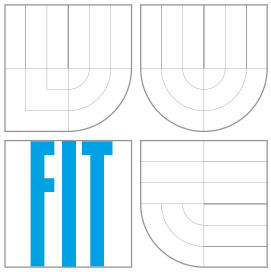
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IMPLEMENTACE GIS SLUŽBY WPS

IMPLEMENTATION OF THE WPS SERVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DUŠAN MAĎARKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2016

Zadání diplomové práce/18997/2015/xmadar01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Maďarka Dušan, Bc.**
Obor: Management a informační technologie
Téma: **Implementace GIS služby WPS**
Implementation of the WPS Service
Kategorie: Modelování a simulace

Pokyny:

1. Prostudujte normy OGC týkající se služeb WMS, WFS a WPS. Zaměřte se na protokol WPS a na jeho integraci se službami WMS a WFS.
2. Navrhněte implementaci služby WPS pro klientskou a serverovou část. Zaměřte se na snadnou integraci klientské části do uživatelských aplikací.
3. Implementujte službu pro klient a server.
4. Testujte službu v různých konfiguracích serverů, vstupních dat a požadavků.

Literatura:

- Technická dokumentace WMS, WFS a WPS na stránkách OGC.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hrubý Martin, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Cieľom tejto diplomovej práce bolo navrhnúť a implementovať webovú mapovú službu Web Processing Service. Účelom tejto služby je poskytovať výpočty s geografickým zameraním v prostredí internetu. V úvodnej fáze bola prevedená analýza existujúcich implementácií mapových služieb a používaných štandardov. Následne boli vyvodené závery a služba bola implementovaná s ohľadom na efektívnosť využitia výpočtových prostriedkov, možnosť rozširovania služby užívateľom a jednoduchosť integrácie do aplikácií typu klient. Súčasťou implementovanej služby je aj rozhranie pre prácu s geografickým informačným systémom GRASS GIS, ktoré prináša možnosť jeho využitia službou pre geografické výpočty. Záverečná kapitola sa venuje testovaniu implementovanej služby z hľadiska funkčnosti i výkonu.

Abstract

The aim of this diploma thesis was to design and implement web mapping service Web Processing Service. The purpose of this service is to provide geospatial oriented calculations on the internet. Introduction part is dedicated to analysis of existing mapping services implementations and defined standards. Thereafter conclusion was drawn and the service was implemented, with respect to computing system efficiency, pluggable design and simplicity of integration in client applications. The interface for work with geographic information system GRASS GIS is also part of the implemented service and it brings the possibility to use the tool from service. The last chapter describes the testing of implemented service in term of functional and performance tests.

Kľúčové slová

OGC, Web Processing Service, WPS, webová služba, geografický informačný systém, GIS, GRASS GIS

Keywords

OGC, Web Processing Service, WPS, web service, geographic information system, GIS, GRASS GIS

Citácia

MAĎARKA, Dušan. *Implementace GIS služby WPS*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hrubý Martin.

Implementace GIS služby WPS

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Martina Hrubého, Ph.D.. Uviedol som všetky literárne pramene, z ktorých som čerpal.

.....
Dušan Maďarka
23. mája 2016

Poďakovanie

Rád by som poďakoval za odbornú pomoc a smerovanie pri práci pánovi Ing. Martinovi Hrubému, Ph.D..

Obsah

Kapitola 1

Úvod

Geografické informačné systémy sú v dnešnej dobe používané v rôznych oblastiach pre širokú škálu úloh. Takisto rastie aj význam poskytovania mapových služieb v prostredí internetu, či už v rámci špecializovaných organizácií alebo pre verejnosť. Čoraz častejšie a širšie využívanie mapových služieb vedie k zvyšovaniu objemu a komplexnosti spracovávaných dát.

V oblasti mapových služieb vzniklo už niekoľko štandardov, najmä od organizácie Open Geospatial Consortium. V rámci tieto práce analyzujem štandardy webových mapových služieb pričom sa zameriavam na službu OGC WPS vo verzii 2.0.

V prípade stále častejšieho používania webových mapových služieb je ich dôležitým faktorom najmä efektívnosť prevádzania výpočtov, využívania dostupných výpočtových prostriedkov a poskytovania podpory pre širokú škálu dátových typov a formátov. Tieto skutočnosti naberajú na dôležitosti v tom prípade, ak sa jedná o službu Web Processing Service pre poskytovanie výpočtov nad geografickými dátami. Existuje viacero dostupných implementácií služby WPS. V rámci tejto práce boli podrobené bližšej analýze v kapitole 2.3.

V rámci tejto diplomovej práce som v súlade so štandardom OGC WPS 2.0 navrhol a implementoval WPS službu, ktorá pozostáva ako zo serverovej tak aj klientskej časti. Vybral som konkrétne aspekty výpočtových služieb, na ktoré sa zameriavam. Hlavným z nich je vysoká efektívnosť WPS služby pri využívaní zdrojov, jednoduché rozhranie pre rozširovanie služby o užívateľské výpočtové procesy a jednoduchá integrácia do klientskej aplikácie. Model implementovanej služby bližšie popisuje kapitola 3.2.1.

Štandard OGC WPS je univerzálnym rozhraním pre podporu výpočtov nad priestorovými dátami. Samotný štandard však nešpecifikuje nijaké spôsoby výpočtu a ani algoritmy, ktoré majú byť použité. Pre tento účel sú služby WPS spravidla doplnené o geografický informačný systém, ktorý zabezpečuje podporu pre správu dát priestorového charakteru, a zároveň i robustné výpočtové jadro pre prevádzanie samotných výpočtov. Z tohto dôvodu je dôležitý spôsob previazania výpočtového jadra a služby WPS. Aj na tento aspekt sa zameriava implementovaná služba. Detaily využitia geografického informačného systému sú popísané v kapitole 3.2.3.

Hlavným cieľom implementovanej WPS služby – SmartWPS – je teda poskytovať sadu preddefinovaných procesov so zameraním na geografické výpočty. Ak sa však na ňu zameriame ako na inštalovateľný produkt, ktorý je z hľadiska zmien aplikácie uzatvorený, je potrebné aby poskytoval rozhranie pre rozširovanie svojej funkcionality o ďalšie procesy. Aj táto skutočnosť bola predmetom implementácie služby a bližšie sa ňou zaoberá kapitola 3.2.4.

Pre demonštráciu implementovanej aplikácie bol súčasťou práce aj návrh a implementácia ukážkových procesov, ktoré služba poskytuje. Implementované procesy sa zameriavajú na demonštráciu rôznych možností využitia služby: pre realizáciu výpočtov v natívne v aplikácii, pre geografické výpočty realizované pomocou GIS nástroja, i pre prácu s dátami uloženými v dátovej základni geografického informačného systému. Bližším popisom navrhnutých a implementovaných užívateľských procesov sa venuje kapitola 3.4.

V poslednom rade je potrebnou súčasťou WPS služby napojenie na klientskú aplikáciu. Takáto aplikácia by mala poskytovať možnosti pre prehliadanie dostupných procesov služby, definovanie ich vstupov, prevádzania výpočtov a prehliadanie výstupov, a to všetko prostredníctvom rozhrania služby WPS. Súčasťou práce je aj návrh a implementácia klientskej aplikácie vo forme programového modulu, ktorý je bližšie popísaný v kapitole 3.3.

Aplikácia implementujúca štandard WPS je službou výpočtového charakteru z čoho plynie potrebný dôraz na jej stabilitu a výkonnosť. Služba prijímajúca požiadavky cez webové rozhranie by mala byť schopná korektne spracovať akúkoľvek prijatú požiadavku a vytvoriť odpoveď v súlade s implementovaným štandardom. Pre tento účel bola implementovaná serverová aplikácia podrobená funkčnému testovaniu – popisuje kapitola 5.1 – ktoré preverilo jej schopnosti vysporiadania sa s neočakávanými stavmi a nesprávnymi požiadavkami.

V rámci práce som sa sústredil aj na testovanie služby z hľadiska výkonu, ktoré ukázalo schopnosti služby obsluhovať veľké množstvo požiadaviek v krátkom časovom intervale. Tomuto druhu testov sa venuje kapitola 5.2.

Kapitola 2

Technický rozbor

Nasledujúca kapitola sa venuje rozboru nadobudnutých poznatkov o technických riešeniach, štandardoch a existujúcich implementáciách služieb WPS. V poslednej podkapitole sú z nich vyvádzané závery a definované konkrétne ciele pre navrhnuté a implementované riešenie.

2.1 Geografické informačné systémy

Foote a Lynch [4] definujú geografický informačný systém ako databázu so špecifickým zameraním na priestorové dáta a vzťahy medzi nimi. Typický geografický informačný systém umožňuje priestorové dáta vkladať, uchovávať, editovať a prevádzať nad nimi analýzy alebo výpočty. Existuje viacero bežne používaných technických riešení GIS.

2.1.1 ArcGIS

ArcGIS je komerčný geografický informačný systém od spoločnosti ESRI. Medzi jeho hlavé výhody patrí prenositeľnosť, dostupnosť pre rôzne platformy (vrátane mobilných zariadení) a robustnosť. Nakoľko sa však jedná o komerčný produkt, nebudem sa ním v rámci tejto práce ďalej zaoberať.

2.1.2 GRASS GIS

Geographic Resources Analysis Support System (GRASS) je open-source geografický informačný systém používaný pre správu geografických dát a ich analýzu, priestorové modelovanie, spracovanie rastrovej grafiky a vytváranie máp. Je dostupný pre všetky bežné platformy.

GRASS GIS obsahuje viac ako 350 modulov renderovanie máp a obrazu, prácu s rastrovými a vektorovými dátami, spracovanie multispektrálneho obrazu a vytváranie, spravovanie a uchovávanie priestorových geografických dát. [13] Poskytuje ako desktopovú aplikáciu s grafickým užívateľským rozhraním, tak aj konzolové rozhranie. Geografické dáta sú usporiadané prostredníctvom interných štruktúr `LOCATION` a `MAPSET`. Podľa [9] je `LOCATION` geografický extent obsahujúci geografické dáta rovnakého súradnicového systému. Technicky vzato sa jedná o priečinok v súborovom systéme obsahujúci definičné súbory popisujúce `LOCATION` (použité súradnicové systémy a mapové projekcie) a zároveň aj samotné geografické dáta – zaradené v jednotlivých `MAPSET`. Ten slúži na kategorizáciu geografických dát podľa potrieb užívateľa. V každom `LOCATION` existuje jeden predvolený `MAPSET` s označením `PERMANENT`. Špecifickým prípadom môže byť používanie GRASS GIS v prostredí zdieľanom

medzi viacerými užívateľmi. Vtedy môžu jednotlivé `MAPSET` slúžiť pre vymedzenie pracovných prostredí pre jednotlivých užívateľov tak, aby počas manipulácie so súbormi medzi nimi nedochádzalo ku konfliktom.

Moduly programu GRASS sú samostatne spustiteľné binárne programy, ktoré zároveň umožňujú dynamické linkovanie. Je teda možné funkcionalitu týchto modulov previazať aj do samostatných novovytvorených aplikácií a tak ju využiť priamo v nich, uvádza [14]. Takisto je možné dostupné moduly rozširovať, prípadne jednoducho dopĺňať o ďalšie pomocou linkovania s už existujúcimi. V dokumentácii GRASS sú popísané spôsoby pre použitie s jazykmi a technológiami PHP, Python, uDig, ABM, C a C++.

2.1.3 Externé použitie nástrojov GRASS GIS

Ako už bolo spomenuté, jednotlivé moduly programu GRASS sú samostatne spustiteľné binárne programy. Tie pre svoju funkciu potrebujú správnu konfiguráciu prostredia, uvádza [8]. Pri samotnom spustení či už textového, alebo grafického rozhrania aplikácie GRASS dôjde ku konfigurácii premenných prostredia, čím sa moduly stávajú spustiteľnými a dostupnými cez konzolové rozhranie. Takisto dôjde ku konfigurácii ciest k užívateľom zvoleným `LOCATION` a `MAPSET`. Tieto špecifické nastavenia sa označujú ako *GRASS session*. Ku jednotlivým modulom je ale možné pristupovať aj bez jej explicitného vytvorenia, a to práve pomocou nastavenia jednotlivých premenných prostredia, uvádza [8]. Pre simulovanie existencie *GRASS session* je potrebné nastaviť:

- Premennú prostredia `GISBASE` ako cestu k inštalačnému priečinku aplikácie GRASS.
- Premennú prostredia `GISRC` ako cestu k súboru obsahujúcemu nastavenia pre voľbu `GISDBASE`[1], `LOCATION.` a `MAPSET`.
- Premenná prostredia `PATH` musí zahŕňať cestu ku adresárom, ktoré obsahujú binárne programy jednotlivých modulov.

2.2 Mapové služby

V oblasti definície a štandardizácie mapových služieb je jednoznačným priekopníkom združenie OpenSource Geospatial Consortium (OGC). Štandardy OGC sa zameriavajú na definície dátových modelov pre reprezentáciu geografických dát, rozhrania webových mapových služieb a implementačné špecifikácie. OGC Web Services sú štandardy vytvorené pre popis webových mapových služieb v prostredí internetu. Jednotlivým štandardom sa venujú nasledovné podkapitoly.

2.2.1 Štandard Web Map Service

Web Map Service (WMS) je štandardizovaná webová služba, ktorej účelom je poskytovať priestorovo referencované rastrové mapové výstupy prostredníctvom protokolu HTTP. [11]

Generovanie máp prebieha dynamicky na základe požiadavky klienta. Je vykonávané mapovým serverom pomocou geografických dát z priestorovej databázy prípadne iného zdroja. Štandard definuje niekoľko druhov dotazov.

Účelom požiadavky `GetCapabilities` je získanie základného popisu o operáciách, ktoré daný mapový server poskytuje. Jedná sa teda najmä o podporované služby, podporované

[1] `GISDBASE`: užívateľský priečinok, v ktorom sú umiestnené jednotlivé `LOCATION`

dátové formáty, mapy a mapové vrstvy, ktoré konkrétny mapový server poskytuje. Nakoľko je tento druh dotazu podporovaný všetkými webovými službami OGC súčasťou požiadavky musí byť aj určenie služby, o ktorú má daný klient záujem. [11]

Požiadavka `GetMap` je používaná pre získanie generovaných rastrových výstupov vo forme bitmapových obrázkov. Požiadavka musí identifikovať konkrétne mapové vrstvy o ktoré má klient záujem. Povinné je takisto určenie ohraničenia oblasti – tzv. *Bounding Box*, ktorú má vygenerovaný výstup zobrazovať. Voliteľné sú potom nastavenia grafického výstupu (napr. priehľadnosť alebo formát rastrovej grafiky).

Voliteľným rozšírením štandardu WMS je operácia `GetFeatureInfo`, ktorá slúži pre získavanie metadát o prvkoch mapových vrstiev.

2.2.2 Štandard Web Feature Service

Základným účelom služby Web Feature Service (WFS) je poskytovať klientovi informácie o geografických prvkoch pochádzajúcich z priestorovej databázy, prípadne iného zdroja. Geografický prvok je priestorovo zaradený objekt, ktorý má definované atribúty s ktorými služba WFS pracuje. Okrem toho táto služba poskytuje ďalšie operácie pre správu klientom definovaných dotazov a voliteľne aj prácu s transakciami a výlučným prístupom. [12] Transakčné operácie sú voliteľným rozšírením služby WFS. Základná funkcionalita je tak doplnená o rozhranie pre pridávanie, mazanie a úpravu geografických prvkov, pričom sú využívané operácie pre zabezpečenie výlučného prístupu.

Ako už bolo spomenuté vyššie, každá z webových služieb OGC poskytuje operáciu `GetCapabilities`. V kontexte WFS slúži táto operácia na získanie základných informácií o poskytovanej službe a zároveň pre špecifikáciu kategórií a schém filtrov týkajúcich sa geografických prvkov.

Ako uvádza špecifikácia služby [12], operácia `DescribeFeatureType` vracia schému geografických prvkov poskytovaných inštanciou služby WFS. Táto schéma definuje požadovanú štruktúru pre vstup a výstup dát geografických prvkov. Definícia schémy pre vstupy je potrebná iba v prípade implementovaného transakčného rozšírenia. Implementácia operácie `DescribeFeatureType` je povinná pre každú službu WFS.

Dopytovanie klienta na informácie o geografických prvkoch sa prevádza pomocou zostavovania dotazov spĺňajúcich konkrétny formát. Správa klientom definovaných dotazov slúži na jednoduchšiu prácu s týmito dotazmi a umožňuje preto dotazy spravovať užívateľom pomocou operácií `CreateStoredQuery`, `DropStoredQuery`, `ListStoredQueries` a `DescribeStoredQueries` — ktoré sú v tomto poradí určené pre vytváranie, odstraňovanie, získanie zoznamu a popis definovaných dotazov.

Operáciou `GetFeature` sa klient dotazuje na informácie o geografických prvkoch. Odpoveď servera obsahuje informácie o geografických prvkoch, ktoré vyhovujú podmienkam dotazu. Dotaz je formulovaný buď použitím predtým definovaného dotazu a doplnením jeho prípadných parametrov, alebo konštrukciou nového ad-hoc dotazu. Operáciu `GetFeature` dopĺňa ešte `GetPropertyValue`, ktorá slúži pre získanie hodnoty niektorého z atribútov geografického prvku.

Ako bolo spomenuté vyššie, transakčné rozšírenie služby WFS umožňuje spravované geografické prvky nielen poskytovať, ale ich aj pridávať, upravovať a odstraňovať. Pomocou operácie `Transaction` klient popisuje transformačné operácie, ktoré sa majú aplikovať na existujúce geografické prvky [12], alebo pomocou nej definuje prvky nové. Nakoľko je služba WFS poskytovaná serverom pre mnoho klientov, je potrebné zabezpečiť pri manipulácii výlučný prístup. To môže byť zabezpečené:

- Automatickým zamykaním geografických prvkov.
- Manuálnym zamykaním s využitím operácie LockFeature.

Tieto jednotlivé prípady závisia od konkrétnej implementácie služby a server o nich musí informovať klienta v rámci odpovede na `GetCapabilities`. Webová služba je sama o sebe nestavová a bez stáleho spojenia a preto pracuje služba WFS s identifikátorom `lockId` (identifikátorom zámku). Ten je po manuálnom zamknutí pomocou operácie `LockFeature` pridelený konkrétnemu klientovi. Prevedenie transakcie na konkrétnom prvku je potom možné iba s použitím prideleného identifikátora, ktorý klient získal. Zámok prvku vyprší buď na základe časovača, alebo dokončením operácie klientom.

2.2.3 Štandard Web Processing Service

Služba Web Processing Service 2.0 (pokiaľ uvedené ďalej bez označenia verzie, jedná sa o verziu 2.0) poskytuje robustný a všestranný protokol pre prevádzanie výpočtových procesov prostredníctvom webových služieb. Takto poskytované služby umožňujú používať metódy bežne definované v prostredí geografických informačných systémov (viď kapitola 2.1). Spravidla sa teda jedná o výpočty nad geografickými dátami, avšak ako uvádza štandard [10] služba umožňuje verejne poskytovať výpočty akéhokoľvek druhu.

Pod pojmom *proces* v kontexte štandardu WPS rozumieme funkciu, ktorá pre každú množinu vstupných argumentov vracia odpovedajúcu množinu návratových hodnôt [10]. Pod jednotlivými vstupnými argumentami rozumieme *vstupy* procesu, pod návratovými hodnotami *výstupy* procesu. Vstupy aj výstupy môžu byť predávané buď hodnotou, alebo odkazom. V prípade potreby je možné použiť hierarchické vnáranie pri vstupoch aj výstupoch, prípadne intervalom definovať povolený počet výskytov konkrétneho vstupu. Uvedené popisuje diagram tried na obrázku č. 2.1.

Každý proces, vstup alebo výstup musí byť jedinečne identifikovaný. V prípade vetvenia vstupov/výstupov musí mať každý vstup/výstup unikátny identifikátor zložený z jeho identifikátora, a zároveň z identifikátorov jeho predchodcov v poradí vnorenia. Identifikátor musí spĺňať špecifikáciu URI[2].

2.2.4 Správa úloh WPS

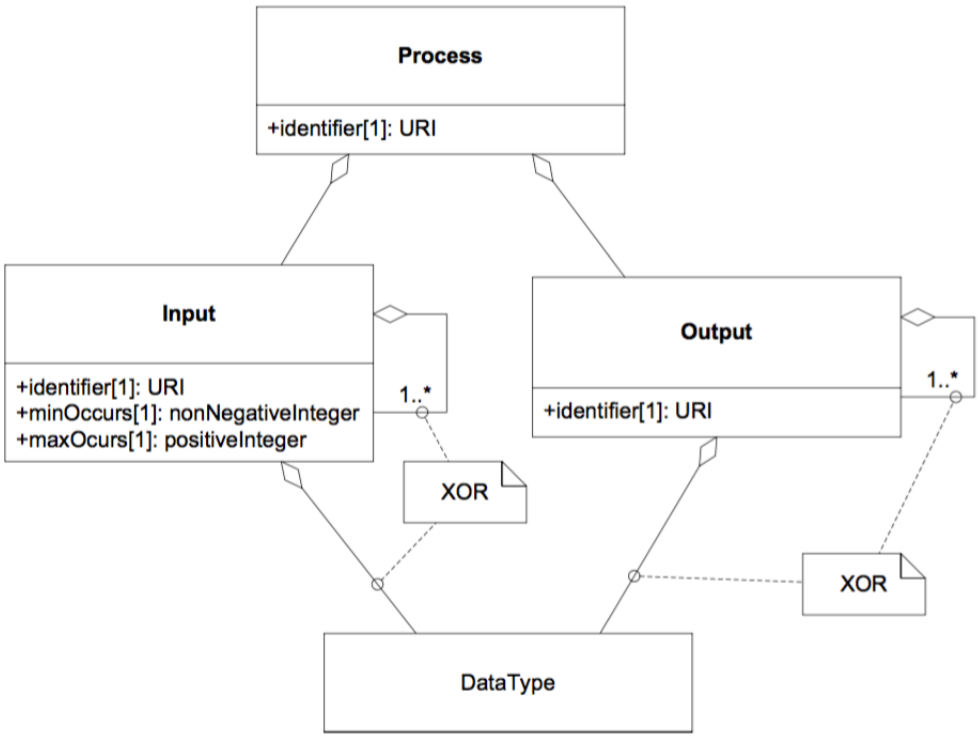
Model služby WPS definuje povinnú funkcionalitu pre každý WPS server. WPS server je webová služba, ktorá poskytuje prístup k preddefinovaným procesom a umožňuje úlohy spúšťať, riadiť a monitorovať ich priebeh. [10] Voliteľné rozšírenie potom umožňuje doplniť túto základnú funkcionalitu o rušenie, spravidla dlhotrvajúcich, bežiacich úloh. Základný výpočtový model WPS znázorňuje Obrázok 2.2.

Pre efektívne využitie výpočtových prostriedkov ako na strane servera tak i klienta, poskytuje štandard WPS prostriedky pre synchrónny alebo asynchrónny mód spustenia úloh. Rozdiel medzi nimi spočíva v spôsobe vrátenia výstupov procesu klientovi.

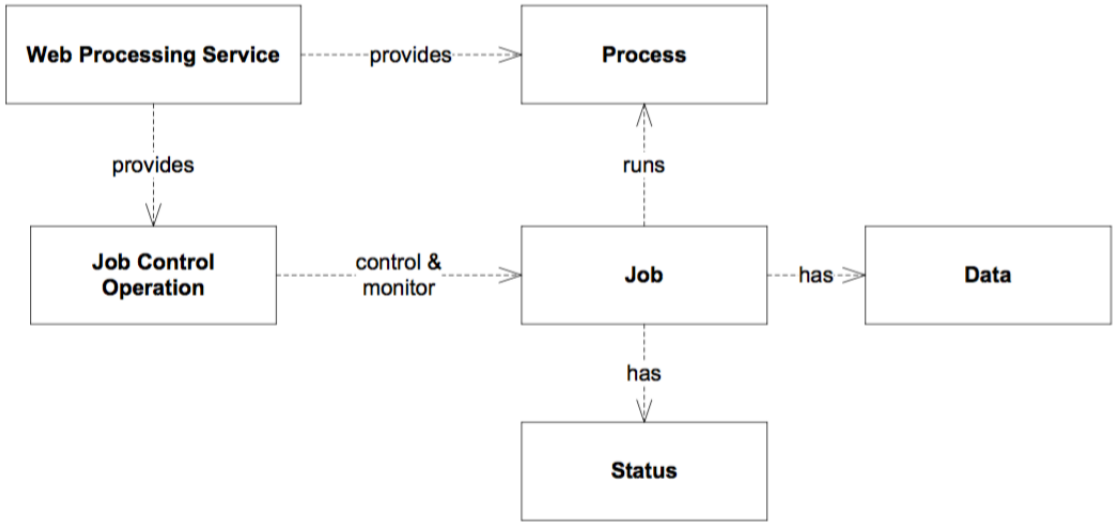
Synchrónny mód vracia výstupy ako odpoveď na HTTP dotaz, ktorým klient žiadal o prevedenie výpočtu (HTTP dotaz obsahoval aj vstupy procesu alebo odkazy na vstupy). Z tohto dôvodu by sa mal používať iba pre úlohy, ktorých spracovanie trvá spravidla krátky čas, aby nedochádzalo k blokovaniu komunikácie klienta.

V prípade použitia *asynchrónneho módu*, klient odošle požiadavku o spustenie procesu, na základe čoho server vytvorí úlohu a odpovie mu správou o jej stave – `StatusInfo`. Táto

[2]URI: Uniform Resource Identifier.



Obr. 2.1: Diagram tried UML popisujúci vzťah procesov a ich vstupov, zo zdroja [10].



Obr. 2.2: Základný výpočtový model služby WPS, zo zdroja [10].

správa informuje klienta, že úloha bola pripravená na spustenie (prípadne už spustená), pričom jej výsledok bude v budúcnosti dostupný, a takisto obsahuje jednoznačný identifikátor použiteľný pre monitorovanie jej stavu. Z bezpečnostných dôvodov štandard Web Processing Service [10] odporúča, aby bola počas prenosu v počítačovej sieti zabezpečená

dôvernosť identifikátora úlohy. Pokiaľ server prevádza výpočet danej úlohy, klient sa priebežne dopytuje na jej aktuálny stav. Stavové informácie `StatusInfo` zahŕňajú:

- Stav úlohy (prijatá, spustená, dokončená, nastala chyba).
- Exspirácia úlohy (čas kedy už nebudú dostupné výstupy spusteného procesu).
- Odhadovaný čas skončenia behu úlohy.
- Navrhovaný čas ďalšieho dotazu na stav úlohy.
- Percentuálny stav výpočtu úlohy.

Ak počas dotazovania klienta dôjde ku zmene stavu na *dokončená*, znamená to, že server dokončil výpočet a klient môže sprístupniť príslušné výsledky (výstupy procesu) pomocou identifikátora úlohy.

Štandard WPS [10] definuje, že aspoň jeden z vyššie uvedených režimov spustenia úloh musí byť implementovaný každým WPS serverom. V prípade, že WPS server podporuje oba modely behu, musí poskytnúť klientovi možnosť voľby, ktorý z týchto modelov chce použiť. V prípade, že klient nezvolí model behu úlohy, server musí *vhodne* zvoliť takýto model automaticky.

2.2.5 Natívne modely procesov WPS

Základná funkcionalita služby WPS vynucuje definíciu vstupov a výstupov pre daný proces a slovného popisu účelu procesu. Ako je uvedené v štandarde [10], voliteľné rozšírenie WPS pre natívne modely procesov poskytuje prostriedky pre:

1. Základnú štruktúru pre popis rozhraní procesov.
2. Množinu dátových typov pre špecifikáciu vstupov a výstupov procesov.
3. Rámec pre definovanie profilov procesov, pre zlepšenie spolupráce rozlične implementovaných procesov medzi WPS servermi.

Natívny model WPS procesu sprísňuje podmienky určené základnou funkcionalitou. Definuje, že popis každého procesu, jeho vstupu a jeho výstupu musí obsahovať identifikátor a ľudským okom čitateľný[3] názov, voliteľne potom kľúčové slová, popis účelu a doplňujúce metadáta. Štandard WPS [10] definuje tieto tri základné dátové typy:

- `ComplexData`, ktorý slúži pre prenos komplexných dát rôzneho druhu (napr. geografické objekty, alebo rastrové mapové vrstvy) a pre tieto účely môže byť ľubovoľne rozširovaný.
- `LiteralData`, ktorý vyjadruje jednoduchú hodnotu a voliteľne popisuje jej jednotku (napr. metre).
- `BoundingBoxData`, definovaný ako minimálne geografické ohraničenie, ktorý je používaný napr. pre výber pracovnej oblasti pre daný výpočet alebo mapové zobrazenie.

[3] Z anglického originálu „human-readable“.

Operácia	Popis operácie
Povinné operácie	
GetCapabilities	Získanie informácií o WPS službe; zoznam procesov so základným popisom.
DescribeProcess	Detailný popis, konkrétneho procesu; špecifikácia jeho vstupov a výstupov.
Execute	Spustenie konkrétneho, procesu; odpoveďou sú jeho výstupy alebo stav vytvorenej úlohy.
Voliteľné operácie	
GetStatus[5]	Získanie stavu o bežiacej úlohe.
GetResult[6]	Získanie výsledku, dokončenej úlohy.
Dismiss[7]	Predčasné prerušenie a ukončenie výpočtu klientom.

Tabuľka 2.1: Prehľad operácií štandardu WPS.

Rozšírenie umožňuje pre dátové typy definovať rôzne spôsoby zápisu a čítania hodnôt *štruktúrou pre definíciu dát*. Pomocou nej je možné špecifikovať typ MIME[4], kódovanie dát alebo ich schémy. Robustnosť špecifikácie dátových typov dodáva štandardu široké možnosti využitia v rôznych oblastiach.

Profily procesov sú návrhy, resp. popisy implementácie procesov, ktorých účelom je možnosť harmonizácie. Slúžia ako referencia pre implementáciu procesu, umožňujú dedičnosť a pomocou definovaných metód popisujú druh a spôsob konkrétneho výpočtu. [10]

2.2.6 Operácie poskytované WPS

Podobne ako aj pri ostatných webových mapových službách OGC, pozostáva rozhranie pre klienta na najnižšej úrovni z operácií. Prehľad jednotlivých operácií a ich stručný popis v kontexte Web Processing Service je uvedený v Tabuľke 2.1.

2.3 Architektúry mapových služieb

Nasledujúca kapitola sa venuje rozboru existujúcich mapových služieb so zameraním na ich architektúru a spôsob implementácie. Sústreďuje sa na služby s podporou Web Processing Service popísané v kapitole 2.2.3.

2.3.1 52°North WPS

52°North WPS je open-source implementácia štandardu OGC WPS vo verzii 1.0.0, vyvinutá organizáciou 52°North. Ako uvádza [15], tento softvér podporuje množstvo vstupných a výstupných formátov a procesov, pričom jeho architektúra ho umožňuje jednoducho rozširovať o ďalšie procesy.

Ako zdroj dát a výpočtových funkcií je možné k službe pripojiť rôzne geografické informačné systémy. Dokumentácia [16] uvádza, že je možné použiť napr. GRASS (kapitola 2.1.2), ArcGIS (kapitola 2.1.1) alebo Sextante. Implementačným jazykom 52°North WPS

[4] MIME: Multipurpose Internet Mail Extensions.
[5] Povinné v prípade podpory asynchrónneho módu spustenia úloh.
[6] Povinné v prípade podpory asynchrónneho módu spustenia úloh.
[7] Voliteľné rozšírenie štandardu WPS.

je Java. Pre internú reprezentáciu geografických dát používa služba knižnicu GeoTools[8]. Pre napojenie na webový server používa služba rozhranie *HttpServlet*.

Výhodou 52°North WPS je jednoznačne jeho robustnosť, množstvo podporovaných formátov, pripojiteľných súčastí, no najmä efektívne spojenie webovej aplikácie s jadrom služby. Z pohľadu užívateľa je dobre vyriešená konfigurácia služby, ktorú je možné prevádzať prostredníctvom webového administračného rozhrania.

2.3.2 PyWPS

Inou implementáciou štandardu OGC WPS je PyWPS. Aktuálna verzia PyWPS 3.2.0 implementuje štandard OGC WPS 1.0.0. Služba je implementovaná v jazyku Python a takisto aj ňou poskytované procesy. Vďaka tomu je možné službu jednoducho rozšíriť o ďalšie procesy.

„PyWPS je možné spustiť ako v prostredí Mod_python *ako CGI, tak pomocou Java servletu cez jython“* uvádza dokumentácia [20]. Služba umožňuje priame napojenie na GRASS GIS (kapitola 2.1.2), ktorý slúži ako zdroj dát a výpočtový prostriedok pre priestorové operácie. Okrem GRASS GIS umožňuje PyWPS aj integrovanie iných súčastí ako Minnesota MapServer, použiteľného napr. pre zobrazovanie výsledkov komplexných dátových typov `ComplexData`. Rozširovanie služby o ďalšie procesy je možné implementovaním modulu s určitým rozhraním v jazyku Python a jeho zavedením do priečinka, ktorý určuje konfigurácia PyWPS, uvádza dokumentácia [21].

Výhodou služby PyWPS je jednoznačne jednoduché rozširovanie o ďalšie procesy, avšak na druhú stranu implementácia v jazyku Python a využitie CGI uberá službe na výkonnosti.

2.4 Synchronizačné mechanizmy

2.4.1 Web

Webová služba je softvérový systém, ktorý je navrhnutý pre podporu interakcie medzi strojmi [7]. Webová služba typicky komunikuje pomocou protokolu HTTP vo formáte XML správ. Špecifikácia protokolu HTTP [3] ho definuje ako *„...bezstavový aplikačný protokol typu dotaz/odpoveď, ktorý používa rozšíriteľnú sémantiku a samopopisujúce správy pre flexibilnú interakciu so sieťovo-založenými hypertextovými informačnými systémami.“*

Asynchrónnej komunikácie v prostredí webových stránok sa bežne dosahuje pomocou skupiny webových technológií, súhrnne známej pod pojmom AJAX[9]. Táto umožňuje asynchrónne dotazovanie sa na webové zdroje a vizualizovanie prijatých výsledkov pomocou úprav DOM[10] elementov.

Typickým zástupcom klienta štandardu WPS v prostredí webových stránok je knižnica OpenLayers[11] v jazyku JavaScript. Okrem štandardu WPS knižnica implementuje aj štandard OGC WMS a WFS, čo z nej robí robustný nástroj, vhodný pre prácu ako s vizualizovanými geografickými dátami, tak s výsledkami operácií, ktoré sú nad nimi prevádzané – a to všetko za pomoci rozhraní webových služieb OGC. Knižnica OpenLayers používa technológiu AJAX pre asynchrónne získavanie informácií z OGC webových služieb pomocou HTTP protokolu pre zabezpečenie efektivity a komfortu užívateľa.

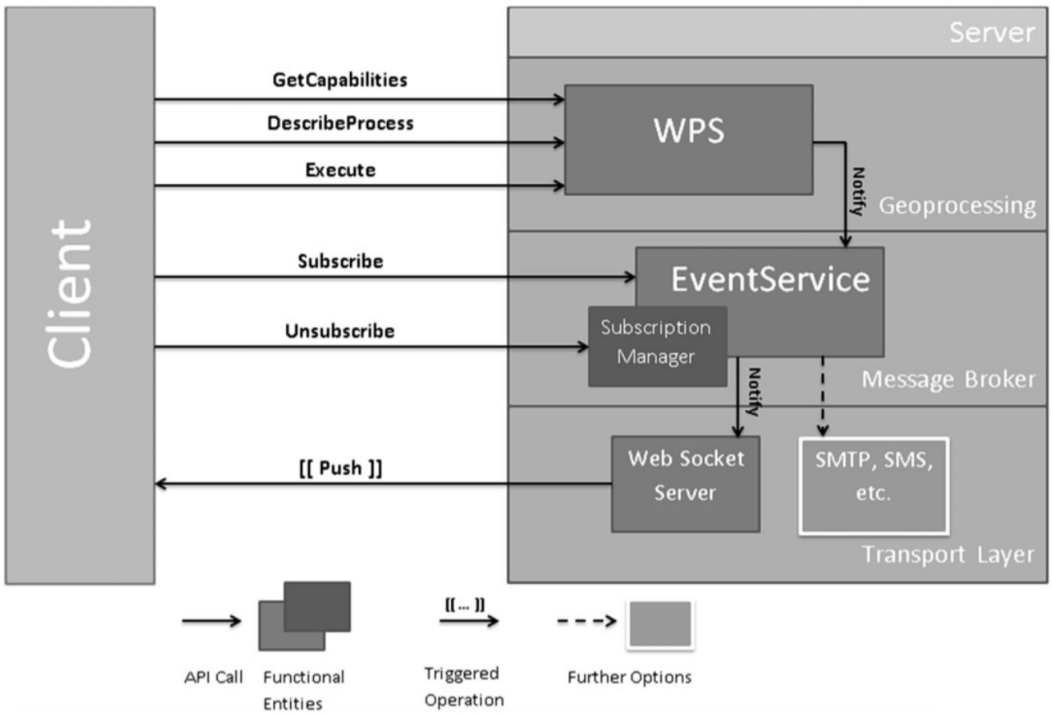
[8] Dostupné na http://www.geotools.org/.
[9] AJAX: Asynchronous JavaScript and XML.
[10] DOM: Document Object Model.
[11] Knižnica OpenLayers je dostupná na http://openlayers.org/.

Alternatívou asynchrónneho dotazovania je použitie protokolu Web Socket, ktorý umožňuje obojsmernú komunikáciu medzi serverom a webovým prehliadačom. Ako uvádza [2] mechanizmus protokolu pozostáva z dvoch častí: *handshake* a prenosu dát. Handshake slúži pre prepnutie komunikácie z režimu používaného pri protokole HTTP na Web Socket. Pokiaľ handshake prebehne správne, TCP spojenie sa neukončí, ale zostane otvorené a môže po ňom prebiehať obojsmerný dátový prenos. Použitím protokolu Web Socket v kontexte WPS sa venuje časť kapitoly 2.4.2.

2.4.2 Asynchrónny mód WPS

Ako popisuje kapitola 2.2.4, štandard WPS podporuje synchrónny a asynchrónny režim behu procesov. V prípade synchrónneho režimu funguje navrátenie výsledku spôsobom dotaz-odpoveď. Naopak pri asynchrónnom režime prebieha výpočet dlhšiu dobu a klient sa pomocou rozhrania Web Processing Service dopytuje na stav úlohy. Klientská časť teda musí pravidelne komunikovať – nadväzovať spojenie a čakať na odpoveď.

Autori článku [19] navrhli a implementovali model asynchrónneho informovania o skončení behu procesu WPS. Klient je o dostupnosti výsledku informovaný prostredníctvom asynchrónnej notifikácie cez Web Socket Protocol. Spojenie Web Socket Protocol je nadviazané registráciou klienta v notifikačnej službe. Notifikačná služba je priamo spojená so serverom WPS a čaká na skončenie behu úlohy. Architektúru celého riešenia popisuje Obrázok 2.3. Medzi výhody tohto riešenia patrí najmä to, že pri asynchrónnom čakaní na správu odpadá réžia, ktorá by inak vznikala neustálym nadväzovaním a ukončovaním sieťového spojenia. Na druhú stranu pri veľkej užívateľskej základni danej WPS služby by bolo potrebné udržovať množstvo otvorených spojení pre notifikačné kanály.



Obr. 2.3: Architektúra asynchrónneho notifikovania o skončení behu WPS procesu; [19].

2.5 Zhodnotenie

Štandard OGC Web Processing Service má široké možnosti uplatnenia vo webových službách. Verzia 2.0.0 okrem iného prináša možnosti použitia v rámci služieb SOA, definovania natívnych modelov procesov a ich profilov. Kľúčovým aspektom služby náročnej na výpočtový výkon však stále zostáva efektívnosť využitia dostupných prostriedkov. Z tohto dôvodu považujem za dôležité inšpirovať sa existujúcimi službami, ktoré nepoužívajú rozhranie CGI pre invokáciu procesov, ale tieto súčasti implementujú iným, efektívnejším spôsobom.

Ďalším nemenej dôležitým aspektom implementácie štandardu WPS je možnosť rozširovania služieb o nové, užívateľom definované procesy. Spôsob rozširovania by nemal byť príliš komplikovaný, no takisto by nemal znižovať efektivitu aplikácie vzhľadom na potrebné zdroje. Rovnako jednoduchá by mala byť inštalácia a konfigurácia samotnej služby a mal by byť kladený dôraz na jej prenositeľnosť.

Keďže WPS je často iba rozhraním medzi klientom a bázou prevádzajúcou (geografické) výpočty, považujem za dôležité pokiaľ možno čo najužšie previazanie s robustným nástrojom pre prevádzanie geografických výpočtov. Takýto nástroj by tiež mal poskytovať možnosť geografické dáta spravovať – poskytovať vstupy pre procesy.

Posledným aspektom, ktorý považujem v oblasti implementácie služieb podľa štandardu OGC WPS za dôležitý, je jednoduchosť a účelnosť klientskej aplikácie. Vzhľadom na to, že sa jedná o služby webové, môže byť vhodné implementovať klientskú aplikáciu vo forme webovej stránky. Zaujímavým riešením však môže byť poskytnúť klientskú aplikáciu vo forme knižnice alebo programového modulu, ktorý by bolo možné začleniť do natívnej aplikácie.

Na základe prevedeného rozboru štandardov mapových služieb, ich existujúcich implementácií, dostupných GIS nástrojov pre geografické výpočty a požiadaviek pre potreby implementácie klientských aplikácií považujem pri implementácii služby WPS za dôležité najmä:

- Implementovať štandard OGC WPS vo verzii 2.0.
- Efektívne implementovať obsluhovanie požiadaviek klientov.
- Integrovať do služby robustný GIS nástroj a použiť ho ako prostriedok pre geografické výpočty.
- Poskytnúť užívateľom infraštruktúru pre zverejnenie vlastných WPS procesov.
- Spraviť implementáciu služby prenositeľnú a jednoducho použiteľnú.
- Zamerať sa na jednoduchú integráciu do klientskej časti a efektívnu komunikáciu medzi klientom a serverom.
- Implementovať klientskú aplikáciu vo forme modulu, použiteľného v natívnej aplikácii.
- Navrhnúť a implementovať do služby ukážkové WPS procesy, ktoré budú účelne demonštrovať funkcionalitu služby.

Kapitola 3

Koncepcia práce

Cieľom tejto práce je implementovať službu WPS zameranú na poskytovanie výpočtovej funkcionality pre jednoduché aj komplexné výpočty, prevažne s priestorovým zameraním. Výsledná služba bude zahŕňať serverovú časť pre poskytovanie výpočtovej funkcionality, ktorú bude možné jednoduchým spôsobom rozširovať. Súčasťou služby bude niekoľko ukážkových WPS procesov demonštrujúcich možnosti použitia služby. Serverová časť bude doplnená o programový modul pre aplikáciu klienta, pomocou ktorého bude možné implementovanú službu využívať.

V nasledujúcich kapitolách sa venujem bližšiemu popisu návrhu architektúry a súčastí rozhrania služby (kapitola 3.1), serverovej aplikácie (kapitola 3.2), ukážkových procesov (kapitola 3.2.4) a klientskej aplikácie (kapitola 3.3).

3.1 Model služby SmartWPS

Vzhľadom na charakter navrhovaného riešenia ako webovej služby, navrhujem riešenie implementovať ako dve spolupracujúce aplikácie:

- Serverovú aplikáciu implementujúcu štandard OGC WPS 2.0 s podporou pre prevádzanie geografických výpočtov.
- Klientskú aplikáciu pre sprístupnenie výpočtovej báze užívateľom demo portálu.

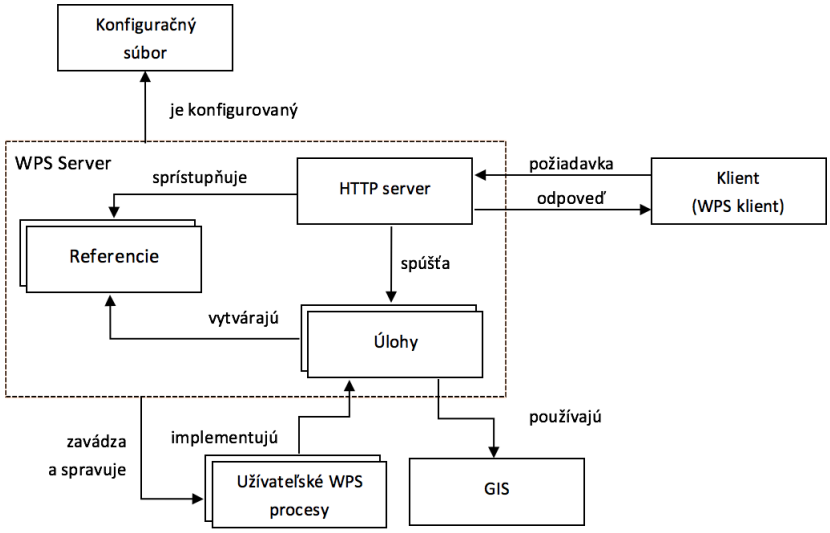
Účel serverovej aplikácie je vymedzený konceptom služby Web Processing Service – poskytovať preddefinovanú, avšak rozšíriteľnú množinu procesov, pomocou ktorých je možné vykonávať geografické výpočty. Tieto procesy bude možné pomocou rozhrania WPS prehliadať, žiadať o ich spustenie pričom definovať ich vstupy a následne pristupovať k nimi vytvoreným výstupom. Popritom bude možné monitorovať priebeh ich výpočtu. Toto jadro pre poskytovanie výpočtovej funkcionality bude vhodne doplnené geografickým informačným systémom pre prevádzanie geografických výpočtov. Celá serverová aplikácia bude konfigurovateľná – bude možné spravovať dostupné procesy, dopĺňať ich o ďalšie procesy a takisto editovať všeobecné informácie o službe.

Aplikácia klienta bude slúžiť pre prehľadné a jednoduché využívanie výpočtovej funkcionality poskytovanej serverom. Klientskú aplikáciu navrhujem ako desktopovú aplikáciu využívajúcu programový modul, ktorý bude poskytovať rozhranie pre komunikáciu so serverom.

3.1.1 Komunikačné rozhranie

Ako komunikačnú medzivrstvu pre poskytovanie výpočtovej infraštruktúry použijem štandard OGC Web Processing Service vo verzii 2.0, ktorého použitie vyplýva zo zadania práce. Z použitia tohto štandardu vyplývajú správy, pomocou ktorých budú vytvorené aplikácie komunikovať.

Získavanie informácií o službe a informácií o dostupných procesoch bude prevádzané pomocou požiadavky `GetCapabilities`. Bližšie popisovanie procesov a získanie informácií o požiadavkách na ich vstupy bude možné pomocou požiadavky `DescribeProcess`. Následné spúšťanie procesov a definovanie ich vstupov bude prevádzané požiadavkou `Execute`. Nakoľko štandard poskytuje možnosť voľby medzi implementáciou synchrónneho a asynchrónneho módu (viď kapitola 2.2.4), v rámci tejto práce sa sústredím najmä na implementáciu módu synchrónneho. Z toho dôvodu odpadá nutná podpora pre tento spôsob výpočtu určených WPS operácií: `GetStatus` a `GetResult`. Všetky spomenuté požiadavky budú implementované v súlade so štandardom WPS 2.0 a sú bližšie popísané v kapitole 2.2.6. Základný model architektúry služby je znázornený na Obrázku 3.1.



Obr. 3.1: Základný model architektúry služby.

3.1.2 Služba z pohľadu poskytovateľa

Poskytovateľ službu WPS spravidla spravuje pomocou konfiguračného súboru. V rámci neho má k dispozícii prostriedky ku:

- Nastavovaniu parametrov behu služby SmartWPS.
- Definícii obecných informácii o poskytovateľovi, účelu služby, organizácii a pod..
- Správe poskytovaných procesov.

Pomocou nastavovania parametrov služby konfiguruje prevádzkovateľ nastavenia, ktoré priamo súvisia s poskytovaním geografických výpočtov a behu procesov.

Priamo zo štandardu WPS 2.0 plynie povinnosť sprístupniť cez rozhranie určité všeobecné informácie o službe WPS. Konkrétne sa jedná o identifikáciu služby, poskytovateľa

služby, stručný popis poskytovaných procesov a jazyk v ktorom sú dané ľudským okom čitateľné informácie k dispozícii. Tieto informácie budú takisto konfigurovateľné prostredníctvom konfiguračného súboru.

Prevádzkovateľ služby bude ďalej spravovať aj nastavenia konfiguračného charakteru. Tie budú použité pre konfiguráciu jednotlivých súčastí služby a jej parametrov. Pre konkrétne procesy, ktoré služba WPS poskytuje sa spravidla predpokladá konfigurácia na úrovni základných úkonov: zavedenia alebo vylúčenia konkrétneho procesu. Uložením informácie o procese do konfiguračného súboru sa povolí zavedenie jeho kompilovanej knižnice do služby WPS. Tejto problematike sa podrobnejšie venuje kapitola 3.2.4. Pomocou jednoduchého rozhrania však bude možné jednotlivé procesy pridávať či odstraňovať aj za behu aplikácie.

3.1.3 Služba z pohľadu užívateľa

Užívateľ služby prostredníctvom desktopovej klientskej aplikácie pristupuje k prehliadaniu a spúšťaniu procesov a následnému prehliadaniu získaných výsledkov. Nakoľko bude aplikácia typu klient komunikovať so serverom iba prostredníctvom WPS operácií, bude možné ju použiť s akoukoľvek službou WPS 2.0 podporujúcou rovnakú podmnožinu operácií.

Z dôvodu komplexnosti používaných geografických dát sa klientská aplikácia nebude zameriavať na rozhranie pre ich zadávanie užívateľom alebo ich vizualizáciu, ale priamo na poskytovanie rozhrania pre službu WPS. To však nebráni zrozumiteľnému využívaniu služby pomocou klientskej aplikácie, nakoľko komplexné geografické dáta je možné na vstup procesov zavádzať pomocou *referencií* (viď kapitola 2.2.3), a rovnakým spôsobom ich aj získavať pre potreby zobrazenia.

3.2 WPS Server

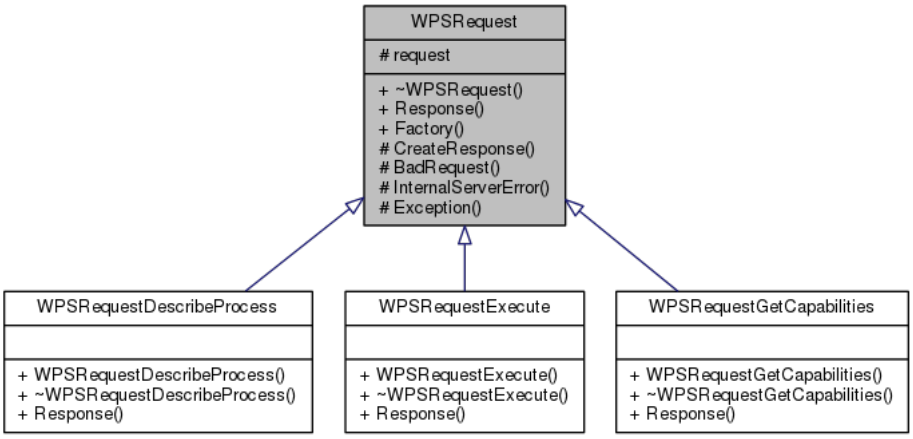
Na základe záverov vyvodených v kapitole 2.5 bude serverová aplikácia implementovaná v jazyku C++. Z charakteru aplikácie vyplýva, že je potrebné vyriešiť spôsob súbežného obsluhovania požiadaviek klientov. Pre zlepšenie efektivity práce aplikácie ju implementujem ako jednoliatu aplikáciu s rozhraním pre webovú službu a internou správou procesov obsluhujúcich prichádzajúce požiadavky. Uvedeným problematikám sa bližšie venuje kapitola 3.2.2. Pre podporu širokej škály výpočtov geografického charakteru bude implementovaná služba previazaná s nástrojom GRASS GIS, čomu sa bližšie venuje kapitola 3.2.3. Možnosť užívateľského rozširovania poskytovaných procesov vyžaduje implementáciu rozhrania, pomocou ktorého budú jednotlivé – samostatne distribuované procesy – využívané pre vykonávanie výpočtu. Problematike zavádzania a implementácie užívateľských procesov sa bližšie venuje kapitola 3.2.4. Poslednou dôležitou súčasťou serverovej aplikácie je konfigurácia, ktorej detailný návrh je popísaný v kapitole 3.5.

3.2.1 Architektúra

Ako už bolo spomenuté, navrhované riešenie sa snaží čerpať z výhod spojenia implementácie, ako služby WPS, tak obsluhy požiadaviek na webovú službu do jedného celku.

Architektúra časti serverovej aplikácie do značnej miery reflektuje konceptuálny model služby WPS definovaný štandardom. V prvom rade sa jedná o definovanie jednotlivých druhov požiadaviek. Diagram dedičnosti tried je znázornený na obrázku č. 3.2. Každá z tried

preťažuje metódu pre získanie odpovede na danú požiadavku. Obsah jednotlivých požiadaviek je uchovávaný v spracovanom formáte XML, čomu sa bližšie venuje kapitola 3.2.2.



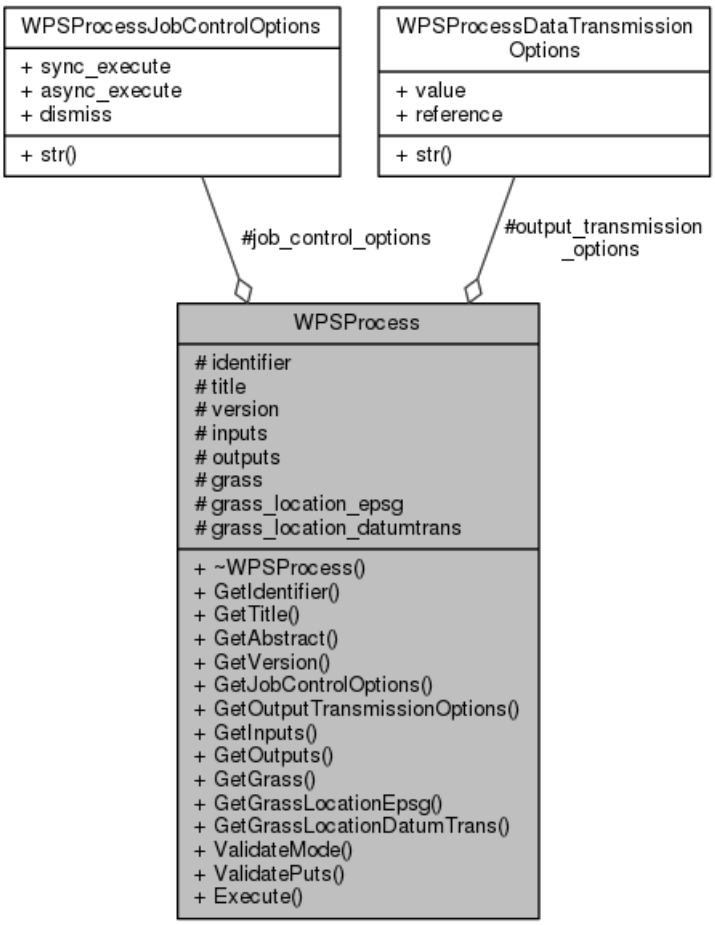
Obr. 3.2: Diagram dedičnosti tried UML pre jednotlivé požiadavky.

Ďalším prvkom je trieda pre proces, ktorá reprezentuje konfiguráciou zavedený, užívateľom vytvorený WPS proces, poskytovaný službou SmartWPS. Diagram triedy `WPSProcess` je znázornený na obrázku 3.3. Trieda `WPSProcess` obsahuje:

1. Základné atribúty WPS procesu ako identifikátor, ľudským okom čitateľný názov, popis funkcionality procesu, verziu a iné.
2. Zoznam dátových štruktúr pre vstupy procesu.
3. Zoznam dátových štruktúr pre výstupy procesu.
4. Dátovú štruktúru pre konfiguráciu možností jeho spustenia – *job control options*.
5. Dátovú štruktúru pre konfiguráciu možností prenosu vstupov a výstupov – *data transmission options*.
6. Voľby týkajúce sa použitia geografického informačného systému GRASS GIS.

Požiadavky na základné atribúty WPS procesu, spomenuté v bode č. 1, sú určené štandardom OGC WPS 2.0. Dátové štruktúry pre konfiguráciu vstupov a výstupov WPS procesov slúžia pre definíciu identifikátora, textového popisu, povolených počtov výskytu a typu dát, pre ktoré je určený. Pre reprezentáciu vstupu procesu je implementovaná trieda `WPSInput`, pre výstup procesu trieda `WPSOutput`, pričom základným rozdielom v ich návrhu sú povolené počty výskytov. Podľa štandardu totiž každý z výstupov musí byť uvedený presne jedenkrát. Pri vstupoch procesu je naopak počet platných výskytov definovaný intervalom, ktorý môže byť užívateľom konfigurovaný v konštruktore procesu. Hierarchia dedičnosti implementovaných tried pre vstupy a výstupy sa nachádza na Obrázku 3.4.

Každý zo vstupov resp. výstupov procesu primárne reprezentuje jeden z dátových typov vstupu resp. výstupu: `LiteralData`, `ComplexData` alebo `BoundingBoxData`. Diagram dedičnosti tried sa nachádza na Obrázku 3.5. Dátový typ `ComplexData` je v rámci implementácie služby SmartWPS podporovaný textovo reprezentovaný formát XML. Použitie formátu XML umožňuje používať aj ďalšie formáty na ňom založené (napr. GML či KML),

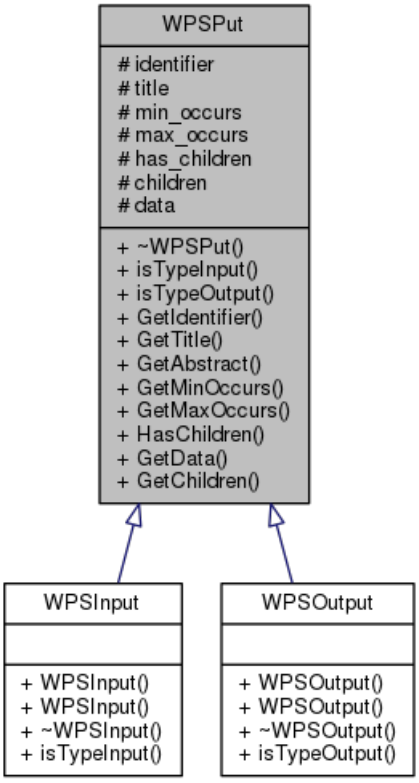


Obr. 3.3: Diagram triedy UML pre proces poskytovaný službou WPS.

ktoré sú bežne používané pre reprezentáciu geografických dát. Pre dátový typ `LiteralData` poskytuje štandard prostriedky pre definíciu primitívneho dátového typu, predvolenej hodnoty a definíciu akceptovaných hodnôt. Túto funkcionalitu implementuje služba SmartWPS a je možné ju využiť pre užívateľské procesy a tým automatizovať validáciu vstupov procesu, čomu sa venuje kapitola 4.3. Podpora pre dátový typ `BoundingBoxData` nie je v rámci služby SmartWPS implementovaná.

Voľby týkajúce sa použitia geografického informačného systému GRASS GIS sú konfigurovateľné pre jednotlivé WPS procesy hneď z viacerých dôvodov. Jedným z nich je, že služba nevyžaduje nutne pre svoju funkčnosť použitie geografického informačného systému – výpočet realizovaný procesom nemusí použiť na získanie výsledku GRASS GIS. Užívateľ tak môže špecifikovať, či ním zavedený proces potrebuje pre svoju funkcionalitu využívať službou konfigurovaný GRASS GIS, alebo nie. Takisto poskytuje jednoduché rozhranie pre určenie vlastností dočasne vytváranej `LOCATION`. Tejto problematike sa bližšie venuje kapitola 3.2.3.

Triedy reprezentujúce WPS procesy a ich atribúty sú inicializované buď pri spustení služby, alebo dodatočne počas jej behu. Každá z tried reprezentujúcich procesy je inštancovaná iba jeden krát, pričom je umiestnená v sklade procesov – triede `WPSProcessesStore`. Pri prijatí ktorejkoľvek požiadavky sa zo skladu procesov získajú požadované informácie.

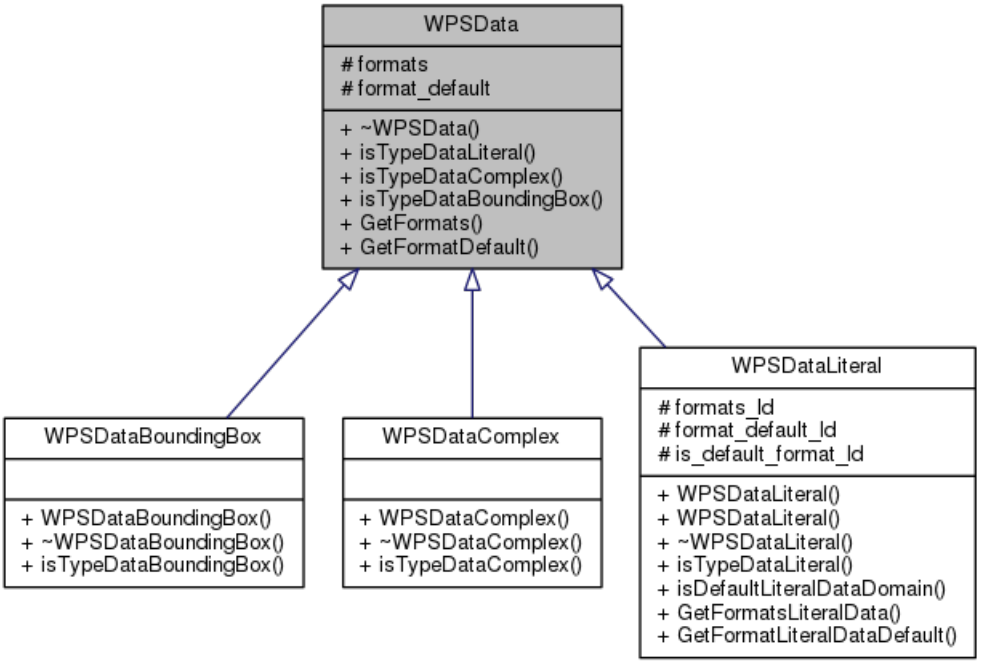


Obr. 3.4: Diagram dedičnosti tried UML pre vstupy procesu.

Pri prijatí požiadavky typu `Execute` je po validácii vstupov a parametrov spustenia inicializovaná úloha príslušného procesu. Samotný model procesu slúži ako šablóna pre vytvorenie úlohy. Pokiaľ daný proces vyžaduje použitie geografického informačného systému, bude v rámci úlohy inicializované aj dočasné prostredie GRASS GIS. Vytvorená úloha úplne zaobaľuje spustenie požadovaného výpočtu a tým vytvára dočasné prostredie pre jeho vykonávanie. Po ukončení úlohy sú klientovi navrátené výsledky výpočtu a dočasne vytvorené prostredie je odstránené.

3.2.2 Infraštruktúra pre webovú službu

Akákoľvek aplikácia typu server je špecifická tým, že musí obsluhovať prichádzajúce požiadavky od viacerých klientov neblokujúcim spôsobom. Komunikačným protokolom v prípade služby WPS je protokol HTTP. Pre tento účel som zvolil multiplatformovú knižnicu v jazyku C *GNU Libmicrohttpd* [5], ktorá umožňuje implementovať konkurentný HTTP server. Knižnica podporuje viacero režimov pre súbežné spracovanie požiadaviek a monitorovanie príchodu požiadaviek. Vzhľadom na charakter implementovanej služby som pre implementáciu WPS služby zvolil režim využívajúci skupinu vlákien – *thread pool* – ktorý je dobre škálovateľný. Počet použitých vlákien bude nastaviteľný pomocou konfiguračného súboru. Pokiaľ ide o režim zachytávania vzniknutých udalostí – prichádzajúcich požiadaviek, najvhodnejšou z prenositeľných variant je podľa dokumentácie [6] režim `poll`. Hlavným dôvodom je jeho efektívnosť a zároveň široká podpora naprieč platformami. Zvolená súčasť tak odpovedá určeným požiadavkám pre škálovateľnosť, efektívnosť a prenositeľnosť



Obr. 3.5: Diagram dedičnosti tried UML pre jednotlivé dátové typy WPS.

implementovanej aplikácie.

Štandard OGC WPS 2.0 vymedzuje dva spôsoby odosielania požiadaviek klientom na server, a to:

- Pomocou metódy GET protokolu HTTP, s využitím KVP[1].
- Pomocou metódy POST protokolu HTTP, kde je požiadavka definovaná v tele metódy vo formáte XML.

V rámci implementovanej aplikácie som sa rozhodol použiť druhý z uvedených spôsobov. Prichádzajúce dáta vo formáte XML je potrebné previesť na internú štruktúru v rámci ktorej prebieha analýza vstupnej požiadavky. Pre tento účel použijem knižnicu *Xerces-C++* [2], pomocou ktorej budú požiadavky spracované do internej DOM reprezentácie. V uvedenom procese spracovania sa v prvom rade identifikuje typ prichádzajúcej požiadavky – konkrétna operácia Web Processing Service, ktorá bude následne spracovaná. Implementačným špecifikám tejto problematiky sa venuje kapitola 4.2.

Ako uvádza technický rozbor v kapitole 2.2.3, štandard OGC WPS definuje dva spôsoby predávania vstupov a výstupov: hodnotou alebo odkazom. Pri predávaní *vstupu* odkazom sa principiálne jedná iba o vytvorenie požiadavky na vzdialený zdroj a získanie jeho obsahu, prípadne spracovanie chybového stavu pri jeho nedostupnosti. Pre predávanie *výstupov* odkazom je však potrebné implementovať mechanizmus, ktorý zabezpečí vytváranie, správu a dostupnosť vytvorených výstupov. Mechanizmus pre správu referencií implementujem ako súčasť služby SmartWPS. Po spracovaní procesu sa jeho výstup, ak o to klient požiadal, uloží do perzistentného úložiska – adresára na disku s konfigurovateľnou cestou. Klient

[1] KVP: Key-Value Pair; informácie o požiadavke kódované v URL.
[2] Dostupné na https://xerces.apache.org/xerces-c/.

obdrží v odpovedi namiesto dát URL adresu, na ktorej bude dostupný ním požadovaný výstup. Tento bude možné získať pomocou metódy GET protokolu HTTP, až po kým nedôjde k automatickému zmazaniu výstupu po určenom čase. Životnosť výstupu bude v tomto prípade pre jednoduchosť rovnaká ako životnosť úlohy pre dané spustenie procesu. Vytváraniu a správe výstupov dostupných odkazom sa z hľadiska implementácie bližšie venuje kapitola 4.6.

3.2.3 Modul pre geografické výpočty

Ako bolo spomenuté v kapitole 2.2.3 samotná WPS služba neurčuje žiadne preddefinované procesy a ani spôsob prevádzania geografických výpočtov. Z tohto dôvodu bolo potrebné zvoliť prostriedok, ktorý potrebnú funkcionalitu výpočtového jadra poskytne. Na základe zhodnotenia som sa rozhodol na tento účel použiť geografický informačný systém GRASS GIS. Hlavným dôvodom je možnosť samostatného využita jeho modulov. Použitie takéhoto robustného nástroja značne rozšíri možnosti implementovanej služby SmartWPS.

Pri používaní zvoleného nástroja je kvôli charakteru služby – aplikácie typu server s konkurentnou obsluhou požiadaviek – potrebné zabezpečiť pre každú bežiacu úlohu samostatné dočasné prostredie, a to i v rámci použitého geografického informačného systému. Na základe analýzy možností nástroja GRASS GIS, bolo pre tento účel možné použiť dva spôsoby:

- Použiť pre každú úlohu dočasne vytvorený `MAPSET` v rámci predpripravenej `LOCATION`, čo by bolo podobné ako v prípade prístupu pre viacerých užívateľov k dátovej základni GRASS GIS (ako popisuje kapitola 2.1.2).
- Použiť pre každú úlohu dočasne vytvorená `LOCATION`.

Aj keď sa použitie dočasného `MAPSET` javí ako efektívnejšia varianta, je potrebné zvážiť fakt, že v rámci jedinej `LOCATION` môžu geografické dáta používať práve jeden súradnicový systém, čo by do značnej miery ovplyvnilo možnosti spracovania dát. Preto som sa rozhodol pre účely dočasného prostredia vytvárať pre každú spustenú úlohu novú `LOCATION`. Autor procesu potom môže špecifikovať aký súradnicový systém a mapovú projekciu má dočasne vytvorená `LOCATION` používať.

Pre sprístupnenie modulov GRASS GIS pre implementovanú službu som navrhol rozhranie, ktoré simuluje spustenie GIS nástroja pomocou mechanizmov popísaných v kapitole 2.1.3. Simulovanie prostredia je realizované nastavením premenných systému. Spustenie konkrétneho príkazu je potom možné priamo v danom prostredí. Pre tento účel implementovaná služba využíva generované skripty v jazyku Bash, ktoré reflektujú užívateľom nastavené prostredie. Nakoľko presná implementácia inicializačného skriptu značne závisí na použitej verzii nástroja GRASS GIS, súčasťou implementovanej služby je iba šablóna tohoto skriptu. Táto je vždy pri použití nástroja GRASS GIS načítaná a jednotlivé parametre spustenia príkazu sú do nej doplené. V nasledovnom odstavci sa nachádza predvolená šablóna skriptu[3].

```
export GISBASE=%1
export PATH=$PATH:$GISBASE/bin:$GISBASE/scripts
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$GISBASE/lib
export GIS_LOCK=$$
```

[3] Šablóna skriptu bola navrhnutá pre použitie s GRASS GIS verzie 6.4.x.

```
export GISRC=.grassrc6
export HOME=/var/www
export USER=www-data
export GROUP=www-data

echo "GISDBASE: %2
LOCATION_NAME: %3
MAPSET: %4
GRASS_GUI: text
" > .grassrc6

%5

$GISBASE/etc/clean_temp
rm -rf /tmp/grass6-$USER-$GIS_LOCK
```

Prvým nahradeným parametrom `%1` je cesta k spustiteľným súborom nástroja GRASS GIS. Nasledovné riadky skriptu pracujú s konfigurovanou premennou prostredia a zprístupňujú tak jednotlivé moduly. Ďalšia časť konfiguruje systémové premenné tak, aby nevznikli problémy s oprávneniami (v tomto prípade sa predpokladá, že aplikácia SmartWPS beží na konkrétnom stroji ako samostatná webová služba). Pri spustení nástroja GRASS GIS je nutné špecifikovať predvolenú `LOCATION` a `MAPSET`, v ktorých bude daný príkaz spustený. V tomto prípade je toto dosiahnuté pomocou zapísania týchto informácií do súboru `.grassrc6`. Pretože je výsledný skript spúšťaný vždy v dočasnom priečinku konkrétnej úlohy, nemôže dôjsť k zámene tohoto súboru. Parameter `%2` určuje cestu k dátovej základni GRASS GIS, parameter `%3` špecifikuje názov použitej `LOCATION` a parameter `%4` názov použitej `MAPSET`. Posledným parametrom je `%5`, za ktorý je nahradený príkaz pre spustenie konkrétneho modulu GRASS GIS doplnený o mechanizmy zachytávajúce jeho výstupy a návratový kód.

Dočasné vytváranie `LOCATION` poskytuje spustenej úlohe priestor pre prevádzanie výpočtov, do ktorého nebudú ostatné úlohy zasahovať. Cielom služby SmartWPS je však nielen poskytnúť priestor pre spustenie procesu a manipulovanie so vstupmi pre neho definovanými, ale aj umožniť prácu s dátami uloženými v dátovom úložisku geografického informačného systému. Z tohoto dôvodu v skripte používané parametre pre `LOCATION` a `MAPSET` bude môcť pre každý spúšťaný príkaz definovať autor procesu. Nebudú teda vždy automaticky nastavené na dočasné štruktúry vytvorené pre práve spustenú úlohu, ako by sa na prvý pohľad dalo predpokladať.

Podstatnou súčasťou prepojenia implementovanej služby s geografickým informačným systémom je aj určenie používaných formátov pre reprezentáciu priestorových dát. Jedným zo štandardizovaných a v prostredí OGC webových služieb tiež často používaných formátov je formát GML[4]. Ako uvádza kapitola 2.2, webové služby OGC presne nešpecifikujú formáty pre reprezentáciu geografických dát. Implementovaná služba SmartWPS pri práci s geografickými dátami v rámci procesov nijako nevyužíva prevod do interných dátových štruktúr, a teda spracovanie geografických dát je plne v réžii implementácie procesu. I napriek tomu však vo väčšine prípadov postačuje využitie textovej reprezentácie geografických dát a použitie formátu GML je vyhovujúce. Podpora tohoto formátu je implementovaná

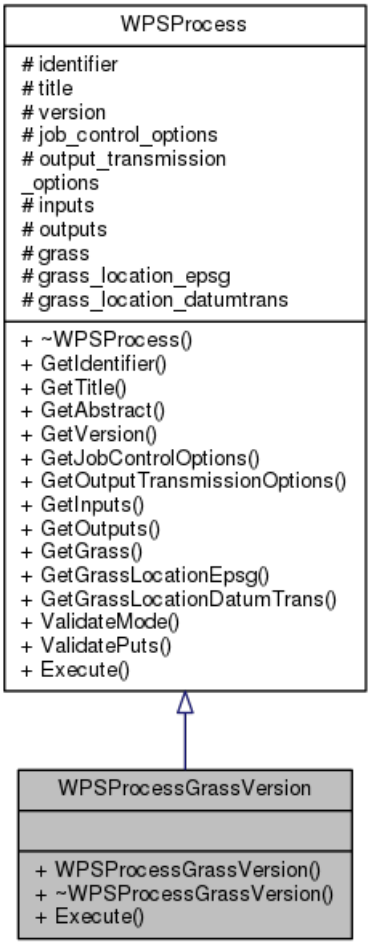
[4] GML: Geography Markup Language.

priamo v systéme GRASS GIS, a preto je tento formát používaný v rámci ukážkových procesov, ktorým sa venuje nasledovná kapitola.

3.2.4 Užívateľské procesy

Výsledná aplikácia by mala umožňovať jednoduché rozširovanie svojej funkcionality o ďalšie WPS procesy. S prihliadnutím na doteraz zvolené technológie – najmä implementačný jazyk a charakter aplikácie ako jednoliateho celku – použijem pre tento účel mechanizmus dynamicky zavádzaných knižníc. Jednotlivé WPS procesy budú implementovať špecifikované rozhranie. Každý z nich bude implementovaný v samostatnej knižnici, ktorej zavedenie po spustení bude vyžiadané povolením procesu v konfiguračnom súbore serverovej aplikácie.

Rozhranie, ktoré budú procesy implementovať do značnej miery reflektuje štandard OGC WPS. V konštruktore príslušnej triedy sa nastavia zdedené atribúty: identifikátor, názov, popis, naplní sa zoznam vstupov a výstupov a používaných formátov. Ďalšou požiadavkou je preťaženie virtuálnej metódy `Execute`, ktorá implementuje výpočtovú funkcionalitu daného procesu. Je volaná po vytvorení a následnom spustení úlohy. Diagram dedičnosti na Obrázku 3.6 popisuje implementáciu rozhrania procesom reprezentovaným triedou `WPSProcessGrassVersion`



Obr. 3.6: Diagram dedičnosti znázorňuje implementovanie rozhrania WPS procesu.

3.3 Klientská aplikácia

Pre účely klientskej aplikácie implementovanej služby SmartWPS, bol navrhnutý a implementovaný programový modul, ktorý je možné jednoducho zaviesť do akejkoľvek klientskej aplikácie implementovanej v prostredí Qt[5]. Voľbou tejto sady knižníc pre implementáciu klientskej aplikácie sledujem hlavne jednoduchú a účelnú tvorbu aplikácie s grafickým užívateľským rozhraním.

Klientská aplikácia pre používanie službou poskytovanej výpočtovej funkcionality musí podporovať:

- Získanie popisných informácií o službe a zoznamu poskytovaných procesov, s využitím WPS operácie `GetCapabilities`.
- Detailné popisovanie procesov a ich vstupných a výstupných štruktúr, s využitím WPS operácie `DescribeProcess`.
- Prostriedky pre spúšťanie procesov a zobrazovanie získaných výsledkov, s využitím WPS operácie `Execute`.

Implementovaný programový modul bude pre jednotlivé úkony konštruovať požiadavky na uvedené operácie a pomocou HTTP požiadavky ich odosielať na službu SmartWPS. Komunikácia medzi serverovou a klientskou časťou tak prebieha výhradne pomocou štandardných operácií Web Processsing Service 2.0. Pre spracovanie odpovedí je potrebné spracovať prijaté XML dáta a odpoveď v zrozumiteľnej forme zobraziť užívateľovi. Implementovaný programový modul je možné použiť samostatne v rámci inej aplikácie, a tým do nej zaniesť podporu pre využívania WPS služby.

3.4 Ukážkové procesy

Jedným z cieľov vymedzených v kapitole 2.5 je aj implementácia ukážkových procesov. Procesy, ktoré môžu byť použité so službou SmartWPS je možné na základe spôsobu ich výpočtu rozdeliť do troch skupín:

- Procesy realizujúce výpočet pomocou kódu v jazyku C++.
- Procesy používajúce pre výpočet sprístupnené moduly geografického informačného systému GRASS GIS.
- Procesy používajúce pre výpočet sprístupnené moduly GRASS GIS a zároveň dátovú základňu GRASS GIS (perzistentne uložené geografické dáta).

Pre ukážku možností služby bol pre každú skupinu navrhnutý a implementovaný jeden proces. Bližšie sa im venujú nasledovné podkapitoly.

3.4.1 Vzdialenosť dvoch zemepisných súradníc

Zástupcom skupiny procesov, ktoré implementujú výpočet iba pomocou jazyka C++ je proces pre výpočet vzdialenosti dvoch bodov určených zemepisnými súradnicami. Pre výpočet vzdialenosti používa *Haversinov algoritmus* [18]. Proces pre svoj výpočet vyžaduje štyri vstupy: zemepisnú šírku prvého bodu, zemepisnú dĺžku prvého bodu, zemepisnú šírku

[5]Dostupné na https://www.qt.io.

druhého bodu a zemepisnú dĺžku druhého bodu. Všetky vstupy používajú dátový typ WPS `LiteralData` a vyžadujú hodnoty OWS dátového typu `Double`. Výstupom je vypočítaná vzdialenosť medzi súradnicami v kilometroch, ktorá je tiež typu `LiteralData` a `Double`. Proces v tomto prípade pre kontrolu vstupov využíva validačné mechanizmy implementované v službe SmartWPS, ktoré sú bližšie popísané v kapitole 4.3.

Haversinov algoritmus použitý pre výpočet vzdialenosti pre zjednodušenie predpokladá guľovú plochu planéty Zem. Postup tak vlastne počíta *ortodrómu*, čo je najkratšia spojnica dvoch bodov na guľovej ploche. Zdroj [17] diskutuje voľbu hodnoty vstupnej konštanty – polomeru Zeme – pričom sa odkazuje na odporúčania *International Union of Geodesy and Geophysics*. Inštitúcia odporúča pre reálne aplikácie používať hodnotu 6371 kilometrov a preto bola táto hodnota použitá aj v implementácii tohoto procesu.

3.4.2 Buffering vstupnej geometrie

Použitie nástroja GRASS GIS robí z implementovanej služby skutočne robustný nástroj použiteľný pre širokú škálu geografických výpočtov. Jedným z takýchto výpočtov je aj tzv. *buffering*. Ako uvádza [1], operácia *buffer* vyžaduje dva parametre – vstupný geometrický objekt a vzdialenosť – a vytvára okolo daného objektu novú geometriu, ktorá je zväčená o vzdialenosť zadanú v metroch. Výsledkom operácie je teda vždy plocha, pričom vstupom môže byť akákoľvek geometria. Implementovaný WPS proces vyžaduje vstupnú geometriu typu plocha.

```
v.in.ogr -o output=input_geometry dsn=./input_file.gml

v.buffer input=input_geometry output=buffered_geometry type=area
        distance=500

v.out.ogr input=buffered_geometry dsn=./output_file format=GML
```

Účelom implementovaného procesu je *buffering* vstupnej geometrie o zadanú vzdialenosť v metroch. Vstupnú geometriu procesu je potrebné pred samotným výpočtom importovať do dátového úložiska GRASS GIS. To je prevedené modulom `v.in.ogr`[6]. Uvedený príkaz je spustený v dočasnom adresári príslušnej úlohy. Následne je prevedené samotný buffering, pomocou modulu `v.buffer`[7], ktorého parameter `distance` obsahuje vzdialenosť v jednotkách mapy, ktoré sú špecifikované súradnicovým systémom `LOCATION`, v ktorej sa príkaz spúšťa. Preto je dôležité, aby mal autor procesu k dispozícii nástroje, pomocou ktorých môže tieto parametre dočasne vytváranej `LOCATION` ovplyvniť. V tomto prípade, ak má byť vzdialenosť zadávaná v metroch, musí byť zvolený súradnicový systém, ktorý takéto jednotky používa. Typickým príkladom je mapová projekcia UTM[8], ktorá je použitá aj v tomto prípade. Príkaz `v.out.ogr`[9] nakoniec do výstupného súboru zapíše vzniknutú geometriu. Z príkazu je zrejmé, že výstupným formátom je GML. Uvedené príkazy sú vzhľadom na charakter procesu vykonávané v dočasne vytvorenej `LOCATION` pre konkrétnu úlohu.

[6] Dokumentácia dostupná na https://grass.osgeo.org/grass64/manuals/v.in.ogr.html.
[7] Dokumentácia dostupná na https://grass.osgeo.org/grass64/manuals/v.buffer.html.
[8] UTM: Universal Transverse Mercator.
[9] Dokumentácia dostupná na https://grass.osgeo.org/grass64/manuals/v.out.ogr.html.

3.4.3 Prekryv vstupnej geometrie s referenčnou

Nástroj GRASS GIS je možné použiť nie len na vykonávanie výpočtov, ale aj na sprístupnenie dátového úložiska. Príkladom takéhoto použitia je zástupca tretej kategórie procesov, ktorého účelom je „priložiť“ vstupnú geometriu na referenčnú a vrátiť výstupnú geometriu obsahujúcu tie prvky referenčnej geometrie, ktoré sa čiastočne alebo úplne prekrývajú so vstupnou geometriou. Pre realizáciu prekryvu – *overlap* použijem modul `v.select`[10], ktorého parametrami sú `ainput` (referenčná geometria) a `binput` (vstupná geometria). Pred samotným výberom prvkov dochádza ešte ku *bufferingu* vstupnej geometrie, kde je použitý rovnaký prístup ako v prípade predchádzajúceho procesu.

```
v.in.ogr -o output=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_input_geometry
        dsn=./input_file.gml

v.buffer input=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_input_geometry
        output=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_buffered_geometry
        type=area distance=30

v.select ainput=ReferencneSidla
        binput=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_buffered_geometry
        output=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_overlap_geometry

v.out.ogr input=tmp_7AB314B0_583A_4B56_8507_C4B59EE2751D_overlap_geometry
        dsn=./output_file.gml
```

Zásadným rozdielom od predchádzajúceho procesu je ale to, že referenčná geometria je umiestnená v stálej a nie dočasnej `LOCATION`. Moduly programu GRASS GIS neumožňujú pri určení dátového vstupu použiť geometriu z inej ako súčasnej `LOCATION` (najmä kvôli potenciálne rozdielnym súradnicovým systémom). Riešením tohoto problému by preto mohlo byť vytvoriť v dočasnej `LOCATION` kópiu referenčnej geometrie, čo je ale vzhľadom na potenciálne veľký objem dát neefektívne. Spúšťanie uvedených príkazov preto prebieha v rámci tej `LOCATION`, v ktorej je umiestnená referenčná geometria. Aby počas behu úlohy nedošlo ku konfliktu medzi názvami používaných geometrií, sú v jednotlivých príkazoch pre spustenie modulov použité identifikátory príslušnej úlohy. Názvy vstupných a výstupných súborov sa však stále nachádzajú v dočasnom adresári pre konkrétnu úlohu, a preto ich nie je nutné nijako špeciálne odlišovať.

3.5 Konfigurácia

Konfigurácia implementovanej služby je jediným nástrojom pre jej správu prevádzkovateľom. Preto musí služba poskytovať jednoduchý a ucelený prostriedok pre tento účel. Predchádzajúce kapitoly spomenuli viacero konkrétnych konfigurovateľných položiek. Tie je možné rozdeliť do týchto kategórií:

1. Informácie o prevádzkovanej službe.
2. Poskytované procesy.

[10] Dokumentácia dostupná na https://grass.osgeo.org/grass64/manuals/v.select.html.

Označenie	Typ	Popis
`title`	reťazec	Názov WPS služby.
`abstract`	reťazec	Popis WPS služby.
`keywords`	zoznam reťazcov	Kľúčové slová WPS služby.
`fees`	reťazec	Poplatky spojené s využívaním WPS služby.
`access_constraints`	reťazec	Podmienky prístupu k WPS službe.
`provider_name`	reťazec	Názov poskytovateľa.
`provider_site`	reťazec	Web poskytovateľa.
`provider_contact_individual_name`	reťazec	Meno a priezvisko kontaktnej osoby poskytovateľa.
`provider_contact_contact_info_electronic_mail_address`	reťazec	Emailová adresa kontaktnej osoby poskytovateľa.
`processes_folder`	cesta k adresáru	Adresár kde sú umiestnené procesy
`processes`	zoznam reťazcov	Procesy, ktoré majú byť inicializované pri spustení služby.

Tabuľka 3.1: Položky konfiguračného súboru zo sekcie `SERVICE`: ich označenie, dátový typ a popis položiek.

3. Parametre webovej služby

4. Parametre nástroja GRASS GIS.

Nasledovné odseky sa venujú popisu konfiguračného súboru, ktorého ukážka sa nachádza v prílohe A. Prvá kategória poskytuje všeobecné informácie o prevádzkovanej WPS službe. Jedná sa najmä o položky, ktoré vychádzajú zo štandardu OGC WPS 2.0. Položky konfiguračného súboru zo sekcie `SERVICE` sú prehľadne popísané v tabuľke č. 3.1. Súčasťou tejto sekcie sú aj položky z druhej kategórie, ktoré umožňujú definovať poskytované procesy. Položka `processes` obsahuje zoznam *identifikátorov knižníc procesov*, ktoré sa očakávajú v priečinku určenom nastavením `processes_folder`. Detailom dynamického zavádzania a použitia *identifikátorov knižníc procesov* sa podrobne venuje kapitola 4.5.

Tretia kategória zahŕňa položky konfigurujúce webovú službu – a teda webový server, ktorý je súčasťou SmartWPS. Všetky nastavenia sú uvedené v tabuľke č. 3.2. Položka `host` je dôležitá najmä pre jej použitie pri konštrukcii URL adries, ktorými sú odkazované výstupy WPS procesov predávané odkazom. K nastaveniam `references_folder` a `tmp_folder` je dôležité dodať, že WPS služba musí mať dostatočné oprávnenia pre zápis do týchto priečinkov.

Posledná kategória obsahuje nastavenia potrebné pre využitie nástroja GRASS GIS. Jej nastavenia sú uvedené v Tabuľke 3.3.

Celá konfigurácia serverovej aplikácie bude zaznamenaná v konfiguračnom súbore, ktorý aplikácia očakáva v koreňovom adresári z ktorého je spúšťaná. Pre spôsob uloženia volieb bol použitý vlastný textový formát, čo je v tomto prípade výhodné pre jeho jednoduchosť. Zároveň použitie vlastného formátu odstraňuje potrebu ďalších komponent, čo prispieva k prenositeľnosti aplikácie.

Označenie	Typ	Popis
`host`	reťazec	Doménové meno pre webovú službu.
`port`	celé číslo; 1 až 65535	Port pre webovú službu.
`thread_pool`	celé číslo	Počet vlákien pre obsluhu požiadaviek.
`tmp_folder`	adresár ku priečinku	Adresár pre dočasné adresáre bežiacich úloh.
`references_folder`	adresár ku priečinku	Adresár pre perzistentné úložisko referencií.
`references_expiration`	celé číslo	Doba životnosti referencií; v minútach.

Tabuľka 3.2: Položky konfiguračného súboru zo sekcie `WEBSERVER`: ich označenie, dátový typ a popis položiek.

Označenie	Typ	Popis
`path`	cesta k adresáru	Adresár spustiteľných súborov aplikácie GRASS GIS.
`gisbase`	cesta k adresáru	Adresár databázy geografických dát GRASS GIS.
`location`	reťazec	Predvolená `LOCATION`.
`mapset`	reťazec	Predvolená `MAPSET`.

Tabuľka 3.3: Položky konfiguračného súboru zo sekcie `GRASS`: ich označenie, dátový typ a popis položiek.

Kapitola 4

Implementácia riešenia

4.1 Technické požiadavky

4.1.1 Serverová aplikácia

Implementovaná serverová aplikácia služby WPS vyžaduje pre svoju funkcionalitu viacero komponent, či už vo forme doplnkových knižníc alebo externého softvéru. Implementácia serverovej aplikácie bola zameraná na cieľovú platformu – operačný systém Linux. V Tabuľke 4.1 uvádzam popis potrebných komponent a zároveň aj názvy ich balíčkov, ktoré sú určené pre distribúciu Ubuntu.

Prvé tri uvedené komponenty sú potrebné priamo pre preklad zdrojových kódov aplikácie, preto po inštalácii nevyžadujú žiadnu ďalšiu konfiguráciu. V prípade nástroja GRASS GIS je potrebné predpripraviť jednu `LOCATION`, ktorá bude slúžiť ako predvolená pre automatické vytváranie ďalších, dočasných pre jednotlivé úlohy. Vytváranie `LOCATION` je vždy spojené s určením jej súradnicového systému a mapovej projekcie. Implementovaná služba SmartWPS neurčuje žiadne konkrétne požiadavky na použitý súradnicový systém a mapovú projekciu u predvolenej `LOCATION`. Jedinou výnimkou je použitie ukážkových implementácií WPS procesov. Niektoré z nich pre zrozumiteľné použitie vyžadujú, aby bola v predvolenej `LOCATION` použitá konkrétna mapová projekcia – EPSG:32633, ktorá pracuje v UTM zóne 33N a súradnicovom systéme WGS84. Detailný popis pre konfiguráciu GRASS GIS vrátane predvoleného regiónu je uvedený nižšie.

```
projection: 1 (UTM)
zone: 33
  north:       5507968.34
  south:       5381527.7
  east:        712139.79
```

Názov súčasti	**Verzia**	**Názov balíka**	**Použitie**
GNU Libmicrohttpd	0.9.49	`libmicrohttpd10`	HTTP server.
Xerces-C++	3.1.3	`libxerces-c3.1`	Práca s XML.
Libcurl	7.35.0	`libcurl4-gnutls-dev`	Odosielanie HTTP požiadaviek.
GRASS GIS	6.4.3	`grass`	GIS pre výpočty.

Tabuľka 4.1: Zoznam potrebných súčastí serverovej aplikácie.

```
  west:         545664.99

  e-w res:      1002.86024096
  n-s res:      1003.49714286

total rows: 126
total cols: 166
total cells: 20,916
```

Primárne podporovanou platformou pre prevádzku serverovej aplikácie je operačný systém Linux. Služba je však plne funkčná aj na operačnom systéme Mac OS X verzie 10.11.4.

4.1.2 Klientská aplikácia

Klientskú aplikáciu a programový modul pre komunikáciu so službou WPS som implementoval v jazyku C++ s využitím multiplatformovej knižnice Qt, verzie 5.6.0. Klientská aplikácia pre svoju funkcionalitu využíva Qt základné komponenty a nevyžaduje tak dodatočnú inštaláciu akýchkoľvek ďalších súčastí.

4.2 Spracovanie požiadaviek

Kľúčovou časťou implementácie serverovej aplikácie je spracovanie požiadaviek vo forme operácií WPS. Nasledujúce podkapitoly popisujú implementáciu spracovania požiadaviek. Kapitola 4.2.1 popisuje obecné spracovanie prostredníctvom webového rozhrania a kapitoly 4.2.2, 4.2.3 a 4.2.4 detailne rozoberajú implementáciu spracovania jedotlivých operácií. Posledná podkapitola 4.2.5 popisuje implementáciu WPS výnimiek.

4.2.1 HTTP požiadavka

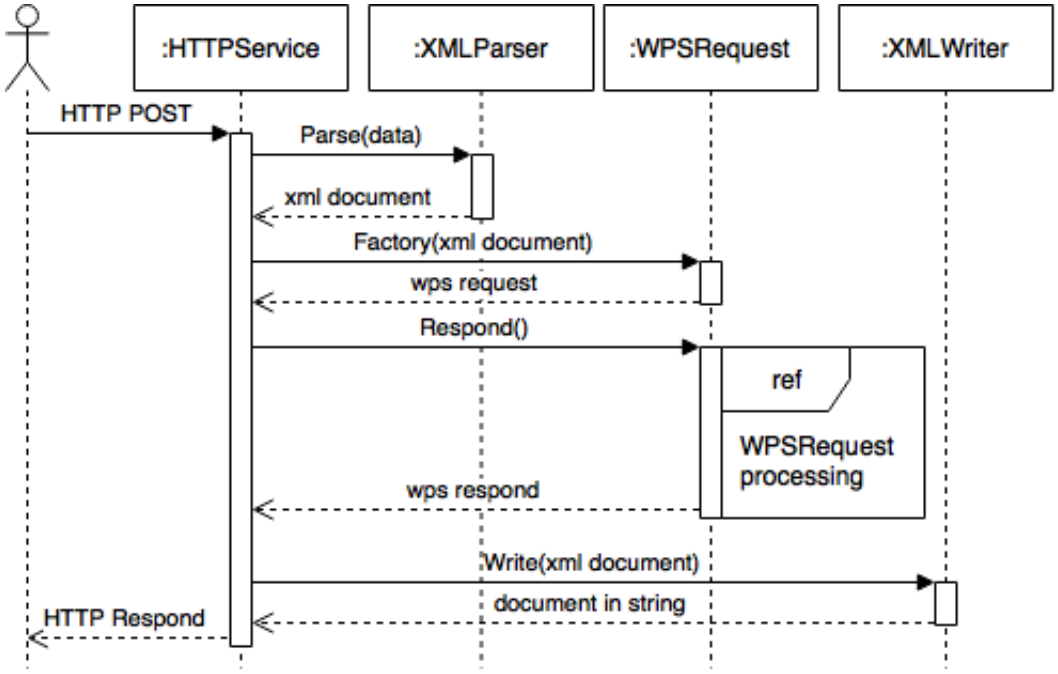
Pre spracovanie HTTP požiadaviek som implementoval wrapper nad rozhraním poskytovaným knižnicou *GNU Libmicrohttpd*. Tento wrapper je reprezentovaný statickou triedou `HTTPServer`.

Pri spustení serverovej aplikácie je trieda `HTTPServer` inicializovaná, čím dôjde k zavolaniu inicializačnej rutiny `MHD_start_daemon` obalenej knižnice. Jej parametrami sú port, režim obsluhy, veľkosť *thread pool* a ukazovatele na funkcie spracovávajúce prichádzajúce požiadavky.

Pri prijatí HTTP požiadavky jej obalená knižnica priradí vlákno z *thread pool*, v ktorom dôjde pomocou volania funkcie `HTTPServer_answer_to_connection` ku získaniu celého tela požiadavky. Následne sa zavolá statická metóda `HTTPServer::Process()`, ktorá obdrží všetky informácie o obdržanej HTTP požiadavke. Serverová aplikácia obsluhuje dve HTTP metódy:

- Metóda *POST*, používanú pre WPS operácie.
- Metóda *GET*, používanú pre získavanie referencií.

Akákoľvek iná metóda je považovaná za nevalidnú požiadavku a webová služba na ňu odpovedá chybovým kódom *405 – Method not allowed*. Problematike spracovania GET metódy a práci s referenciami sa podrobne venuje kapitola 4.6.



Obr. 4.1: Sekvenčný diagram spracovania HTTP požiadavky.

Telo prichádzajúcej požadavky typu HTTP *POST* sa očakáva vo formáte XML. Metóda `HTTPServer::Process()` sa pokúsi o spracovanie prijatej požiadavky do interne reprezentovaného formátu používaného knižnicou *Xerces-C++*. Pre tento účel sa vytvorí inštancia triedy `XMLParser`, ktorá umožňuje takéto spracovanie. Jej metóda `Parse` vráti buď spracovaný objekt typu `DOMDocument`, alebo nulový ukazovateľ, čo značí nevalidnosť prijatých dát. V takomto prípade spracovanie požiadavky končí a `HTTPServer` odpovedá chybovým kódom *400 – Bad Request*.

Validne spracovaný XML dokument je ďalej podrobený analýze na typ operácie. K tomuto účelu slúži statická metóda `WPSRequest::Factory()`, ktorá vykoná test koreňového XML elementu a určí o akú WPS operáciu sa jedná. Následne inicializuje príslušný objekt (spôsob je zrejmý z diagramu dedičnosti uvedenom na Obrázku 3.2), prípadne vráti nevalidnú hodnotu (nulový ukazovateľ). V prípade, že XML dokument nie je validná požiadavka na WPS operáciu, odpovedá webový server chybovým kódom *400 – Bad Request*. Validná požiadavka na WPS operáciu reprezentovaná inštanciou abstraktnej triedy `WPSRequest` môže byť ďalej spracovaná metódou `WPSRequest::Respond()`. Spracovanie podľa druhu operácie popisujú ďalšie kapitoly. Sekvenčný diagram spracovania HTTP požiadavky a vytvorenia `WPSRequest` je znázornený na obrázku č. 4.1.

Spracovanie požiadavky metódou `WPSRequest::Respond()` vracia inštanciu triedy `WPSRequest`, ktorá obsahuje dva atribúty: HTTP návratový kód a XML dokument. Dôvodom začlenenia HTTP návratového kódu do objektu určujúceho odpoveď je to, že počas spracovania odpovede na požiadavku môže dôjsť k vyvolaniu *WPS výnimky*. Popisom spracovania výnimiek a ich jednotlivým druhom sa venuje kapitola 4.2.5. XML dokument je reprezentovaný ukazovateľom na objekt `DOMDocument`, ktorý aplikácia pomocou triedy `XMLWriter` opätovne prevedie na textovú reprezentáciu. Obsah odpovede vo forme reťazca sa spolu s HTTP kódom odpovede predá funkcii `HTTPServer_send_page()`, ktorá zaobaľuje volania

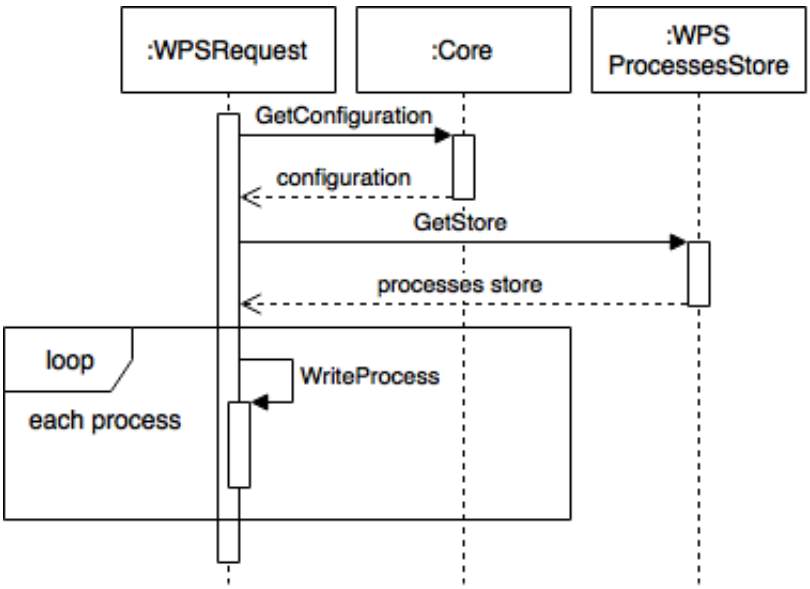
knižničných funkcií *GNU Libmicrohttpd* pre odoslanie HTTP odpovede.

V poslednej fáze dôjde vo funkcii `HTTPServer_send_page()` ku kopírovaniu reťazcovej formy odpovede do buffera pomocou `funkcie MHD_create_response_from_buffer()` vracajúcej typ `MHD_Response`. V prípade ak na začiatku bola prijatá požiadavka obsahujúca nepodporovaný typ HTTP metódy, musí byť podľa štandardu HTTP takáto odpoveď doplnená o hlavičku `Allow`. Jej hodnotu nastaví služba SmartWPS na zoznam podporovaných metód – metódy *GET* a *POST*. Vytvorenú odpoveď `MHD_Response` spolu s HTTP kódom odpovede pripraví na odoslanie funkcia knižnice *GNU Libmicrohttpd* `MHD_queue_response`. Ďalšie spracovanie odpovede prevádza táto knižnica. Tá klientovi odošle odpoveď s poskytnutým telom a HTTP kódom, pričom ju doplní o niektoré štandardne používané HTTP hlavičky.

Jedným z ukazovateľov funkcie, ktoré boli použité pri inicializácii HTTP servera funkciou `MHD_start_daemon` je aj ukazovateľ na funkciu `HTTPServer_request_completed`. Tejto účelom je najmä korektne dealokovať použité zdroje – v tomto prípade pole znakov použité pre telo odpovede. Po navrátení z tejto funkcie je spracovanie HTTP požiadavky dokončené a všetky dočasne vytvorené dátové štruktúry sú dealokované.

4.2.2 Operácia `GetCapabilities`

Validne spracovaný XML dokument obsahujúci WPS operáciu `GetCapabilities` nedefinuje žiadne konkrétne parametre, ktoré by bolo potrebné analyzovať. Ako znázorňuje sekvenčný diagram na Obrázku 4.2, k vytvoreniu odpovede potrebuje `WPSRequest` získať konfiguráciu služby a zoznam všetkých dostupných WPS procesov. Sekvenčný diagram nadväzuje na diagram na obrázku č. 4.1.



Obr. 4.2: Sekvenčný diagram spracovania operácie `GetCapabilities`.

Z objektu `CoreConfiguration`, ktorého atribúty sú konfigurované z konfiguračného súboru pri spustení služby, sa získajú všeobecné informácie o službe. Pre jednotlivé inštancie abstraktnej triedy `WPSProcess` sa pristúpi k atribútom popisujúcim daný WPS proces a tie sa v požadovanom formáte spolu s informáciami o službe vložia do výstupného

XML dokumentu. Ukážky vložených informácií o službe v podobe elementov `ows:Service-Identification` a `ows:ServiceProvider` sa nachádzajú v prílohách B.1.1 a B.1.2. Ukážka elementu `wps:ProcessSummary` popisujúceho stručné informácie o konkrétnom procese je uvedená v prílohe B.1.3.

Schéma výsledného XML dokumentu je v súlade so štandardom OGC WPS 2.0, pričom jeho konštrukcia prebieha pomocou dosadzovania do šablóny. Pomocou metódy `WPSRequest::CreateReponse()` – vracajúcou inštanciu triedy `WPSRespond` – dôjde k vytvoreniu odpovede obsahujúcej XML dokument a návratový HTTP kód. V prípade úspešného spracovania požiadavky je ako návratový kód použitá hodnota *200 – OK*. Inicializovaný objekt `WPSRespond` je vrátený ako odpoveď na volanie metódy `WPSRequest::Respond()`. Obsluha WPS operácie `GetCapabilities` je týmto ukončená a riadenie je opäť predané komponente `HTTPServer`, ktorá prevedie odoslanie odpovede klientovi.

4.2.3 Operácia `DescribeProcess`

Požiadavka WPS operácie `DescribeProcess` obsahuje zoznam identifikátorov procesov, ktorých detailný popis má byť súčasťou odpovede. Z tohoto dôvodu je XML dokument podrobený analýze v rámci spracovania `WPSRequestDescribeProcess`.

Z vnútornej reprezentácie XML dokumentu triedou `DOMDocument` sú extrahované jednotlivé identifikátory WPS procesov. XML element popisujúci identifikátor je znázornený nižšie. Sekvenčný diagram spracovania operácie `DescribeProcess` sa nachádza na obrázku 4.3, pričom pre zjednodušenie nezobrazuje prípad pre viac vyžiadaných procesov.

```
<ows:Identifier>http://example.com/smartwps/distance</ows:Identifier>
```

Následne je pre každý identifikátor vyhľadaný príslušný proces – invokovaním metódy `WPSProcessesStore::Get()` s parametrom identifikátora. Z úložiska procesov `WPSProcessesStore` je vrátený buď ukazovateľ na príslušný objekt `WPSProcess` alebo hodnota `NULL`. V takomto prípade sa pre zadaný identifikátor procesu nenašiel žiaden zaregistrovaný `WPSProcess`, preto spracujúca metóda vyvolá WPS výnimku *No such process*. Spracovanie výnimiek bližšie popisuje kapitola 4.2.5.

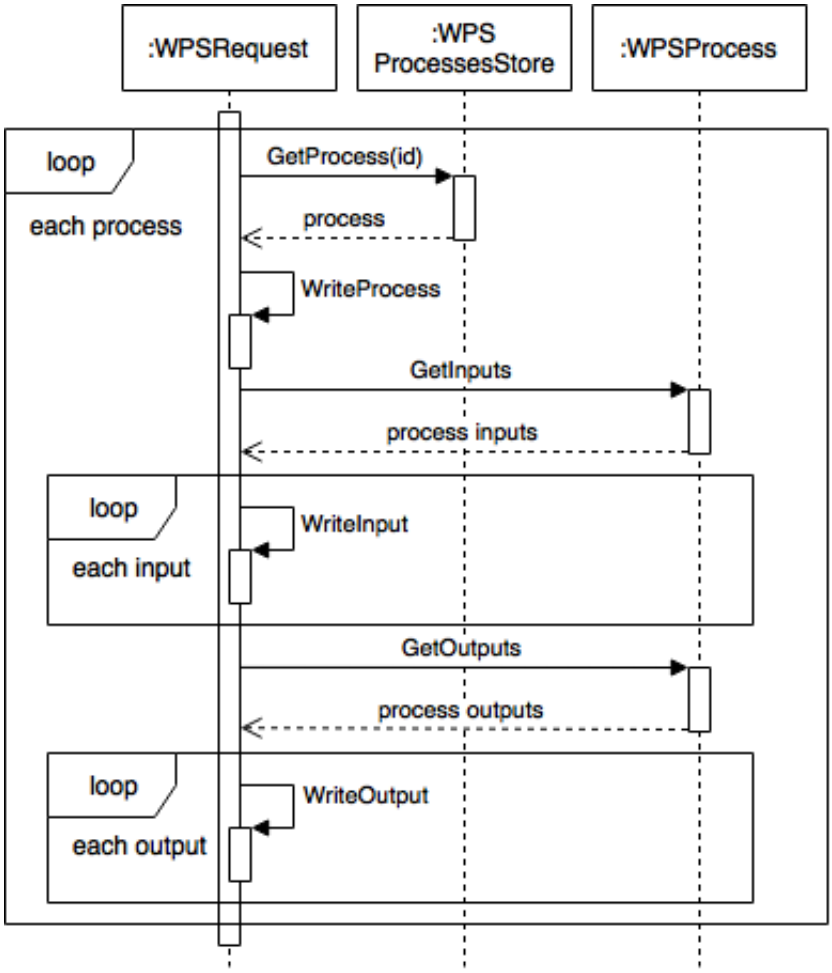
Získaný proces je potrebné pretransformovať do výstupného XML elementu, ktorý bude obsahovať jeho identifikátor, opisné informácie, možnosti spustenia, verziu a zoznamy vstupov a výstupov. Tie ďalej musia obsahovať podporované WPS dátové typy a formáty. Pomocou sady metód triedy `WPSRequest`, počínajúc metódou `ReponseProcessOffering`, sú dosadzovaním do šablóny konštruované popisné elementy jednotlivých procesov. Ukážka takto zkonštruovaného elementu pre konkrétny proces je uvedená v prílohe B.2.1.

Po zkonštruovaní výstupného XML dokumentu je rovnako ako pri operácii `GetCapabilities` vytvorený objekt `WPSRequest`. Nasledovné spracovanie odpovede teda prebieha rovnako ako je uvedené v závere kapitoly 4.2.2.

4.2.4 Operácia `Execute`

Spracovanie WPS operácie `Execute` zahŕňa vykonanie viacerých netriviálnych rutín. V prípade synchrónneho režimu spustenia procesu, ktorý služba SmartWPS implementuje, je potrebné aby všetky obslužné rutiny boli vykonané v rámci odpovede na jedinú požiadavku odoslanú klientom.

Nasledujúce odstavce popisujú postup spracovania požiadavky, avšak samostné spustenie procesu vyžaduje aj prácu s externými objektami – s dočasnými zložkami a súbormi. Z tohoto hľadiska popisuje spustenie WPS procesu kapitola 4.4.1.



Obr. 4.3: Sekvenčný diagram spracovania operácie `DescribeProcess`.

Incializovaný objekt `WPSRequestExecute` vykoná v metóde `WPSRequestExecute::Respond()` analýzu XML dokumentu požiadavky. Z reprezentácie XML dokumentu triedou `DOMDocument` sú extrahované pomocou metód knižnice *Xerces-C++* nasledovné vlastnosti žiadaného spustenia procesu:

- Režim spustenia.
- Identifikátor spúštaného procesu.
- Vstupy procesu:
  - Identifikátor vstupu.
  - Hodnota vstupu alebo URL – predanie referenciou.
- Výstupy procesu:
  - Identifikátor výstupu.
  - Spôsob predania výstupu.

Režim spustenia je reprezentovaný výčtovým typom `WPSProcessExecutionMode`, ktorého hodnoty sú `wps_exec_sync` alebo `wps_exec_async`. Reprezentujú *synchrónny* a *asynchrónny* režim spustenia. WPS klient môže použiť aj hodnotu `auto` – vtedy SmartWPS zvolí *synchrónny* režim spustenia. Identifikátor spúšťaného procesu je reprezentovaný rovnakým XML elementom ako v prípade operácie `DescribeProcess`. Pokiaľ daný proces nebol v službe zaregistrovaný, spracovanie požiadavky vráti WPS výnimku *No such process*. V opačnom prípade sa získa ukazovateľ na príslušný `WPSProcess`, s ktorým pracuje služba ďalej.

Vstupy WPS procesu sú spracovávané z jednotlivých XML elementov `wps:Input`. Dva fragmenty XML dokumentu definujúce vstup sú uvedené nižšie. Prvý z nich určuje konkrétne dáta pre vstup procesu, obsiahnuté v elemente `wps:Data`, ktoré služba očakáva v znakovej forme a následne ich extrahuje do reťazca. Pre tento účel inicializuje inštanciu triedy `WPSParamIn`[1]. V prípade, že daný vstup je definovaný referenciou pomocou URL – znázorňuje druhý fragment XML dokumentu – je inicializovaný opäť objekt `WPSParamIn`, ktorý sa pokúsi získať obsah HTTP dokumentu dostupného na URL adrese prítomnej v atribúte `xlink:href`. Pre tento účel použije služba HTTP metódu *GET*. Pri bezchybnom získaní HTTP dokumentu je za hodnotu `WPSParamIn` použitý jeho obsah. Pri nedostupnosti zdroja dôjde ku vyvolaniu programovej výnimky, ktorej spracovanie inicializuje WPS výnimku *Data not accessible*.

```
<wps:Input id="ORIGINAL">
    <wps:Reference
        xlink:href="http://example.com/geo/geometry.gml"/>
</wps:Input>

<wps:Input id="WITH">
    <wps:Data>Sidla</wps:Data>
</wps:Input>
```

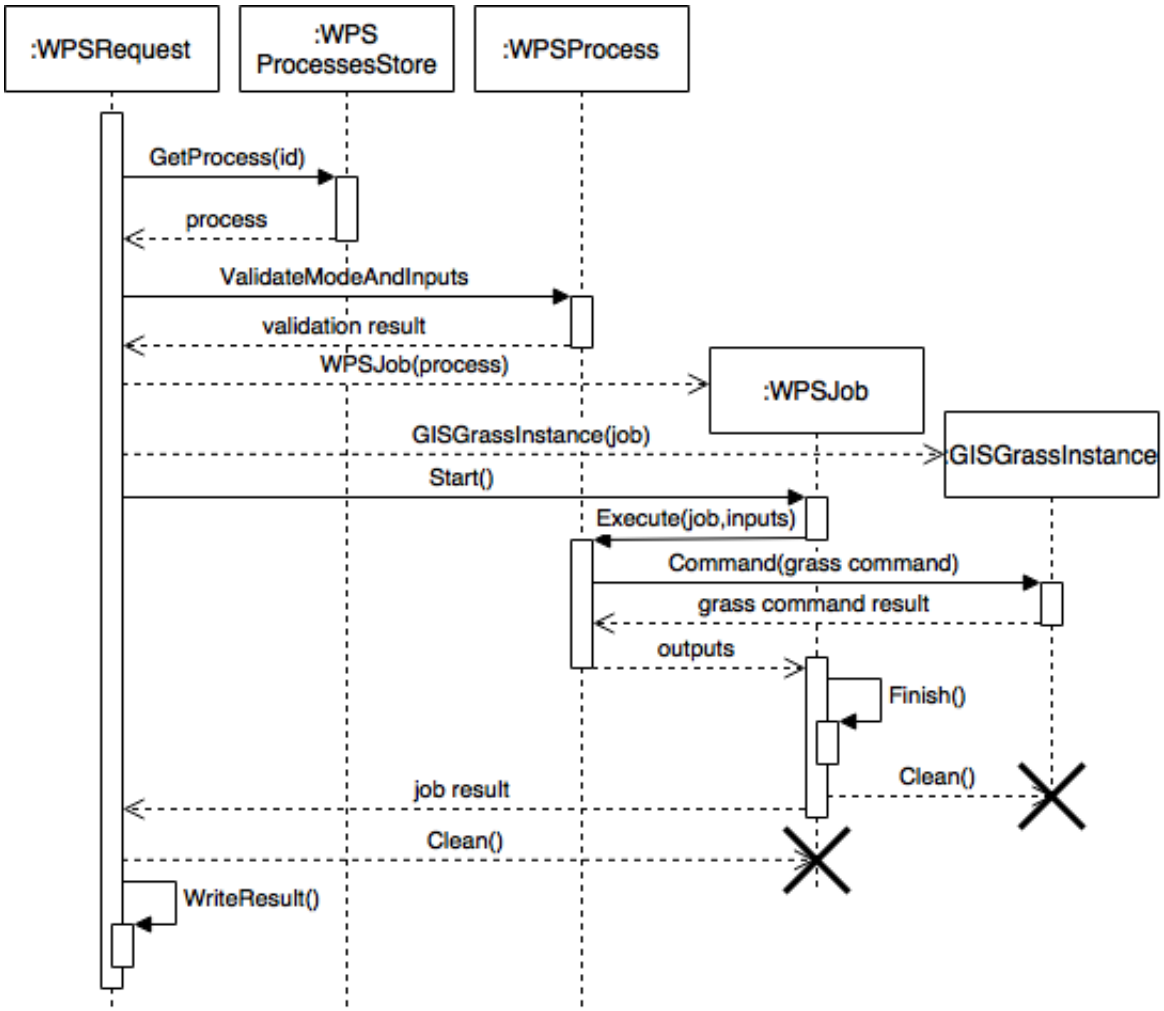
Z XML elementu `wps:Output` je extrahovaný identifikátor daného vstupu a z atribútu `transmission` spôsob navrátenia výsledku. Ten môže nadobúdať hodnoty `value`, pre vrátenie výstupu hodnotou, alebo `reference` pre vrátenie URL referencie na hodnotu výstupu. Tieto informácie sú reprezentované jednotlivými inštanciami triedy `WPSParamOut` vo forme jej atribútov.

```
<wps:Output id="DISTANCE" transmission="value"/>

<wps:Output id="OVERLAPPED" transmission="reference"/>
```

Ako je zrejmé aj zo sekvenčného diagramu na Obrázku 4.4, ďalším krokom je validácia parametrov spustenia procesu. V prvom kroku dôjde ku kontrole `WPSProcessExecutionMode` pre daný proces. Požiadavkou definovaná hodnota je porovnaná voči povoleným hodnotám, ktoré určuje proces na základe atribútu `job_control_options` triedy `WPSProcess`. Overenie režimu spustenia prevádza metóda `WPSProcess::ValidateMode()` abstraktnej triedy `WPSProcess`. V prípade nepodporovaného režimu spustenia je vyvolaná WPS výnimka *No such mode*.

[1] `WPSParamIn` a `WPSInput` sú súvisiace, nie však totožné triedy. `WPSParamIn` slúži pre uchovanie hodnoty zadaného vstupu pri spúšťaní procesu pričom `WPSInput` určuje jeho identifikátor atribúty, formáty a dátový typ.



Obr. 4.4: Sekvenčný diagram spracovania operácie `Execute`.

Druhý krok kontroly parametrov spustenia procesu validuje zadané vstupy a výstupy. Aby bola práca služby v súlade so štandardom OGC WPS 2.0, musí byť skontrolovaná prítomnosť jednotlivých vstupov a výstupov, a zároveň vylúčená prítomnosť vstupov a výstupov nepodporovaných. Metóda `WPSProcess::ValidatePuts()` abstraktnej triedy `WPSProcess` po zavolaní porovná identifikátory v zozname vstupov procesu `WPSInput` s tými uloženými v požiadavke, uloženými v zozname objektov `WPSParamIn`. Rovnakým spôsobom sa porovnajú požadované výstupy `WPSParamOut` voči výstupom definovaným procesom `WPSOutput`. Pri vstupoch procesu sa navyše pracuje s intervalovým ohraničením povoleného počtu ich vloženia. Výsledok tejto validácie sa riadi nasledovným rozdelením:

1. V prípade vstupu `WPSParamIn`, ktorý sa nepodarilo nájsť medzi definovanými vstupmi `WPSInput`, dôjde k vyvolaniu výnimky *No such input*.

2. V prípade uvedenia vstupu `WPSParamIn` nepovolený počet krát (určený intervalom), dôjde k vyvolaniu výnimky *Too many inputs*.

3. V prípade, že hodnota uvedeného vstupu `WPSParamIn` typu `LiteralData` nespĺňa

podmienky určené definovaným vstupom `WPSInput` je vyvolaná WPS výnimka *Wrong input data*.

4. V prípade výstupu `WPSParamOut`, ktorý sa nepodarilo nájsť medzi definovanými vstupmi `WPSOutput`, dôjde k vyvolaniu výnimky *No such output*.

5. V prípade uvedenia vstupu `WPSParamOut` nepovolený počet krát (iný ako práve jedenkrát), dôjde k vyvolaniu výnimky *Too many outputs*.

Podmienke uvedenej v bode č. 3 – validácii dátového typu `LiteralData` – sa bližšie venuje kapitola 4.3. V prípade nesplnenia ani jednej z uvedených podmienok pre všetky vstupy alebo výstupy sú vstupy a výstupy *validné*.

Po úspešnej validácii režimu spustenia, vstupov a výstupov procesu môže byť týmito informáciami inicializovaná úloha – `WPSJob`. Parametrami pri jej inicializácii sú spomenuté validované hodnoty a samotný objekt `WPSProcess` o ktorého spustenie je žiadané.

Počas inicializácie `WPSJob` je generovaný náhodný UUID[2] identifikátor úlohy verzie 4. Pre inicializovanú úlohu je vytvorený dočasný adresár. Problematike dočasne vytváraných adresárov pre úlohy sa bližšie venuje kapitola 4.4.1.

Ku inicializácii komunikačného rozhrania s geografickým informačným systémom GRASS GIS dochádza v prípade vyžiadania daným procesom. Sekvenčný diagram na Obrázku 4.4 zobrazuje prípad, kedy je nástroj GRASS GIS inicializovaný. Inštancia triedy `GISGrassInstance` je previazaná s objektom `WPSJob` a pripravená na použitie počas spustenia procesu. Dočasným štruktúram vytváraným v prostredí nástroja GRASS GIS sa detailne venuje kapitola 4.4.2. V prípade neúspešnej inicializácie `GISGrassInstance` je vyvolaná WPS výnimka *Internal server error*.

Ukazovateľ na inicializovanú úlohu je uložený do skladu úloh `WPSJobsStore` pomocou metódy `Register()`, čím dôjde k *zaregistrovaniu* úlohy. Po registrácii môže byť spustená metódou `WPSJob::Start()`. Tá po zavolaní nastaví interný príznak spustenia úlohy `is_started` a invokuje preťaženú metódu priradeného procesu `WPSProcess::Execute()`. Úloha tak prejde do spusteného stavu a vlákno vykonáva programový kód užívateľského procesu.

Počas výkonu kódu procesu môže užívateľský kód pristupovať ku inicializovanej komponente `GISGrassInstance` a používať jej metódy:

- `Command()` pre vykonávanie príkazov nástroja GRASS GIS.
- `ReadFile()` pre čítanie súboru z dočasného priečinka úlohy.
- `WriteFile()` pre zápis súboru do dočasného priečinka úlohy.

Od užívateľského procesu sa očakáva, že v metóde `Execute` implementovaného procesu `WPSProcess` (alebo v iných metódach volaných z metódy `Execute`) nastaví hodnoty výstupov procesu. Užívateľský proces nerozlišuje medzi predaním výstupu hodnotou alebo referenciou – to za neho rieši služba po skončení behu úlohy.

Po navrátení z metódy `Execute` je úloha dokončená. Zavolanie metódy `WPSJob::Finish()` spustí záverečnú rutinu úlohy. Počas nej je inicializovaný *expiration time* – čas platnosti výstupov úlohy, ktorý je súčasťou odpovede. Jeho hodnota je závislá na konfiguračnom súbore. Zároveň sú jednotlivé výstupy `WPSParamOut`, ktoré sú požadované vo forme referencie, uložené do interného úložiska referencií čomu sa bližšie venuje kapitola 4.6.

[2] UUID: Universally unique identifier.

Označenie	HTTP kód	Popis
NoSuchProcess	400	Pre zadaný identifikátor neexistuje proces.
NoSuchMode	400	Proces nepodporuje zadaný režim spustenia.
NoSuchInput	400	Zadaný vstup neexistuje.
NoSuchOutput	400	Zadaný výstup neexistuje.
DataNotAccessible	400	Referencovaný vstup nie je dostupný.
SizeExceeded	400	Veľkosť vstupných parametrov je príliš veľká.
TooManyInputs	400	Bolo zadaných príliš mnoho vstupov.
TooManyOutputs	400	Bolo zadaných príliš mnoho výstupov.
ServerBusy	503	Server je preťažený.
StorageNotSupported	400	Nepodporovaný typ predania výstupov.
NoSuchFormat	400	Bol použitý nepodporovaný formát.
WrongInputData	400	Niektoré zo vstupov obsahujú nevalidné dáta.

Tabuľka 4.2: Zoznam a popis WPS výnimiek.

Dokončená úloha `WPSJob` je odregistrovaná zo skladu úloh `WPSJobsStore`. Registrovanie a odregistrovanie úlohy slúži pre prípadné získanie prístupu k úlohe v prípade asynchrónneho výskytu udalosti, ktorá vyžaduje spracovanie. Typickým zástupcom takejto udalosti je požiadavka o *ukončenie služby*. V takomto prípade musí byť úloha korektne ukončená a odstránené všetky dočasne vytvorené štruktúry.

Prevzaté výstupy úlohy reprezentované zoznamom inštancií tried `WPSParamOut` sú vložené do šablóny výstupného XML dokumentu. Každý výstup je zaobalený v XML elemente `wps:Output`, ktorý obsahuje hodnotu výstupu alebo URL adresu referencie. XML dokument odpovede obsahuje aj UUID spustenej úlohy v elemente `wps:JobID` a čas exspirácie úlohy v elemente `wps:ExpirationDate`. Ukážky fragmentov XML dokumentov obsahujúce elementy `wps:Result` sa nachádzajú v prílohách B.3.1 a B.3.2.

Z vytvoreného XML dokumentu je vytvorený objekt `WPSRequest` a spracovanie odpovede pokračuje ako je uvedené v závere kapitoly 4.2.2.

4.2.5 Spracovanie výnimiek

Koncept WPS výnimiek pochádza priamo zo štandardu OGC WPS 2.0. Štandard určuje konkrétne výnimky, ktorými služba reaguje na nesprávne stavy alebo chybové udalosti počas spracovania operácií.

Väčšina výnimiek už bola v kontexte popísaná v predchádzajúcich kapitolách. Tabuľka 4.2 súhrnne popisuje všetky štandardom definované WPS výnimky.

Vyvolaním WPS výnimky sa preruší vytváranie odpovede `WPSRespond`. Namiesto bežne vytvorenej odpovede v podobe odpovedí na jednu z operácií je použitý špecifický XML

dokument popisujúci kód a popis výnimky. WPS výnimky tiež ovplyvňujú HTTP kód odpovede na požiadavku.

Pre vyvolanie je použitá metóda `WPSRequest::Exception()`, ktorej jediným parametrom je kód výnimky, zvolený z výčtového typu

Spracovanie jednotlivých výnimiek typu `WPSException` prebieha podobne ako spracovanie XML dokumentu pre odpoveď. Do pripravenej šablóny sú metódou `WPSExceptionGenerator::Throw()` vložené jednotlivé parametre výnimky: označenie a text. Fragment XML dokumentu obsahujúci element `wps:ExceptionReport` je obsahom prílohy B.4.1. Vytvorená odpoveď je potom doplnená o príslušný chybový HTTP kód a spracovaná rovnako ako iná `WPSRequest`. Spracovanie je popísané v závere kapitoly 4.2.2.

4.3 Validácia `LiteralData`

Štandard OGC WPS 2.0 spoločne so štandardom OGC OWS poskytuje možnosti určenia a definovania dátových typov. Pre dátový typ `LiteralData` poskytuje tiež štandard možnosti pre určenie povolených hodnôt a tak umožňuje ich validáciu pri spracovaní vstupu.

Dátový typ `LiteralData` je určený primitívnym dátovým typom a množinou povolených hodnôt. Služba SmartWPS implementuje dva druhy určenia povolených hodnôt `LiteralData`. Prvým z nich je povolenie akejkoľvek hodnoty daného primitívneho dátového typu. Druhým je určenie hodnôt buď intervalom (týka sa číselných primitívnych dátových typov `Integer`, `Decimal`, `Double` a `Float`), alebo výčtom hodnôt (týka sa primitívneho dátového typu `String`).

Pokiaľ `LiteralData` povoľuje akúkoľvek hodnotu, musí byť možné zadaný vstup procesu previesť na hodnotu takéhoto typu. V prípade, že prevod vstupu nie je možný, je počas validácie vyvolaná výnimka *Wrong input data*.

Intervalové určenie hodnôt pre číselné primitívne dátové typy je odosielané klientovi v odpovedi na operáciu `DescribeProcess`. Fragment XML dokumentu, ktorý je súčasťou detailného popisu procesu znázorňuje element `LiteralDataDomain`.

```
<LiteralDataDomain default="true">
    <ows:AllowedValues>
        <ows:Range>
            <ows:MinimumValue>-90.000000</ows:MinimumValue>
            <ows:MaximumValue>90.000000</ows:MaximumValue>
        </ows:Range>
    </ows:AllowedValues>
    <ows:DataType ows:reference=
             "http://www.w3.org/2001/XMLSchema#double">
        Double
    </ows:DataType>
    <ows:DefaultValue>0.000000</ows:DefaultValue>
</LiteralDataDomain>
```

Minimálna a maximálna povolená hodnota je určená elementom `ows:Range`. Primitívny dátový typ je zvolený z množiny primitívnych dátových typov. Predvolená hodnota je určená elementom `ows:DefaultValue`.

Výčtové určenie povolených hodnôt pre primitívny dátový typ `String` je určené analogicky. Fragment XML elementu `ows:AllowedValues` je znázornený nižšie. Hodnota primitívneho dátového vstupu musí byť jednou z určených povolených hodnôt.

```
<ows:AllowedValues>
    <ows:Value>Sidla</ows:Value>
    <ows:Value>Vodohospodarstvo</ows:Value>
</ows:AllowedValues>
```

V prípade nesplnenia požiadaviek na povolené hodnoty je pre akýkoľvek primitívny dátový typ vyvolaná WPS výnimka *Wrong input data*.

4.4 Správa dočasne vytváraných prostriedkov

Spustenie procesu je v prípade služby Web Processing Service kľúčovým prvkom. Vytvorená úloha musí byť korektne inicializovaná, riadená a následne ukončená, pričom sa všetky jej dočasné štruktúry uvoľnia[3]. Tejto problematike so zameraním na dočasné adresáre a dočasne vytvárané štruktúry nástroja GRASS GIS sa venujú nasledovné podkapitoly.

4.4.1 Dočasné prostriedky pre úlohy

Po prijatí WPS požiadavky typu `Execute` a korektnom spracovaní parametrov spustenia je inicializovaná úloha. Každá úloha je jednoznačne identifikovaná náhodne vygenerovaným UUID verzie 4. Pre prácu so súbormi počas behu úlohy je v dočasnom adresári určenom konfiguráciou vytvorený podadresár.

Vykonávaný proces môže pripravený podadresár využívať počas behu príslušnej úlohy. Jeho účelom je primárne poskytnúť priestor pre prácu so súbormi obsahujúcimi geografické dáta určené pre GRASS GIS, no je možné ho použiť pre akýkoľvek účel. Cestu ku dočasnému adresáru získava proces prostredníctvom inicializovaného objektu bežiacej úlohy. Príslušná úloha tiež poskytuje rozhranie pre prácu s dočasným adresárom a jeho obsahom. Po skončení výpočtu procesu dôjde k odstráneniu dočasného adresára. V prípade vzniku chyby počas behu, alebo vyvolania inej udalosti (napr. ukončenia služby) je adresár takisto odstránený a úloha uvoľní všetky použité zdroje.

Z uvedeného popisu mechanizmov pre dosiahnutie dočasného prostredia pre jednotlivé úlohy je zrejmé, že sa nijako nesústreďujú na dosiahnutie bezpečne[4] oddeleného priestoru pre jednotlivé úlohy. Implementovaná služba si nekladie za cieľ oddeliť pracovné prostredia na bezpečnej úrovni, pretože počíta s predpokladom, že prevádzkovateľ dôveruje zavádzaným a spúšťaným WPS procesom.

4.4.2 Rozhranie pre moduly GRASS GIS

Rozhranie pre implementovanie WPS procesov umožňuje špecifikovať či daný proces požaduje pre realizáciu svojho výpočtu GRASS GIS. V prípade, že tomu tak je, pri vytváraní úlohy incializuje služba dočasné štruktúry potrebné pre použitie tohoto nástroja.

Základom vytvoreného dočasného prostredia je dočasná `LOCATION`. Jej inicializácia je plne v réžii SmartWPS – užívateľské procesy jej vytvorenie nemusia nijako inciovať. Vytvorenie `LOCATION` prebieha za pomoci modulov GRASS GIS. Aby bolo možné spustiť príkaz pre vytvorenie dočasnej `LOCATION`, musí už nejaká `LOCATION` v rámci dátovej základne

[3] Až na prípadné výstupy procesu predávané referenciou.

[4] Na úrovni oprávnení.

GRASS GIS existovať. Pre tento účel služba potrebuje tzv. *predvolenú* `LOCATION` (a takisto aj `MAPSET`), z ktorej je tento príkaz spustený[5].

```
g.proj -c location=7AB314B0-583A-4B56-8507-C4B59EE2751D
```

Spustením uvedeného príkazu vytvorí aplikácia dočasnú `LOCATION`, ktorej mapová projekcia a súradnicový systém bude rovnaký, ako mala predvolená `LOCATION`. Prepínač `-c` použitý v príkaze určuje vytvorenie novej `LOCATION`. Rozhranie pre implementovanie WPS procesov však umožňuje špecifikovať aj iný súradnicový systém a mapovú projekciu dočasne vytváranej `LOCATION`. Príkaz v takom prípade vyzerá nasledovne.

```
g.proj -c epsg=32633 datumtrans=1
       location=7AB314B0-583A-4B56-8507-C4B59EE2751D
```

Dôležitým krokom pri vytváraní dočasnej `LOCATION` je, v prípade použitia GRASS GIS verzie 6, použitie výlučného prístupu. Program GRASS GIS tejto verzie vyžaduje aby ku jednej a tej istej `MAPSET` pristupoval v jednom čas iba jeden užívateľ. Nakoľko sa uvedené týka použitia uvedených príkazov, umožňuje služba SmartWPS prístup ku vytváraniu dočasných štruktúr iba jednej bežiacej úlohe zároveň.

Po skončení výpočtu je dočasná `LOCATION` jednoducho odstránená z dátovej bázy GRASS GIS – `LOCATION` je technicky vzato adresár. Týmto i pri použití nástroja GRASS GIS procesom úloha uvoľní všetky použité zdroje.

4.5 Dynamické zavádzanie užívateľských procesov

Implementovaný mechanizmus pre zavádzanie užívateľských procesov pozostáva z dvoch častí:

1. Dynamického zavedenia externej knižnice.
2. Vytvorenie inštancie triedy zavedeného procesu, a jeho registrovanie.

Dynamické zavádzanie knižnice je implementované s využitím POSIX knižnice `dlfcn.h`. Volaním funkcie `dlopen` služba otvorí príslušnú knižnicu zo zadaného súboru. Pre presnejšie zavádzanie užívateľských procesov sú procesy zavádzané na základe *identifikátora knižnice procesu*. Identifikátor knižnice procesu používa implementovaná služba pre správu aktuálne zavedených dynamických knižníc. Spravidla je tento identifikátor odvodený od názvu alebo identifikátora WPS procesu použitý pre jeho invokáciu službou. Názov zavádzanej knižnice musí obsahovať identifikátor knižnice procesu a presne spĺňať nasledovný formát[6].

```
libsmartwps_process_%identifikátor knižnice procesu%.(so|dylib)
```

Po otvorení knižnice zo súboru služba zavolaním funkcie `dlsym` vyhľadá adresu symbolu. Vzhľadom na to, že WPS proces je trieda implementovaná v jazyku C++ a knižnica `dlfcn.h` je určená pre použitie s jazykom C, obsahuje užívateľský proces incializačnú funkciu. Jej názov je opäť určený identifikátorom knižnice procesu. Funkcia `dlsym` sa použije pre vyhľadanie a zavolanie tejto funkcie. Príklad takejto funkcie je uvedený nižšie.

[5] GRASS GIS od verzie 7 poskytuje pre tento účel jednoduchší spôsob – vytvoriť dočasnú `LOCATION` je možné vytvoriť bez potreby predvolenej. Implementáciou služby SmartWPS sa však zameriavam aj na podporu GRASS GIS verzie 6.

[6] Prípona knižnice je určená operačným systémom, na ktorom bola serverová aplikácia kompilovaná.

```
extern "C" std::shared_ptr<%názov triedy WPS procesu%>
create__%identifikátor knižnice procesu%()
{
    return std::make_shared<%názov triedy WPS procesu%>();
}
```

Inicializačná funkcia vytvorí ukazovateľ na inštanciu triedy príslušného procesu a ten vráti. Následne je tento proces zaregistrovaný do zoznamu procesov. V prípade problému s otvorením súboru knižnice alebo vyhľadaním inicializačnej funkcie je prevádzkovateľ služby informovaný o chybe.

Služba SmartWPS umožňuje dynamické zavádzanie užívateľských procesov realizovať buď pri spustení – zadaním identifikátora knižnice procesu do konfiguračného súboru – alebo počas behu služby zadaním príkazu na štandardný vstup. Spustená aplikácia tento príkaz spracuje a pokúsi sa o načítanie a zaregistrovanie procesu. Podobne je možné WPS proces za behu služby aj odregistrovať. Túto funkcionalitu pokladám za dôležitú, nakoľko prispieva ku jednoduchéu experimentovaniu so službou a implementácii vlastných procesov užívateľom.

```
load %identifikátor knižnice procesu%

unload %identifikátor knižnice procesu%
```

4.6 Správa referencií

Potrebnou súčasťou WPS služby je, vzhľadom na vyžadovanú podporu predávania výstupov procesu odkazom, aj možnosť vystavovať a odkazovať výstupné dáta. Samotný štandard nijako nešpecifikuje ako by mala byť táto funkcionalita implementovaná.

Pre tento účel som implementoval jednoduché úložisko výstupných dát, ktoré umožňuje:

- Z výstupov WPS procesov vytvárať perzistentne uložené dátové súbory.
- Prostredníctvom HTTP serveru na požiadanie sprístupňovať obsah týchto súborov.
- Automaticky riadiť odstraňovanie týchto súborov na základe vypršania ich životnosti.

Po dokončení výpočtu procesu sa tie získané výstupné dáta ktoré majú byť vrátené hodnotou vložia priamo do odpovede na požiadavku `Execute`. Výstupy, ktoré majú byť navrátené odkazom uloží správa referencií do adresára nastaveného konfiguráciou. Namiesto dát je potom do odpovede vložený URL odkaz, pod ktorým budú tieto výstupné dáta dostupné počas ich životnosti.

Adresár referencií obsahuje okrem jednotlivých dátových súborov aj *riadiaci súbor*, v ktorom sú evidované dostupné výstupné dáta. Evidencia zahŕňa názov príslušného súboru a čas kedy dôjde k *vypršaniu* dát – vtedy služba dáta automaticky odstráni. Životnosť výstupných dát je nastaviteľná prostredníctvom konfiguračného súboru.

Klientská, alebo akákoľvek iná aplikácia (webový prehliadač, iná WPS služba) sprístupní príslušnú referenciu odoslaním HTTP GET požiadavky na obdržanú URL. O spracovanie tejto požiadavky sa v SmartWPS postará vytvorený webový server. Odpoveďou sa sprístupní príslušná referencia odoslaním dátového súboru. V prípade nedostupnosti požadovanej referencie odpovie webový server chybovým kódom *404 – Not Found*.

Implementované úložisko dát používa pre evidenciu dátových súborov *riadiaci súbor*. K nemu však, vzhľadom na konkuretný charakter WPS služby môžu pristupovať súčasne

viaceré aktívne vlákna aplikácie. Potenciálnym problémom som zabránil použitím mechanizmov operačného systému pre zabezpečenie výlučného prístupu.

4.7 Klientská aplikácia

Implementovaná aplikácia s grafickým užívateľským rozhraním poskytuje rozhranie pre prehliadanie procesov, vykonávanie procesov a získavania ich výstupov. Príloha C.1 zobrazuje prvú obrazovku klientskej aplikácie, ktorá slúži pre prehliadanie procesov a základných informácií o službe. Informácie zobrazené na tejto obrazovke sú získané pomocou operácií `GetCapabilities` a `DescribeProcess`.

Po voľbe procesu užívateľ zadáva parametre do textových polí pre zvolený proces. Druhá obrazovka klientskej aplikácie sa nachádza v prílohe C.2. Ako parameter použije užívateľ buď referenciu na dáta (pokiaľ je pri textovom poli znázornená značka „ref“), alebo hodnotu. Následne odošle požiadavku pre vykonávanie procesu, čím sa odošle požiadavka operácie `Execute`. Získané výstupy sú zobrazené v pravej časti obrazovky. V prípade získania referencie môže užívateľ sprístupniť jej obsah priamo vo webovom prehliadači.

Kapitola 5

Testovanie riešenia

Jedným z cieľov práce bolo navrhnutú a implementovanú službu testovať v rôznych konfiguráciách servera a takisto vstupných dát. V tejto kapitole sa sústredím ako na testovanie z pohľadu funkčnosti, tak i z pohľadu výkonnosti riešenia.

Prvou kategóriou prevedených testov boli testy zamerané na funkčnosť riešenia. Aplikácia SmartWPS musí, ako webová aplikácia pracujúca s mnohými komponentami, byť schopná vysporiadať sa ako s internými chybovými stavmi, tak aj s nesprávnymi požiadavkami pochádzajúcimi od potenciálneho klienta. Cieľom týchto testov je overiť či je implementovaná aplikácia schopná zvládnuť nesprávne definované požiadavky od klienta, a za druhé či je schopná riadiť prepojenie na externé súčasti – geografický informačný systém – korektným spôsobom.

Cieľom druhej kategórie testov je experimentálne overiť výkonnosť implementovanej aplikácie. Testovanie výkonnosti je zamerané na rôzne druhy požiadaviek od WPS klienta a zároveň rôzne konfigurácie serverovej apikácie.

Cieľom prevedených testov nebolo testovanie použitých súčastí tretích strán.

5.1 Testovanie funkčnosti

Ako bolo naznačené v predchádzajúcom úvode kapitoly, testovanie funkčnosti sa zameriava na:

- Správne spracovanie chybných požiadaviek od klienta.
- Riadenie prepojenia na externú súčasť – geografický informačný systém GRASS GIS.

Pre uvedené oblasti boli súhrnne prevedené tri testy, ktorých cieľ, priebeh i zhodnotenie výsledku je popísané v nasledovných podkapitolách.

Test č. 1 – Nesprávne požiadavky klienta

Motivácia a očakávaný výsledok

Cieľom testu je overiť, či je implementovaná aplikácia schopná sa korektne vysporiadať s prijatím nesprávne formulovaných požiadaviek. Cieľom je sledovať nielen ošetrenie chybového stavu, ale aj správne oznámenenie chyby v súlade s popisom chybových kódov protokolu HTTP.

Číslo testu	HTTP požiadavka		Odpoveď	Cieľ
	Metóda	Popis tela	HTTP kód	
1	*PUT*	Ľubovoľný obsah.	*405 Method not allowed*	Nepodporovaná metóda.
2	*POST*	Odoslaný obsah HTML formulára.	*400 Bad request*	Nepodporovaný obsah požiadavky.
3	*POST*	XML dokument nepodporovaný WPS.	*400 Bad request*	Nevyhovujúci XML dokument.
4	*GET*	–	*404 Not found*	Neexistujúca referencia.

Tabuľka 5.1: Požiadavky použité pre Test č. 1.

Priebeh testovania

V bode č. 3 boli použité rôzne špecifické požiadavky, ktorých popis sa nachádza v Tabuľke 5.1.

1. Serverová aplikácia je správne inicializovaná a spustená.
2. HTTP server obsluhujúci požiadavky má dostatočnú voľnú kapacitu pre spracovanie požiadavky od klienta.
3. Klient odošle špecifickú HTTP požiadavku.
4. Serverová aplikácia inicializuje spracovanie požiadavky.
5. Serverová aplikácia spracuje prichádzajúcu požiadavku a odpovie na ňu.
6. Klient obdrží odpoveď a z jej obsahu a HTTP kódu je schopný posúdiť úspešnosť jej spracovania.

Zhodnotenie

Prijatá požiadavka je pri spracovaní vždy podrobená analýze – overuje sa HTTP metóda a spracováva sa XML dokument alebo požaduje zdroj – referencia. Služba však detailne rozlišuje medzi jednotlivými chybovými stavmi a tak vždy klientovi HTTP kódom popíše úspešnosť spracovania jeho požiadavky. Zároveň služba spracuje a vyhodnotí aj chybnú požiadavku, ktorá nenaruší jej beh. Z tohoto vyplýva, že služba tomuto testu *vyhovela*.

Test č. 2 – Nesprávne parametre WPS operácie

Motivácia a očakávaný výsledok

Cieľom testu je overiť schopnosť služby spracovávať WPS výnimky tak ako ich popisuje štandard OGC WPS 2.0. Na rozdiel od Testu č. 1 sú teda požiadavky správne definované, pri ich spracovaní sa však predpokladá výskyt chybového stavu. Služba by podľa definície štandardu mala byť schopná správne informovať WPS klienta o vzniknutej WPS výnimke.

Číslo testu	Požiadavka		Výnimka	Cieľ
	Operácia	**Popis**		
1	*DescribeProcess*	Žiadosť o popis procesu podľa jeho identifikátora.	*NoSuchProcess*	Neregistrovaný WPS proces.
2	*Execute*	Žiadosť o spustenie procesu asynchrónne.	*NoSuchMode*	Nepodporovaný režim spustenia tejto služby.
3	*Execute*	Nesprávna hodnota pre dátový typ LiteralData.	*WrongInputData*	Validovanie vstupov typu `LiteralData`
4	*Execute*	Vyžiadanie neexistujúceho výstupu.	*NoSuchOutput*	Neexistujúci výstup procesu.

Tabuľka 5.2: WPS operácie použité pri Teste č. 2.

Priebeh testovania

V bode č. 3 boli použité rôzne špecifické XML dokumenty popisujúce WPS operácie. Ich popis sa nachádza v Tabuľke 5.2.

1. Serverová aplikácia je správne inicializovaná a spustená.
2. HTTP server obsluhujúci požiadavky má dostatočnú voľnú kapacitu pre spracovanie požiadavky od klienta.
3. Klient odosiela špecifickú požiadavku v tvare konkrétnej WPS operácie.
4. Serverová aplikácia inicializuje spracovanie požiadavky.
5. Serverová aplikácia spracuje prichádzajúcu WPS operáciu a odpovie na ňu správne definovaným XML dokumentom.
6. Klient obdrží odpoveď a z XML dokumentu, ktorý je jej obsahom, je schopný posúdiť stav spracovania WPS operácie.

Zhodnotenie

Validne prijatá WPS operácia je podrobená analýze s ohľadom na ňou požadovaný výsledok. V prípade, že danú požiadavku nie je možné z istého dôvodu uspokojiť, je klientovi namiesto výstupu vrátený popis WPS výnimky definovanej štandardom. Nakoľko výsledky testu ukázali, že služba je schopná správne rozlíšiť jednotlivé chybové stavy a pre odpoveď použiť WPS príslušnú výnimku určenú štandardom WPS, služba tomuto testu *vyhovela*.

Test č. 3 – Konfigurácia nástroja GRASS GIS

Motivácia a očakávaný výsledok

Implementovaná služba poskytuje široké rozhranie pre konfiguráciu použitia externého nástroja GRASS GIS. Účelom služby je poskytovať rozhranie pre vykonávanie procesov pomo-

cou definovaných operácií. Nevynucuje preto použitie externých nástrojov – GRASS GIS – jednotlivými procesmi, no spokytuje rozhranie, ktoré to umožňuje.

Cieľom tohoto testu je overiť schopnosť služby SmartWPS správne riadiť využívanie geografického informačného systému GRASS GIS ako externej komponenty. V nasledujúcom teste sú podrobené analýze prípady kde sa predpokladá vznik neočakávaného stavu, ale i prípady kedy sa očakáva korektné spracovanie požiadavky od užívateľa.

Priebeh testovania

Test prebiehal dvoma spôsobmi a to buď s očakávaným využitím komponenty GRASS GIS pri výpočte procesu, alebo s očakávaným prevedením výpočtu iba za pomoci natívneho kódu.

Pre obidve varianty prebiehalo testovanie nasledovným spôsobom. V bode č. 4 je popísaný rozdiel medzi variantami. Popis špecifickej konfigurácie aplikácie je uvedený v Tabuľke 5.3.

1. Serverová aplikácia je špecificky konfigurovaná, korektne inicializovaná a spustená.
2. Služba obdrží požiadavku o vykonanie WPS operácie `Execute`.
3. Prijatá požiadavka je validná a preto úspešne prejde kontrolou identifikátora procesu, režimu spustenia a aj vstupov a výstupov.
4. Požadovaný proces pre svoj výpočet:
   (a) *Vyžaduje* použitie komponenty GRASS GIS.
   (b) *Nevyžaduje* použitie komponenty GRASS GIS.
5. Serverová aplikácia spustí úlohu pre daný proces.
6. Výpočet úlohy je ukončený a je formulovaná odpoveď pre klienta.
7. Klient obdrží odpoveď na WPS operáciu `Execute`.

Zhodnotenie

Na základe výsledkov prevedeného testu je zrejmé, že služba SmartWPS je schopná riadiť používanie externého nástroja GRASS GIS, no zároveň umožňuje aj využívanie služby bez jeho konfigurácie či inštalácie na cieľovom systéme. Ošetrenie chyby pri nesprávne konfigurovanej komponente a jej poznačenie do záznamov o behu služby je z administračného hľadiska pre prevádzkovateľa postačujúce. Na druhú stranu korektné prevedenie procesu, ktorý pre svoj beh nevyžaduje nástroj GRASS GIS ukazuje, že služba sa pre bežiacu úlohu ani nepokúšala vytvoriť dočasné prostredie – dočasnú `LOCATION`. Služba teda pracuje s externou komponentou v súlade s požiadavkami, čím tomuto testu *vyhovela*.

5.2 Testovanie výkonnosti

Testovanie výkonnosti implementovanej služby bolo prevedené prostredníctvom experimentálneho spúšťania WPS procesov, pri rôznej záťaži služby. Spúšťanie WPS procesov bolo prevádzané pomocou WPS operácie `Execute`. Pri špecifických konfiguráciách služby a špecifickej frekvencii boli prevedené experimenty s dvomi druhmi procesov:

Číslo testu	Vyžaduje GIS	Konfigurácia GRASS GIS v službe SmartWPS	Výsledok	Cieľ
1	áno	Nástroj nie je nainštalovaný, alebo je nesprávne nastavená cesta k nemu.	Chyba; detailný popis v logu služby.	Spracovanie chybového stavu vzniknutého nesprávnym nastavením.
2	áno	Konfigurovaná predvolená `LOCATION` neexistuje.	Chyba; detailný popis v logu služby.	
3	nie	Nástroj nie je nainštalovaný, alebo je nesprávne nastavená cesta k nemu.	Korektné prevedenie výpočtu.	Pre proces nevyžadujúci komponentu, táto nie je inicializovaná.
4	nie	Konfigurovaná predvolená `LOCATION` neexistuje.	Korektné prevedenie výpočtu.	

Tabuľka 5.3: Špecifické konfigurácie služby použité pri Teste č. 3.

- Procesy, ktorých výpočet prebieha pomocou natívneho kódu.
- Procesy, ktoré pre výpočet využívajú GRASS GIS.

Pre jednotlivé skupiny procesov a konfigurácie služby bola sledovanou hodnotou priemerná doba obsluhy požiadavky. Jednotlivým skupinám procesov sa ďalej venujú nasledovné kapitoly.

Pre generovanie HTTP požiadaviek na službu som implementoval skript, ktorý umožňuje konfiguráciu jednotlivých parametrov odosielania. Pre odosielanie požiadaviek a meranie času využíva nástroj `curl`. Testovací skript je súčasťou obsahu priloženého disku CD.

Test č. 4 – Meranie výkonu procesu implementovaného natívne

Motivácia a očakávaný výsledok

Cieľom testu je zistiť ako efektívne je schopná služba SmartWPS vykonávať procesy implementované v natívnom kóde, s rôznym nastavením veľkosti skupiny vlákien.

Predpokladom je, že priemerná doba výpočtu tejto kategórie procesov bude podstatne nižšia ako pri druhej kategórii procesov. Zároveň predpokladám, že doba obsluhy bude závisieť od konfigurácie služby, konkrétne od parametra vplývajúceho na efektívnosť súbežného spracovania procesov – počtu použitých vlákien.

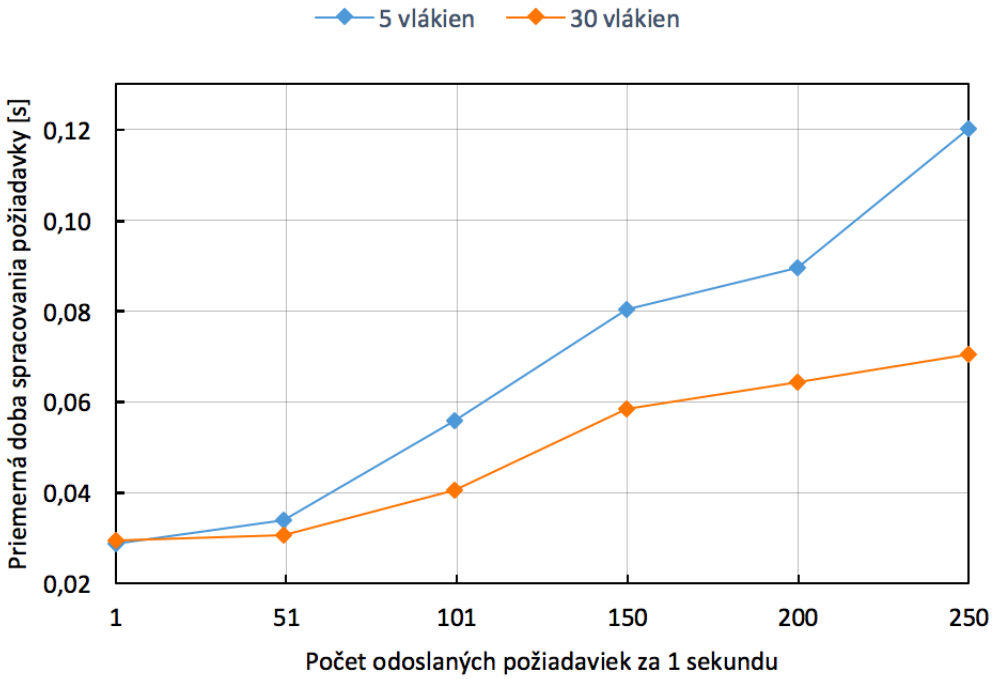
Priebeh testovania

Za zástupcu tejto skupiny procesov som zvolil proces *Vzdialenosť dvoch zemepisných súradníc*, ktorý je bližšie popísaný v kapitole 3.4.1. Test prebiehal nasledovným spôsobom:

1. Serverová aplikácia je špecificky konfigurovaná, inicializovaná a spustená.
2. Testovací skript generuje HTTP požiadavky s validným obsahom, ktoré žiadajú spustenie WPS procesu *Vzdialenosť dvoch zemepisných súradníc*.
3. Klient odosiela požiadavky so špecifickou frekvenciou, pričom každú z nich spustí na pozadí.

4. Po odoslaní požiadavky klient začne merať dobu jej spracovania.
5. Serverová aplikácia spracováva požiadavky.
6. Pre spracovanie požiadaviek sa používa najviac konfiguráciou zadaný počet vlákien.
7. Serverová aplikácia po dokončení výpočtu odošle výsledok klientovi.
8. Klient dokončí meranie doby spracovania a výsledok poznačí do výstupného súboru.

Postup testu bol niekoľko krát zopakovaný pre získanie priemernej doby obsluhy požiadavky. Zároveň testovanie prebiehalo s rôznou frekvenciou odosielania požiadaviek a pre rôzne konfigurovanú službu. Výsledky boli vynesené do grafu, ktorý sa nachádza na Obrázku 5.1.



Obr. 5.1: Závislosť doby obsluhy požiadavky od zaťaženia služby, pri dvoch rôznych konfiguráciách služby. Použitý proces je implementovaný natívne.

Zhodnotenie

Nameraná priemerná doba obsluhy požiadavky na výpočet procesu závisí na vyťažení služby v súlade s vysloveným predpokladom. Zvýšenie počtu obsluhujúcich vlákien v konfigurácii služby sa prejavilo znížením priemernej doby obsluhy. Zároveň je zrejmé, že aplikácia je schopná spracovávať aj relatívne veľké objemy požiadaviek. Z tohoto hľadiska služba testu *vyhovela*.

Test č. 5 – Meranie výkonu procesu využívajúceho nástroj GRASS GIS

Motivácia a očakávaný výsledok

Cieľom testu je zistiť ako efektívne je schopná služba SmartWPS vykonávať procesy implementované pomocou nástroja GRASS GIS, s rôznym nastavením veľkosti skupiny vlákien.

Predpokladom je, že priemerná doba výpočtu tejto kategórie procesov bude podstatne vyššia ako pri predchádzajúcej kategórii procesov. Zároveň predpokladám, že doba obsluhy bude značne ovplyvnená:

- Nutnou inicializáciou dočasného prostredia pre GRASS GIS, ktorá musí prebiehať s výlučným prístupom.
- Získavaním vstupu pre výpočet pomocou referencie.

Vzhľadom na uvedenú skutočnosť predpokladám, že navýšenie počtu použitých vlákien nemusí priniesť značné zrýchlenie obsluhy požiadaviek, nakoľko v rámci aplikácie dochádza k blokujúcim úkonom.

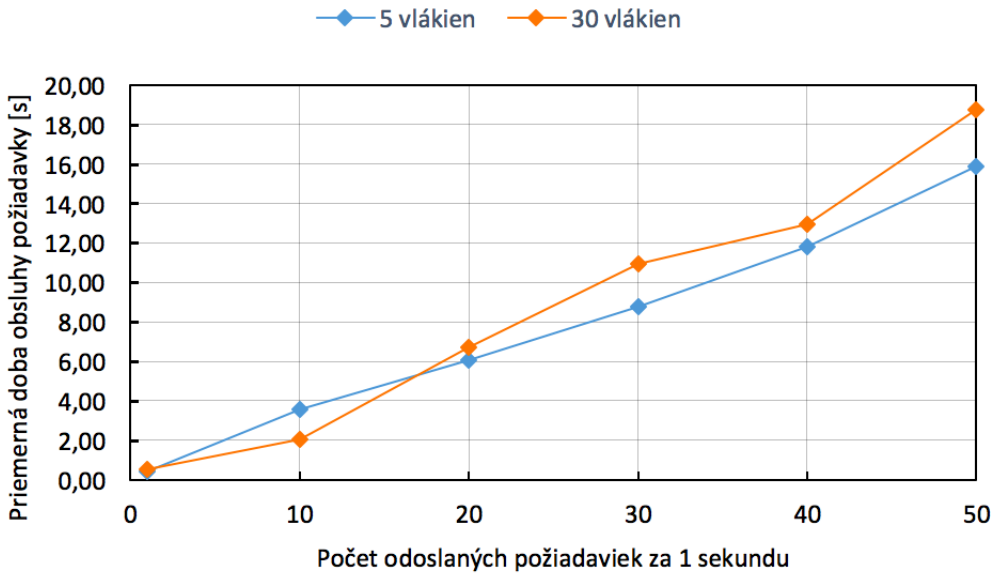
Tento test si nekladie za cieľ testovať efektívnosť prevádzania výpočtov v nástroji GRASS GIS. Cieľom je testovať najmä efektívnosť práce s prostredím nástroja, ktoré prevádza implementovaná služba: vyvorenie dočasnej `LOCATION`, mechanizmus spúšťania príkazov a iné.

Priebeh testovania

Za zástupcu tejto skupiny procesov som zvolil proces *Buffering vstupnej geometrie*, ktorý je bližšie popísaný v kapitole 3.4.2. Test prebiehal nasledovným spôsobom:

1. Serverová aplikácia je špecificky konfigurovaná, inicializovaná a spustená.
2. Testovací skript generuje HTTP požiadavky s validným obsahom, ktoré žiadajú spustenie WPS procesu *Buffering vstupnej geometrie*.
3. Klient odosiela požiadavky so špecifickou frekvenciou, pričom každú z nich spustí na pozadí.
4. Po odoslaní požiadavky klient začne merať dobu jej spracovania.
5. Serverová aplikácia spracováva požiadavky.
6. Pre spracovanie požiadaviek sa používa najviac konfiguráciou zadaný počet vlákien.
7. Serverová aplikácia po dokončení výpočtu odošle výsledok klientovi.
8. Klient dokončí meranie doby spracovania a výsledok poznačí do výstupného súboru.

Postup testu bol niekoľkokrát zopakovaný pre získanie priemernej doby obsluhy požiadavky. Testovanie rovnako ako pri predchádzajúcom teste prebiehalo s rôznou frekvenciou odosielania požiadaviek smerovaných na rôzne konfigurovanú službu. Výsledky boli vynesené do grafu na Obrázku 5.2.



Obr. 5.2: Závislosť doby obsluhy požiadavky od zaťaženia služby, pri dvoch rôznych konfiguráciách služby. Použitý proces využíva pre výpočet nástroj GRASS GIS.

Zhodnotenie

Nameraná priemerná doba obsluhy požiadavky na výpočet procesu závisí na vyťažení služby v súlade s vysloveným predpokladom. Priemerná doba obsluhy je podstatne vyššia ako v prípade Testu č. 4.

Zvýšenie počtu dostupných vlákien v konfigurácii služby sa neprejavilo na zefektívnení práce služby. Je preto zrejmé, že réžia pre prípravu dočasného prostredia nástroja GRASS GIS je veľmi veľká. Na druhú stranu pri použití iného, náročnejšieho druhu výpočtu by použitím externého nástroja, ktorý je pre tento účel optimalizovaný, pravdepodobne došlo k úspore. Táto skutočnosť by však musela byť podrobená ďalším testom.

Na základe potvrdenia vyslovených predpokladov služba tomuto testu *vyhovela*.

Kapitola 6

Záver

V rámci tejto práce som na základe analýzy existujúcich služieb a požiadaviek zadania vytvoril návrh služby SmartWPS, ktorú som následne implementoval. Táto služba umožňuje poskytovať užívateľom definované výpočty geografického charakteru cez štandardizované rozhranie. Navrhnutá služba podporuje štandard OGC WPS vo verzii 2.0.

Počas analýzy situácie v prostredí webových mapových služieb som identifikoval ich kľúčové aspekty, ktoré som zohľadnil pri návrhu služby. Aplikáciu pre server som implementoval ako jednoliatu službu, ktorá poskytuje rozhranie pre prevádzanie výpočtov a zároveň v sebe implementuje aj podporné prostriedky: webový server, správu užívateľských procesov a perzistentné úložisko referencií. Serverovú aplikáciu som implementoval so zameraním na efetivitu využitia dostupných prostriedkov a s dôrazom na konfigurovateľnosť zo strany prevádzkovateľa.

S ohľadom na charakter služby ako prostriedka pre poskytovanie výpočtov geografického charakteru som sa sústredil aj na doplnenie služby o robustný GIS nástroj. Pre tento účel som použil nástroj GRASS GIS, pričom som do implementovanej služby zaviedol rozhranie pre jednoduché využívanie funkcionality tohoto nástroja. Toto rozhranie umožňuje nielen prevádzať výpočty, ale aj GRASS GIS použiť ako databázu pre perzistentné ukladanie geografických dát.

Pre potreby užívateľského rozširovania funkcionality služby na strane prevádzkovateľa som implementoval rozhranie, ktoré umožňuje vytvárať, dynamicky zavádzať a používať pre výpočet užívateľom definované WPS procesy. Tieto procesy je možné pridávať a odstraňovať i počas behu služby SmartWPS. Táto skutočnosť značne prispieva ku konfigurovateľnosti a rozšíriteľnosti služby.

Implementovanou ukážkou užívateľských procesov sú zástupcovia troch kategórií. Prvou z nich sú procesy, ktoré implementujú výpočet procesu natívne, druhou a treťou potom procesy využívajúce externý nástroj GRASS GIS. Ten je použitý buď len pre samotný výpočet, alebo aj ako dátová báza pre ukladanie geografických dát. Implementovanými WPS procesmi demonštrujem široké možnosti spracovania užívateľských procesov, ktoré služba SmartWPS poskytuje.

Klientskú aplikáciu som implementoval vo forme programového modulu, ktorý je doplnený o jednoduché grafické užívateľské rozhranie. Podporuje všetky operácie určené štandardom OGC WPS 2.0 pre prehliadanie, popisovanie a spúšťanie službou poskytovaných WPS procesov.

Implementovaná služba bola podrobená testovaniu sériou základných testov. Kvalitatívne testovanie overilo funkčnosť aplikácie a jej schopnosť korektne sa vysporiadať s chybnými vstupmi. Testovanie výkonnosti zase ukázalo priepustnosť služby za rôznych podmie-

nok.

Počas ďalšieho vývoja služby SmartWPS by bolo vhodné zamerať sa na asynchrónny režim spustenia WPS procesov. Služba by tak mohla realizovať aj časovo náročné výpočty, ktoré nie je možné realizovať synchrónnym modelom výpočtu. V tomto smere by tak zároveň služba doplnila podporu štandardu OGC WPS 2.0 o jej voliteľnú súčasť.

Za ďalšiu možnosť rozvoja implementovanej služby pokladám rozšírenie množiny podporovaných formátov geografických dát a primitívnych dátových typov. To by zo služby spravilo robustnejší nástroj, v ktorom by užívateľ pri implementácii vlastných WPS procesov mohol plne využívať širokú škálu formátov a primárne sa tak sústrediť iba na efektívne riešenie svojej problematiky.

Literatúra

[1] Albrecht, J.: Key Concepts & Techniques in GIS. 2007.

[2] Fette, I.; Melnikov, A.: *RFC 6455: The WebSocket Protocol*. Internet Engineering Task Force, 2011.

[3] Fielding, R.; Reschke, J.: *RFC 7230: Hypertext Transfer Protocol (HTTP/1.1)*. Internet Engineering Task Force, 2014.

[4] Foote, K. E.; Lynch, M.: *Geographic Information Systems as an Integrating Technology: Context, Concepts, and Definitions [online]*. Department of Geography, University of Texas at Austin, 2014-09-11 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL http://www.colorado.edu/geography/gcraft/notes/intro/intro.html

[5] Free Software Foundation, Inc.: *Libmicrohttpd – GNU Project – Free Software Foundation [online]*. Free Software Foundation, Inc., 2016-05-07 [cit. 2016-05-08], [Online; navštívené 8.5.2016].
URL https://www.gnu.org/software/libmicrohttpd/

[6] Grothoff, C.: *The GNU libmicrohttpd Library Reference Manual [online]*. Free Software Foundation, Inc., 2015-12-18 [cit. 2016-05-08], [Online; navštívené 8.5.2016].
URL
https://www.gnu.org/software/libmicrohttpd/manual/libmicrohttpd.html

[7] Haas, H.; Brown, A.: *Web Services Glossary [online]*. W3C, 2004-02-11 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[8] Neteler, M.: *Working with GRASS without starting it explicitly – GRASS-Wiki [online]*. OSGeo, 2015-10-30 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL https://grasswiki.osgeo.org/w/index.php?title=Working_with_GRASS_without_starting_it_explicitly&oldid=22083

[9] Neteler, M.: *Location and Mapsets – GRASS-Wiki [online]*. OSGeo, 2015-12-01 [cit. 2016-05-01], [Online; navštívené 1.5.2016].
URL https://grasswiki.osgeo.org/w/index.php?title=Location_and_Mapsets&oldid=22111

[10] Open Geospatial Consortium: *OGC WPS 2.0 Interface Standard*. Open Geospatial Consortium, 2015.

[11] Open Geospatial Consortium Inc.: *OpenGISm® Web Map Server Implementation Specification*. Open Geospatial Consortium Inc., 2006.

[12] Open Geospatial Consortium Inc.: *OpenGIS Web Feature Service 2.0 Interface Standard*. Open Geospatial Consortium Inc., 2010.

[13] OSGeo: *GRASS GIS – General overview [online]*. OSGeo, 2015-02-24 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL https://grass.osgeo.org/documentation/general-overview/

[14] Petráš, V.: *Development – GRASS-Wiki [online]*. 2015-02-16 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL https://grasswiki.osgeo.org/w/index.php?title=Working_with_GRASS_without_starting_it_explicitly&oldid=22083

[15] Pross, B.: *52°North Web Geoprocessing Service [online]*. 52°North, 2015-03-09 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL https://wiki.52north.org/bin/view/Geoprocessing/52nWebProcessingService?rev=88

[16] Pross, B.: *WPS Architecture Section [online]*. 52°North, 2015-12-11 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL
https://wiki.52north.org/bin/view/Geoprocessing/WPSArchitecture?rev=4

[17] Rosetta Code: *Haversine formula [online]*. Rosetta Code, 2016-04-20 [cit. 2016-05-09], [Online; navštívené 9.5.2016].
URL https://rosettacode.org/wiki/Haversine_formula

[18] Veness, C.: *Calculate distance, bearing and more between Latitude/Longitude points [online]*. Movable Type Scripts, 2015 [cit. 2016-05-09], [Online; navštívené 9.5.2016].
URL http://www.movable-type.co.uk/scripts/latlong.html

[19] Westerholt, R.; Resch, B.: *Asynchronous Geospatial Processing: An Event-Driven Push-Based Architecture for the OGC Web Processing Service*. John Wiley & Sons Ltd, 2014.

[20] Čepický, J.: *PyWPS 3.2.0 Documentation [online]*. 2010 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL http://geopython.github.io/pywps/doc/build/html/

[21] Čepický, J.: *PyWPS Process [online]*. 2010 [cit. 2016-01-07], [Online; navštívené 7.1.2016].
URL http://geopython.github.io/pywps/doc/build/html/process/index.html

Prílohy

Zoznam príloh

Príloha A

Konfiguračný súbor

```
SERVICE
title = "SmartWPS - WPS 2.0"
abstract = "Master thesis."
keywords = "wps", "thesis"
fees = "NONE"
access_constraints = "NONE"
provider_name = "Provider name"
provider_site = "http://example.com/"
provider_contact_individual_name = "Name Surname"
provider_contact_contact_info_electronic_mail_address = "mail@example.com"

processes_folder = "./processes"
processes = "grass_buffer", "grass_overlaps", "distance"

WEBSERVER
host = "localhost"
port = 17000
thread_pool = 5
tmp_folder = "./tmp"
references_folder = "./references"
references_expiration = 1440

GRASS
path = "/usr/lib/grass64"
gisbase = "/home/USER_NOT_CONFIGURED/"
location = "SmartWPSLocation"
mapset = "PERMANENT"
```

Príloha B

Ukážky WPS operácií

B.1 Operácia GetCapabilities

B.1.1 Ukážka elementu ows:ServiceIdentification

```
<ows:ServiceIdentification>
    <ows:Title>SmartWPS - WPS 2.0</ows:Title>
    <ows:Abstract>Master thesis.</ows:Abstract>
    <ows:Keywords>
        <ows:Keyword>wps</ows:Keyword>
        <ows:Keyword>thesis</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WPS</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
</ows:ServiceIdentification>
```

B.1.2 Ukážka elementu ows:ServiceProvider

```
<ows:ServiceProvider>
    <ows:ProviderName>Provider name</ows:ProviderName>
    <ows:ProviderSite
        xlink:href="http://example.com/"/>
    <ows:ServiceContact>
        <ows:IndividualName>Name Surname</ows:IndividualName>
        <ows:ContactInfo>
            <ows:Address>
                <ows:ElectronicMailAddress>
                mail@example.com
            </ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
</ows:ServiceProvider>
```

B.1.3 Ukážka elementu `wps:ProcessSummary`

```
<wps:ProcessSummary
        jobControlOptions="sync-execute"
        processVersion="1.0.0">
    <ows:Title>GRASS buffering</ows:Title>
    <ows:Identifier>
        http://example.com/smartwps/grass_buffer
    </ows:Identifier>
</wps:ProcessSummary>
```

B.2 Operácia `DescribeProcess`

B.2.1 Ukážka elementu `wps:Process`

```
<wps:Process>
    <ows:Title>GRASS buffering</ows:Title>
    <ows:Abstract>Buffering using GIS GRASS.</ows:Abstract>
    <ows:Identifier>
        http://example.com/smartwps/grass_buffer
    </ows:Identifier>
    <wps:Input>
        <ows:Title>GML geometry to buffer.</ows:Title>
        <ows:Abstract/>
        <ows:Identifier>ORIGINAL</ows:Identifier>
        <wps:ComplexData>
            <wps:Format
                mimeType="text/plain"/>
            <wps:Format
                default="true"
                encoding="UTF-8"
                mimeType="application/xml"/>
        </wps:ComplexData>
    </wps:Input>
    <wps:Input>
        <ows:Title>Buffer size.</ows:Title>
        <ows:Abstract/>
        <ows:Identifier>DISTANCE</ows:Identifier>
    </wps:Input>
    <wps:Output>
        <ows:Title>Random GML geometry.</ows:Title>
        <ows:Abstract/>
        <ows:Identifier>BUFFERED</ows:Identifier>
    </wps:Output>
</wps:Process>
```

B.3 Operácia Execute

B.3.1 Ukážka elementu wps:Result – hodnota výstupu

```
<wps:Result
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">

    <wps:JobID>5EA288DC-A050-4CAC-A2FD-AD49AE91B940</wps:JobID>
    <wps:ExpirationDate>2016-05-20T15:17:14.000Z</wps:ExpirationDate>
    <wps:Output id="VERSION">
        <wps:Data>GRASS 6.4.3 (2013) </wps:Data>
    </wps:Output>
</wps:Result>
```

B.3.2 Ukážka elementu wps:Result – referencia výstupu

```
<wps:Result
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">

    <wps:JobID>F2229105-53F7-447B-8824-072A742D8557</wps:JobID>
    <wps:ExpirationDate>2016-05-20T14:47:06.000Z</wps:ExpirationDate>
    <wps:Output id="BUFFERED">
        <wps:Reference
            xlink:href="http://localhost:17000/
            reference/58679ace-6257-4c4e-81ad-bb1a5c73d9a2"/>
    </wps:Output>
</wps:Result>
```

B.4 WPS Výnimka

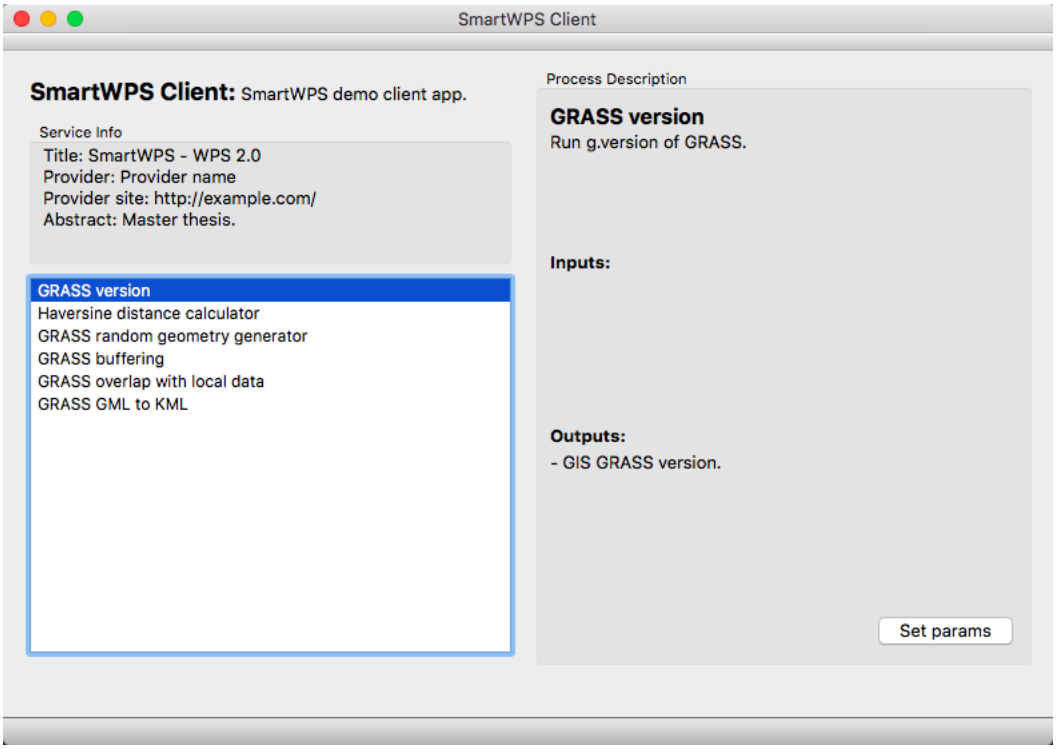
B.4.1 Ukážka elementu ows:Exception

```
<ows:Exception exceptionCode="NoSuchProcess">
    <ows:ExceptionText>
        One of the identifiers passed does not match
        with any of the processes offered by this server.
    </ows:ExceptionText>
</ows:Exception>
```

Príloha C

Ukážky klientskej aplikácie

C.1 Prehliadanie procesov



C.2 Spustenie procesu