodetických sítí.

Kapitola 2

Použité technologie

V této kapitole jsou představeny technologie a knihovny použité při vývoji webové aplikace. Protože jich je použito mnoho, jsou podrobněji popsány jen některé z nich. Některé technologie a zvláště jejich použité verze jsou záměrně obsahově zvýrazněny. Důraz je kladen na technologie, které jsou nějakým způsobem výjimečné v kontextu vytvořené aplikace.

2.1 Programovací jazyky

2.1.1 Jazyk Java

Jazyk Java je objektově orientovaný programovací jazyk syntaxí podobný jazykům C a C++. Hlavním mottem jazyka je: „write once, run anywhere“ umožňující vývojářům použít stejný zdrojový kód na všech dnes populárních platformách¹ bez nutnosti překladu pro každou platformu zvlášť. Překlad zdrojového kódu se neprovádí přímo do strojového kódu ale do tzv. (*bytecode*), který běží v prostředí virtuálního stroje JVM (*Java Virtual Machine*) bez ohledu na architekturu počítače. Základní vlastnosti tohoto jazyka jsou:

- **jednoduchý** – syntaxí je podobný jazykům C a C++. Odpadly složité konstrukce a zároveň byla přidána jiná užitečná rozšíření.
- **objektově orientovaný** – umožňuje objektově orientované programování díky podpoře polymorfismu, dědičnosti a zapouzdření.
- **interpretovaný** – není překládán přímo do strojového kódu, ale do *bytecode*, který je interpretován virtuálním strojem JVM.

¹Tedy na všech operačních systémech, kde je k dispozici JVM

- **robustní** – je určený pro vývoj vysoce spolehlivého softwaru.
- **vícevláknový** – je schopný pracovat s více výpočetními vlákny najednou.
- **výkonný** – přestože je to jazyk interpretovaný, ztráta výkonu oproti kompilovaným jazykům není tak významná díky tzv. „just-in-time“ kompilátorům².
- **bezpečný** – obsahuje vlastnosti, které chrání počítač v síťovém prostředí.
- **elegantní** – je snadno čitelný, vyžaduje ošetření výjimek a typovou kontrolu.

Dalšími silným prvkem jazyka je automatická správa paměti tzv. *garbage collection*. Již nepoužívané části paměti jsou automaticky uvolněny pro další použití. V prvních verzích Javy to byl značný výkonnostní problém. Problém byl z velké části vyřešen zavedením nových *garbage collection* algoritmů a rozdělením paměti do tzv. generací, ve kterých jsou použity jiné algoritmy. Objekty jsou pak přesouvány mezi generacemi podle délky svého života.

Historie

Jazyk byl vyvinut v devadesátých letech týmem pod vedením Jamese Goslinga ve firmě Sun Microsystems³. Původně se jazyk jmenoval Oak (pojmenovaný podle stromu před Goslingovou kanceláří). Současný název se často odvozuje od slangového označení kávy. Původně byl jazyk určen pro zařízení jako jsou videorekordéry nebo set-top boxy. Rozmach *World Wide Web* byl impulsem i pro firmu Sun, která další vývoj Javy soustředila na Internet a internetový prohlížeč podporující Javu. První verze byla vydána roku 1995 s označením Java 1.0. Nejrozšířenější webové prohlížeče do sebe zahrnuly podporu Javy a Java se stala velice populární. Ve verzi 1.2 byla Java rozdělena do několika edic⁴:

- Java SE (Standard Edition) – základní edice vyvíjena od začátku. Obsahuje široké spektrum užitečných tříd a metod.
- Java EE (Enterprise Edition) – určená pro tvorbu velkých podnikových aplikací a informačních systémů, obsahuje všechny třídy SE edice.

²Je to speciální typ překladače, který překládá za běhu programu často využívané části kódu přímo do nativního strojového kódu a tím je dosaženo vyššího výkonu aplikace.

³V roce 2010 byla firma Sun Microsystems koupena společností Oracle Corporation, která tak převzala celé jejich portfólio včetně Javy.

⁴V roce 2006 bylo číslování verzí Javy i edic změněno kvůli marketingovým záměrům na uvedenou podobu.

- Java ME (Micro Edition) – určená pro vývoj softwaru na mobilní telefony, PDA nebo set-top boxy.
- Java Card – užívaná např. v SIM kartách nebo kartách do bankomatů.

Postupem doby do Javy přibývaly nové prvky a vylepšení. V roce 2006 Sun uvolnil většinu zdrojových kódů, na které vlastnil práva. Aktuální verze Java 7 byla vydána již společností Oracle v roce 2011.[3]

Platforma

Jazyk Java je pouhou součástí Java platformy. Celá platforma se skládá z mnoha softwarových produktů a specifikací, které dohromady vytvářejí vývojářské prostředí pro vytváření multiplatformních aplikací. Srdcem celé platformy je již zmíněný virtuální stroj JVM. Jazyk Java je jedním z mnoha jazyků, které lze přeložit do *bytecode*. Existují *bytecode* překladače pro jiné populární jazyky jako je například JavaScript, Python nebo Ruby. Dále existují jazyky, které jsou nativně vytvořeny pro běh v JVM (Scala, Clojure, Groovy, ...). Z toho všeho lze usoudit, že je Java platforma velice úspěšná. Mnoho úspěšných aspektů bylo inspirací při vytváření novější platformy .NET firmou Microsoft, která plně podporuje pouze platformy Windows.

Existuje několik různých implementací Java platformy. Nejznámější a nejvíce používanou je implementace firmy Oracle (dříve Sun). Další hojně užívanou platformou především v *open source* prostředích je OpenJDK. Dnes nejrozšířenější mobilní operační systém Android používá jazyk Java pro vývoj klientských aplikací. Nelze jej však plně řadit do implementací Java platformy, protože využívá vlastní virtuální stroj zvaný Dalvik VM a také vlastní knihovnu tříd. Tato skutečnost byla již několikrát předmětem soudních sporů mezi firmou Google zastřešující vývoj Androidu a firmou Oracle držící práva k platformě Java.[3]

Java v číslech

V dnešní době je jedním z nejvíce používaných jazyků na světě. Vyskytuje se jak na malých mobilních zařízeních, tak i na rozsáhlých serverových clusterech.

- Na 1,1 miliardě osobních počítačů lze spustit aplikaci v Javě,
- 930 milionů stažených Java prostředí každý rok,
- 3 miliardy mobilních telefonů používá Javu,

- 100% Blu-ray přehrávačů obsahují Javu,
- 1,4 miliard karet (Java Card) je vyrobeno každý rok.[6]

2.1.2 HTML

HTML je značkovací jazyk určený pro tvorbu webových stránek, které jsou posléze zobrazovány webovými prohlížeči. Pomocí tohoto jazyka lze zobrazovat texty, obrázky nebo jiný audio/vizuální obsah na webu. Zjednodušeně řečeno se jedná o sadu standardizovaných většinou párových značek, které formátují dokument do požadovaného stavu. Aktuální specifikace je HTML 4.01 zveřejněná v prosinci roku 1999, ale intenzivně se pracuje na specifikaci HTML5, která je již částečně implementována mnoha prohlížeči. Původně se předpokládalo, že HTML nahradí specifikace XHTML⁵, ale v roce 2007 vznikla pracovní skupina s cílem vytvořit další verzi HTML. Společně s CSS a některým ze skriptovacích jazyků (nejčastěji JavaScript) lze vytvořit rozsáhlé dynamické weby.

Po zadání adresy prohlížeč přijme od webového serveru odpověď ve formě HTML dokumentu. Ze značek v tomto dokumentu je vytvořen tzv. HTML DOM, což je objektová reprezentace HTML v paměti. Specifikace DOM je nezávislá na platformě či jazyku a umožňuje přistupovat k dokumentu jako ke stromu. Je tedy možné následně modifikovat obsah, strukturu nebo styl dokument nebo jen jeho částí.

HTML 5

HTML5 je nejnovější specifikace HTML ve stádiu návrhu. Tato specifikace je rozdělena do několika na sobě víceméně nezávislých celků (tzv. API). Důraz je kladen na dnes nejčastěji používaná multimédia se zachováním čitelnosti jak pro běžné uživatele, tak pro zařízení jako jsou prohlížeče, *parser*y a tak dále. HTML5 přináší mimo jiné tyto novinky:

- nové značky pro lepší strukturu webu,
- *drag-and-drop* funkce,
- `<video>` a `<audio>` elementy pro nativní vkládání videí a hudby,
- `<canvas>` element pro interpretaci vektorové grafiky a následnou editaci pomocí skriptovacího jazyka,

⁵Striktnější forma HTML založená na XML.

- nové formulářové objekty pro snadnější vkládání například data nebo emailové adresy,
- geolokace nebo možnost pracovat *offline* pomocí tzv. *cachování* a mnoho dalšího.

Společně s CSS a JavaScriptem vytváří silnou trojici pro vývoj vysoce kvalitních webových aplikací současnosti a blízké budoucnosti bez nutnosti použití dalších nástrojů (Flash, ...).[7]

2.1.3 CSS

CSS znamená v překladu kaskádové styly, je to jazyk určený pro popis zobrazení webových dokumentů napsaných značkovacím jazykem. Nejčastěji je používán pro definování stylu HTML nebo XHTML webových dokumentů. Může však být použit i v případě jakéhokoliv XML jako je např. SVG. Je navržen tak, aby byl používán paralelně s dokumenty, které jsou vytvořeny pomocí značkovacího jazyka. V praxi to znamená, že v jednom souboru je pouze obsah a v dalším jsou definována pravidla, jak tento obsah má být prezentován uživateli.

Syntax jazyka je velice jednoduchá. Skládá se ze selektoru (často název prvku v dokumentu nebo jeho třída) a bloku pravidel definující způsob vykreslení prohlížečem na základě určených priorit. Syntax jazyka je od první verze stejná, ale jednotlivá pravidla jsou přidávána nebo odebrána ze specifikace. Poslední vydaná specifikace je CSS 2.1. V současné době se pracuje na specifikaci úrovně 3, kterou postupně implementují jednotlivé prohlížeče. Existuje i úroveň 4, která však není dosud žádným prohlížečem implementována.

Na základě pravidel lze v dokumentu mimo jiné upravit:

- barvy a styly písem nebo pozadí,
- pozici a viditelnost jednotlivých elementů,
- velikost a druh ohraničení,
- velikost jednotlivých prvků,
- zarovnání textů a podobně.

Výhodou tohoto způsobu formátování obsahu před formátováním v samotném HTML je například snazší údržba. Stačí pouze předefinovat pravidlo a změna se projeví u všech prvků se společným selektorem. Dále je možné vytvořit několik různých stylů pro každé zařízení zvlášť (tisk, mobil, prohlížeč, ...) nebo možnost *cachování* souboru se stylem a tím

tak rychlejší načtení stránek. Hlavní nevýhodou kaskádových stylů je rozdílná podpora ze strany prohlížečů. Často se stává, že jsou jednotlivá pravidla rozdílně zobrazena v některém z prohlížečů. Nevýhodou je rovněž neschopnost pracovat s rodičovskými prvky vybraného elementu.

CSS3

Vývoj této specifikace byl zahájen roku 2005, oproti CSS2 je specifikace rozdělena do několika modulů. Tato technologie přináší řadu nových pravidel pro obsahu dokumentů:

- zaoblené rohy,
- stínování písma a blokových elementů,
- animované přechody mezi jednotlivými stavy elementu,
- animace obrázků nebo nastavení průhlednosti,
- 2D a 3D transformace.[3]

2.1.4 JavaScript

Jazyk JavaScript je interpretovaný jazyk původně vyvinutý jako část webových prohlížečů. Je to dynamický, slabě typový, skriptovací jazyk se syntaxí podobnou jazyku C. Mnoho lidí jej chybně zaměňuje za jazyk Java. Mají ale podobné pouze některé jmenné konvence a podobnou syntaxi. Přestože je nejčastěji využíván ke skriptování na straně klienta na webových stránkách, používá se i mimo ně. Za zmínku stojí třeba platforma Node.js, která umožňuje vytvářet aplikace na straně serveru, nebo tvorba *widgetů* na plochu různých platforem.

Majoritní význam má ale při tvorbě dynamických webových aplikací. Pomocí JavaScriptu lze snadno měnit HTML DOM a tím tak interaktivně měnit obsah nebo vzhled stránek. Dále je možné získávat asynchronně data ze serveru (technologie AJAX), validovat vstupní hodnoty formulářů a podobně. Aktuální stabilní specifikace JavaScriptu vychází ze standardu ECMAScript-262. Pro zjednodušení vývoje existuje mnoho knihoven. V následujícím výčtu jsou uvedeny některé z nich:

- jQuery
- Dojo

- Prototype
- YUI
- MooTools

Knihovna jQuery je použita v aplikaci WebGama a je podrobně popsána v části 2.2.5.

AJAX

AJAX je technologie používaná na straně klienta k vytváření asynchronních webových aplikací. S touto technologií může aplikace odesílat data na server a obratem přijímat data ze serveru na pozadí existující webové stránky. Data jsou získávána pomocí objektu XMLHttpRequest, ale není nutné používat formát XML ani dotazy odesílat asynchronně. V mnoha případech je místo XML použit formát JSON, který syntaxí vychází právě z JavaScriptu a v mnoha ohledech předčí formát XML. Je snadno čitelný a zároveň není tolik „upovídaný“ jako XML.[3]

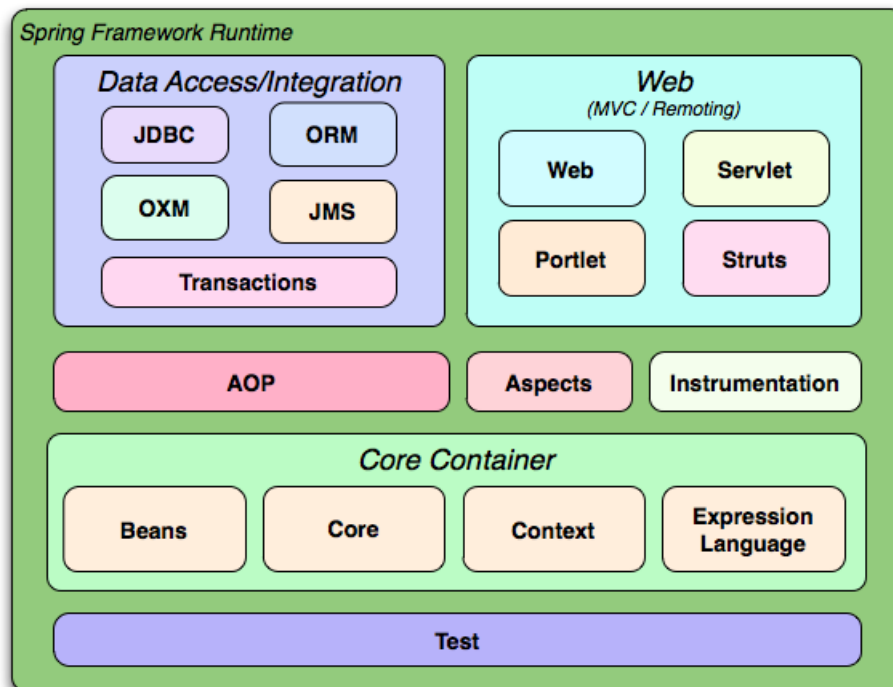
2.2 Knihovny

V této sekci jsou uvedeny knihovny třetích stran, které jsou použity v aplikaci. Jedná se až na jednu výjimku o knihovny v jazyce Java. Výjimkou je knihovna jQuery, která je knihovnou JavaScriptovou.

2.2.1 Spring Framework

Je populární open source framework pro vývoj enterprise aplikací v Javě. Poskytuje komplexní infrastrukturu pro vývoj Java aplikací. Spring se stará o infrastrukturu, je tedy možné se plně zabývat vývojem dané aplikace. Základem celého Springu je IOC kontejner, který řeší závislosti mezi objekty na základě konfigurace mimo samotný kód. Takto nakonfigurovaný projekt je pak snáze udržitelný, rozšiřitelný a čitelnější. Jednotlivé beans jsou zpravidla singletony, není nutné vytvářet pokaždé nový objekt, jen se objektu přiřadí ta správná závislost. Mimo to Spring Framework obsahuje další užitečné knihovny, které jsou schématicky rozděleny do několika modulů, jak je naznačeno na obrázku 2.1. V Javě existuje mnoho frameworků pro tvorbu enterprise nebo webových aplikací, ale tento je populární právě proto, že obsahuje komplexní řešení tvorby celé aplikace. Základní prvky frameworku mohou být použity na jakýkoliv druh aplikace. Framework začal být velice

populární hlavně kvůli své podpoře vývoje webových aplikací jako protiváha k stávající těžkopádné EJB technologii.[4]



Obrázek 2.1: Přehled modulů Spring Framework [4]

Historie

První verze Spring Framework byla napsána Rodem Johnsonem, který jej publikoval jako ukázkové části ve své knize *Expert One-on-One J2EE Design and Development* v říjnu 2002. První verze byla veřejně publikována pod Apache 2.0 licencí v červnu následujícího roku. Spring se postupem času stával čím dál více populární a ceněný vývojářskou komunitou. Současná verze je 3.2.2 s plánovaným vydáním verze 4.0 koncem roku 2013.[3]

IOC a jádro

Základem frameworku je takzvaný IOC kontejner. Ve Spring Framework je ve formě tzv. *dependency injection*. Doslovný překlad „vstřikování závislostí“ částečně vystihuje jeho účel. Je to proces, v kterém objekty definují svoje závislosti na jiné objekty, s kterými spolupracují. IOC kontejner automaticky doplní (vstřikuje) tyto závislosti na základě

externích konfiguračních metadat. V rozsáhlých aplikacích se stává zdrojový kód čitelnějším a zároveň více efektivním, protože jsou jednotlivé objekty poskytnuty jen tam, kde jsou zrovna potřeba.

Tato funkčnost je zajištěna moduly *Bean* a *Core*. Modul *Context* staví na právě zmíněných modulech a rozšiřuje jejich funkčnost například o internacionalizaci a další prvky. Základním bodem tohoto modulu je rozhraní *ApplicationContext*.

DAO

Framework také obsahuje širokou podporu pro práci s daty. Zejména je široce podporována práce s databázemi. Spring Framework nativně podporuje práci s JDBC. JDBC je API programovacího jazyka Java, které definuje způsob, jímž klient přistupuje do databáze. Rozhraní je implementováno subjekty, které provozují jednotlivé databázové systémy. Není tedy nutné znát API jednotlivých databází.

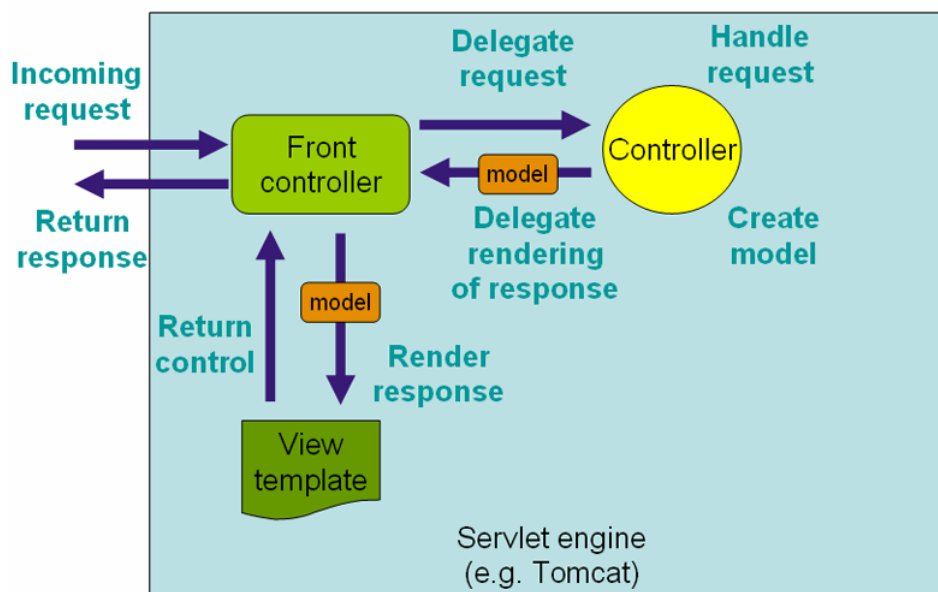
Na základě návrhového vzoru Šablona (Template) usnadňuje vývojářům nízkoúrovňovou práci s rozhraním JDBC a odbourává nutnost psaní zdlouhých a opakujících se konstrukcí. Také podporuje populární API pro objektově relační mapování jako jsou Hibernate, JPA nebo MyBatis⁶. Podporuje i uchovávání dat v XML pomocí technologií jako je například JAXB a v neposlední řadě podporuje práci s transakcemi.

Web

Webová část Spring Framework je zastoupena Spring MVC frameworkem. Tento framework je založen na paradigmatu Model-View-Controller, který je někdy řazen mezi návrhové vzory. Původně nebylo vůbec plánováno vytvoření vlastního webového frameworku, ale nespokojenost s webovým frameworkem Struts donutila vývojáře změnit názor. Implementace je založena na Servlet API. Středobodem frameworku je objekt *DispatcherServlet*, což je de facto *controller*, kterému jsou předávány klientské požadavky. Na obrázku 2.2 je schématicky naznačena infrastruktura Spring MVC.

Po přijetí deleguje *DispatcherServlet* požadavek ke zpracování *controlleru*. Na základě konfigurace a logiky je v *controlleru* zpracován *model* a jméno souboru obsahujícího šablonu pro publikování odpovědi. Tato smíšená odpověď je opět předána *DispatcherServletu* a je dále delegována do procesoru, který na základě šablony a dynamických informací z modelu vytvoří požadovanou odpověď (*view*). Tato odpověď je pak přeposlána klientovi opět *DispatcherServletem*.

⁶Dříve známý jako iBatis.



Obrázek 2.2: Infrastruktura Spring MVC [4]

V následujícím výčtu jsou vypsány některé vlastnosti Spring MVC:

- podpora tvorby aplikací v REST stylu,
- snadná lokalizace,
- snadná validace klientských vstupů,
- podpora nahrávání souborů na server,
- jednoduché zacházení s výjimkami nebo cookies,
- snadná integrace s JSP, Velocity nebo Freemarker,
- podpora asynchronního zpracování a mnoho dalšího.

Testování

Mimo výše uvedené moduly Spring dále zahrnuje širokou podporu pro testování. S pomocí knihoven JUnit nebo TestNG dokáže otestovat téměř všechny vrstvy aplikace, nově i modul Spring MVC.[4]

2.2.2 Spring Security

Spring Security je široce přizpůsobitelný framework pro implementaci autentizace, autorizace a mnoha dalších bezpečnostních prvků. V současnosti je víceméně standardem pro zabezpečení webových Spring aplikací. Předností projektu je krátká a nenáročná konfigurace základního zabezpečení aplikace.

Knihovna byla vytvořena v roce 2003 ještě pod jménem Acegi Security. Projekt byl poprvé publikován pod Apache licencí v březnu 2004. Do rodiny Spring projektů byl přijat v roce 2008 a jeho název byl změněn na Spring Security. V současnosti je framework vyvíjen a komerčně podporován divizí SpringSource a je jedním z nejvyzrálejších Spring projektů.

Vlastnosti

Projekt obsahuje řadu bezpečnostních vlastností:

- podpora autentizace pomocí webového formuláře nebo certifikátu,
- podpora databází, LDAP nebo CAS,
- podpora autorizace na základě definovaných rolí,
- podpora limitace souběžných sezení na definovaný počet,
- podpora šifrování hesla,
- podpora trvalého přihlášení,
- podpora OpenID⁷ autentizace a další.

Zabezpečení pomocí Spring Security se konfiguruje v XML souboru. Pomocí vytvořeného XML *namespace* lze obsloužit některé běžné praktiky zabezpečení aplikace. Například element `<form-login />` se jednoduše nastavuje autentizace pomocí webového formuláře.

Celý proces zabezpečení běží na principu takzvaného řetězu filtrů, což je posloupnost objektů, kterým prochází klientský požadavek krok za krokem. Tato posloupnost je pečlivě seřazena tak, aby vyhovovala bezpečnostnímu scénáři. Každý z těchto kroků má definovanou implicitní implementaci. Pokud je nutné některý krok přizpůsobit vlastním potřebám, stačí pouze dědit z abstraktního předka daného kroku a v konfiguraci novou implementaci zařadit do daného řetězu. [5]

⁷Způsob autentizace uživatele bez nutnosti tvorby vlastního autentizačního řešení.

2.2.3 Apache Velocity

Velocity je jednoduchý *template engine* založený na jazyku Java, který je schopný odkazovat na Java objekty a pomocí toho je zobrazit v prostém textu, HTML, SQL a tak dále. Použití Velocity umožňuje oddělit zdrojový kód Javy od vlastních webových stránek. To umožňuje web designerům odkazovat na metody definované ve zdrojovém kódu bez nutné znalosti principů programování v Javě. Syntaxi je velice snadné se naučit a implementovat. Do HTML stránek se pouze přidávají odkazy na dynamické informace. V ukázce 2.1 je naznačeno odkazování na objekty pomocí znaku \$. Znakem # se volají jazykové konstrukce nebo předem vytvořená makra. Zbytek je už klasické HTML. Po zpracování *template engine* je klientovi odeslána čistá HTML stránka obohacená o dynamické prvky.

Ukázka 2.1: Velocity Template syntaxe

```
<html>
<body>
Hello $customer.Name!
<table>
#foreach( $mud in $mudsOnSpecial )
  #if ( $customer.hasPurchased($mud) )
    <tr>
      <td>
        $flogger.getPromo( $mud )
      </td>
    </tr>
  #end
#end
</table>
```

Velocity je svobodný software pod záštitou *Apache Software Foundation*. Je významnou alternativou k *JavaServer Pages* technologii nebo PHP. Projekt má širokou podporu vývojářské komunity. Je široce podporován jak Springem, tak i jinými webovými frameworky. Generování webových stránek není jeho jedinou doménou, neboť bývá často používán i při generování emailů. [9]

VelocityTools

Pro usnadnění vývoje v Apache Velocity byla vytvořena sada nástrojů nazývaná VelocityTools. Je to knihovna pomocných tříd, které jsou zejména určeny pro závěrečné změny zobrazovaných dat pomocí Apache Velocity. Jedná se o třídy, které pomáhají například

formátovat datum do zadaného tvaru nebo zaokrouhlovat číselné položky na určený počet míst a podobně.

2.2.4 JUnit

JUnit je nejpopulárnější *framework* pro jednotkové testování programů psaných v jazyce Java. Jednotkové testování je testování malých částí kódu obvykle pouze jednotlivých metod nebo tříd na rozdíl od integračních testů, které testují celé komponenty nebo integraci mezi nimi. Pomocí jednotkového testování lze naprogramovat scénář a očekávaný výsledek metod a tím tak určit jejich správné fungování. Dělají-li se výraznější změny ve zdrojovém kódu, lze pak snadno odhalit případné chyby ve stávajícím zdrojovém kódu opakovaným spuštěním testů. Jednotkové testování je základem TDD agilní programovací techniky, kde se nejdříve napíše testovací scénář, a poté se implementuje samotná metoda.

Pro úspěšnost byl tento framework portován i do dalších jazyků jako jsou třeba C++, PHP nebo Objective-C, tato rodina je pak nazývána jako xUnit⁸. [3]

2.2.5 jQuery

jQuery je malá, rychlá a funkčně obsáhlá JavaScriptová knihovna, která významně usnadňuje skriptování na straně klienta. V současnosti se jedná o nejpopulárnější JavaScriptovou knihovnu používanou více jak 55% nejnavštěvovanějších webových stránek. Jedná se o open source software licencovaný pod MIT licencí. Tvůrcem knihovny je John Resig. V dnešní době je vyvíjen týmem vývojářů vedeným Davem Methvinem. Knihovna plně podporuje nepoužívanější prohlížeče dnešní doby. Pomocí ní lze snadno vytvářet dynamické webové aplikace.

jQuery nabízí řadu funkcí, některé z nich jsou uvedeny v následujícím výčtu:

- výběr DOM elementů založeném na CSS selektorech,
- AJAX,
- manipulace s CSS,
- procházení a změna DOM,
- obsluha událostí,

⁸Původní framework SUnit byl napsán pro jazyk Smalltalk Kentem Beckem.

- tvorba efektů a animací.

Dále poskytuje možnost vytvářet *pluginy* nad jQuery knihovnou. Existuje nespočet různých knihoven. Příkladem může být knihovna jQuery UI.

jQuery UI

jQuery UI je JavaScriptová knihovna vytvořená nad knihovnou jQuery. Zabývá se pokročilými UI elementy jako jsou například nástroje pro výběr data, *progressbary* nebo modální dialogová okna. Dále obsahuje metody pro pokročilou interakci jednotlivých objektů (posouvání, roztahování, selektivní výběr, ...). V neposlední řadě rozšiřuje knihovnu jQuery o širokou škálu grafických efektů. [3]

2.2.6 Ostatní

Quartz

Quartz je plnohodnotná služba pro plánování úkolů. Pomocí Quartz lze spouštět v aplikaci jednotlivé úkoly na základě nastavené periody nebo okamžiku v čase.

Apache Commons

Je sada knihoven užitečných tříd a funkcí, kterou zastřešuje Apache Software Foundation. Snahou je vytvořit a udržovat opakovaně využitelné komponenty, které by měly mít minimum dalších závislých knihoven.

Hibernate Validator

Pod pojmem Hibernate se většinou vybaví knihovna pro objektově relační mapování, nicméně celý projekt zahrnuje daleko více. V tomto případě je to knihovna, která ulehčuje validování jednotlivých objektů. Knihovna je implementací tzv. *JSR 303: Bean Validation* specifikace. Validace fungují na základě anotování členských proměnných doménových tříd.

Joda Time

Knihovna Joda Time je knihovna tříd zabývající se časem. Vznikla kvůli nekvalitně navrženým třídám `java.util.Date` a `java.util.Calendar`⁹. Knihovna také umí snadno pracovat s různými systémy kalendářů nebo s rozdílnými časovými zónami.

⁹Java 8 nabídne úplně nové časové API inspirované právě knihovnou Joda Time

Google Guava

Google Guava poskytuje řadu tříd a funkcí vyvíjenou vývojáři firmy Google, které jsou používány v jejich projektech založených na Javě. Poskytuje nástroje pro ulehčení běžných operací, vylepšuje Java kolekce nebo zahrnuje užitečné funkce jako je například podpora *cachování* nebo *hashování*.

SLF4J

Pomocí SLF4J lze snadno logovat události aplikace. Rozhraní staví na návrhovém vzoru Fasáda (Facade). Je tedy nutné zahrnout do *classpath* jednu z možných implementací jako jsou například knihovny `log4j` nebo `logback`.^[3]

2.3 Programové vybavení

2.3.1 PostgreSQL

PostgreSQL je objektově relační databázový systém (DBMS¹⁰) primárně vyvíjen pro unixové systémy, ale existují i verze pro Windows. Je vydáván pod MIT licenci, tudíž se jedná o *free a open source* software. V současné době je to jedna z mála *free* databází, která může směle konkurovat proprietární databázi Oracle Database společnosti Oracle. Zejména projekt PostgreSQL Plus obsahující navíc také nástroje pro snadnější administraci a škálovatelnost je schopnou alternativou. V tabulce 2.1 jsou uvedeny vybrané limity databázového systému PostgreSQL.

Tabulka 2.1: Limity PostgreSQL DBMS [8]

Limit	Hodnota
Maximální velikost databáze	bez limitu
Maximální velikost tabulky	32 TB
Maximální velikost řádku	1.6 TB
Maximální velikost položky	1 GB
Maximální počet řádků v tabulce	bez limitu
Maximální počet sloupců v tabulce	250 - 1600
Maximální počet indexů pro tabulku	bez limitu

¹⁰V odborné literatuře často uváděn jako systém řízení báze dat (SRDB)

Historie

PostgreSQL byl vyvinut z projektu Ingres vyvíjeným na univerzitě v Berkeley v roce 1985. V té době byl nazýván pouze Postgresem¹¹ a měl vyřešit problémy tehdejších databázových systémů. V roce 1995 byl nahrazen interpret dotazovacího jazyka QUEL interpretem jazyka SQL. Na tento popud byla databáze přejmenována na PostgreSQL. V roce 1997 byla uvolněna první verze nesoucí označení 6.0. Od té doby je tento software vyvíjen skupinou firem a dobrovolnými vývojáři po celém světě.[3]

Vlastnosti

PostgreSQL podporuje většinu SQL standardů. Silně odpovídá ANSI-SQL:2008 standardu zahrnující datové typy INTEGER, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, TIMESTAMP. Také podporuje ukládání velkých textových a binárních dat (BLOB, CLOB). Dále zahrnuje cizí klíče, *subselecty*, pohledy, joiny, triggerů a uložené procedury. Uložené procedury lze psát v mnoha jazycích například v Javě, C/C++, Pythonu nebo ve vlastním PL/pgSQL, který silně připomíná procedurální jazyk PL/SQL firmy Oracle.

PostgreSQL je rovněž bezpečný transakční databázový systém. Je plně ACID kompatibilní. Podporuje několik druhů indexů pomocí algoritmů B-tree, Hash, GiST, GIN uvedených níže.

- vícesloupcové indexy – indexy vytvořené nad více sloupci dohromady,
- unique indexy – indexy zajišťující jedinečnost ve sloupci,
- funkční indexy – indexy na základě výsledku nějaké funkce nebo skalárního výrazu,
- částečné indexy – indexy vytvořené pouze nad určitou částí dat.

Pro zajištění rychlého dotazování při velkém objemu dat lze použít partitioning. Ten je v PostgreSQL implementován pomocí dědičnosti, což je jeden z důvodů, proč je PostgreSQL označován jako objektově orientovaný. Pro většinu uvedených databázových technologií lze vyhledat detailnější informace v práci[1].

Pro práci s tímto databázovým systémem jsou připravena rozhraní pro C, C++, PHP, Qt, ODBC, **JDBC**. Poslední zmiňované rozhraní používá vytvořená aplikace ke komunikaci s databází. Více informací o PostgreSQL a JDBC lze nalézt rovněž v bakalářské práci[1].[8]

¹¹Název vznikl ze slovního spojení post-Ingres.

Nástroje

Primárním ovládacím nástrojem je řádkový program `psql`, pomocí něhož lze zadávat dotazy přímo z konzole nebo ze souborů. Existují i nástroje s grafickým uživatelským rozhraním. Za zmínku stojí desktopový `pgAdmin` nebo webový `phpPgAdmin`. Často nejschůdnějším řešením bývají proprietární nástroje, které obsahují jak administrační prvky, tak i pokročilé datové modelování či generování reportů.

2.3.2 Apache Tomcat

Apache Tomcat je open source webový server a *servlet container* implementující *Java Servlet* a *JavaServer Pages* technologie vytvořené firmou Sun Microsystems v roce 1999. Je kompletně napsaný v jazyce Java. Slouží k publikování webových aplikací v Javě založených na servlet a JSP technologiích. Obsahuje nástroje pro snadnou administraci a správu aplikací, ale i možnost konfigurace pomocí XML konfiguračních souborů. Je to jeden z nejrozšířenějších open source serverů pro Java aplikace. Podporuje JNDI, SSL nebo komunikaci s jiným webovým serverem pomocí AJP protokolu. Skládá se ze tří základních komponent:

1. Catalina
2. Coyote
3. Jasper

Catalina

Catalina je základním prvkem celého Tomcatu. Slouží jako *servlet container*. Pokud startujeme Tomcat, ve skutečnosti startujeme komponentu Catalina. Zpracovává servlety, spravuje práva uživatelů a jejich hesla, loguje události do souborů, nastavuje společné vlastnosti pro všechny aplikace a tak dále.

Coyote

Coyote je komponenta serveru zajišťující komunikaci po HTTP 1.1 protokolu. Coyote naslouchá na předem určeném TCP portu a čeká na příchozí spojení. Příchozí požadavky předává Tomcat *engine* na zpracování a po zpracování odešle klientovi odpověď.

Jasper

Jasper je *engine*, který zpracovává JSP soubory a kompiluje je do Java servletů, kterým rozumí Catalina. Je schopný detekovat změny v JSP souborech a překompilovat je za běhu.[3]

2.3.3 Ostatní

Apache Maven

Maven je software určený pro automatickou správu vytvářené aplikace. Celý projekt stojí na pluginech. Jeho konfiguračním prvkem je XML soubor `pom.xml`. V tomto souboru lze nakonfigurovat například:

- proces buildování projektu a jeho distribuce,
- spouštění jednotkových a integračních testů,
- generování dokumentací,
- prostředí pro oblíbená IDE.

Zásadní funkcí je správa závislostí na knihovny třetích stran. Jednotlivé prvky, někdy označované jako *Artifacts*, jsou identifikovány pomocí `<groupId>` a `<artifactId>`. Maven tyto závislosti automaticky stáhne z centrálního nebo z vlastních repositářů a nainstaluje je.

Apache HTTP Server

Tento server je od roku 1996 nejoblíbenějším webovým serverem na trhu. Je vyvíjen pro širokou škálu platforem od unixových systému po Windows. V minulosti hrál klíčovou roli na vzestupu *World Wide Web*. [3]

Kapitola 3

Architektura aplikace

V rámci projektu GNU Gama bylo vytvořeno několik grafických rozhraní. Většina však byla aplikacemi s nutností instalace a s podporou jen pro vybrané operační systémy. Za zmínku stojí projekty Rocinante a QGama, které od sebe dělí zhruba deset let od jejich obhajoby. Oba projekty jsou napsány v jazyce C++ a využívají grafickou knihovnu Qt. První zmíněný je počinem Ing. Jana Pytla, Ph.D., druhý Ing. Jiřího Nováka, který byl představen v diplomové práci v loňském roce.

Webový systém WebGama představený v následujících kapitolách se snaží být spíše alternativou. V této kapitole budou prakticky popsány některé z vlastností systému s odkazy na teoretické základy v minulé kapitole. V následující kapitole budou představeny návody doplněné ukázkami přímo z prostředí webového systému WebGama.

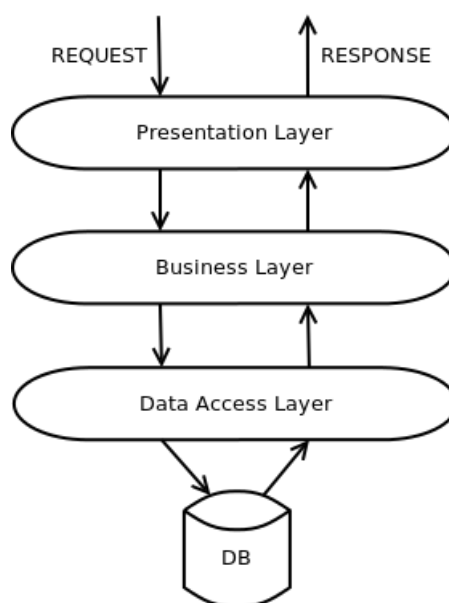
3.1 Úvod do architektury

Systém WebGama je webovou aplikací napsanou v jazyku Java. Aplikace běží v prostředí *servlet containeru* Apache Tomcat. Aplikace využívá Spring Framework, který je postaven na myšlence návrhu vícevrstevné aplikace. Vícevrstevná aplikace není celistvým programem, ale sadou několika vrstev, které spolupracují pouze se sousedními vrstvami skrze rozhraní a o ostatních nevědí nic. V takto navržené aplikaci lze snadno měnit nebo přidávat další vrstvy bez nutnosti přepracování celé aplikace. Klíčovým prvkem je Java rozhraní. Pomocí rozhraní lze skrýt implementace jednotlivých vrstev a tím tak umožnit jejich snadné změny bez ovlivnění dalších částí aplikace.

V návrhu webového systému WebGama byla použita třívrstvá architektura. Jednotlivé vrstvy jsou uvedené v následujícím výčtu:

1. **Perzistentní** – Označovaná též jako datová vrstva. Tato vrstva se stará o ukládání a získání dat z různých typů úložišť. Ve velkých aplikacích bývá úložištěm zpravidla databázový systém, ale je možné data ukládat také například do souborů.
2. **Aplikační** – Často bývá označována anglickým spojením *business logic*. Je to v podstatě jádro celé aplikace. V této vrstvě sídlí aplikační logika. Zajišťuje přístup k datům komunikací s datovou vrstvou a po případném zpracování je odesílá prezentační vrstvě.
3. **Prezentační** – Tato vrstva se stará o interakci s uživatelem často skrze grafická uživatelská prostředí. Může obsahovat validační prvky zadávaných vstupů.

Protože se jedná o webovou aplikaci využívající HTTP protokol, který je postaven na modelu požadavek/odpověď, jednotlivé uživatelské interakce prostupují všemi vrstvami, jak je naznačeno na obrázku 3.1.



Obrázek 3.1: Třívrstvá architektura webové aplikace

Interakce mezi jednotlivými vrstvami je řízena pomocí Spring IOC kontejneru. Podrobnosti se lze dočíst v sekci 3.3 o aplikační vrstvě.

Takto navržená aplikace je základním prvkem pro horizontální a vertikální škálování. Vertikální škálování se typicky týká přidávání hardwarových komponent, například dalších procesorů nebo pamětí. Obecně se tato forma škálování týká pouze jednoho operačního systému. Horizontální škálování obvykle znamená připojení nových výpočetních strojů do

clusteru, které dohromady zvyšují výpočetní výkon. Tato forma škálování typicky zahrnuje více operačních systémů sídlících na samostatných serverech.

3.2 Perzistentní vrstva

Perzistentní vrstva zajišťuje manipulaci s daty. Tato vrstva se skládá z takzvaných DAO objektů. DAO je objekt, který poskytuje abstraktní rozhraní pro manipulaci s daty. Pro uchovávání dat byl vybrán objektově relační databázový systém PostgreSQL¹. Vyšší vrstvy aplikace volají DAO objekty, které poskytují specifické operace nad daty bez nutnosti znalostí detailů o uvedené databázi.

Pro každý doménový objekt (například třída `User`) je zpravidla vytvořen jeden DAO objekt (`UserDao`), který zajišťuje manipulaci s daty příslušné tabulky (`users`) vytvořené v databázi. DAO objekty jsou vytvořeny pomocí modulu Spring JDBC, který usnadňuje práci s rozhraním JDBC. V každém DAO je vytvořena sada metod, které obsluhují jednotlivé operace nad určitou tabulkou databáze. Příklady jednotlivých zpravidla DML operací jsou naznačeny v následujícím výčtu.

- Získávání dat – pomocí SQL příkazu `SELECT`,
- Vkládání dat – pomocí SQL příkazu `INSERT`,
- Mazání dat – pomocí SQL příkazu `DELETE`,
- Úprava dat – pomocí SQL příkazu `UPDATE`.

Získaná data pomocí DAO objektů jsou mapována do doménových objektů, které jsou přístupné ostatním vrstvám aplikace.

3.2.1 Databáze

V databázovém systému PostgreSQL verze 8.4 byla vytvořena sada tabulek. Databázové schéma a souhrnné počty objektů v databázi jsou uvedeny v příloze A. Schéma bylo navrženo do jisté míry dle prvních třech normálních forem. Centrálním prvkem první poloviny je tabulka uživatelů `users`. K této tabulce jsou přidruženy další tabulky, které zajišťují základní funkce týkající se uživatelské správy a s ní spojených vlastností. Příkladem může

¹Oobecné informace o PostgreSQL lze nalézt v sekci 2.3.1.

být tabulka *authorities*, která uchovává informace o uživatelské přístupové roli. Na základě příslušné role může uživatel vykonávat pouze explicitně povolené úkony.

Centrální tabulkou pomyslné druhé části je tabulka výpočtů *calculations*. Na tuto tabulku je napojena vstupní (*inputs*) a výstupní část (*outputs*) výpočtu. Ze vstupní části je vytvořen strom dalších tabulek korespondující se vstupním XML formátem projektu GNU Gama a datovou strukturou aplikace. Schéma bylo navrženo tak, aby bylo možné co nejjednodušší parsovat a znovu sestavovat vstupní GNU Gama XML dávky s ohledem na normalizaci databáze.

Pro urychlení dotazování byla vytvořena sada indexů, které byly vytvořeny zejména nad sloupci s identifikátorem záznamu tabulky, protože bylo selektováno zejména podle těchto sloupců. Dále byla vytvořena funkce v jazyku PL/pgSQL, která mimo jiné automaticky přidává roli nově zaregistrovanému uživateli. K této funkci byly vytvořeny příslušné *triggery*. Databázové schéma ve zdrojových kódech na přiloženém CD v inicializačním souboru `initdb.sql`.

Spojení s databází

Spojení aplikace s databází bylo realizováno pomocí JDBC ovladače databázového systému PostgreSQL s podporou nejnovější JDBC 4 specifikace. Toto spojení bylo automaticky spravováno samotným *servlet containerem* Apache Tomcat. V konfiguračním souboru `server.xml` byl nastaven takzvaný *connection pool* s explicitně zadaným chováním jako je například maximální počet aktivních nebo nečinných spojení. Podrobnosti o *connection pool* jsou uvedeny níže. Dále byly nastaveny parametry spojení, například *connection string* či přístupové údaje do databáze. Takto nastavený zdroj dat (`DataSource`) byl distribuován do kontextu webové aplikace s využitím JNDI. V jednotlivých DAO objektech byl tento `DataSource` využíván.

Coonection Pool

Při přístupu do databáze je obvykle nejvíce časově náročnou akcí navazování spojení. Tento problém řeší *connection pool*, který je vytvořen na základě návrhového vzoru Fond (*Pool*). Jedná se o sadu předem vytvořených konexí, která nejsou po vykonání dotazů uzavřena, ale zůstanou uchována. Při každém dalším přístupu do databáze jsou tyto konexe znovu použity a tím je tak docíleno výrazného zrychlení interakce s databázovým systémem. Je vytvořeno několik implementací *connection poolingu* v Javě. V Apache Tomcat je použita Apache Commons DBCP knihovna.

3.2.2 Spring JDBC

Spring JDBC usnadňuje práci s rozhraním JDBC. Rozhraní JDBC je podrobně popsáno v práci [1]. Spring JDBC odstraňuje nutnost psát dokola opakující se konstrukce. V tabulce 3.1 jsou akce, které jsou automaticky prováděny Spring JDBC abstrakcí, a o které se musí postarat sám vývojář.

Tabulka 3.1: Výhody použití Spring JDBC

Akce	Spring	Vývojář
Definování parametrů spojení s databází		✓
Otevírání spojení	✓	
Tvorba SQL dotazů		✓
Deklarování parametrů a poskytování jejich hodnot		✓
Příprava a spouštění dotazů	✓	
Vytváření cyklů pro procházení výsledkem	✓	
Zpracovávání výjimek	✓	
Manipulování s transakcemi	✓	
Uzavírání spojení	✓	

Z této tabulky je patrné, že o většinu těchto konstrukcí se postará Spring Framework. Vývojář je odpovědný pouze za nadefinování SQL dotazů a tím, jak bude s výsledkem naloženo. Základním prvkem Spring JDBC je objekt typu `JdbcTemplate`. [4]

JdbcTemplate

Je vytvořený podle návrhového vzoru Šablona (*Template*). Výrazně ulehčuje práci s JDBC. Je nejpopulárnějším způsobem přístupu do databáze modulu Spring JDBC. Stará se o vytváření a uvolňování zdrojů a tím tak pomáhá vyvarovat se častým chybám jako je neuzavřené spojení. Provádí základní operace s databází jako je dotazování, aktualizování záznamů nebo spouštění procedur přes sadu členských metod. Na ukázce 3.1 je zobrazen příklad jedné z členských metod třídy `JdbcTemplate`.

Ukázka 3.1: Příklad užití typu `JdbcTemplate`

```
public User findUserByUsername(String username) {  
  
    String sql = "SELECT * FROM users WHERE username = ?";  
  
    User user = getJdbcTemplate().queryForObject(sql,  
        new Object[] { username }, new UserMapper());  
}
```

```
    return user;  
}
```

V ukázce je metoda, která hledá uživatele na základě jeho uživatelského jména. Je pouze nutné napsat SQL dotaz ve formě `PreparedStatement`². Jak je z uvedené ukázky patrné, stačí pouze zavolat jednu z členských metod třídy `JdbcTemplate` a poskytnout parametry. Dále je potřeba zmínit objekt typu `UserMapper`, který implementuje rozhraní `RowMapper`. Tento objekt pomáhá mapovat jednotlivé atributy výsledných záznamů do členských proměnných doménových objektů aplikace.

Transakce

Protože se jednotlivé úkony dějí v aplikační vrstvě, také transakce jsou vymezeny v této vrstvě. Metody aplikační vrstvy, které využívají DAO objekty jsou označeny anotací `@Transactional`. Spring se pomocí nakonfigurovaného správce transakcí postará, aby jednotlivé úkony byly zahrnuty do stejné transakce. Tento způsob deklarace transakcí v aplikační vrstvě oproti deklaraci v perzistentní vrstvě je výhodný zejména v případech, kdy zápis do databáze je podmíněný jiným úkonem v téže metodě. Příkladem může být metoda, která registruje uživatele, obsahující navíc také oznámení po emailu. Pokud z jakéhokoliv důvodu selže odeslání emailu, nový uživatel není do databáze zapsán, protože metoda je brána jako jedna transakce a je tedy atomická.

3.3 Aplikační vrstva

Celá aplikace stojí na Spring IOC kontejneru, který byl ve zkratce představen v minulé kapitole. Tento kontejner reprezentuje rozhraní `ApplicationContext`, které je zodpovědné za instanciaci a konfiguraci objektů, které se nazývají *bean*³. Kontejner má informace jaké objekty instanciovat, případně konfigurovat čtením konfiguračních metadat. Tato metadata mohou být reprezentována pomocí XML, Java anotací nebo samotným Java kódem. Jinými slovy v těchto konfiguračních souborech lze explicitně nastavit vzájemné závislosti mezi objekty pro každý objekt zvlášť.

²Informace o předpřipravených dotazech lze nalézt v práci [1].

³Název nejspíše pochází z anglického slova *coffee beans*, které znamená v překladu kávové boby. Zde si lze všimnout abstrakce: Java (program) = káva, beans = kávové boby (objekty), ze kterých je káva vytvořena.

Z důvodu větší přehlednosti byla většina beanů konfigurována pomocí XML. Příklad XML bean konfiguračního souboru je na ukázce 3.2.

Ukázka 3.2: Příklad XML bean konfiguračního souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">

  <bean id="userManager"
    class="cz.cvut.fsv.webgama.service.impl.UserManagerImpl">
    <property name="userDao" ref="userDao" />
    <property name="authorityDao" ref="authorityDao" />
    <property name="mailManager" ref="mailManager" />
  </bean>

  <bean id="mailManager"
    class="cz.cvut.fsv.webgama.service.impl.MailManagerImpl">
    <property name="addressFrom" value="{mail.address.from}" />
    <property name="mailSender" ref="mailSender" />
    <property name="userDao" ref="userDao" />
  </bean>

  <bean id="userDao" class="cz.cvut.fsv.webgama.dao.jdbc.JdbcUserDao">
    <property name="dataSource" ref="dataSource" />
  </bean>

  <bean id="authorityDao"
    class="cz.cvut.fsv.webgama.dao.jdbc.JdbcAuthorityDao">
    <property name="dataSource" ref="dataSource" />
    <property name="roleDao" ref="roleDao" />
    <property name="userDao" ref="userDao" />
  </bean>

</beans>
```

Tato ukázka je pouze ilustrační, konkrétní konfigurační soubory jsou daleko delší a bývají dost často rozděleny do několika souborů se společnou vlastností. Na ukázce jsou patrné vazby mezi objekty. Atribut `id` je identifikátorem objektu v rámci aplikačního kontextu a atribut `class` je konkrétní třída. Element `property` odkazuje na členskou proměnnou se

jménem v atributu `name` a atribut `ref` odkazuje na identifikátor jiné beanu z aplikačního kontextu. Například třída `UserManagerImpl`, která je implementací rozhraní pro správu uživatelů, obsahuje odkazy na DAO objekty a správce emailové služby. Všechny beanu jsou singletony, pokud není explicitně uvedeno jinak. Sám IOC kontejner efektivně dosazuje potřebné závislosti do jednotlivých beanů.

V novějších verzích Spring Frameworku byly představeny nové způsoby konfigurace beanů. Příkladem je v prezentační vrstvě použitá konfigurace pomocí anotací. Jednotlivé třídy jsou oannotovány speciálními anotacemi. Spring skenuje zadané balíčky a pomocí těchto anotací rozezná vazby mezi objekty.

Diagram vazeb jednotlivých beanů je v příloze C.

3.3.1 Parsování GNU Gama XML

Výměnným formátem mezi webovou aplikací WebGama a projektem GNU Gama je jeho vstupní XML dávka. Krátký příklad vstupní XML dávky je v ukázce 1.1. Pomocí této dávky jsou distribuována data ze vstupních formulářů webové aplikace do programu `gama-local`, který posléze zadanou síť vyrovná. Předávání vstupních dat je zajištěno předáváním vytvořené XML vstupní dávky programu `gama-local`, který je spuštěn v externím procesu. Bližší informace o spouštění externích procesů z Java aplikace je v následující podsekcí.

Všechny stavy GNU Gama vstupní XML dávky musely být rozparsovány a uloženy do vytvořené datové struktury. Datová struktura vstupní části výpočtů vyrovnání byla inspirována DTD souborem `gama-local.dtd` nalezeném v distribuci projektu GNU Gama. Pokud bylo měření zadáno pomocí webových formulářů, musela být naopak vstupní dávka patřičně sestavena a odeslána k vyrovnání.

Pro parsování XML dávky byla zvolena technologie Java StAX. Jako další možnosti se nabízely technologie SAX nebo DOM. Technologie DOM načte celé XML do paměti, což je při velkých dávkách velice neefektivní. Oproti tomu SAX i StAX jsou parsery streamové. Java StAX má několik výhod oproti technologii SAX:

- Novější technologie – StAX je považována za nástupce technologie SAX.
- Snadnější práce s *pull* parserem — SAX je *push* parserem, kde *handler* je volán parserem.
- Možnost zpětného sestavování XML – SAX parser umí pouze číst XML, v aplikaci je nutné XML také sestavovat.

- Výběr ze dvou odlišných API – StAX umožňuje parsovat XML buď pomocí Iterator API nebo Cursor API.

Pro rozhraní StAX existuje několik implementací. V aplikaci byla použita implementace firmy Oracle zahrnutá v balíčku `javax.xml`. Dále bylo pro parsování vybráno Iterator API, které je ve většině případů doporučovaným řešením. Tento způsob dělí XML na takzvané události (*events*). Příkladem události může být například počátek elementu nebo znaky uvnitř elementu. Parser prochází jednu událost po druhé a rozřazuje data do správných objektů.

Ve webové aplikaci byly vytvořeny dvě metody, kde první z nich XML rozkládá (*parsing*) na objekty. Tato metoda byla použita při importu již vytvořených dávek do systému WebGama. Druhá metoda naopak XML dávku sestavuje z vytvořené datové struktury. Dávka je pak odesílána programu `gama-local`, který se postará o vyrovnaní a vrátí výsledek vyrovnaní.

3.3.2 Spouštění externích procesů

Komunikace mezi webovou aplikací a programem pro vyrovnaní místních geodetických sítí `gama-local` probíhá na základě spouštění tohoto programu v externím procesu. Pro tuto příležitost má Java připravenou třídu typu `Process`.

Nejprve je nutné sestavit seznam uživatelem zadaných parametrů spouštěného výpočtu. Tyto parametry jsou reprezentovány přepínači jako je například přepínač `--language`, který nastavuje jazyk výstupu. Kromě uživatelem vybraných parametrů jsou automaticky přidány přepínače zajišťující výstup ve všech dosud podporovaných formátech. Aktuálně podporované výstupní formáty jsou uvedeny v části 1.2.4. Program `gama-local` tak při každém spuštění vyrovnaní vytvoří několik dočasných souborů s výstupy v uvedených formátech.

Takto vytvořený seznam parametrů programu je připojen za cestu ke GNU Gama programu `gama-local` a cestu ke vstupní XML dávce. Pomocí objektu `ProcessBuilder`, který v konstruktoru přebírá vytvořený seznam parametrů, je sestaven proces připravený ke spuštění. Nejprve je nadefinován pracovní adresář (v našem případě adresář `/tmp`), ve kterém jsou prováděny všechny operace. Poté je možné spustit připravený proces. Po spuštění jsou otevřeny vstupní a chybový proud, které čekají na výstup z programu. Po dokončení vyrovnaní programem `gama-local` je očekávána návratová hodnota, podle které je rozhodnuto, jak bude s výsledkem naloženo. Pokud se při vyrovnaní neobjeví chyba,

nastaví se výpočtu příznak „vyrovnáno“ a výsledky výstupních souborů jsou uloženy do databáze.

3.3.3 Automaticky spouštěné procedury

S předchozí části je patrné, že je v některých případech nutné vykonávat části kódu opakovaně v zadaných intervalech. Tyto úkony často slouží zejména k údržbě prostředí aplikace a jsou spouštěny přímo z aplikace a nikoliv z externích programů typu `cron`. V následujícím výčtu jsou uvedeny konkrétní úkony aplikace WebGama, které jsou spouštěny automaticky v zadané době.

- promazávání dočasných souborů vytvořených při práci s programem `gama-local`,
- promazávání neaktivních uživatelů, kteří se neprokázali platnou emailovou adresou,
- promazávání starých záznamů sloužících pro verifikování platných emailových adres.

Quartz Scheduler

Výše uvedené úkony jsou automaticky spouštěny pomocí open source knihovny Quartz Scheduler (zkráceně Quartz) vydávané pod Apache licenci. Tato knihovna byla vybrána díky nativní podpoře Spring Frameworkem. Quartz je vyvíjen komerční firmou Terracota a je vhodný jak pro malé aplikace, tak pro velká serverová řešení. Zavádí základní pojmy úloha (*job*) a spoušť (*trigger*). *Job* je spustitelný úkol, který může být naplánován, zatímco *trigger* poskytuje plán, kdy je tento *job* spuštěn. Tím, že jsou tyto dva objekty oddělené, je možné naplánovat pro jednu úlohu několik *triggerů*.

Nastavení plánování uvedených úloh nevyžadovalo žádné programátorské úsilí. Díky podpoře poskytované Springem stačilo pouze vytvořit *jobs* registrováním vykonávajících metod ve Spring XML konfiguračních souborech. Poté byly vytvořeny *triggery* se zadaným plánem pomocí `cron` syntaxe a nakonec byly *triggery* zaregistrovány v objektu typu Scheduler pomocí Spring abstrakce. Konkrétní plány uvedených úloh jsou uvedeny v konfiguračním souboru aplikace v příloze B.

3.3.4 Emailová služba

Pro resetování zapomenutého hesla a dalších funkcí byla vytvořena malá emailová služba. Služba opět využívá abstrakce poskytované frameworkem Spring. Nejprve je nutné zahrnout

JavaMail API do *classpath*, které je součástí edice JavaEE. Pomocí tohoto API lze odesílat emailové zprávy přímo z kódu aplikace. V následujícím výčtu jsou uvedeny konkrétní zprávy automaticky odesílané z aplikace na adresu uvedenou při registraci.

1. potvrzení platnosti emailové adresy,
2. vyžádané resetování hesla,
3. vyžádaná upomínka zapomenutého uživatelského jména.

3.4 Prezentační vrstva

Prezentační vrstva aplikace stojí na modulu Spring MVC a generování HTML stránek pomocí Apache Velocity. Teoretické základy výše uvedených technologií jsou uvedeny v kapitole 2. Celá prezentační vrstva je postavena na architektuře *Model-View-Controller*. Tato architektura bývá někdy řazena mezi návrhové vzory. Odděluje datový model, uživatelské rozhraní a aplikační logiku. Uživatelské rozhraní (*View*) je zajištěno projektem Apache Velocity. Aplikační logiku (*Controller*) obstarávají controllery modulu Spring MVC, které volají manažery z níže postavené aplikační vrstvy. Datový model (*Model*) prostupuje všemi vrstvami aplikace.

3.4.1 Spring MVC

V dnešní době je jedním z nejpoužívanějších webových frameworků ve světě Java. Poskytuje širokou škálu funkcionality a podpory dalších technologií. Tato technologie byla vybrána hned z několika důvodů. Hlavním důvodem byla nativní podpora IOC. Dalším důvodem byl idiom MVC. V neposlední řadě je nutné zmínit podporu výstupu v několika různých formátech (HTML, XML, ...), zvláště pak ve formátu JSON, který se hojně využívá v technologii AJAX.

Celá technologie je založena na MVC, schéma technologie je na obrázku 2.2. Logika prezentační vrstvy se odehrává v *controllerech*. Spring pozná, že je třída *controllerem*, pokud je označena anotací `@Controller`. Tyto třídy obsahují zpravidla jednu nebo více metod, které mapují jednotlivé požadavky (*HTTP requests*) pomocí anotace `@RequestMapping`. Příklad této metody je na ukázce 3.3.

Ukázka 3.3: Příklad POST metody Controlleru

```
@RequestMapping(value = { "/personal" }, method = RequestMethod.POST)
public ModelAndView modifyUser(@Valid @ModelAttribute("user")
    UserForm userForm, BindingResult result, HttpServletRequest request) {

    String username = request.getUserPrincipal().getName();
    userForm.setUsername(username);

    if (result.hasErrors()) {
        return new ModelAndView("/account/personal/personal");
    }

    userManager.updateUser(userForm);

    activityManager.recordActivity(username, "activity.user.changed");
    return new ModelAndView("/account/personal/personal", "success", true);
}
```

Z ukázky je patrné, že se jedná o metodu mapující HTTP POST *request*, který modifikuje osobní informace uživatele na základě odeslaných parametrů z formuláře. Stěžejním objektem je objekt typu `ModelAndView`. Tento objekt může být naplněn daty (*Model*), které jsou delegovány spolu s definicí cesty k souboru s HTML šablonou (*View*)⁴, kde je vytvořena HTML stránka obohacená o data. `ModelAndView` není jedinou třídou, kterou může controller vrátet. Další možnosti jsou uvedeny v dokumentaci[4].

Pokud je metoda označena anotací `@ResponseBody` a v *classpath* projektu je zahrnuta správná závislost, může controller vrátet jakýkoliv objekt, který je posléze převeden například do tvaru formátu JSON. Tento způsob je vhodným a zároveň poměrně rychlým řešením AJAX volání, protože jsou odeslána pouze čistá data bez okolního HTML formátování a podobně.

REST

Celá aplikace je založena částečně na architektuře REST. Cílem REST je vytvořit rozhraní, které mimo jiné usnadňuje manipulaci s daty nebo zvyšuje bezpečnost. Tato architektura je nezávislá na protokolu, nejčastěji je používána v rámci protokolu HTTP. Dle této architektury by měly být všechny zdroje jednoznačně identifikovány pomocí URI a mělo by být možné s nimi manipulovat pomocí CRUD metod. V našem případě lze jednotlivé zdroje identifikovat pomocí adresy URL a manipulovat pomocí HTTP GET a

⁴V našem případě se jedná o soubor `personal.vm`, což je soubor v jazyce *Velocity Template Language*.

POST metod.

3.4.2 Apache Velocity

Standardní technologií pro vývoj webových aplikací v Javě je technologie JavaServer Pages, zkráceně JSP. Spring MVC podporuje několik technologií pro generování výstupu (JSP, Tiles, Velocity, FreeMarker, ...). Výběr padl na projekt Apache Velocity. V následujícím výčtu jsou uvedeny jeho silné stránky.

- striktní oddělení logiky od šablon,
- rychlost zpracování,
- profesionální komunita vývojářů,
- snadné osvojení technologie,
- snáze pochopitelné pro webové designéry než JSP.

V aplikaci WebGama byl tento projekt použit pro generování výstupných HTML stránek. Byla vytvořena sada HTML souborů (šablon) obohacených o prvky jazyka VTL⁵. Tyto soubory byly zpravidla umístěny do adresářů korespondujících s obsluhovanou URL adresou webové aplikace. Na základě definování cesty v controlleru byl vybrán správný HTML soubor, který byl interpretován Velocity *template engine*m a odeslán klientovi jako výsledná HTML stránka.

Zmíněný Jazyk VTL je velice jednoduchý. Syntaxí trochu připomíná jazyk Java. Pomocí tohoto jazyka byla doplněna případně modifikována data přímo v HTML souborech. Tento jazyk podporuje základní konstrukce známé z jiných jazyků jako jsou podmínky nebo cykly. Není tedy problém dosáhnout kýžených výsledků na základě definování pravidel přímo v šabloně. Příklad syntaxe lze nalézt v ukázce 2.1.

3.4.3 Uživatelské prostředí

Grafické uživatelské rozhraní aplikace bylo upraveno pomocí kaskádových stylů. Dynamické prvky rozhraní byly vytvořeny pomocí skriptů jazyka JavaScript.

⁵Velocity lze použít i například pro generování dynamických XML nebo SQL skriptů. Lze jej tedy použít i v jiném než jen v jazyce HTML

Styl

Pomocí pravidel jazyka CSS byl vytvořen jednotný layout pro celou aplikaci. Veškerá pravidla byla vytvořena speciálně pro tento projekt. Nebyl použit žádný z dostupných CSS frameworků. Barevné schéma bylo částečně inspirováno oficiálními barvami ČVUT. Při vývoji byl kladen důraz na kompatibilitu s dnešními nejčastěji používanými prohlížeči. Tato skutečnost je však velice časově náročná, proto se v některých (zejména v těch starších) prohlížečích mohou objevovat drobné vizuální rozdíly. V aplikaci byly použity prvky z nové specifikace CSS3 jako jsou stíny nebo gradienty.

Pro projekt bylo vytvořeno vlastní grafické logo a také sada ikon pro názornější zadávání jednotlivých parametrů měření.

Skriptování

Samotné HTML a CSS by nenavozovalo dostatečně dojem, že se jedná o plně dynamickou webovou aplikaci. Za plně dynamickou webovou aplikaci lze pokládat aplikaci s podporou skriptování na straně klienta. V aplikaci WebGama toho bylo docíleno přidáním skriptů v jazyce JavaScript. Vývoj byl do jisté míry inspirován přístupem *Unobstrusive JavaScript*, což je způsob skriptování, kde je separována funkcionálnost napsána v JavaScriptu od obsahu stránek v HTML. Takto vyvíjené stránky jsou přehlednější a tím tak snadněji udržitelné.

Většina vytvořených skriptů využívá funkce JavaScriptové knihovny jQuery. Základní informace o této knihovně lze nalézt v kapitole 2. Celá aplikace je závislá na použití JavaScriptu. Pokud by byla podpora JavaScriptu vypnuta, většina vlastností aplikace by byla nefunkční. Příkladem může být interaktivní zadávání nových vyrovnaní, kde jsou skrze toolbar interaktivně přidávána nebo odebrána nová měření a parametry měřené lokální sítě. To způsobuje velké problémy při interpretaci odeslaných dat serverové části aplikace, protože jsou jednotlivé elementy reprezentovány svým identifikátorem. Tyto identifikátory musely být při každém odebrání nějakého prvku přepočítány, aby si udržely vnitřní konzistenci. Na to byly použity takzvané *traversing* funkce knihovny jQuery, které dokáží filtrovat objekty v rámci HTML DOM stromu.

Další hojně využívanou funkcí byla funkce `ajax()`, která je základem všech AJAX požadavků knihovny jQuery. Zde je možné uvést jako příklad správce vyrovnaní, kde jeho veškerá interakce se serverem je prováděna přes AJAXová volání. Výhody toho přístupu jsou zejména v rychlosti zpracování a uživatelsky příjemnější interakci.

V aplikaci byly použity dva pluginy knihovny jQuery. Prvním je knihovna SlideJS, pomocí které je vytvořena slideshow na úvodní stránce. Druhým z pluginů je knihovna

jQuery UI. Pomocí této knihovny byly vytvořeny některé ovládací prvky aplikace jako je třeba posouvání toolbaru či efekt „tahací harmoniky“ ve správci vyrovnaní.

3.4.4 Zabezpečení

Aplikace je zabezpečena pomocí knihovny Spring Security, více informací o knihovně lze nalézt v 2.2.2. Podstatná část aplikace je chráněná proti neautorizovanému vstupu. Pokud chce klient využívat služeb systému WebGama, musí si nejdříve projít registrací. Při registraci zadává povinně uživatelské jméno, heslo a emailovou adresu. Tyto údaje jsou uloženy v databázi. Postup při registraci je popsán v další kapitole.

Celý proces zabezpečení stojí na definování pravidel, která URL jsou povolena pro anonymní vstup uživatelů a pro která je nutné přihlášení. Přihlašování se provádí pomocí webového formuláře. Pro úspěšné přihlášení je nutné znát uživatelské jméno a heslo zvolené při registraci. Jestliže uživatel svoje uživatelské jméno zapomene, je možné si heslo znovu poslat na emailovou adresu zvolenou při registraci. Pokud však uživatel zapomene přihlašovací heslo, je situace složitější. Hesla jsou totiž v databázi šifrována, proto je uživateli vygenerováno nové heslo, které je odesláno na jeho emailovou adresu. Je na uživateli, zda si svoje nové heslo po přihlášení změní či nikoliv.

Šifrování hesla

Při registraci je zadané heslo zašifrováno pomocí jednocestné hašovací funkce SHA-256 s náhodně vygenerovanou solí (*salt*)⁶. Takto zašifrované heslo je uloženo do databáze. Při použití pouhé hašovací funkce je pro jeden řetězec pouze jeden výstupný haš. Útočník, který získá jedno heslo, může určit i ostatní sestavením takzvané *rainbow table*. Proto se přidává náhodná sůl, která zajistí, že hašované heslo je při každém průchodu hašovací funkcí rozdílné.

Funkce *Remember-Me*

Další implementovanou funkcí, kterou poskytuje knihovna Spring Security, je funkce *Remember-Me*. Uživatel se může přihlásit se zaškrtnutou položkou „Zůstat přihlášen“. Při příští návštěvě systému již nemusí zadávat přihlašovací údaje a je automaticky přihlášen. Tato funkcionality se docílí tím, že si WebGama uloží vygenerovaný haš u sebe na serveru i

⁶Kryptografická sůl je několik náhodných bitů, které slouží jako doplněk k jednocestné funkci, aby její výstup měl mnoho možných variant.

u klienta ve formě *cookies*. *Cookies* jsou malé části dat, které server posílá prohlížeči, který si je uloží. Při každé další návštěvě je prohlížeč posílá zpět serveru. Pokud se uložená data shodují, není nutné po uživateli chtít opětovné přihlášení.

Uživatelské role

Systém WebGama je navrhnout tak, aby podporoval různé uživatelské role s rozdílnými přístupovými právy. V současné době podporuje pouze roli klasického uživatele, roli administrátora s neomezenými právy a anonymní přístup do veřejné části systému.

3.4.5 Lokalizace

WebGama je lokalizována do dvou jazyků. Defaultním jazykem celého systému je anglický jazyk, podporován je i český jazyk. Celá lokalizace funguje na implementaci rozhraní `MessageSource`. Spring poskytuje vlastní implementaci toho rozhraní. Tato implementace poskytuje možnost přepnutí lokalizace bez nutnosti restartovat celou aplikaci.

Celá lokalizace stojí na takzvaných lokalizačních *bundlech*, což jsou textové soubory, kde jsou v páru identifikační kód a příslušný lokalizační řetězec. Pro každou jazykovou mutaci je vytvořen jiný soubor se stejnými kódy, ale s odlišným překladem řetězců. Příklad formátu zápisu dvou lokalizačních *bundleů* je v ukázce 3.4.

Ukázka 3.4: Příklad dvou lokalizačních *bundleů*

```
# messages.properties
user.password=Password
user.password.confirm=Confirm password
user.street=Street

# messages_cs_CZ.properties
user.password=Heslo
user.password.confirm=Potvrdit heslo
user.street=Ulice
```

Tyto řetězce jsou pak získávány pomocí metody `MessageSource.getMessage()` na základě uvedeného identifikačního kódu a příslušného lokalizačního identifikátoru (pro češtinu to je `cs_CZ`). Tato metoda je zpravidla volána v šablonách Velocity *template engine* pomocí připravených maker.

Aplikace je nakonfigurována tak, aby rozpoznala lokalizaci operačního systému, kterou pak implicitně nastaví celé webové aplikaci. Pokud však tato lokalizace není podporována,

je nastavena základní lokalizace, kterou je angličtina. Kdykoliv je možné lokalizaci explicitně změnit. Při příští návštěvě je nastavena poslední vybraná lokalizace. Tato funkčnost je dostupná pouze v případě, že je v prohlížeči povoleno ukládání cookies.

3.4.6 Validace

Pro zajištění konzistence v datech jsou jednotlivé formulářové elementy validovány. Tyto validace zajišťují, aby do systému a jeho výpočtů nevstupovaly chybné hodnoty. V aplikaci jsou použity dvě technologie plně podporované Springem.

- rozhraní `Spring Validator`
- *JSR-303 Bean Validation*

Jednotlivé technologie se shodně volají v `Controlleru` pomocí anotace `@Valid`. Příklad použití je uveden v ukázce 3.3. *Controller* tak pozná, zda jsou vstupní hodnoty v souladu s validačními pravidly. V opačném případě uživateli vrátí formulář k opravě s vyznačenými chybami.

Spring Validator

Při použití této technologie je pouze nutné implementovat rozhraní `Validator` z balíčku `org.springframework.validation`. Toto rozhraní obsahuje dvě základní metody:

- `supports(Class)` – testuje, zda tento `Validator` může validovat třídu uvedenou v argumentu metody,
- `validate(Object, Errors)` – metoda, kde se nachází logika samotné validace.

Implementováním těchto metod je vytvořen příslušný `Validator`, který lze pak volat v *Controlleru*, který obsluhuje příslušnou POST metodu. Příklad jednoduchého `Validatoru` je v ukázce 3.5.

Ukázka 3.5: Implementace Spring Validator

```
public class UploadValidator implements Validator {  
  
    @Override  
    public boolean supports(Class<?> clazz) {  
        return UploadForm.class.isAssignableFrom(clazz);  
    }  
}
```

```
}

@Override
public void validate(Object target, Errors errors) {

    UploadForm uploadForm = (UploadForm) target;

    // check if file was added
    if (uploadForm.getFile().isEmpty()) {
        errors.rejectValue("file", "Empty", "no file was chosen");
    }

    // check if file is smaller than 10MB
    if (uploadForm.getFile().getSize() > 10485760L) {
        errors.rejectValue("file", "Larger", "file is larger than 10MB");
    }
}
}
```

V této ukázce je validován formulářový objekt `UploadForm` určený pro upload vstupních dávek. Metoda `validate` obsahuje dvě podmínky. První testuje, zda je vybrán nějaký soubor, a druhá testuje, zda vybraný soubor nepřesahuje 10 MB. V případě selhání validace je objektu typu `Errors` oznámeno jméno chybného pole, chybový kód a implicitní chybová zpráva. Tato zpráva lze jednoduše lokalizovat definováním chybového kódu a pole v lokalizačním souboru. Více o lokalizaci lze nalézt v předešlé sekci.

JSR-303

Předešlý způsob je komplexním řešením definování jednoduchých i složitých validací. Pro každý objekt je zvykem definovat vlastní `Validator`. To je při velkém počtu objektů velice neproduktivní. Proto byla využita v tomto projektu i varianta validací pomocí tzv. *Bean Validation*. Tato metoda se skládá z pouhého přiřazení příslušné anotace k členské proměnné validovaného objektu. Konfigurací této metody je pouhé přidání knihovny do *classpath*, která tento typ validací implementuje. Spring MVC tuto knihovnu rozpozná a automaticky umožní její podporu. Pro tento projekt byla vybrána knihovna `Hibernate Validator`. Knihovna obsahuje některé základní již implementované validační anotace. V projektu byly tyto předpřipravené anotace použity. Některé z nich jsou uvedeny v následujícím výčtu:

- `@NotBlank` – kontroluje, zda v anotovaném poli není null nebo prázdný řetězec,

- `@Length()` – kontroluje, zda řetězec nepřekročil zadaný počet znaků,
- `@Max()` – kontroluje, zda je v poli číslo a nepřekročí zadanou hodnotu,
- `@Email` – kontroluje, zda je v poli správně naformátována emailová adresa.

Samozřejmě specifikace tohoto způsobu dovoluje definování specifických anotací pro vlastní potřebu a mnoho dalšího. V dnešní době již existuje aktualizovaná specifikace JSR-349 *Bean Validation 1.1*.

3.5 Testování

Testování aplikace bylo provedeno pomocí jednotkového testování. Jednotkové testování je vhodné pro testování funkčnosti malých částí kódu, zpravidla pouhých metod. Tyto testy pak mohou být automaticky spouštěny například při *buildu* aplikace. V ideálním případě by měly být jednotky na sobě nezávislé a izolované od okolního prostředí. Z tohoto důvodu se často vytvářejí speciální pomocné objekty, které se nazývají *mock objects*⁷. Jednotkové testování má následující přednosti.

- Možnost odhalení chyb již při psaní funkčního kódu.
- Větší pravděpodobnost odhalení chyby při změnách logiky nebo refaktORIZACI kódu.
- Zjednodušená tvorba integračních testů.
- Může být bráno jako druh dokumentace, vývojáři mohou snadněji poznat chování aplikace z testovacích scénářů.

Ve webové aplikaci byl pro jednotkové testování použit framework JUnit. Základní informace o této knihovně lze nalézt v sekci 2.2.4.

3.5.1 JUnit

JUnit ve verzi 4.11 byl použit z důvodu jeho silné podpory Spring Frameworkem a snadné integraci do jeho testovacího kontextu. Tento testovací framework ve verzi 4.x využívá anotací, které značí testovací metody. Protože je předpokládáno, že jednotlivé metody jsou nezávislé, může být pořadí vykonání metod libovolné. Vybrané anotace a jejich účel jsou uvedené v tabulce 3.2.

⁷*Mock object* je pomocný objekt, který imituje pravou instanci třídy v případě, že je tuto instanci složité sestavit nebo pracuje-li s externími zdroji (databáze, HTTP, ...) a tak dále.

Tabulka 3.2: Dostupné anotace frameworku JUnit 4.x

Anotace	Popis
@Test	Označuje testovací metodu.
@Before	Tato metoda je spouštěna před každou testovací metodou. Takto oannotovaná metoda je často využívána při načítání dat nebo inicializacích.
@After	Tato metoda je spouštěna po každé testovací metodě. Takto oannotovaná metoda se používá například při mazání dočasných dat.
@BeforeClass	Tato statická metoda je spuštěna pouze jednou před všemi testy. Je používána při časově náročných operacích jako je například otevírání databázových spojení.
@AfterClass	Tato statická metoda je spuštěna pouze jednou po dokončení všech testovacích metod. Je používána například při odpojení z databáze.
@Ignore	Tato metoda je přeskočena a není zahrnuta do výsledku.

V testovacích metodách jsou pak volány statické metody třídy `Assert`, které porovnávají aktuální hodnotu s očekávanou hodnotou. Jména těchto metod typicky začínají slovem „assert“. Příklady těchto metod mohou být například `assertTrue`, `assertEquals` nebo `assertNotNull`. Pokud se aktuální a očekávaná hodnota liší, je vyhozena výjimka `AssertionException`.

Příklad jednoduchého jednotkového testu je na ukázce 3.6, kde je testována metoda, která se stará o mazání dočasných souborů.

Ukázka 3.6: Jednotkové testování pomocí JUnit

```
public class FileEraserTest {

    private FileEraser fileEraser;

    @Before
    public void setUp() throws Exception {
        fileEraser = new FileEraser();

        for (int i = 0; i < 10; i++) {
            File file = new File("/tmp/"+Generator.generateInputFilename("foo"));
            file.createNewFile();
        }
    }

    @Test
    public void testEraseTemporaryFiles() {
        int i = fileEraser.eraseTemporaryFiles();
    }
}
```

```
        assertEquals(10, i);  
    }  
}
```

Testování bylo spouštěno při buildu celého projektu pomocí Apache Maven. V `pom.xml` byl nakonfigurován plugin *surefire*, který umožňuje spouštět testování pomocí příkazu `mvn test`.

3.5.2 Selenium

Selenium je výborným nástrojem pro testování prezentační vrstvy webové aplikace. Jedná se o soubor několika nástrojů pro spouštění automatizovaných testů uživatelského rozhraní webové aplikace. Tyto nástroje jsou uvedeny v následujícím výčtu.

- **Selenium IDE** – Je plugin pro Firefox. Je to nejjednodušší způsob spouštění testů. Dokáže nahrávat, editovat nebo ladit spouštěné testy.
- **Selenium Remote Control** – Je testovací nástroj, který umožňuje vytvářet testy v několika jazycích včetně Javy. Tento nástroj obsahuje server, který spouští podporované webové prohlížeče.
- **Selenium WebDriver** – Novější přístup k testování než u RC. Není nutné používat server ke spouštění testů.
- **Selenium Grid** – umožňuje spouštět testy současně na několika strojích. Testování je pak rychlejší.

Celý proces testování je založen na definování postupu při práci s uživatelským rozhraním. Například při použití nástroje Selenium IDE by mohl být postup takto:

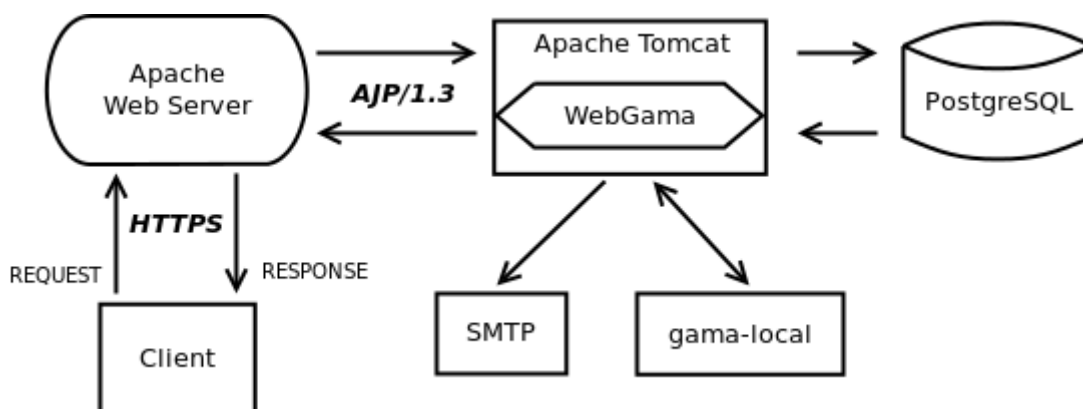
1. otevři stránku s adresou *www.google.com*,
2. napiš řetězec „gama fsv cvut cz“ do elementu s id *gbqfq*,
3. vyber link s textem „Studijní program Geodézie a kartografie – GeoWikiCZ“.

Takto vytvořené scénáře jsou spouštěny i několikrát za sebou. Výsledky testů jsou porovnány s očekávaným chováním aplikace a podle toho je rozhodnuto, zda je test úspěšný či nikoliv.[10]

Bohužel kvůli nedostatku času nemohlo být toto testování zahrnuto do diplomové práce.

3.6 Produkční prostředí

Webová aplikace WebGama běží na jednom ze školních serverů. Konkrétně na serveru geo102 udržovaném katedrou mapování a kartografie fakulty stavební. Na serveru, který disponuje šestnácti výpočetními jádry a 47 gigabajty operační paměti, běží operační systém Debian GNU/Linux verze 6.0 s kódovým označením *squeeze*. Celé prostředí se skládá z několika softwarových součástí naznačených na obrázku 3.2.



Obrázek 3.2: Schéma produkčního prostředí

Schéma popisuje cestu směřující od klientského požadavku až po obdržení odpovědi ze serveru. Protože se na tomto serveru nacházejí další webové projekty běžící na Apache Web Server (dále jen Apache), který naslouchá na standardních portech pro HTTP komunikaci (port 80 pro HTTP a port 443 pro HTTPS), bylo nutné použít tento server pro komunikaci s klientem. Požadavky odeslané na adresu `webgama.fsv.cvut.cz` Apache automaticky směřuje pomocí protokolu AJP/1.3 na spuštěnou instanci serveru Apache Tomcat (dále jen Tomcat), ve kterém běží webová aplikace WebGama. Tomcat požadavek zpracuje a vrací odpověď serveru Apache, který ji odešle klientovi. Toto řešení přináší ještě navíc několik dalších benefitů jako je možnost použití módů dostupných pro Apache nebo možnost vytvoření *clusteru* z několika instancí Tomcatu, kterým Apache rozesílá požadavky dle nastaveného scénáře⁸.

V Tomcatu běžící aplikace dále komunikuje s databázovým systémem PostgreSQL a odesílá emailové zprávy skrze fakultní SMTP server. Na serveru je zkompileována aktuální verze programu `gama-local` projektu GNU Gama. Aplikace jej pro každý výpočet volá

⁸Více informací na <http://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html>

v externím procesu. Podrobnější konfigurace jednotlivých součástí je uvedena v následující podsekcí.

3.6.1 Konfigurace software

Apache Web Server

Z výše uvedených skutečností vyplývá nutnost použití tohoto serveru jako vstupní brány pro zpracování Tomcatem. Nejprve bylo nutné zaregistrovat `webgama.fsv.cvut.cz` doménu ve fakultních DNS systémech, aby ukazovala na IP adresu serveru. Pro tuto doménu byly vytvořeny dva virtuální servery každý pro jeden port. Ve virtuálním serveru pro port 80 bylo nastaveno permanentní přesměrování na adresu: `https://webgama.fsv.cvut.cz`. Tuto adresu obsluhuje druhý virtuální server. Aby bylo možné komunikovat přes SSL vrstvu, musel být vygenerován certifikát pomocí programu `openssl`. Tento certifikát byl podepsán certifikační autoritou TERENA. Certifikát a hierarchie certifikačních autorit byly nahrány na server a namapovány v konfiguraci virtuálního serveru. Pomocí modulu `mod_jk` bylo nastaveno spojení s Tomcatem.

Apache Tomcat

Tomcat byl nastaven tak, aby naslouchal pouze na portu 8009 protokolu AJP. Konektor zajišťující komunikaci po HTTP protokolu byl zakázán. Dále byl vytvořen takzvaný *Connection Pool* skrze JNDI, který si sám Tomcat spravuje podle potřeby a nastavených preferencí. Více informací v sekci o perzistentní vrstvě 3.2.

PostgreSQL

V již používaném databázovém systému byl vytvořena nová databáze jménem `webgama`. Poté byl vytvořen uživatel a heslo s právy čtení a zápisu. V této databázi byly vytvořeny databázové objekty viz sekce 3.2 a databázové schéma A.

SMTP Server

Webová aplikace využívá fakultního SMTP serveru pro rozesílání emailových zpráv na adresy povinně registrované pro každého uživatele. Tyto emailové zprávy jsou určeny například k verifikování identity uživatele nebo pro resetování zapomenutého hesla. Jelikož se server fyzicky nachází v síti fakulty, je možné využít fakultního SMTP serveru

`smtp.fsv.cvut.cz` naslouchajícího na portu 25.

`gama-local`

Jak již bylo uvedeno v první kapitole, `gama-local` je řádkovým programem, který provádí výpočet vyrovnání. V konfiguračním souboru aplikace B je definována cesta k tomuto programu. Aktuální verze programu `gama-local` je označena číslem 1.13f. Více o programu a jeho použití v sekci 1.2.

Kapitola 4

Popis aplikace

Tato kapitola si dala za cíl provést uživatele jednotlivými kroky aplikací od počáteční registrace až po konečný export výsledků vyrovnání sítě.

Byla vyvíjena a testována pro pět nejvíce používaných webových prohlížečů dnešní doby, které jsou uvedeny níže ve výčtu seřazeném sestupně dle aktuálního podílu na trhu. Je více než pravděpodobné, že bude aplikace fungovat i na jiných než uvedených prohlížečích. Pro tyto prohlížeče nebyla aplikace testována a není tedy možné zajistit její správnou funkčnost. Ještě je nutné podotknout, že aplikace je primárně určena pro počítače. Na mobilních zařízeních jako je například mobilní telefon nebo tablet webová aplikace funguje, ale není tak uživatelsky přívětivá.

1. Google Chrome
2. Mozilla Firefox
3. Internet Explorer
4. Safari
5. Opera

Je doporučeno mít uvedené prohlížeče aktualizovány na poslední dostupnou verzi. V předešlých verzích se lze setkat s odlišnostmi v zobrazení některých částí aplikace. To je zapříčiněno nekompatibilitou s aktuálně podporovanou specifikací CSS3.

Pro správné fungování aplikace je nutné mít povolenou podporu skriptování jazyka JavaScript. Většina webových prohlížečů ji povoluje implicitně, v opačném případě aplikace uživatele upozorní na úvodní stránce, že je nutné podporu povolit. Návod, jak si podporu JavaScriptu povolit, lze najít v dokumentaci jednotlivých prohlížečů.

Aplikace je dostupná z webové adresy:

`http://webgama.fsv.cvut.cz/`

Po zadání adresy se objeví úvodní stránka aplikace, která seznamuje uživatele s aplikací a jejími vlastnostmi. Na obrázku 4.1 je zobrazena její horní část.



Obrázek 4.1: Úvodní stránka aplikace WebGama

Vzhled aplikace je vytvořen pomocí fixního layoutu. To znamená, že na všech zařízeních má aplikace konstantní šířku. Je tedy možné vytvořit kvalitní vzhled i pro majitele zařízení s nižším rozlišením. Jako nejmenší bylo bráno v potaz rozlišení 1024x768 obrazových bodů.

Aplikace se skládá z hlavičky, těla aplikace a patičky. Hlavička a patička (není vidět na předešlém obrázku) provází uživatele ve všech částech aplikace. Je připnuta k horní respektive dolní části aktivní plochy prohlížeče. Kliknutím na logo aplikace se uživatel vrátí na hlavní stránku aplikace. Uživatel má možnost také změnit jazyk kliknutím na odkaz „Lokalizace“. Aplikace je plně lokalizována do českého a anglického jazyka.

V první sekci bude popsána registrace nového uživatele a jeho následné přihlášení do systému WebGama.

4.1 Registrace nového uživatele

Aby bylo možné využívat funkcí systému WebGama, je nejprve nutné projít jednoduchou registrací nového uživatele. Na registrační formulář je možné se dostat přes velké tlačítko „Zaregistrovat Zdarma“ uprostřed úvodní stránky nebo kliknutím na odkaz „Registrovat“ v hlavičce aplikace. Formulář je rovněž dostupný z adresy <https://webgama.fsv.cvut.cz/register>. Na obrázku 4.2 je formulář zobrazen.

Registrace

Tučně jsou zvýrazněna požadovaná pole

Uživatelské Informace

Uživatelské jméno: diplomant **E-mail:** diplomant@fsv.cvut.cz

Heslo: ●●●●●●●● **Potvrdit heslo:** ●●●●●●●●

Osobní Informace

Křestní jméno: Příjmení:

Telefon:

Adresa

Ulice: Číslo popisné:

Město: Praha PSČ:

Stát: Česká Republika

Registrovat Reset

Obrázek 4.2: Registrační formulář aplikace WebGama

Ve formuláři je nutné vyplnit první čtyři pole. Uživatel je identifikován svým uživatelským jménem v rámci celé aplikace. Dále je nutné vyplnit platnou emailovou adresu. Aplikace se posléze pokusí zadanou emailovou adresu ověřit zasláním potvrzovací zprávy. Obě hesla se musí shodovat a být alespoň 6 znaků dlouhá. Není nutné si dělat obavy o bezpečnost nebo únik uživatelova hesla. Hesla jsou uložena v zašifrované podobě, takže jej nemohou přečíst ani provozovatelé aplikace. Další pole obsahují osobní informace. Není nutné je, vyplňovat jsou určena pouze ke statistickým účelům. Ještě je nutné zdůraznit, že zadaná data nebudou v žádném případě poskytnuta třetím osobám.

Všechna pole jsou validována. Pokud uživatel zadá chybná data nebo data v chybném formátu, je upozorněn pomocí chybové hlášky. Příklad chybného vstupu je na obrázku 4.3.

The screenshot shows a web form titled "Registrace". At the top, a red error bar states "Formulář obsahuje chyby". Below this, a note says "Tučně jsou zvýrazněna požadovaná pole". The form is divided into two columns under the heading "Uživatelské Informace". The left column contains fields for "Uživatelské jméno:" (filled with "diplomant") and "Heslo:" (empty). The "Heslo:" field has two red error messages: "alespoň 6 znaků" and "hesla se neshodují". The right column contains fields for "E-mail:" (filled with "www.fsv.cvut.cz") and "Potvrdit heslo:" (empty). The "E-mail:" field has a red error message: "chybná emailová adresa". The "Potvrdit heslo:" field has a red error message: "hesla se neshodují".

Obrázek 4.3: Chybný vstup registračního formuláře aplikace WebGama

Pokud uživatel vyplní všechna pole korektně, potvrdí registraci tlačítkem „Registrovat“, je přesměrován na stránku s upozorněním, že jeho účet byl vytvořen. Dále ho upozorní na to, že mu byla odeslána emailová zpráva s potvrzovacím odkazem. Klikne-li na tento odkaz, jeho účet bude aktivován. V opačném případě účet zůstane v neaktivním stavu a po třech dnech je vymazán z databáze.

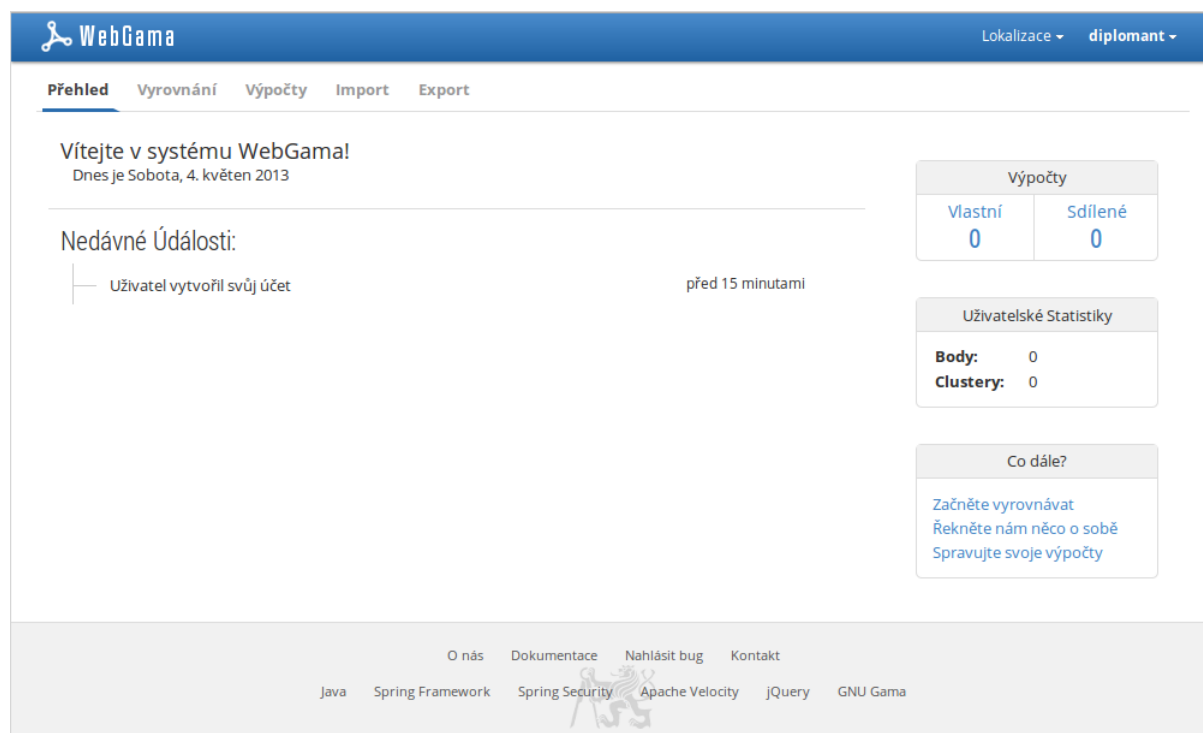
Má-li uživatel účet již aktivní, je možné se přihlásit do systému WebGama pomocí odkazu „Přihlásit“, který je uveden v hlavičce. Ukázka přihlašovacího formuláře je na obrázku 4.4.

The screenshot shows a web form titled "Přihlásit". It contains two input fields: "Uživatelské jméno (zapomenuté uživatelské jméno)" filled with "diplomant" and "Heslo (zapomenuté heslo)" filled with ten dots. Below these fields is a checkbox labeled "Zůstat přihlášen" which is checked. At the bottom is a blue button labeled "Přihlásit". A mouse cursor is pointing at the button, and a yellow tooltip box next to it says "Kliknutím se přihlašte".

Obrázek 4.4: Přihlašovací formulář aplikace WebGama

V přihlašovacím formuláři se vyplňuje uživatelské jméno a heslo. Pokud se přihlašovací

údaje neshodují se záznamy v databázi nebo je uživatelský účet stále neaktivní, je uživatel upozorněn chybovou hláškou. Dále je možné zaškrtnout pole „Zůstat přihlášen“ a při příští návštěvě se již uživatel nemusí znovu přihlašovat. Na přihlašovacím formuláři lze také nalézt odkazy „Zapomenuté uživatelské jméno“ a „Zapomenuté heslo“. Tyto odkazy přesměrují uživatele na příslušné formuláře. Zadá-li uživatel úspěšně přihlašovací údaje, je posléze uvítán na hlavní stránce systému WebGama 4.5.

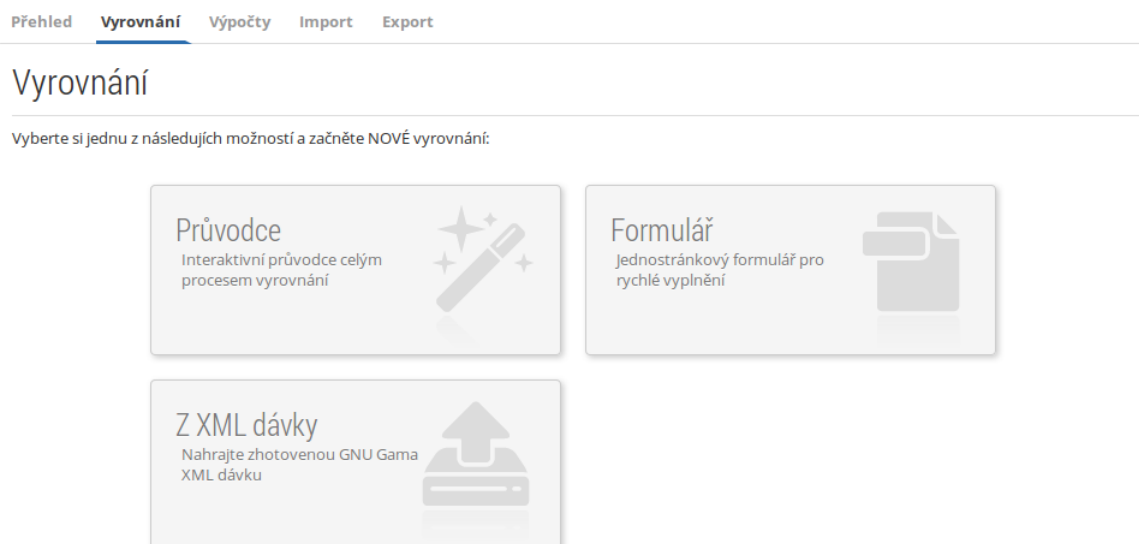


Obrázek 4.5: Hlavní stránka aplikace WebGama

Hlavní stránka systému informuje uživatele o provedených úkonech jako je například vytvoření uživatelského účtu. V levé části je uvedeno počítadlo výpočtů a souhrnné statistiky všech výpočtů. V horní části je ovládací menu. V pravé části hlavičky je zobrazeno uživatelské jméno, po kliknutí na něj se zobrazí nabídka možností. V této nabídce je možné nalézt správu vlastního účtu či možnost odhlášení z aplikace. V následující sekci bude popsáno založení nového vyrovnání.

4.2 Založení nového vyrovnání

Nové vyrovnání lze založit pomocí volby „Vyrovnání“ v ovládacím menu aplikace. Tím se uživatel dostane na stránku s výběrem způsobu zadání nového vyrovnání, jak je naznačeno na obrázku 4.6.



Obrázek 4.6: Možné způsoby zadání nového vyrovnání

Kliknutím na jednu z uvedených dlaždic se uživatel dostane na vybraný formulář. Výběr je z tří možností. Jednou z možností je volba „Formulář“, která přesměruje uživatele na formulář, kde uživatel vyplní všechna data na jedné stránce. Další možností je import již hotové GNU Gama XML vstupní dávky. Poslední možností je interaktivní průvodce, který rozfázuje zadávání jednotlivých parametrů do několika kroků. Tento způsob bude popsán v další části textu.

Průvodce dělí zadávání vyrovnání na čtyři kroky. Nejprve je nutné definovat síť jak je uvedeno na obrázku 4.7. Každá položka obsahuje nápovědu, která je zobrazena najetím myši nad její popis. Tlačítkem „Další“ posune zadávání na další krok.

Dalším krokem je zadání parametrů lokální sítě. Tento krok je zobrazen na obrázku 4.8. V položkách, kde uživatel nevyplnil žádná data, jsou zobrazeny implicitní hodnoty. Uživatel také může navigovat formulář na předešlý krok tlačítkem „Předchozí“. Dosud zadaná data budou zachována.

Následujícím krokem je zadávání bodů, které vstupují do vyrovnání. V tomto kroku

Definice sítě 1/4

Nadefinujte Vaši místní síť.

Tučně jsou zvýrazněna požadovaná pole, pro nápovědu najedte myši na popisek.

Definice sítě

Orientace Os XY: Jih-Západ

Epocha: 0

Úhly: Pravostranné

Pravostranné definuje měřené úhly a/nebo směry proti směru hodinových ručiček, hodnota
Levostranné definuje měřené úhly a/nebo směry po směru hodinových ručiček

Popis sítě

Popis: Ukázkové vyrovnání - květen 2013

Další

Obrázek 4.7: Zadání definice lokální sítě

Parametry sítě 2/4

Parametrizujte Vaši místní síť.

Tučně jsou zvýrazněna požadovaná pole, pro nápovědu najedte myši na popisek.

Parametry sítě

Stř. Chyba Apriorní: 1.0

Hladina Spolehlivosti: 0.95

Tolerance: 1000

Typ Stř. Chyby: Apriorní

Aktualizovat Souřad.: Ne

Volitelné parametry

Stř. Chyba Směrů:

Stř. Chyba Délek: 20

Stř. Chyba Úhlů: 15

Stř. Chyba Zenit. Úhlů:

Předchozí

Další

Obrázek 4.8: Zadání měřených bodů lokální sítě

přichází na řadu *toolbar*, který se schovává na pravé straně okna prohlížeče. Pomocí tohoto *toolbaru* se interaktivně vkládají jednotlivé objekty. Na obrázku 4.9 je naznačen tento krok

spolu s popisovaným *toolbarem*. Kliknutím na tlačítko „Bod“ se přidá na konec seznamu objekt Bod. Do tučně zvýrazněných polí je potom nutné vyplnit příslušné údaje, jinak WebGama nepustí formulář na další případně předchozí krok. Objekty je možné odebrat pomocí křížku v pravém horním rohu objektu. Pokud je uživatel hotov se zadáváním bodů, přistoupí k poslednímu kroku, kterým je zadávání skupin měření (takzvaných clusterů).

Body 3/4

Přidejte všechny body, které vstupují do vyrovnání.

Tučně jsou zvýrazněna požadovaná pole, pro nápovědu najedte myši na popisek.

Body

Toolbar

Bod

Bod: 1

X: 1118103.84 Y: 668559.14 Z: Fix: Adj: XY

Bod: 2

X: 1117697.19 Y: 667132.98 Z: Fix: Adj: XY

Bod: 3

X: 1119159.92 Y: 667054.59 Z: Fix: Adj: XY

Bod: 4

X: 1119260.13 Y: 667932.57 Z: Fix: Adj: xy

Předchozí Další

Obrázek 4.9: Zadání parametrů lokální sítě

Na obrázku 4.10 je zobrazena pouze část poslední kroku. Je možné si povšimnout, že *toolbar* je pohyblivý. Lze jej chytnout za záhlaví a přetáhnout do jakéhokoliv místa aktivní plochy prohlížeče. Pro opětovné ukotvení na původní místo je nutné stisknout křížek umístěný na pravé straně záhlaví.

Zadávání jednotlivých měření do vybraných observací se provádí označením příslušné observace. Označení se provede kliknutím na záhlaví objektu nebo vstupem do jednoho z textových polí objektu. Při označení objektu měření se označí i příslušná observace. Samozřejmostí je odebírání jakéhokoliv objektu. Kliknutím na tlačítko „Potvrdit“ je nakonfigurovaná síť uložena do databáze.

Obrázek 4.10: Zadání clusterů lokální sítě

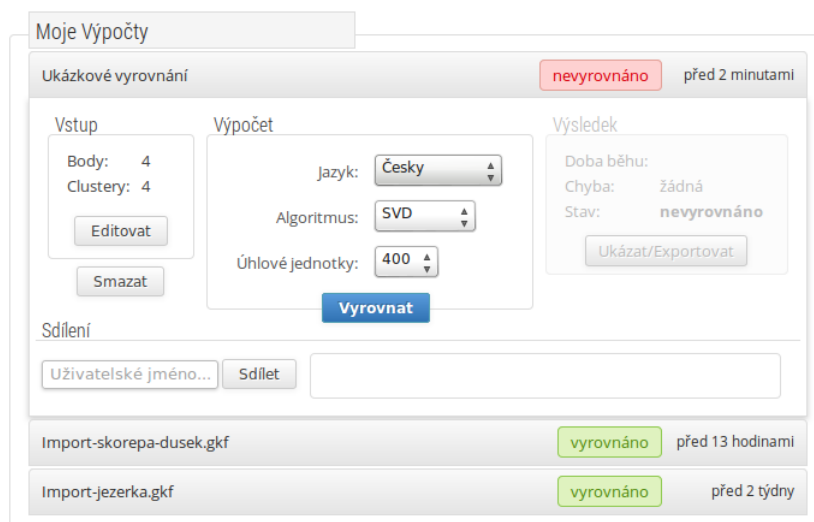
4.3 Správa výpočtů

Po uložení sítě do databáze je uživatel navigován do správce výpočtů. Je možné se do něj dostat i přes hlavní ovládací menu kliknutím na položku „Výpočty“. Z toho je možné spravovat veškeré dosud uložené výpočty uživatele. Obrázek 4.11 ilustruje seznam vlastních výpočtů.

Kliknutím na záhlaví jednotlivých výpočtů se objeví detaily vybraného výpočtu. Nalevo je shrnuta vstupní část výpočtu. Pod počtem bodů a clusterů je tlačítko „Editovat“, pomocí něhož lze jednotlivé sítě měnit. Na výběr je jedna ze dvou dříve uvedených možností. Výpočty lze editovat jak pomocí průvodce, tak i pomocí jednostránkového formuláře. Postup editace je shodný jako v případě založení nového vyrovnání sítě známého z předešlé sekce.

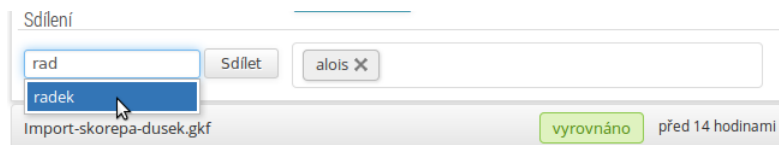
Ve střední části detailu vybraného výpočtu je krátký formulář. Tlačítkem „Vyrovnat“ se odešle vytvořená lokální síť spolu s parametry k výpočtu programem `gama-local` projektu GNU Gama. Lze si vybrat z jedenácti jazyků, do kterých bude výsledek přeložen. Dále je pak možné zvolit algoritmus a úhlové jednotky výsledku. Projde-li výsledek bez

chyby, změní se značka v záhlaví na vyrovnáno a výsledek je připraven k exportu. Pokud však GNU Gama zjistí nějakou chybu, WebGama zobrazí v pravé části červený trojúhelník, pod kterým se zobrazí chybová zpráva. V tomto případě musí uživatel lokální síť upravit.



Obrázek 4.11: Správce výpočtů WebGama

Dalším prvkem je sdílení výpočtů mezi uživateli. Ve spodní části výpočtu je textové pole, do kterého se zapisuje uživatelské jméno uživatele, se kterým chce vlastník sdílet svůj výpočet jak je zobrazeno na obrázku 4.12. Tlačítkem „Sdílet“ přidá vybraného uživatele do seznamu sdílení. Kdykoliv je možné odebrat práva k výpočtu kliknutím na křížek vedle uživatelského jména. Vybranému uživateli se sdílený výpočet objeví v sekci „Sdílené Výpočty“ ve správci výpočtů. Má však pouze omezená práva k výpočtu.



Obrázek 4.12: Sdílení výpočtů WebGama

Pro lepší orientaci ve svých výpočtech je možné si jednotlivé výpočty pojmenovat. Najetím myši na název výpočtu v záhlaví se po pravé straně názvu objeví ikona ve tvaru tužky. Kliknutím na tuto ikonu se objeví dialog, pomocí kterého lze jednoduše změnit název výpočtu.

Kliknutím na tlačítko „Ukázat/Exportovat“ je uživatel přesměrován na stránku s výsledkem vyrovnání. Detaily lze nalézt v následující sekci.

4.4 Export výsledků

Zobrazit výsledek vyrovnání lze buď pomocí tlačítka u vybraného výpočtu ve správci výpočtů nebo pomocí položky „Export“ v hlavním ovládacím menu. Na obrázku 4.13 je tabulka vlastních i sdílených výpočtů po kliknutí na položku „Export“.

Export

Vlastník	Stav	Název	Jazyk	Algoritmus	Čas
diplomant	vyrovnáno	Ukázkové vyrovnání	Česky	svd	2013-05-04 14:54:36
diplomant	nevyrovnáno	Import-prostorovka.gkf	Česky	svd	2013-05-04 14:52:43
alois	vyrovnáno	Lojzovo vyrovnání	Španělsky	cholesky	2013-05-02 08:22:17

Tip: Nevyrovnané výpočty nemohou být exportovány.

Obrázek 4.13: Výběr výpočtů připravených k exportu

Po kliknutí na jeden z řádků je uživatel přesměrován na stránku uvedenou na obrázku 4.14. Na této stránce jsou v horní části uvedeny obecné parametry výpočtu. Ve střední části se nachází textový náhled celého výsledku vyrovnání sítě. Lze jej například využít pro rychlou kontrolu výsledku vyrovnání bez nutnosti stažení výsledku do vlastního počítače. Sekce „Download“ ve spodní části stránky je určena k exportu a stažení výsledku vyrovnání v několika textových formátech. Kliknutím na jednu z uvedených dlaždic je stažen výsledek do souboru ve zvoleném formátu.

4.5 Smazání uživatelského účtu

Uživatel může svůj účet smazat nebo změnit osobní informace případně heslo. Kliknutím na uživatelské jméno v hlavičce aplikace, a poté na položku „Účet“ je zobrazena správa uživatelského účtu. Kliknutím na položku „Smazat Účet“ se zobrazí nabídka uvedená na obrázku 4.15. Smazání účtu je nevratný proces. Veškerá data spojená s účtem jsou vymazána z databáze.

Export

Ukázkové vyrovnaní

Parametry

Název: Ukázkové vyrovnaní
Vlastník: diplomant
Jazyk: Česky
Algoritmus: svd
Úhlové jednotky: 400

Textový Náhled

```
Vyrovnaní místní geodetické sítě verze: 1.13f-svd / GNU g++
*****
http://www.gnu.org/software/gama/

Přibližné souřadnice určovaných bodů nahrazeny vyrovnanými
*****

Počet iterací linearizace: 5

Test chyby z linearizace
*****

Diference výpočtu vyrovnaných měření z oprav a z vyrovnaných souřadnic
*****
```

i	stanovisko	cíl	merena hodnota = [mm cc] ==	v [cc] == [mm] =	diference
9	3	1	57.631000	-4.317	0.000
		2	úhel		0.001

Download

Kliknutím na jednu z možností stáhnete výsledek vyrovnaní v následujících formátech:

XML

<?xml>

TEXT

abcde

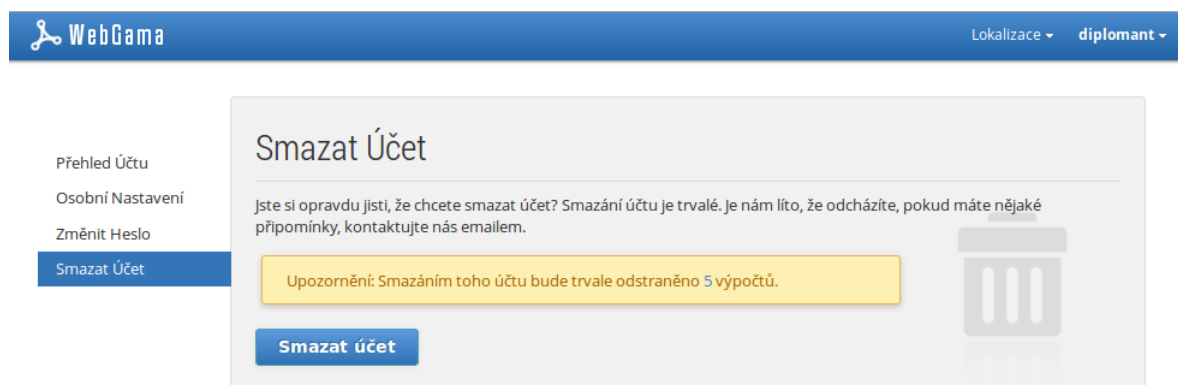
HTML

<html>

SVG



Obrázek 4.14: Export výsledku výpočtu



Obrázek 4.15: Smazání uživatelského účtu

Závěr

Cílem této diplomové práce bylo navrhnout a následně implementovat uživatelsky přívětivý webový systém pro projekt vyrovnání lokálních geodetických sítí GNU Gama. Vytyčené cíle práce byly úspěšně naplněny. Webová aplikace WebGama je přístupná na adrese:

`https://webgama.fsv.cvut.cz`

Aplikace je založena na aplikačním frameworku Spring a využívá databázového systému PostgreSQL pro ukládání dat. Umožňuje uživatelům snadné zadávání jednotlivých vyrovnání. Dále poskytuje možnost ukládat geodetické sítě a spravovat je pomocí intuitivního uživatelského rozhraní. Aplikace byla vyvinuta a testována pro dnes nejčastěji používané prohlížeče.

Text práce byl zaměřen na popis technologií a architektury webové aplikace WebGama. V první části byl představen projekt GNU Gama. V další kapitole byly teoreticky popsány technologie použité při vývoji aplikace. Následně byla popsána architektura aplikace. Poslední kapitola seznamuje uživatele s jednotlivými kroky vytvořené aplikace.

Vytvořená aplikace může být snadno rozšířena o další funkce a vlastnosti. Příkladem může být podpora více druhů vstupních a výstupních formátů dat nebo nové způsoby zadávání lokálních geodetických sítí.

Použité zdroje

- [1] SYNEK, Jan. *Popis a využití technologií relačních databází v geoinformatice*. Praha, [2011]. Bakalářská práce. ČVUT, FSV, katedra mapování a kartografie.
- [2] ČEPEK, Aleš. *GNU Gama Manual* [online]. [2000], 24.7.2012 [cit. 2013-04-12]. Dostupné z: <http://www.gnu.org/software/gama/manual/gama.html>
- [3] *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-22]. Dostupné z: <http://www.wikipedia.org/>
- [4] SPRINGSOURCE. *Spring Framework Reference Documentation* [online]. [2013] [cit. 2013-04-08]. Dostupné z: <http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/html/>
- [5] SPRINGSOURCE. *Spring Security Reference Documentation* [online]. [2013] [cit. 2013-04-08]. Dostupné z: <http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity.html>
- [6] ORACLE. *Learn About Java Technology* [online]. [2010] [cit. 2013-04-23]. Dostupné z: <http://www.java.com/en/about/>
- [7] W3C. *HTML Specification* [online]. [2013] [cit. 2013-04-23]. Dostupné z: <http://www.w3.org/html/>
- [8] *About PostgreSQL* [online]. [1996] [cit. 2013-04-25]. Dostupné z: <http://www.postgresql.org/about>
- [9] *Velocity User Guide* [online]. [2010] [cit. 2013-04-26]. Dostupné z: <http://velocity.apache.org/engine/devel/user-guide.html>
- [10] *Selenium Documentation* [online]. [2013] [cit. 2013-04-26]. Dostupné z: <http://docs.seleniumhq.org/docs/>

Příloha A

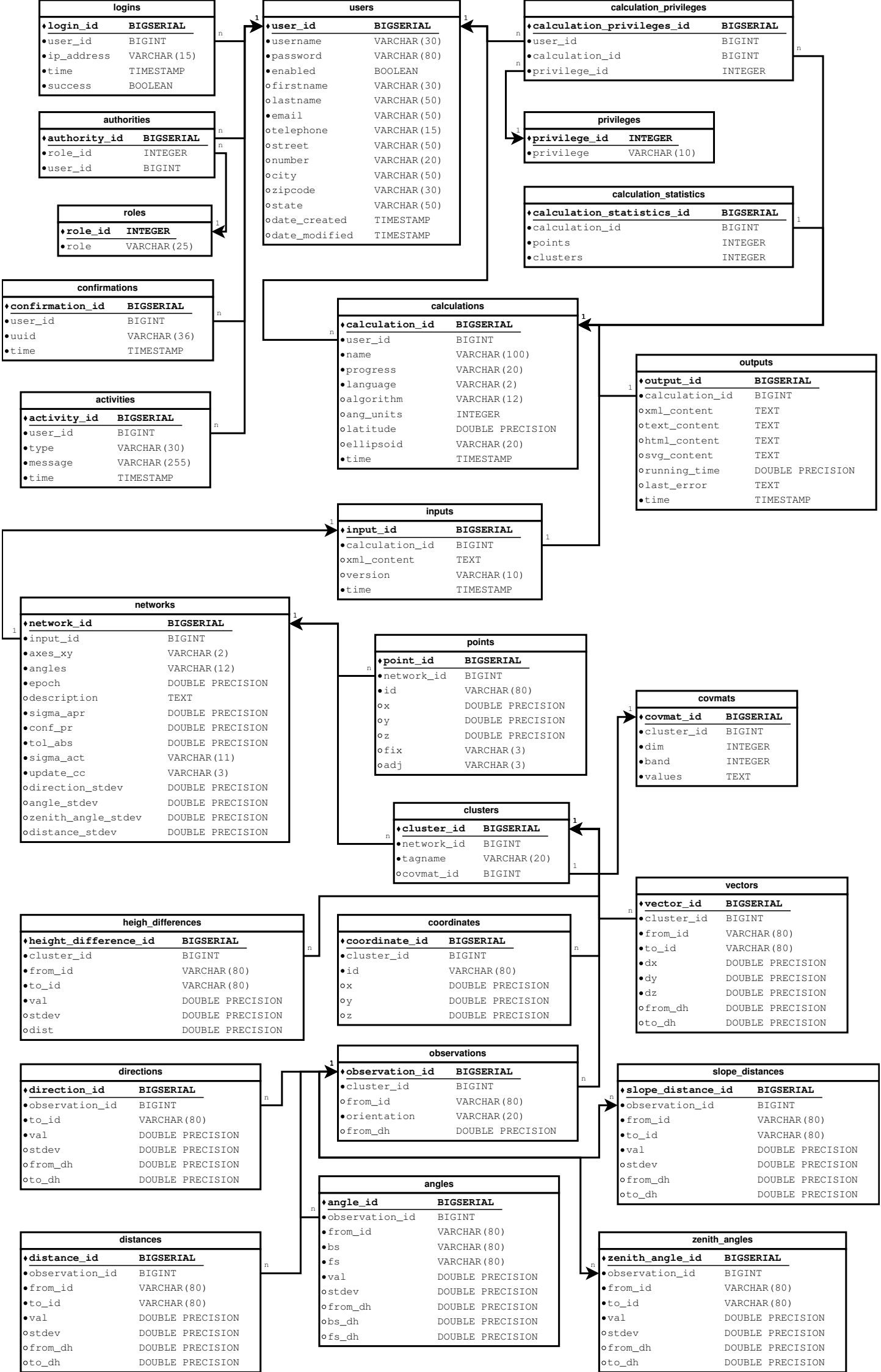
Databázové schéma

Následující tabulka uvádí počty objektů vytvořených v databázovém systému PostgreSQL.

Tabulka A.1: Souhrnný přehled objektů v databázi

Objekt	Počet
Tabulka	26
Sekvence	24
Index	29
Funkce	1
Trigger	2

Na další stránce je zobrazeno schéma databáze s naznačenými vazbami mezi tabulkami.



Příloha B

Konfigurace aplikace

Ukázka B.1: Konfigurační soubor aplikace WebGama

```
#GNU Gama binary location
gama.filepath=/opt/apache-tomcat-7.0.27/bin/gama-local

#Maximum size of files imported to Webgama in bytes (-1 = no limit)
import.maximum.file.size=10485760

#Mail address appeared in FROM
mail.address.from=noreply@fsv.cvut.cz

#FSv SMTP server
production.mail.host=smtp.fsv.cvut.cz
production.mail.port=25

#QUARTZ JOBS
#clearing old email confirmation records
quartz.clearConfirmations.cron=0 1 0 1/1 * ? *
quartz.clearConfirmations.startDelay=30000
#deleting temporary files used in GNU Gama
quartz.eraseTmp.cron=0 21,51 * 1/1 * ? *
quartz.eraseTmp.startDelay=35000
#deleting inactive users
quartz.deleteDisabledUsers.cron=0 8 2 1/1 * ? *
quartz.deleteDisabledUsers.startDelay=40000
```


Příloha C

Schéma beanů

