

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
KATEDRA MAPOVÁNÍ A KARTOGRAFIE



Doktorská disertační práce

**Využití webových služeb v Katastru
nemovitostí**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
KATEDRA MAPOVÁNÍ A KARTOGRAFIE



Ing. Radek Chromý

**Využití webových služeb v Katastru
nemovitostí**

Using web services in Cadastre

DISERTAČNÍ PRÁCE K ZÍSKÁNÍ AKADEMICKÉHO TITULU PH.D.

DOKTORSKÝ STUDIJNÍ PROGRAM: GEODÉZIE A KARTOGRAFIE

STUDIJNÍ OBOR: GEODÉZIE A KARTOGRAFIE

ŠKOLITEL: PROF. ING. ALEŠ ČEPEK, CSc.

Praha, 15. února 2008

Poděkování

V první řadě bych chtěl poděkovat svému vynikajícímu školiteli a člověku prof. Ing. Alešovi Čepkovi, CSc. Prof. Čepek je skvělý a zapálený odborník ve svém oboru, který mi poskytoval vše potřebné pro úspěšné studium a to nejen po stránce vědecké, ale i po stránce lidské. Díky tomu se mé studium stalo báječnou kapitolou mého života a dalo vzniknout mnoha nezapomenutelným okamžikům.

Dále bych chtěl poděkovat celé své široké rodině za veškerou podporu, kterou mi během celého studia poskytovali.

V neposlední řadě bych chtěl poděkovat všem kolegům a kamarádům, kteří mi nejednou rádu, nápadem či připomínkou přispěli při psaní této práce.

Děkuji všem profesionálním programátorům a znalcům objektově orientovaného programování za shovívavost při studiu mých zdrojových kódů.

Text této práce byl vysázen systémem L^AT_EX. Děkuji všem, kteří se podílejí na jeho vývoji a umožňují tím kvalitní sazbu dokumentů.

Obsah

1	Úvod	1
2	Úvod do webových služeb	3
2.1	Architektura orientovaná na služby - SOA	4
2.2	Architektura webových služeb	5
3	XML	7
3.1	Seznámení s XML	7
3.2	Jmenné prostory	8
3.3	Schémata dokumentů	8
3.3.1	XML Schéma (WXS)	9
3.3.1.1	Jednoduché datové typy	10
3.3.1.2	Komplexní datové typy	11
3.3.1.3	Připojení schématu k dokumentu	12
3.3.2	Relax NG	13
3.3.2.1	Základní vzory	14
3.3.2.2	Pokročilé vzory	15
3.3.2.3	Datové typy	16
3.4	Zpracování XML	17
3.4.1	DOM	18
3.4.2	SAX	19
3.5	XML a Java	20
3.5.1	Java API for XML Processing (JAXP)	21
3.5.2	Java API for XML-based RPC (JAX-RPC)	22
3.5.3	Java API for XML Messaging (JAXM)	23
3.5.4	Java API for XML Registries (JAXR)	24
3.5.5	Java API for XML Binding (JAXB)	26
4	Webové služby	27
4.1	SOAP – Simple Object Access Protocol	27
4.1.1	Vznik SOAPu	27
4.1.2	Protokol SOAP	29
4.1.3	SOAP Fault	31
4.2	WSDL	31
4.2.1	Styly WSDL	33

4.3 UDDI	35
4.4 ebXML	35
5 Současný stav využití webových služeb	38
5.1 Open GIS Consortium	38
5.2 EULIS	40
5.3 INSPIRE	42
6 ISKN	44
6.1 Systém ISKN	45
6.1.1 Rozdělení ISKN	46
6.1.1.1 Aplikace PA — Příprava aktualizace KN	46
6.1.1.2 Aplikace AK — Aktualizace KN	46
6.1.1.3 Aplikace PU — Poskytování údajů	47
6.1.1.4 Aplikace DP — Dálkový přístup	48
6.1.1.5 Aplikace VP — Vzdálený přístup	48
6.1.1.6 Aplikace PP — Poskytování a přebírání podkladů	48
6.1.1.7 Aplikace ST — Sumarizace a statistiky	49
6.1.1.8 Aplikace BP — Bodová pole	50
6.1.1.9 Aplikace SC — Správa číselníků	50
6.1.1.10 Aplikace EX — Vazby na externí systémy	50
6.1.1.11 Aplikace TO — Technicko-organizační zabezpečení provozu ISKN	50
6.1.2 ISKN v číslech	51
6.2 Výmenný formát ISKN – NVF (nový výmenný formát)	51
6.2.1 Struktura NVF	51
6.2.1.1 Hlavička souboru	52
6.2.1.2 Datové bloky	53
6.2.1.3 Koncový znak	54
6.2.2 Typy exportů	54
7 Webové služby pro přístup k datům KN	57
7.1 Současný stav	57
7.1.1 Dálkový přístup	58
7.1.1.1 Webové služby dálkového přístupu	59
7.1.2 Nahlížení do KN	62
7.2 Analýza	62
7.2.1 Uživatelské scénáře	65
7.2.2 Návrh webových služeb	67
7.2.3 Výstupní formát	68
7.2.3.1 Zajímavé rysy XML NVF	69
7.2.3.2 Výstupní formát pro webové služby	70
7.2.4 Zabezpečení	73
7.3 Webové služby	74
7.3.1 Použité technologie	74

7.3.2	Vstupní parametry	75
7.3.3	Chybová hlášení	77
7.3.4	Výstupní formát	78
7.3.5	Získání informací o parcele	79
7.3.5.1	Informace o parcele – jednoznačný identifikátor	79
7.3.5.2	Informace o parcele	80
7.3.5.3	Vyhledání parcel	81
7.3.6	Získání informací o budově	82
7.3.6.1	Informace o budově – jednoznačný identifikátor	83
7.3.6.2	Informace o budově	84
7.3.6.3	Vyhledání budov	85
7.3.6.4	Vyhledání budov – jednoznačný identifikátor parcely	86
7.3.7	Získání informací o jednotce	86
7.3.7.1	Informace o jednotce – jednoznačný identifikátor	87
7.3.7.2	Informace o jednotce	88
7.3.7.3	Vyhledání jednotek	90
7.3.7.4	Vyhledání jednotek – jednoznačný identifikátor budovy	90
7.3.8	Získání informací o řízení	91
7.3.8.1	Informace o řízení – jednoznačný identifikátor	91
7.3.8.2	Informace o řízení	92
7.3.9	Nalezení nejbližšího definičního bodu	94
7.3.9.1	Nalezení nejbližšího definičního bodu parcely	94
7.3.9.2	Nalezení nejbližšího definičního bodu budovy	94
7.3.10	Získání informací o číselnících	95
7.3.10.1	Informace o číselníku	95
7.3.10.2	Seznam číselníků	95
7.3.11	Tvorba klientské aplikace	96
7.3.11.1	Klasický klient	96
7.3.11.2	Mobilní klient	98
8	Závěr	106
8.1	Vlastní řešení	106
8.2	Další vývoj	108
Literatura		110
A	Seznam skupin datových bloků VF ISKN	113
B	Uživatelské scénáře	119
B.1	Uživatelé	119
B.1.1	Registrace	119
B.1.2	Zrušení registrace	120
B.2	Získání informací o parcele	120
B.2.1	Informace o parcele – jednoznačná identifikace	120
B.2.2	Informace o parcele	121
B.2.3	Vyhledání parcel	122

B.3	Získání informací o budově	124
B.3.1	Informace o budově – jednoznačná identifikace	124
B.3.2	Informace o budově	125
B.3.3	Vyhledání budov	126
B.3.4	Vyhledání budov – jednoznačná identifikace parcely	127
B.4	Získání informací o jednotce	128
B.4.1	Informace o jednotce – jednoznačná identifikace	128
B.4.2	Informace o jednotce	129
B.4.3	Vyhledání jednotek	130
B.4.4	Vyhledání jednotek – jednoznačná identifikace budovy	131
B.5	Získání informací o řízení	132
B.5.1	Informace o řízení – jednoznačná identifikace	132
B.5.2	Informace o řízení	133
B.6	Nalezení nejbližšího definičního bodu	134
B.6.1	Nalezení nejbližšího definičního bodu parcely	134
B.6.2	Nalezení nejbližšího definičního bodu budovy	135
B.7	Získání informací o číselnících	136
B.7.1	Seznam číselníků	136
B.7.2	Informace o číselníku	136
C	XML schéma výstupního formátu webových služeb	138
D	WSDL – popis webových služeb	148
E	Mapa webových služeb	160
F	Obsah přiloženého CD	161

Seznam obrázků

2.1	Základem je XML (Zdroj [17]).	4
2.2	Service Oriented Architecture - SOA (Zdroj [33]).	5
2.3	Architektura webových služeb (Zdroj [33]).	6
3.1	Hierarchie zabudovaných datových typů. (Zdroj [38]).	11
3.2	Schéma zpracování XML dokumentu pomocí parseru.	18
3.3	Schéma objektového přístupu ke XML dokumentu.	19
3.4	Schéma událostního přístupu ke XML dokumentu.	20
3.5	Architektura JAXP (Zdroj [27]).	21
3.6	JAX-RPC model (Zdroj [27]).	22
3.7	Stavební bloky architektury JAXM (Zdroj [27]).	24
3.8	Architektura JAXR (Zdroj [27]).	25
3.9	Architektura JAXB (Zdroj [27]).	26
4.1	Vývoj SOAP (Zdroj [27]).	28
4.2	Struktura SOAP zprávy (Zdroj [27]).	29
4.3	Struktura WSDL (Zdroj [27]).	32
4.4	Architektura ebXML (Zdroj [27]).	36
5.1	OpenGIS Architektura (Zdroj [30]).	39
5.2	Architektura systému EULIS (Zdroj [11]).	41
5.3	Architektura služeb INSPIRE (Zdroj [12]).	43
6.1	Zjednodušené logické schéma ISKN.	45
6.2	Hlavní menu aplikace ISKN.	47
6.3	Zjednodušené schéma Dálkového přístupu.	49
7.1	Výpis z KN podepsán elektronickou značkou.	59
7.2	Počet uživatelů DP (Zdroj [39]).	60
7.3	Vývoj počtu přístupů k aplikaci Nahlížení do KN (Zdroj [39]).	62
7.4	Objekt Parcela a její vybrané vazby v systému ISKN.	64
7.5	Fungování webových služeb.	66
7.6	Webové služby pro parcely.	79
7.7	Vyhodnocení výstupního formátu z webových služeb <i>infoOParceleID infoOParcele</i> . Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.	81

SEZNAM OBRÁZKŮ

7.8	Webové služby pro budovy.	83
7.9	Vyhodnocení výstupního formátu z webových služeb <i>infoOBudoveID</i> a <i>infoOBudove</i> . Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.	85
7.10	Webové služby pro jednotky.	87
7.11	Vyhodnocení výstupního formátu z webových služeb <i>infoOJednotceID</i> a <i>infoOJednotce</i> . Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.	89
7.12	Webové služby pro řízení.	91
7.13	Vyhodnocení výstupního formátu z webových služeb <i>infoORizeniID</i> a <i>infoORizeni</i> . Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.	93
7.14	Web Service Proxy	96
7.15	Tvorba klientské části pomocí průvodce v JDeveloperu (v 11.1.1.0)	97
7.16	Vytvořené zdrojové kódy klientské části webové služby	98
7.17	Úvodní menu aplikace.	99
7.18	Zadání pro službu <i>infoOParcele</i>	100
7.19	Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.	100
7.20	Zadání pro službu <i>infoOBudove</i>	101
7.21	Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.	102
7.22	Zadání pro službu <i>infoOJednotce</i>	103
7.23	Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.	103
7.24	Zadání pro službu <i>infoORizeni</i>	104
7.25	Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.	105

Seznam tabulek

3.1	Počet opakování vzoru.	15
3.2	Rozdíly mezi SAX a DOM. (Zdroj [18]).	18
3.3	Java API pro XML (Zdroj [27]).	20
3.4	Mapování datových typů mezi Javou a XML (Zdroj [27]).	23
6.1	Datové skupiny (Zdroj [8]).	54
6.2	Hodnoty atributů stav dat a příznak kontextu (Zdroj [8]).	55
7.1	Shrnutí výhod a nevýhod jednotlivých variant pro řešení webových služeb.	65
7.2	Využití datových bloků pro výstupy webových služeb.	72
7.3	Vstupní parametry webových služeb.	76
7.4	Parametry webové služby <i>infoOParceleID</i>	80
7.5	Parametry webové služby <i>infoOParcele</i>	80
7.6	Parametry webové služby <i>vyhledaniParcel</i>	82
7.7	Parametry webové služby <i>infoOBudoveID</i>	83
7.8	Parametry webové služby <i>infoOBudove</i>	84
7.9	Parametry webové služby <i>vyhledaniBudov</i>	86
7.10	Parametry webové služby <i>vyhledaniBudovParID</i>	86
7.11	Parametry webové služby <i>infoOJednotceID</i>	88
7.12	Parametry webové služby <i>infoOJednotce</i>	88
7.13	Parametry webové služby <i>vyhledaniJednotek</i>	90
7.14	Parametry webové služby <i>vyhledaniJednotekBudID</i>	90
7.15	Parametry webové služby <i>infoORizeniID</i>	91
7.16	Parametry webové služby <i>infoORizeni</i>	92
7.17	Parametry webové služby <i>nejblizsiDefBodPar</i>	94
7.18	Parametry webové služby <i>nejblizsiDefBodBud</i>	94
7.19	Parametry webové služby <i>vratCiselnik</i>	95
7.20	Parametry webové služby <i>vratSeznamCiselniku</i>	96

Kapitola 1

Úvod

Cílem této práce je navržení a realizování programového přístupu k výstupům a službám v rozsahu, jaký poskytuje aplikace „Nahlížení do KN“¹.

Aplikace „Nahlížení do KN“ je velmi oblíbenou aplikací, která zdarma poskytuje služby a výstupy z katastru nemovistostí (dále jen KN). Tyto výstupy jsou poskytovány ve formátu HTML. Formát HTML je velmi vhodný jako prezentační formát v internetovém prostředí, ale jeho možné další automatizované zpracování je velmi obtížné.

Navržený alternativní přístup realizovaný pomocí webových služeb, které budou poskytovat výstupy ve snadno automatizovaně zpracovatelném formátu, umožní uživatelům získané výstupy snadno dále zpracovávat a následně analyzovat. A to vše v jejich vlastních informačních systémech. Technologie webových služeb totiž umožňuje bezproblémovou integraci webových služeb do vlastních informačních systémů zákazníků.

Práce je strukturována způsobem, který umožní každému čtenáři postupné pochopení pojmu webových služeb a dalších technologií, které s touto problematikou úzce souvisejí.

První část práce obsahuje základní seznámení s technologií webových služeb. Je zde popsán krátký přehled technologií, které předcházeli jejich vzniku. Dále jsou zde uvedeny základní prvky a vlastnosti webových služeb. Tato část je velmi důležitá z hlediska pochopení vlastního pojmu „co je webová služba“.

Druhá část je věnována jazyku XML, ne však jazyku jako takovému, protože je již dosud znám a rozšířen, ale spíše technologiím, který tento jazyk rozšiřuje a doplňuje. Dále je zde věnován širší prostor pro seznámení s možnostmi jeho zpracování, jak po teoretické stránce, tak po praktické stránce pomocí programovacího jazyka JAVA.

Třetí část rozšiřuje část první a věnuje se technologii webových služeb více podrobněji a komplexněji. Jsou zde velmi podrobně popsány a vysvětleny jednotlivé prvky, které tvoří architekturu webových služeb.

Čtvrtá část popisuje využití webových služeb v rámci různých zahraničních projektů. Webové služby jsou stále více oblíbenou technologií a tak téměř nemožné sledovat všechny

¹Nahlížení do KN
URL:<<http://nahlizenidokn.cuzk.cz/>>

oblasti, kde jsou webové služby použity. Proto je možné, že některé informace nebudou plně aktuální a že někteří čtenáři mohou znát aktuálnější informace.

Pátá část popisuje Informační systém katastru nemovitostí (ISKN). ISKN je jedním z nejrozsáhlejších informačních systémů státní správy v České republice. Systém je zde velmi podrobně popsán, jak z hlediska funkčnosti, tak z hlediska technologické infrastruktury. Dále je zde popsán výměnný formát katastru nemovitostí (NVF), který slouží k výměně dat mezi různými informačními systémy a ISKN. Struktura NVF posloužila jako základ ke tvorbě výstupního formátu pro webové služby.

Závěrečná část je vyvrcholením celé práce. Nejdříve začíná shrnutím stávajících možností pro získání dat z KN. Jsou popsány zejména možnosti získání dat z aplikací „Dálkový přístup“² a „Nahlížení do KN“. Toto shrnutí naznačuje, kde je možný prostor pro vytvoření alternativního přístupu v podobě webových služeb. Dále je část rozdělena na část analytickou a na část praktickou.

Část analytická je zaměřena na detailní analýzu možných variant pro tvorbu webových služeb. Jsou vytvořeny uživatelské scénáře, které chování systému služeb plně popisují. Dále jsou zde analyzovány různé možnosti tvorby výstupního formátu, které budou webové služby poskytovat. Také jsou analyzovány možnosti chybových hlášení.

V části praktické jsou webové služby vytvářeny a to na základě přechozích analýz. Webové služby byly implementovány v programovacím jazyce JAVA. Celkem vzniklo 17 webových služeb, rozdělených do 6 základních skupin. Skupiny zahrnují služby podle obsahu poskytovaných dat. V rámci této části byl vytvořen popis webových služeb – WSDL. Pro výstupní formát bylo vytvořeno XML schéma, které plně popisuje jeho obsah a tvar.

²Nahlížení do KN
URL:<<https://katastr.cuzk.cz/>>

Kapitola 2

Úvod do webových služeb

Web Services lze přeložit jako webové služby. Mnozí si pod tímto pojmem představí pouze webové stránky, které poskytují nějaké služby či firmy, které vytvářejí WWW stránky. V této práci budou webové služby představeny jako nová technologie pro vzdálené volání funkcí v distribuovaných systémech.

Pomocí webových služeb lze zveřejnit vlastní funkce a poskytnout je k použití ostatním uživatelům pro jejich klientské aplikace pomocí standardního webového protokolu.

Z historie již známe některé technologie pro vzdálené volání funkcí jako např. RPC (Remote Procedure Call), CORBA (Common Object Request Broker Architecture) či RMI (Remote Method Invocation). Tyto starší technologie byly poměrně složité a to bránilo jejich snadnému rozšíření. Oproti tomu webové služby jsou založeny na zcela otevřených standardech a snaží se v maximální míře využít již stávající technologie, zejména XML(eXtended Markup Language) [1], jsou platformově zcela nezávislé jak na operačních systémech, tak na programovacích jazycích.

Následující definice je převzata od konsorcia W3C [37]:

”A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”

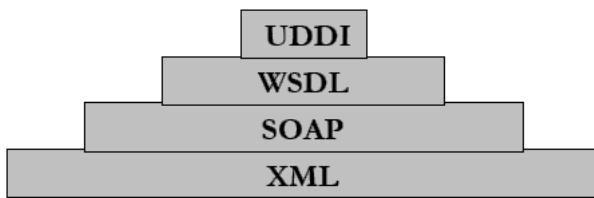
- W3C Definition -

”Webová služba je softwarový systém, vytvořený tak, aby podporoval interakci počítač-počítač, tak aby byly schopné navzájem komunikovat. Má rozhraní popsané v počítačově zpracovatelném formátu (konkrétně WSDL [3]). Ostatní systémy komunikují s webovou službou způsobem definovaným jejím popisem pomocí SOAP [16]-zpráv, které jsou posílané pomocí standardního webového

protokolu HTTP ve formátu XML v souladu s ostatními Web-orientovanými standardy.”

- W3C Definice -

Pro vzájemnou komunikaci je doporučen protokol SOAP (Simple Object Access Protocol), ale může být použit i protokol XML-RPC nebo lze využít funkce protokolu HTTP GET/POST. Transportní vrstvu tvoří některý z protokolů HTTP(S), FTP, SMTP atd. (typicky však HTTP(S)). Pro popis rozhraní služby je doporučeno použití strojově zpracovatelné XML gramatiky zapsané v jazyce WSDL (Web Service Description Language). Pro vyhledávání a publikaci služeb slouží logicky centralizované adresáře služeb označované jako UDDI (Universal Description, Discovery and Integration) [35], více viz [21]. Pro ilustraci lze použít následující schéma na obrázku č. 2.1:



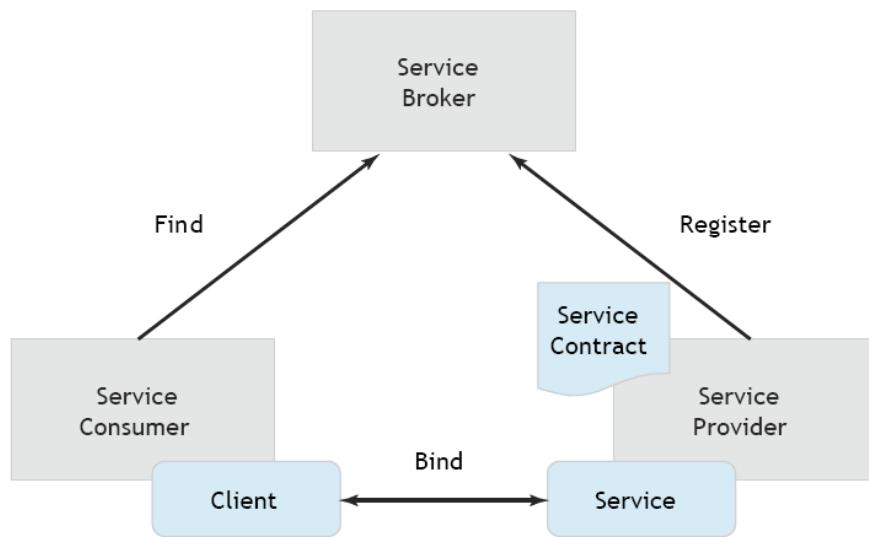
Obrázek 2.1: Základem je XML (Zdroj [17]).

Koncepce služby (service) je klíčem k pochopení webových služeb. Tato koncepce je založena na Architektuře orientované na služby - SOA (Service Oriented Architecture)[14].

2.1 Architektura orientovaná na služby - SOA

SOA je zatím posledním označením architektury aplikací, princip této architektury je znázorněn na obrázku č. 2.2, zaměřené na sdílení a opakování používání kódu. V rámci SOA jsou aplikace budovány pomocí standardních rozhraní, založených na XML, např. SOAP či WSDL. SOA definuje, jak jsou tyto služby lokalizovány, prováděny, spravovány, monitorovány a zabezpečovány.

Je to architektura volně provázaných webových služeb, kde služby mohou běžet asynchronně. Tyto služby lze vzájemně propojit a tím vytvořit vlastní aplikaci. Obecně lze říci, že SOA se obejde i bez webových služeb. Ty zajišťují zejména flexibilitu a zjednoduší sdílení služeb. Bez těchto klíčových vlastností jiné pokusy o realizaci SOA selhaly, např. CORBA (Common Object Request Broker Architecture) či DCOM (Distributed Component Object Model), více viz [13].



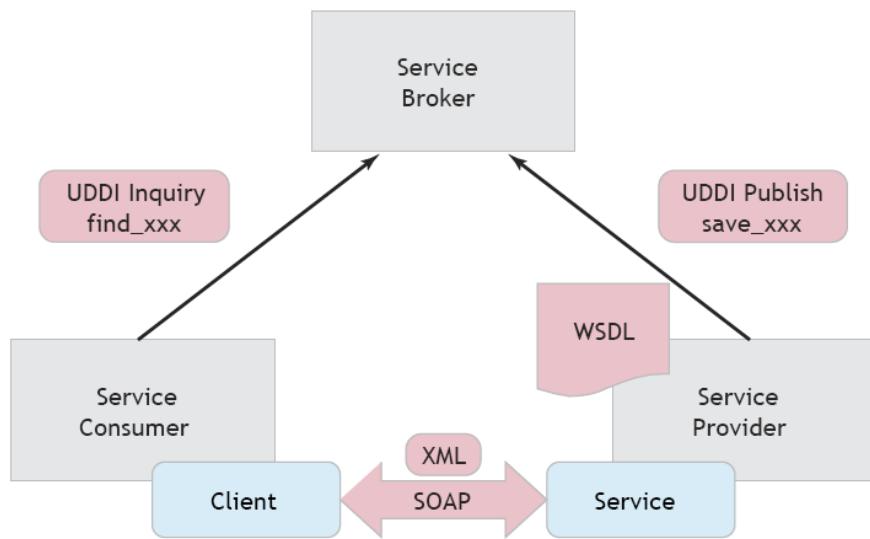
Obrázek 2.2: Service Oriented Architecture - SOA (Zdroj [33]).

Jádrem SOA jsou tři elementy, které definují síťovou službu:

1. Service Provider (poskytovatel služby) - poskytovatel služby registruje popis služby v registru služeb,
2. Service Consumer (odběratel služby) – využívá registr ke zjištění existence služby (jednu službu může poskytovat více poskytovatelů), po nalezení služby se k ní dokáže klientsky připojit a realizovat výměnu zpráv,
3. Service Broker (registr služeb) – registruje popis služby – ”zlaté stránky” webových služeb.

2.2 Architektura webových služeb

Tato architektura je založená na SOA a její princip je znázorněn na obrázku č.2.3:



Obrázek 2.3: Architektura webových služeb (Zdroj [33]).

1. Service Provider (poskytovatel služby) – poskytovatel služby registruje popis služby ve formátu WSDL, v registru služeb – UDDI,
2. Service Consumer (odběratel služby) – využívá registr ke zjištění existence služby (jednu službu může poskytovat více poskytovatelů), po nalezení popisu služby – WSDL se k ní dokáže klientsky připojit a realizovat výměnu zpráv pomocí protokolu SOAP,
3. Service Broker (registrování služeb – UDDI) – registruje popis služby – ”zlaté stránky” webových služeb.

Kapitola 3

XML

Jazyk XML [1] je již natolik známý, že ho není potřeba podrobněji představovat. V kapitole budou uvedeny pouze základní charakteristiky XML, podrobněji budou popsány speciálnější vlastnosti a možné zpracování XML dokumentů.

3.1 Seznámení s XML

Pokud bychom chtěli stručně vysvětlit, jak XML vypadá, dalo by se říci, že XML se podobá HTML. Oba jazyky patří do společné rodiny značkovacích jazyků, tj. popisují data pomocí značek (tags). Oba jazyky XML i HTML jsou odvozeny od jazyka SGML a jejich příbuznost je proto na první pohled zřejmá. Na rozdíl od HTML ale XML umožňuje definovat vlastní sadu značek. V podstatě tedy navrhujeme vlastní jazyk pro popis našich strukturovaných dat. Pro formální definici uživatelských XML formátů, strukturovaných dat, lze využít jazyky pro popis schématu dokumentu jako jsou DTD (Document Type Definition) [1], XML Schema [38] či Relax NG [4].

Výhody pro použití XML pro popis dat jsou především tyto:

- jde o obecně uznávaný a rozšířený standard nezávislý na operačním systému,
- nástupce HTML jako jazyka internetových stránek,
- specifikace XML je stručná a poměrně jednoduchá,
- specifikace XML je volně (zadarmo!) k dispozici na webu konsorcia W3C [37],
- čitelný textový dokument => možná editace v obyčejném textovém editoru,
- možnost zobrazení v prohlížeči webových stránek (Internet Explorer, Netscape, Opera, ...),
- dokument je strukturován podle obsahu, NE podle zobrazení => uloží více informací,

- snadná konverze do dalších formátů (HTML, WML, RTF, PDF, ...) pomocí jazyka XSLT,
- existují volně dostupné „parsery“ (viz kapitola 3.4) a další nástroje pro zpracování XML dokumentů,
- XML má vyřešenu problematiku kódování národních abeced,
- jde o jazyk s pevnou a jasně definovanou syntaxí.

Jisté potíže při praktické implementaci XML mohou vyplynout z následujících skutečností:

- vytvoření formální definice (DTD, XML schéma) není zcela jednoduchou záležitostí,
- prakticky je nereálné pracovat s XML bez využití kvalitního parseru (volba vhodného parseru přitom není zcela samozřejmá; z programátorského úhlu pohledu není ani použití parseru úlohou pro začátečníka).

3.2 Jmenné prostory

Jazyk XML je rozšiřitelný značkovací jazyk. Pokud tedy navrhнемe vlastní sadu značek, lze říci, že tyto značky jsou v dokumentu unikátní. Ale co když dojde ke spojení více různých XML dokumentů? Může dojít ke konfliktu názvů značek a parser (viz kapitola 3.4) pro jejich zpracování, nebude schopen rozlišit, kterou značku (element) má jak a k čemu použít. Proto existují jmenné prostory [2], které dávají značkám globálnější charakter. Hlavní vlastnosti jmenných prostorů:

- rozlišení elementů a atributů, které mohou mít stejný název (mohou pocházet z různých zdrojů),
- zastřelení elementů a atributů pocházejících z jedné aplikace (snadnější zpracování).

Použití jmenných prostorů je jednoduché, každý element či atribut je přiřazen k určitému jmennému prostoru, který je jednoznačně identifikován URI (Uniform Resource Identifier) adresou [36]. Pro snazší použití v XML dokumentu je pro tuto URI adresu deklarována určitá předpona – prefix. Plná URI adresa je uvedena na začátku dokumentu a dále je používán pouze prefix. Tento prefix je uveden před každým elementem či atributem.

3.3 Schémata dokumentů

Aby bylo možné s XML dokumenty plnohodnotně pracovat, je potřeba nadefinovat strukturu (schéma) dokumentu. Tato struktura přesně definuje jaké elementy, atributy (včetně jejich typů a obsahů) lze v dokumentu použít a jak je vzájemně kombinovat. Pokud je tedy struktura dobře nadefinovaná, můžeme dokument bez potíží syntakticky

kontrolovat (validovat), dále vhodně zpracovávat, případně v dokumentech vyhledávat, formátovat atd. Pro tvorbu těchto struktur (schémat) slouží jazyky pro popis (definování) schématu dokumentu XML.

Nejstarším známým jazykem pro popis schématu dokumentů je DTD (Document Type Definition), který popisuje přímo specifikace jazyka XML. DTD je vhodné pro menší a méně složité XML dokumenty. DTD pro současné aplikace již nevyhovuje, protože bylo navrženo zejména pro dokumenty textové povahy. DTD nepodporuje jmenné prostory a nedovoluje určení datového typu pro obsah elementů a atributů.

Dnes nejpoužívanějším jazykem je XML Schéma (WXS) [38], které je obecně uznávaným standardem – doporučením od konsorcia W3C¹. XML Schéma umožňuje použití jmenných prostorů, poskytuje bohatší škálu primitivních datových typů, dovoluje odvozovat nové typy pomocí dědičnosti. Nevýhodou XML schémat je složitá specifikace (specifikace obsahuje přes 200 stran) a obtížnější implementace.

Alternativou k XML Schéma je jazyk Relax NG (REgular LAnguage for XML Next Generation) [4].

Následující kapitoly 3.3.1 – XML Schéma (WXS) a 3.3.2 – Relax NG nebyly napsány s cílem vyčerpávajícího popisu jejich vlastností a možností. Cílem kapitol je seznámení s hlavními rysy a vlastnostmi těchto jazyků pro tvorbu schémat. V obou kapitolách je uvedeno množství příkladů, které prezentují jejich možnosti praktického využití. Kompletní popis, včetně mnoha praktických ukázek, je volně k dispozici na internetu, viz [24, 34].

3.3.1 XML Schéma (WXS)

Schéma je vlastně XML dokument, který obsahuje speciální elementy, které patří do jmenného prostoru <http://www.w3.org/2001/XMLSchema>. Pro tento jmenný prostor (viz kapitola 3.2) se nejčastěji používá prefix `xs` nebo `xsd`.

Schéma definuje pro každý element datový typ (datové typy jsou základem WXS). Datové typy mohou být jednoduché nebo komplexní. Jednoduchý datový typ je používán pro hodnoty jako je číslo, řetězec, datum atd. Komplexní datový typ se skládá z více elementů a atributů. Z příkladu 3.1 je dobře patrné, že element `bud` je komplexní datový typ, naproti tomu např. element `stav_dat` je jednoduchý datový typ typu `string`.

Příklad 3.1: XML schéma

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="bud" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="stav_dat" type="xs:string" minOccurs="0"/>
```

¹Proces vývoje W3C doporučení:

Submission (návrh) ⇒ Note (poznámka) ⇒ Working Draft (WD) (pracovní návrh) ⇒ Candidate Recommendation (CR) (kandidát na specifikaci) ⇒ Proposed Recommendation (PR) (navržená specifikace) ⇒ W3C Recommendation (REC) (specifikace, v terminologii W3C doporučení)

```

<xs:element name="datum_vzniku" type="xs:string" minOccurs="0"/>
<xs:element name="priznak_kontextu" type="xs:string" minOccurs="0"/>
<xs:element name="rizeni_id_vzniku" type="xs:long" minOccurs="0"/>
<xs:element name="typbud_kod" type="xs:string" minOccurs="0"/>
<xs:element name="caobce_kod" type="xs:string" minOccurs="0"/>
<xs:element name="cislo_domovni" type="xs:string" minOccurs="0"/>
<xs:element name="zppybu_kod" type="xs:string" minOccurs="0"/>
<xs:element name="tel_id" type="xs:long" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="id" form="unqualified" type="xs:long"
    use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>

```

Schématu z příkladu 3.1 odpovídá XML dokument zobrazený v následujícím příkladě 3.2.

Příklad 3.2: XML dokument

```

<bud id="382720210">
  <stav_dat>-1</stav_dat>
  <datum_vzniku>17.03.1995 00:00:00</datum_vzniku>
  <priznak_kontextu>3</priznak_kontextu>
  <rizeni_id_vzniku>2303515210</rizeni_id_vzniku>
  <typbud_kod>1</typbud_kod>
  <caobce_kod>58611</caobce_kod>
  <cislo_domovni>316</cislo_domovni>
  <zppybu_kod>3</zppybu_kod>
  <tel_id>751671210</tel_id>
</bud>

```

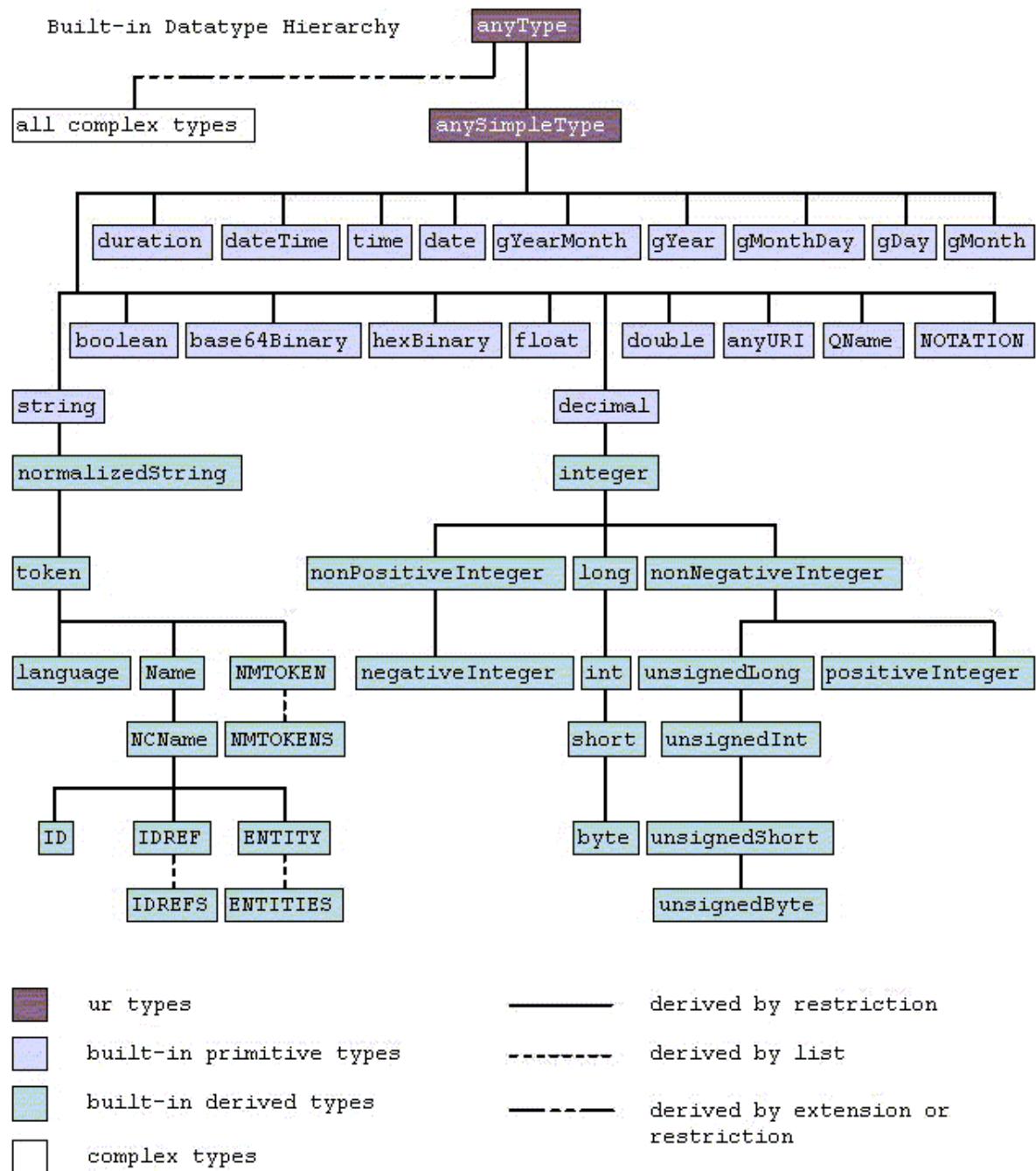
3.3.1.1 Jednoduché datové typy

Výhodou WXS je možnost definovat vlastní datové typy odvozením z již existujících datových typů. Lze použít několik desítek (cca 50) zabudovaných jednoduchých datových typů viz obrázek 3.1 nebo typy uživatelsky definované.

Nové datové typy lze odvodit pomocí:

- restrikce (Restriction) – omezení hodnot,
- seznamu (List) – vytvořením seznamu hodnot daného typu,
- spojení (Union) – spojením několika typů,
- rozšířením (Extension) – rozšířením existujících typů.

Nejčastěji používaným typem pro odvození datového typu je restrikce. Na existující datový typ lze aplikovat několik integritních omezení a tím zúžit obor hodnot.



Obrázek 3.1: Hierarchie zabudovaných datových typů. (Zdroj [38]).

3.3.1.2 Komplexní datové typy

K vytvoření struktury dokumentu se používají komplexní datové typy. Komplexní datový typ se může skládat z elementů a atributů. U elementů lze definovat jejich pořadí, povinnost či volitelnost, zda se mohou opakovat a kolikrát. Počet opakování definují atributy `minOccurs` a `maxOccurs`. V příkladě 3.1 existuje komplexní datový typ `bud`, který

se může opakovat v libovolném počtu (`maxOccurs="unbounded"`). V tomto komplexním datovém typu (`<xs:complexType>`) se vnořené elementy, např. `stav_dat`, `datum_vzniku`, `priznak_kontextu` atd., musejí vyskytovat v daném pořadí (`<xs:sequence>`).

Určení pořadí elementů lze dosáhnout třemi způsoby pomocí elementů:

- sequence – prvky se vyskytují v daném pořadí,
- all – prvky se vyskytují v libovolném pořadí,
- choice – vyskytuje se jeden z uvedených prvků.

Atributy se v definici komplexního typu vyskytují až na konci, lze definovat jejich povinnost či volitelnost. Na příkladu 3.1 lze najít povinný (`use="required"`) atribut `id` (`name="id"`) typu `long` (`type="xs:long"`).

3.3.1.3 Připojení schématu k dokumentu

Pro připojení schématu k dokumentu se používají globální atributy patřící do jmenovaného prostoru `http://www.w3.org/2001/XMLSchema-instance`. Pokud jsou v dokumentu použity jmenné prostory, je nutno použít atribut `schemaLocation`. Tento atribut obsahuje URI jmenného prostoru a umístění schématu, viz příklad 3.3.

Příklad 3.3: Použití atributu `schemaLocation`

```
<wsvfkiskn
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:xmlns:nahlichenidokn.cuzk.cz:wsvfkiskn"
    xsi:schemaLocation="urn:xmlns:nahlichenidokn.cuzk.cz:wsvfkiskn
    RizeWSVFK_v0.2.xsd">
    ...
</wsvfkiskn>
```

Jestliže v dokumentu nejsou použity jmenné prostory, je nutno použít atribut `noNamespaceSchemaLocation`. Použití je zobrazeno v následujícím příkladě 3.4

Příklad 3.4: Použití atributu `noNamespaceSchemaLocation`

```
<wsvfkiskn
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="RizeWSVFK_v0.2.xsd">
    ...
</wsvfkiskn>
```

3.3.2 Relax NG

Relax NG je nejnovějším, zde popisovaným, jazykem pro tvorbu schémat XML dokumentů. Byl vyvinut konsorciem OASIS², v roce 2001 získal status specifikace OASIS a od roku 2003 je i normou ISO/IEC³ (č.19757-2).

Stejně jako WXS je schéma vytvořené pomocí Relax NG XML dokument. Tento XML dokument také tvoří speciální elementy, které patří do jmenného prostoru <http://relaxng.org/ns/structure/1.0>. O proti WXS, které je založeno na datových typech, je Relax NG založen na vzorech.

Schéma vytvořené pomocí Relax NG (viz příklad 3.5 – schématu odpovídá XML dokument znázorněný v příkladě 3.2) je vzorem dokumentu, který se skládá z jednotlivých vzorů pro elementy, atributy a textové uzly. Tyto vzory lze kombinovat do uspořádaných i neusporádaných skupin, lze určovat jejich volitelnost či povinnost, nebo určovat počet možných opakování.

Příklad 3.5: Relax NG schéma

```
<element xmlns="http://relaxng.org/ns/structure/1.0" name="bud">
  <attribute name="id">
    <text/>
  </attribute>
  <element name="stav_dat">
    <text/>
  </element>
  <element name="datum_vzniku">
    <text/>
  </element>
  <element name="priznak_kontextu">
    <text/>
  </element>
  <element name="rizeni_id_vzniku">
    <text/>
  </element>
  <element name="typbud_kod">
    <text/>
  </element>
  <element name="caobce_kod">
    <text/>
  </element>
  <element name="cislo_domovni">
    <text/>
  </element>
  <element name="zpvybu_kod">
    <text/>
  </element>
```

²The Organization for the Advancement of Structured Information Standards
URL: <<http://www.oasis-open.org/home/index.php>>

³International Organization for Standardization
URL: <<http://www.iso.org/iso/home.htm>>

```
<element name="tel_id">
  <text/>
</element>
</element>
```

Zajímavou vlastností Relax NG je možnost zápisu schématu v kompaktní (textové) syntaxi (viz příklad 3.6).

Příklad 3.6: Kompaktní syntaxe

```
element bud {
  attribute id { text },
  element stav_dat { text },
  element datum_vzniku { text },
  element priznak_kontextu { text },
  element rizeni_id_vzniku { text },
  element typbud_kod { text },
  element caobce_kod { text },
  element cislo_domovni { text },
  element zpvybu_kod { text },
  element tel_id { text }
}
```

Tato syntaxe je lépe čitelná a její tvorba je snadnější. Obě syntaxe (XML a kompaktní) lze mezi sebou převádět a není tedy problém tvořit schéma v jedné nebo v druhé syntaxi. XML zápis má obvykle souborovou koncovku ***.rng**, pro kompaktní zápis je obvykle použita souborová koncovka ***.rnc**.

3.3.2.1 Základní vzory

Přehled základních vzorů:

- element – vzor pro elementy, obsah elementu může obsahovat další vzory.
- attribute – vzor pro atributy.
- text – vzor se používá pro definici obsahu elementu (také atributu), který už neobsahuje další podelementy.
- optional – vzor definuje volitelnost či povinnost obsahu vzoru, viz tabulka 3.1.
- oneOrMore – vzor definuje opakování obsahu vzoru, viz tabulka 3.1.
- zeroOrMore – vzor definuje opakování obsahu vzoru či jeho úplné vypuštění, viz tabulka 3.1.

Počet opakování	RNG	RNC
1	-	-
0 nebo 1	optional	?
1 a více	oneOrMore	+
0, 1 nebo více	zeroOrMore	*

Tabulka 3.1: Počet opakování vzoru.

3.3.2.2 Pokročilé vzory

Přehled pokročilých vzorů:

- choice – Ze skupiny vzorů uvnitř tohoto elementu se bude v dokumentu vyskytovat právě jeden.
- group – Vzor pro seskupení vzorů.
- interleave – Vzory uvnitř tohoto elementu se mohou vyskytovat v libovolném pořadí.
- mixed – Obsah elementu může obsahovat jak text, tak také další vnořené vzory.

Pro názornost použití některých pokročilých vzorů je uveden následující příklad 3.7. Příklad je zapsán v obou možných tvarech (XML syntaxe, kompaktní syntaxe).

Příklad 3.7: XML syntaxe a kompaktní syntaxe

```

<element name="jmeno">
  <choice>
    <text/>
    <group>
      <element name="krestni_jmeno">
        <text/>
      </element>
      <optional>
        <element name="rodne_prijmeni">
          <text/>
        </element>
      </optional>
      <element name="prijmeni">
        <text/>
      </element>
    </group>
  </choice>
</element>

```

```

element jmeno {
  text | (
    element krestni_jmeno{text},
    element rodne_prijmeni{text}?,
    element prijmeni{text}
  )
}

```

Schématu z příkladu 3.7 odpovídají všechny zobrazené části XML dokumentu v příkladě 3.8.

Příklad 3.8: Části XML dokumentu

```

...
<jmeno>Jan</jmeno>
...
...
<jmeno>
  <krestni_jmeno>Jan</krestni_jmeno>
  <rodne_prijmeni>Pytel</rodne_prijmeni>
  <prijmeni>Pytel</prijmeni>
</jmeno>
...
...
<jmeno>
  <krestni_jmeno>Jan</krestni_jmeno>
  <prijmeni>Pytel</prijmeni>
</jmeno>
...

```

3.3.2.3 Datové typy

Relax NG mnoho zabudovaných datových typů nenabízí (resp. nabízí dva zabudované datové typy **string** a **token**). Strategií Relax NG je použití již existujících datových typů z externích knihoven. Nejčastěji jsou použity datové typy z WXS. Výhodou takového řešení je možnost kontroly obsahu hodnot elementů a atributů stejně jako u WXS.

Datový typ je určen vzorem **data**, jehož atribut **type** určuje datový typ. Knihovna datových typů je určena pomocí dalšího atributu **datatypeLibrary**. Příklad 3.9 definuje element **poradi**, datového typu **decimal** z datových typů WXS.

Příklad 3.9: Definice datového typu

```

<element name="poradi">
  <data type="decimal"
        datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
</element>

```

Příklad 3.10 je příklad 3.9 zapsaný v kompaktní syntaxi.

Příklad 3.10: Definice datového typu – kompaktní syntaxe

```
element cena { xsd:decimal }
```

Pro každý datový typ je možné určit další zpřesňující parametry. Jestliže použijeme datové typy WXS, lze definovat většinu integritních omezení jako u WXS. Pro nastavení parametrů datového typu se používá element **param** s atributem **name**. Příklad 3.11 zobrazuje schéma, kdy kniha musí mít název o délce mezi 5 a 30 znaky, její cena se pohybuje mezi 100 až 500 a její URL adresa musí začínat <http://>.

Příklad 3.11: Schéma "Kniha"

```

<element name="kniha" xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    <element name="nazev">
        <data type="string">
            <param name="minLength">5</param>
            <param name="maxLength">30</param>
        </data>
    </element>
    <element name="cena">
        <data type="decimal">
            <param name="minInclusive">100</param>
            <param name="maxExclusive">500</param>
        </data>
    </element>
    <element name="url_adresa">
        <data type="anyURI">
            <param name="pattern">http://.*</param>
        </data>
    </element>
</element>

```

A nyní zápis příkladu 3.11 v kompaktní syntaxi – příklad 3.12.

Příklad 3.12: Schéma "Kniha" – kompaktní syntaxe

```

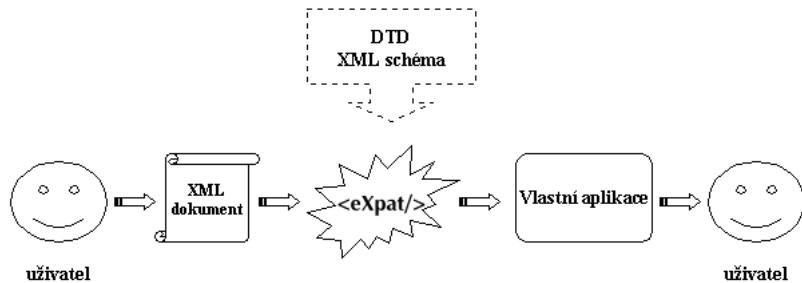
element kniha {
    element nazev {
        xsd:string { minLength = "5" maxLength = "30" }
    },
    element cena {
        xsd:decimal { minInclusive = "100" maxExclusive = "500" }
    },
    element url_adresa {
        xsd:anyURI {pattern = "http://.*"}
    }
}

```

3.4 Zpracování XML

Pro zpracování XML dokumentu si můžeme napsat vlastní program, který by četl data ze souboru a zkoumal, jestli se jedná o element, atribut či vlastní data. Tento způsob je však velmi časově náročný a není vůbec snadný pro každého programátora. Proto existují na Internetu knihovny (volně ke stažení), které umějí zpracovat XML dokument a oddělit tak programátory od specifikací syntaxe XML. V dnešní době jsou tyto knihovny dostupné v mnoha programovacích jazycích (C, C++, Java a atd.). Tyto knihovny — **Parsery** jsou softwarové komponenty, které dokáží pracovat s XML dokumenty, tzn. umí číst a interpretovat syntaxi jazyka XML a mnohé z nich umějí zároveň kontrolovat validitu XML

dokumentu oproti DTD nebo XML schématu.



Obrázek 3.2: Schéma zpracování XML dokumentu pomocí parseru.

V dnešní době existují dva hlavní přístupy při parsování (analyzování) XML dokumentu. První přístup je založen na principu vytvoření stromové hierarchické struktury dokumentu – objektové rozhraní. Druhý přístup je založen na principu vytvoření událostí z dokumentu – událostní rozhraní (pomocí událostí reagujeme na elementy, atributy či entity vyskytující se v dokumentu). Tyto dva přístupy spolu nijak nesoupeří, protože slouží odlišným potřebám. Mnohé parsery XML podporují obě rozhraní. Základní rozdíly mezi událostním rozhraním (SAX) a objektovým rozhraním (DOM) jsou dobře viditelné v následující tabulce 3.2.

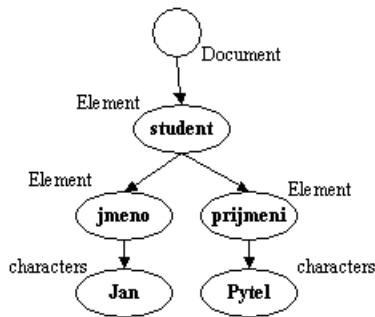
SAX	DOM
Neukládá data z dokumentu XML během zpracování.	Vytváří kopii stromu dokumentu v paměti.
Neobsahuje žádnou podporu pro modifikaci nebo zapisování do dat v dokumentu XML.	Strom dokumentu uložený v paměti lze modifikovat.
Data dokumentu se zpřístupňují již během jeho zpracování analyzátem.	Celý dokument se musí nejprve zpracovat a teprve potom je strom dokumentu přístupný klientské aplikaci.

Tabulka 3.2: Rozdíly mezi SAX a DOM. (Zdroj [18]).

3.4.1 DOM

Standardem pro objektové rozhraní je DOM (Document object model) [26] – objektový model dokumentu, podporovaný konsorciem W3C [37]. V tomto rozhraní je XML dokument reprezentován jako objekt se stromovou hierarchickou strukturou, která je tvořena vsemi elementy v dokumentu. V tomto objektu každý element odpovídá uzlu stromu. Rozhraní DOM obsahuje funkce, které nám dovolují se v takto vytvořeném objektu velmi jednoduše

pohybovat z uzlu do uzlu, provádět různé operace (např. výpočty), přidávat nebo rušit jednotlivé uzly a atributy.



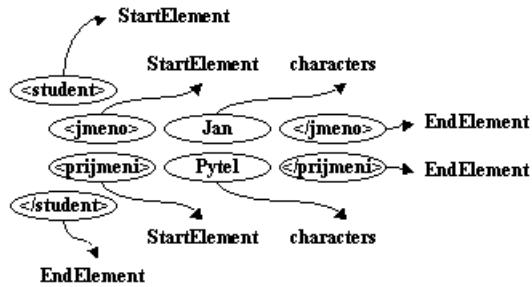
Obrázek 3.3: Schéma objektového přístupu ke XML dokumentu.

Výhodou tohoto rozhraní je, že nemusíme dokument procházet od začátku do konce, ale jak potřebujeme. Nevýhodou je velká spotřeba paměti pro sestavování stromu (vytvoření objektů) a tím také ztráty výkonnosti oproti událostnímu rozhraní. Toto rozhraní se nejvíce používá pro aplikace manipulující s dokumentem XML, jako jsou např. prohlížeče, editory, procesory XSL apod.

3.4.2 SAX

Asi nejznámějším typem událostního rozhraní je SAX (Simple API for XML) [28]. SAX byl vyvinutý společným úsilím členů e-mailové konference XML-DEV a upravený Davidem Megginsonem. Původně byl SAX navržen pro Javu, ale existují jeho implementace pro mnoho dalších jazyků. V současné době se již používá novější verze rozhraní SAX2, která podporuje jmenné prostory a některé další užitečné vlastnosti. V tomto rozhraní se nesestavuje strom objektů. Jakmile parser narazí na element či atribut při načítání XML dokumentu, vygeneruje příslušnou událost (event-driven). Jednotlivé události vyjadřují např. začátek či konec elementu, atributy, entity nebo obsah elementů.

Pomocí rozhraní vytvoříme vazbu mezi těmito událostmi. To znamená, že vytvoříme sadu funkcí, které se budou volat v okamžiku, kdy parser vygeneruje událost (začátek či konec elementu, atributy, entity nebo obsah elementů). Jako parametry se těmto funkcím předávají např. název elementu, název atributu apod. Výhodou tohoto rozhraní je jeho výkonnost a malá spotřeba paměti. Další výhodou je možnost zpracovávat dokument již během jeho načítání. Toto rozhraní se používá pro zpracování rozsáhlých souborů a u serverů (zpracování více dokumentů najednou). Nevýhodou je scházející navigace v dokumentu, na rozdíl od DOM.



Obrázek 3.4: Schéma událostního přístupu ke XML dokumentu.

3.5 XML a Java

XML a Java jsou technologie, které jsou založeny na otevřených standardech. Díky tomu přinášejí možnost velmi dobré vzájemné spolupráce.

Pro práci s XML v jazyce Java existuje několik skupin rozhraní, které jsou označeny jako JAX APIs (Java API for XML). V současné době existuje pět takovýchto skupin. Jejich přehled a základní popis je uveden v tabulce 3.3.

Java API	popis
Java API for XML Processing (JAXP)	API pro čtení, zapisování, manipulaci a transformaci XML dat
Java API for XML-based RPC (JAX-RPC)	API pro volání vzdálených procedur
Java API for XML Messaging (JAXM)	API pro synchronní či asynchronní posílání a přijímání XML zpráv pomocí SOAP
Java API for XML Registries (JAXR)	API pro komunikaci a přístup k registru webových služeb
Java API for XML Binding (JAXB)	API pro vytvoření vazeb mezi XML dokumenty a Javaovskými třídami

Tabulka 3.3: Java API pro XML (Zdroj [27]).

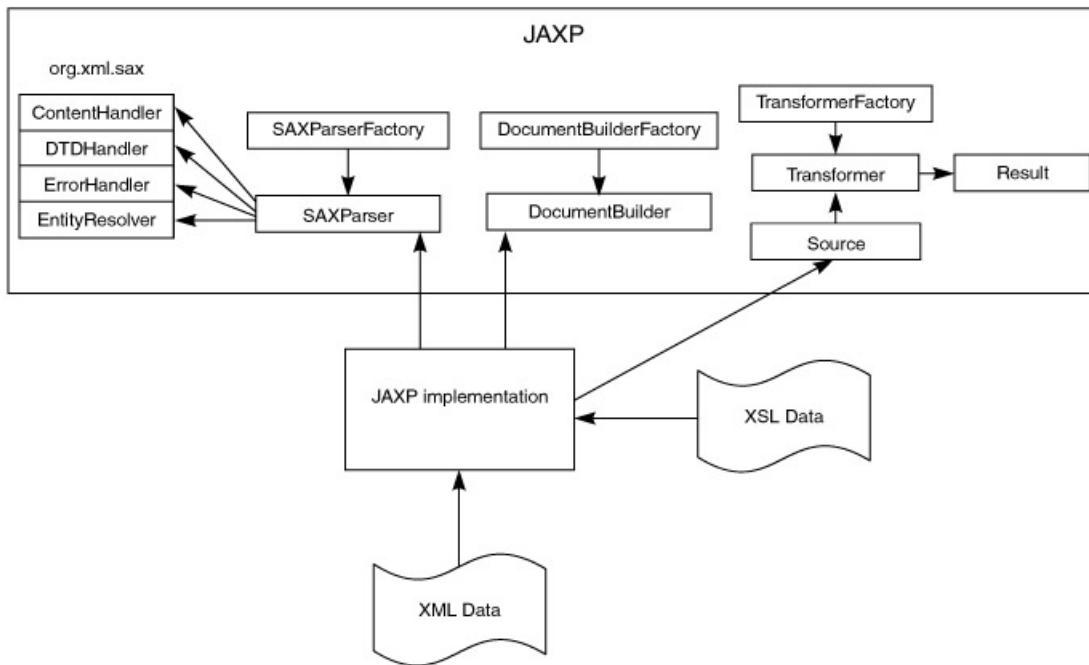
3.5.1 Java API for XML Processing (JAXP)

Java API for XML Processing (JAXP)⁴ umožňuje aplikacím číst, zapisovat, zpracovávat a transformovat XML data. XML data lze parsovat pomocí SAX (viz kapitola 3.4.2) či DOM (viz kapitola 3.4.1), transformace XML se provádí pomocí XSLT. Díky standardizovanému rozhraní lze kdykoliv změnit parser či XML procesor pro transformaci dat.

JAXP tvoří tyto balíčky (package):

- javax.xml.parsers – balíček definuje API pro zpracování XML,
- javax.xml.transform – balíček definuje API pro transformaci XML,
- org.xml.sax – balíček definuje API pro zpracování XML pomocí SAX,
- org.w3c.dom – balíček definuje API pro zpracování XML pomocí DOM.

Na následujícím obrázku 3.5 je zobrazen základní přehled architektury JAXP. Od verze



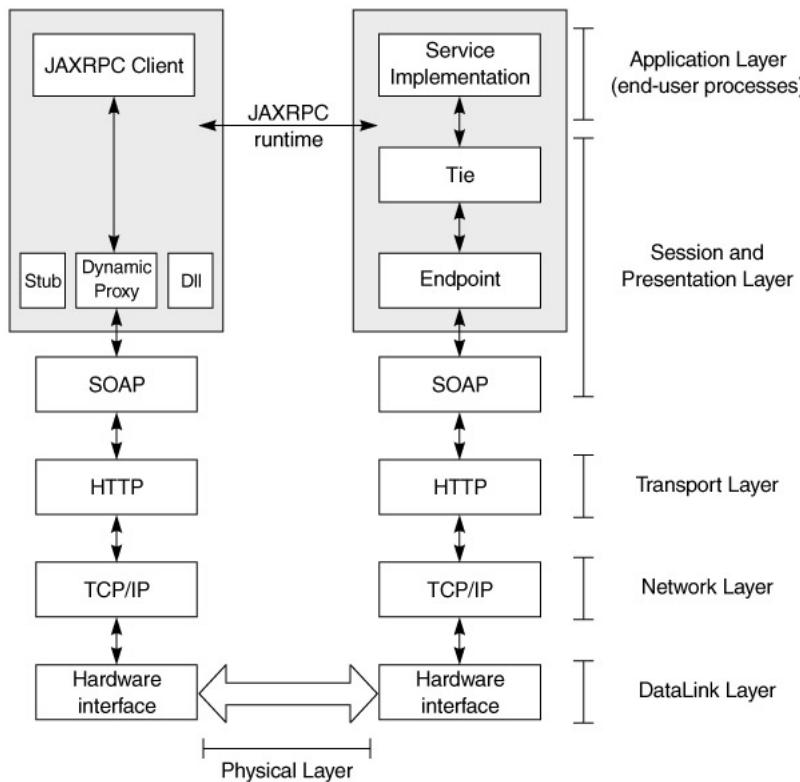
Obrázek 3.5: Architektura JAXP (Zdroj [27]).

Javy 1.4 je JAXP standardním API pro práci s XML.

⁴JAXP Reference Implementation
URL: <<https://jaxp.dev.java.net/>>

3.5.2 Java API for XML-based RPC (JAX-RPC)

JAX-RPC umožňuje vytvářet a zpřístupňovat webové služby za pomocí vzdáleného volání procedur – metod (RPC – Remote Procedure Calls) založených na XML. Webové služby mohou komunikovat pomocí mnoha různých protokolů, JAX-RPC využívá SOAP jako aplikační protokol a HTTP protokol jako transportní.



Obrázek 3.6: JAX-RPC model (Zdroj [27]).

Nevýhodou RPC je nutnost programovat komplikovanou nízkoúrovňovou síťovou komunikaci přímo pomocí socketů. Naproti tomu JAX-RPC vývojáře odstíní od detailů SOAPI, protože běhové prostředí (runtime environments) JAX-RPC – služba/klient vykonává na mapování mezi vzdálenými voláními metod – funkcí a SOAP zpráv. Pro přístup k webovým službám JAX-RPC poskytuje následující možnosti:

- Pomocí statických *stubs* (lokální objekty, které reprezentují vzdálené služby). *Stub* lze vygenerovat z definice rozhraní webové služby nebo z WSDL dokumentu.
- Pomocí dynamické proxy (Dynamic Proxy). Vazební objekt vzniká až za běhu aplikace, ale klient musí znát předem rozhraní webové služby.
- Pomocí rozhraní dynamického vyvolání (Dynamic Invocation Interface – DII). V tomto případě probíhá komunikace mezi webovou službou a klientem výhradně na dyna-

mické bázi. Klient tedy nemusí předem vědět jméno ani rozhraní webové služby, se kterou bude komunikovat.

Hlavním předpokladem úspěšné spolupráce klientské části a serverové části je, že prostředí JAX-RPC na obou stranách má správně a shodně nakonfigurovány type-mapping registry, které určují vzájemné mapování mezi datovými typy jazyka Java a XML datovými typy použitými v SOAP zprávách a WSDL dokumentech. Pro možné mapování JAX-RPC podporuje množinu primitivních datových typů jazyka Java, JAX-RPC podporuje také malou množinu standardních Javovských tříd, v tabulce 3.4 je zobrazeno mapování mezi Javou a XML . Proces kódování primitivních datových typů a tříd do jejich XML reprezentace se označuje jako *serializace*, proces dekódování se označuje jako *deserializace*.

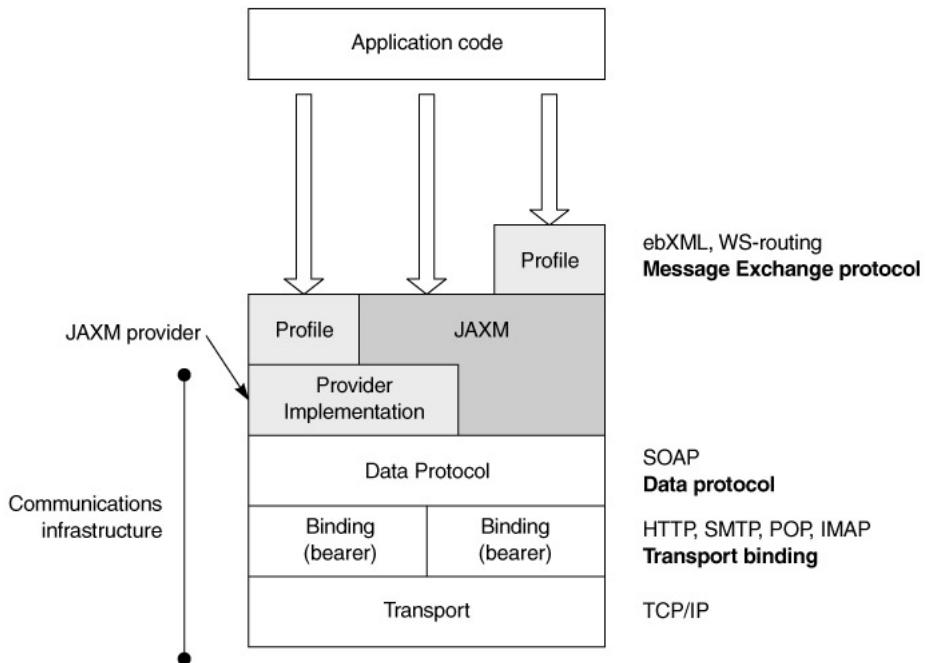
Java datový typ	XML datový typ
Boolean	xsd:boolean
Byte	xsd:byte
Short	xsd:short
Int	xsd:int
Long	xsd:long
Float	xsd:float
Double	xsd:double
byte[]	xsd:base64Binary
Byte[]	xsd:base64Binary
java.lang.String	xsd:string
java.math.BigInteger	xsd:integer
java.math.BigDecimal	xsd:decimal
java.util.Calendar	xsd:dateTime
java.util.Date	xsd:dateTime
javax.xml.namespace.QName	xsd:QName

Tabulka 3.4: Mapování datových typů mezi Javou a XML (Zdroj [27]).

Na obrázku 3.6 je zobrazen model JAX-RPC. Model obsahuje několik základních komponent a vrstev. Jak je z obrázku patrné, tak vývojáři, kteří použijí JAX-RPC, jsou úplně odstíněni od složitostí RPC. Plně postačuje, když se postará o aplikační vrstvu a o prezentaci vrstvu, o všechno ostatní se postará JAX-RPC.

3.5.3 Java API for XML Messaging (JAXM)

Java API for XML Messaging (JAXM) bylo navrženo pro vytváření podnikových aplikací (B2B). JAXM umožňuje aplikacím posílat a přijímat zprávy ve formátu XML, ale i v jiných formátech v rámci komunikačních infrastruktur založených na protokolech HTTP, SMTP a FTP. Zprávy mohou být posílány a přijímány synchronně či asynchronně. Pro komunikaci je použit Simple Object Access Protocol (SOAP) (viz 4.1).



Obrázek 3.7: Stavební bloky architektury JAXM (Zdroj [27]).

API je rozděleno na dvě části:

1. javax.xml.messaging – balíček definuje vlastní jádro JAXM,
2. javax.xml.soap – balíček implementuje SOAP with Attachments API for Java (SAAJ)⁵. SAAJ slouží pro vytváření, zpracování, posílání a přijímání SOAP zpráv s přílohou (Attachment).

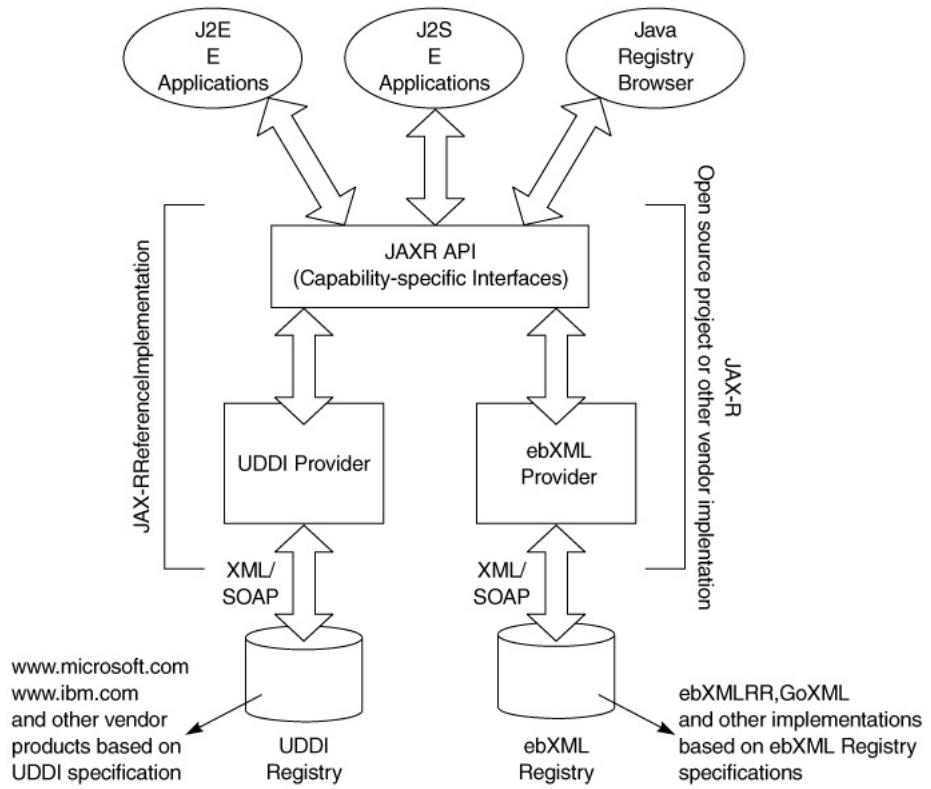
3.5.4 Java API for XML Registries (JAXR)

Java API for XML Registries (JAXR)⁶ je jednotné standardizované API pro přístup k různým registrům webových služeb. Registry uchovávají informace o zaregistrovaných webových službách a poskytují tyto informace potencionálním klientům. Jedná se tedy o jakýsi rejstřík pro vyhledávání a uchovávání informací o službách.

Dnes existuje několik různých (a vzájemně nekompatibilních) registrů, mezi nejpouží-

⁵SAAJ Standard Implementation
URL: <<https://saaj.dev.java.net/>>
⁶Java API for XML Registries (JAXR)
URL: <<http://java.sun.com/webservices/jaxr/>>

vanější registry patří UDDI⁷ (viz 4.3) a ebXML⁸ (viz 4.4).



Obrázek 3.8: Architektura JAXR (Zdroj [27]).

JAXR API vytváří abstraktní vrstvu, která tvoří spojení mezi aplikacemi a registry webových služeb, viz obrázek 3.8, který prezentuje JAXR architekturu.

API tvoří tyto základní balíčky:

- javax.xml.registry – balíček definuje rozhraní tříd pro připojení a pro práci s registry (vyhledávání a správa služeb),
- javax.xml.registry.infomodel – balíček obsahuje rozhraní objektů, které jsou uložené v registru (jak s registrem pracovat).

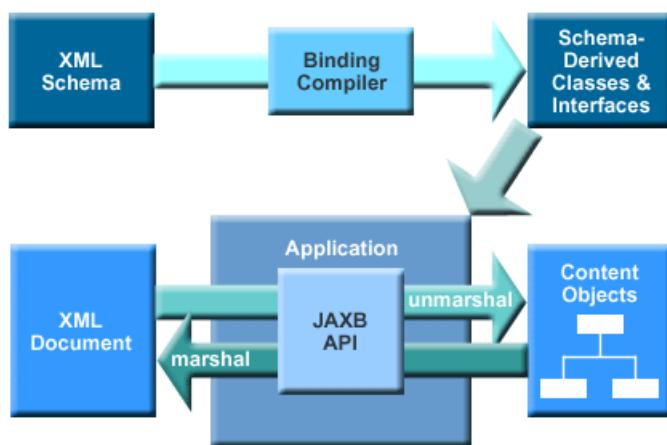
⁷The Universal Description, Discovery and Integration
URL: <<http://www.uddi.org/>>

⁸Electronic Business using eXtensible Markup Language
URL: <<http://www.ebxml.org/>>

3.5.5 Java API for XML Binding (JAXB)

Java API for XML Binding (JAXB)⁹ umožňuje z XML schémat vygenerovat Java třídy, na které lze namapovat XML dokument. Lze postupovat i opačně, tedy z Java tříd vygenerovat XML dokument, který vyhovuje danému schématu. Díky tomu nemusíme o XML téměř nic vědět a přesto s ním můžeme plnohodnotně pracovat ve vygenerovaných Java třídách.

Na obrázku 3.9 je zobrazena architektura JAXB. Obrázek schématicky zobrazuje postup pro vygenerování Java tříd z XML (unmarshal) a opačný postup pro vygenerování XML (marshal).



Obrázek 3.9: Architektura JAXB (Zdroj [27]).

Pro správné namapování XML na třídy je nutné znát DTD nebo schéma a dále je nutné znát tzv. "binding schema", ve kterém je uložena definice mapování mezi XML dokumentem a budoucí třídou. Tyto dva soubory jsou předány komplikátoru, který vygeneruje zdrojové kódy Java tříd.

⁹JAXB Reference Implementation
URL: <<https://jaxb.dev.java.net/>>

Kapitola 4

Webové služby

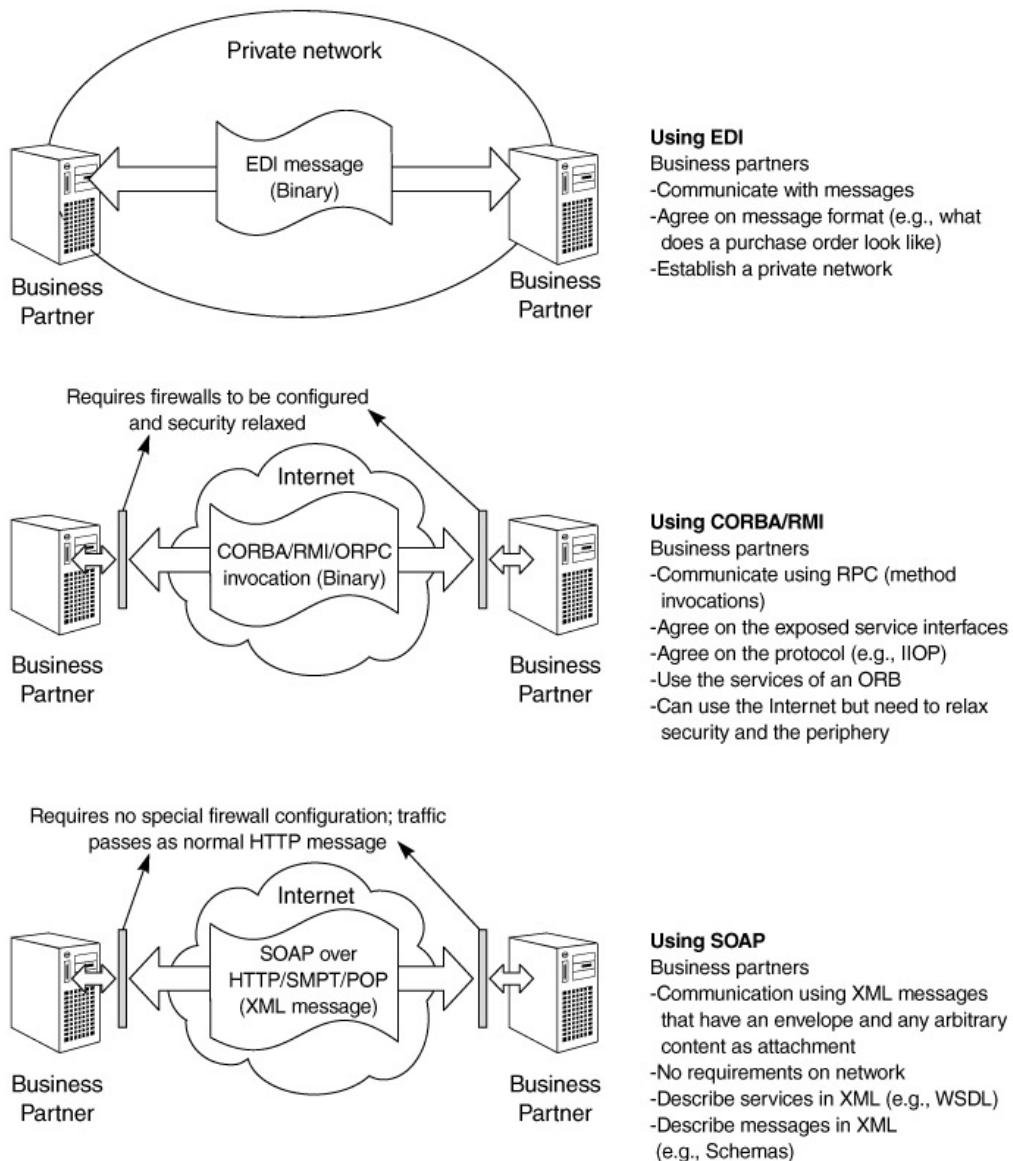
4.1 SOAP – Simple Object Access Protocol

4.1.1 Vznik SOAPOu

Před vznikem SOAPOu existovaly (a stále existují) dvě hlavní možnosti (viz obrázek 4.1) pro vzájemnou komunikaci mezi geograficky nebo fyzicky rozdělenými systémy.

- První možností je komunikace pomocí EDI (Electronic Data Interchange). EDI je elektronická výměna standardních strukturovaných zpráv mezi dvěma aplikacemi dvou subjektů (obchodních partnerů nebo také částí jedné firmy). V systémech EDI tedy spolu přímo komunikují počítačové aplikace nebo informační systémy. EDI je zřejmě nejstarší technologií umožňující elektronický obchod. Příklady využití EDI komunikace jsou např. v automobilovém průmyslu (standard ODETTE) nebo v bankovnictví (standard SWIFT). Nevýhodou je poměrně složitá implementace a existence mnoha standardů.
- Druhou možností je komunikace pomocí technologií pro vzdálené volání funkcí, např. CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation) či DCOM (Distributed extension of the Component Object Model). Každá tato technologie má svůj vlastní komunikační protokol (IIOP, JRMP, ORPC), který je postaven nad protokolem TCP/IP (Transmission Control Protocol / Internet Protocol). Využití protokolu TCP/IP pro komunikaci mezi aplikacemi/systémy je velmi výhodné, ale zpracování této komunikace opět závisí na implementačních technologiích, které jsou pro volání funkcí použity. Tzn., že implementace technologie CORBA může komunikovat s technologií CORBA, RMI může komunikovat s RMI a DCOM může komunikovat s DCOM. Ale napříč těmito technologiemi to jde velmi obtížně nebo vůbec.

Z předchozího popisu obou možností komunikace vyplývá několik zásadních nevýhod. Po vzniku jazyka XML následně vznikla myšlenka na posílání XML dat protokolem HTTP.



Obrázek 4.1: Vývoj SOAP (Zdroj [27]).

Vzniklo několik definic, např. XML-RPC, WDDX (Web Distributed Data Exchange), XMI (XML Metadata Exchange) a další. Dalším požadavkem na přenášená XML data byla možnost vzdáleného volání procedur. Z iniciativy firem Microsoft a IBM vznikl z protokolu XML-RPC¹ protokol SOAP (Simple Object Access Protocol)².

¹XML-RPC Home Page

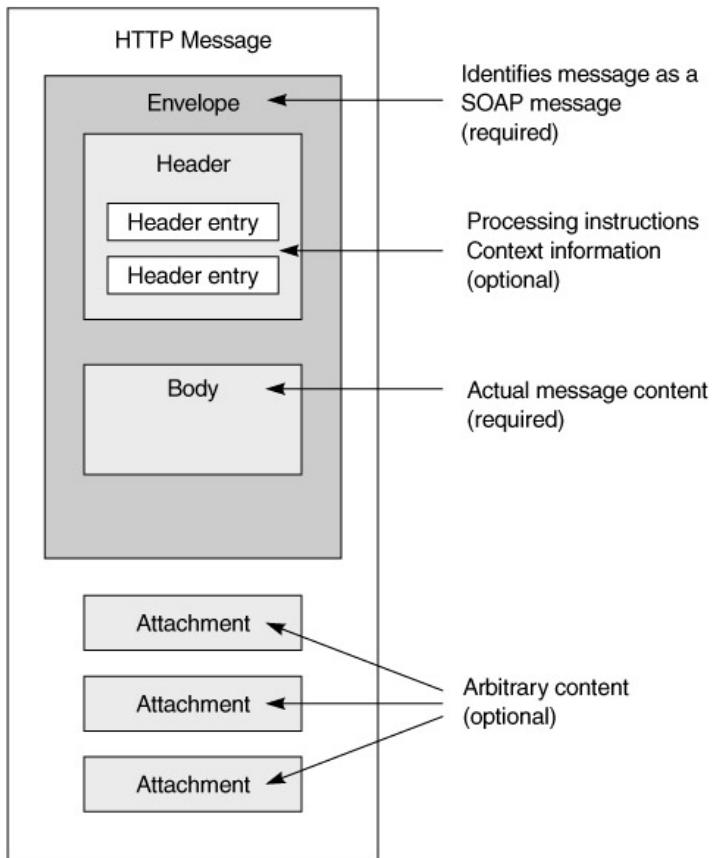
URL: <<http://www.xmlrpc.com/>>

²SOAP Specifications

URL: <<http://www.w3.org/TR/soap/>>

4.1.2 Protokol SOAP

SOAP [16] je protokol pro posílání XML zpráv – SOAP zpráv (viz obrázek č. 4.2) pomocí protokolu HTTP a je základem webových služeb. Princip volání je následující: webový server, na kterém je umístěna webová služba, čeká na požadavky klientů a v okamžiku, kdy přes HTTP, většinou pomocí metody POST, přijde (request) SOAP zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán (ve formě SOAP zprávy) zpět klientovi jako odpověď (response).



Obrázek 4.2: Struktura SOAP zprávy (Zdroj [27]).

SOAP zprávy jsou validní XML dokumenty, které mají společnou strukturu (viz obrázek č. 4.2). SOAP zpráva obsahuje kořenový element *Envelope* (obálku), která je dále rozdělena na dvě části: volitelnou *Header* (hlavičku) a povinnou *Body* (tělo). Tělo zprávy nese vlastní informace, které webová služba dokáže zpracovat, hlavičky zpráv obsahují metainformace. Data nemusí být obsažena jen v těle zprávy, pro přenos větších objemů dat (zejména binárních) lze použít data „přibalená“ jako příloha/y – *Attachment*. *Attachment* je tvořen hlavičkou a vlastními daty. Hlavička specifikuje MIME typ přenášených dat. Na zpracování a přijímání těchto zpráv lze s výhodou použít API SAAJ – SOAP with Attachments API

for Java³.

Příklad 4.1 prezentuje zavolání webové služby protokolem SOAP. Je volána operace *Secti*, která má dva vstupní parametry *a* a *b*. Výsledek je zobrazený v následujícím příkladě 4.2, kdy výsledkem je součet těchto dvou parametrů.

Příklad 4.1: HTTP request

```
HTTP request header
POST / HTTP/1.1
Content-Type: text/xml; charset="utf-8"
Content-Length: 513
SOAPAction:
User-Agent: Java1.3.1_04
Host: localhost:8080
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns0="http://project5/types/">
  <soap:Body xmlns:ns1="http://project5/types/">
    <ns1:Secti>
      <ns1:a>2</ns1:a>
      <ns1:b>2</ns1:b>
    </ns1:Secti>
  </soap:Body>
</env:Envelope>
```

Příklad 4.2: HTTP response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
SOAPAction: ""
Transfer-Encoding: chunked
Date: Thu, 03 Oct 2006 17:52:05 GMT
Server: Apache Coyote HTTP/1.1 Connector [1.0]

<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns0="http://project5/types/">
  <env:Body>
    <ns0:SectiResponse>
      <ns0:result>4</ns0:result>
    </ns0:SectiResponse>
  </env:Body>
```

³SAAJ Standard Implementation
URL: <<https://saaj.dev.java.net/>>

```
</env:Envelope>
```

4.1.3 SOAP Fault

SOAP *Fault* (chybová značka/element) je specifický typ XML elementu, který je umístěn přímo v těle – *Body* návratové SOAP zprávy (viz příklad 4.3). Pro lepší pochopení lze říci, že SOAP *Fault* je obdobou výjimek v programovacích jazycích.

Tato chybová značka informuje o chybě, která vznikla v průběhu zpracování požadavku. Chybová značka může obsahovat tři základní části:

- element *faultcode* určuje, kde došlo k chybě,
- element *faultstring* obsahuje chybové hlášení, které je srozumitelné lidem,
- element *faultactor* obsahuje URI, které je součástí SOAP zprávy, vede k místu, kde byla chybná zpráva vygenerována.

Chybová značka může obsahovat ještě jednu speciální značku *detail*, která může obsahovat libovolný XML dokument a většinou podrobněji popisuje chybový stav (viz příklad 4.3).

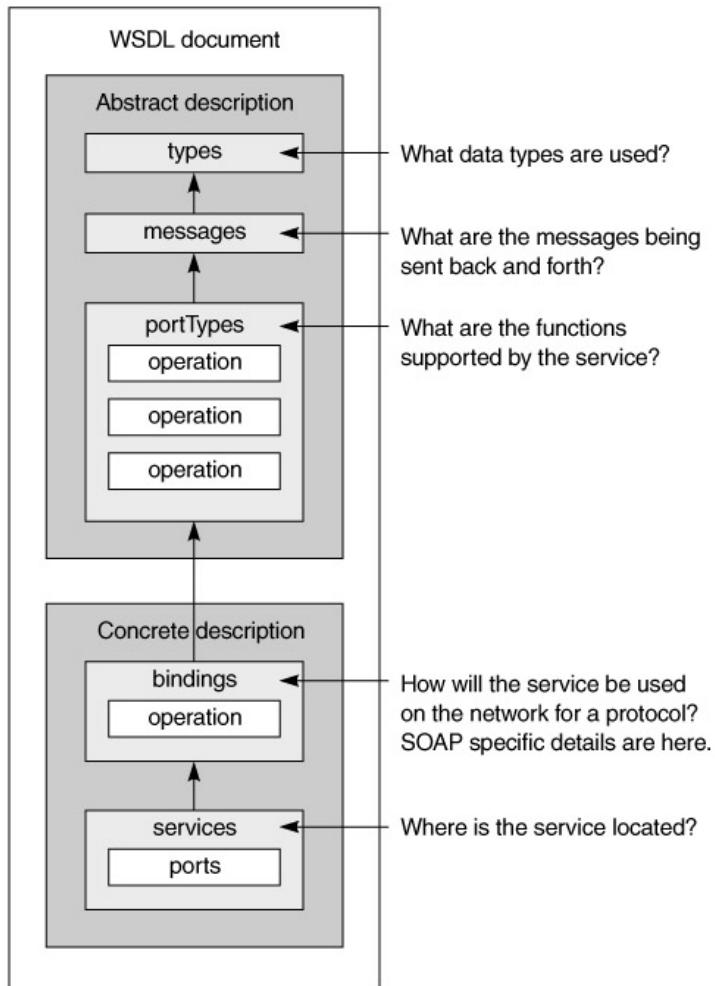
Příklad 4.3: Chybová zpráva protokolu SOAP

```
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns0="http://project5/types/">
<env:Body>
  <env:Fault>
    <faultcode
      xmlns:ans1="http://localhost:8888/MyWebService1">ans1:ServerFailed
    </faultcode>
    <faultstring>hodnota zadaneho parametru neodpovida pozadovane hodnote
    </faultstring>
    <faultactor>http://localhost:8888/MyWebService1/Secti</faultactor>
    <detail>
      <Chyba>Nespravna hodnota parametru. Parametr muze obsahovat pouze
        cislo!</Chyba>
    </detail>
  </env:Fault>
</env:Body>
</env:Envelope>
```

4.2 WSDL

Abysme mohli vzdáleně volat funkce, musíme znát, jaké funkce chceme volat, jaké mají parametry a jaké vrací hodnoty. K tomuto účelu slouží WSDL [3] – jazyk pro popis webo-

vých služeb. Je opět založený na XML. Vznikl sloučením tří jazyků firem IBM, Microsoft a Ariba s názvy NASSL, SCL a SDI.



Obrázek 4.3: Struktura WSDL (Zdroj [27]).

Dokument WSDL plně webovou službu popisuje, jestliže máme WSDL, můžeme začít s webovou službou aktivně pracovat. Mnohá dnešní vývojová prostředí zvládají na základě WSDL automaticky vygenerovat kód pro volání služby. Dále lze WSDL vygenerovat ze stávajícího kódu a poskytnout ho případným uživatelům. Schéma je zobrazeno na obrázku č. 4.3.

WSDL dokument může obsahovat pět částí (viz obrázek č. 4.3):

- část *types* definuje použité datové typy pomocí XML Schema,
- část *messages* definuje typy zpráv, které služba přijímá či vysílá,
- část *portTypes* obsahuje jednu nebo více značek *operation*, které definují podporované

operace, jako vstupní a výstupní zprávy s možnými chybovými zprávami – *Fault*, operace má kromě jména také určen jmenný prostor,

- část *bindings* definuje jakými komunikačními protokoly lze službu používat,
- část *services* definuje, kde se služba nachází – na jaké adrese je k dispozici.

4.2.1 Styly WSDL

Rozhraní webové služby ve WSDL lze popsat čtyřmi různými způsoby, protože část *bindings* specifikuje pro každou operaci dva atributy – *style* a *use*. Atribut *style* může nabývat hodnot *rpc* nebo *document*, atribut *use* může nabývat hodnot *literal* nebo *encoded*. Existují tedy čtyři kombinace atributů *style/use*:

- RPC/encoded (příklad 4.4)
 - určen pro volání operací,
 - informace o typu je obsažena v SOAP zprávě,
 - zprávu lze obtížně validovat proti Schema,
 - umožňuje cyklické odkazy mezi přenášenými objekty a polymorfismus.
- RPC/literal (příklad 4.5)
 - určen pro volání operací,
 - není obsažena informace o typu,
 - zprávu lze obtížně validovat proti Schema,
 - neumožňuje cyklické odkazy mezi přenášenými objekty a polymorfismus.
- Document/encoded – nelze rozumně použít.
- Document/literal (příklad 4.6)
 - určen pro přenos XML,
 - není obsažena informace o typu,
 - zprávu lze snadno validovat proti Schema,
 - nevyskytuje se název operace.

Následující příklady jsou převzaty z tutoriálu Martina Kuby⁴.

⁴Martin Kuba
Ústav výpočetní techniky, Masarykova univerzita.
URL: <<http://www.ics.muni.cz/makub/>>

Příklad 4.4: RPC/encoded

WSDL :	SOAP :
<message name="myMethodRequest">	
<part name="x" type="xsd:int"/>	<soap:body>
</message>	<myMethod>
	<x xsi:type="xsd:int">5</x>
	</myMethod>
	</soap:body>

Příklad 4.5: RPC/literal

WSDL :	SOAP :
<message name="myMethodRequest">	
<part name="x" type="xsd:int"/>	<soap:body>
</message>	<myMethod>
	<x>5</x>
	</myMethod>
	</soap:body>

Příklad 4.6: Document/literal

WSDL :	SOAP :
<types>	
<schema>	<soap:body>
<element name="x" type="xsd:int"/>	<x>5</x>
</schema>	</soap:body>
</types>	
<message name="myMethodRequest">	
<part name="params" element="x"/>	
</message>	

Dnes je doporučován styl Document/literal wrapped (příklad 4.7), který sloučuje výhody stylu Document/literal a navíc obsahuje název operace.

Příklad 4.7: Document/literal wrapped

WSDL :	SOAP :
<types>	
<schema>	<soap:body>
<element name="myMethod"/>	<myMethod>
<complexType>	<x>5</x>
<sequence>	</myMethod>
<element name="x" type="xsd:int"/>	</soap:body>
</sequence>	
</complexType>	
</element>	
</schema>	
</types>	
<message name="myMethodRequest">	
<part name="parameters" element="myMethod"/>	
</message>	

4.3 UDDI

UDDI [35] nabízí veřejnou databázi, do které mohou tvůrci webových služeb ukládat popisy služeb. Uživatelé mohou tuto databázi prohlížet a hledat podle popisů, názvů či poskytovatelů požadovanou webovou službu. UDDI můžeme přirovnat ke zlatým stránkám webových služeb. V této databázi lze vyhledávat podle klíčových slov, podle oborů či podle služeb (které jsou poskytovány), nebo dokonce podle geografického místa.

Zápis do databáze UDDI lze rozdělit na tři části. „Bílé stránky“ popisují firmu nabízející službu: název, adresa, kontakt atd. „Žluté stránky“ zahrnují průmyslové kategorie založené na standardních systematikách, jako je North American Industry Classification System (NAICS)⁵ a Standard Industrial Classification (SIC) System⁶. „Zelené stránky“ podrobně popisují rozhraní služby. Popis, jakým způsobem jsou služby v UDDI definovány, je uveden v XML dokumentu zvaném Type Model (tModel). Nejčastěji tModel obsahuje WSDL soubor, který popisuje rozhraní služby.

Tato veřejná databáze má dvě hlavní nevýhody. První nevýhodou je neaktuálnost a špatná kvalita záznamů. Druhou nevýhodou je důvěryhodnost poskytovatelů služeb.

Podle [25] tři největší provozovatelé veřejných UDDI rejstříků – IBM, Microsoft a SAP, vypnuli své věřejně přístupné rejstříky v lednu 2006. Důvodem bylo, že většina záznamů v rejstřících byla v neutěšeném – nefunkčním stavu.

Jinou možností, než použítí UDDI, skýtá možnost použití WSIL – Web Services Inspection Language⁷ (jazyk pro popis služeb poskytovaných určitou firmou či institucí). WSIL byl definován firmami IBM a Microsoft v roce 2001, ale jeho úspěch je zatím v nedohlednu. WSIL dokument je vždy pojmenován *inspection.wsil* a je umístěn v kořenovém adresáři web serveru příslušného poskytovatele služeb. WSIL tedy umožňuje procházet množinu webových služeb nabízených na určitém serveru poskytovatelem.

4.4 ebXML

Zakladateli iniciativy ebXML (Electronic Business XML) jsou UN/CEFACT⁸ a OASIS⁹.

Cílem ebXML¹⁰ je vytvoření obecných a veřejných struktur, které popisují obchodní

⁵North American Industry Classification System (NAICS)
URL: <<http://www.census.gov/epcd/www/naics.html>>

⁶Standard Industrial Classification (SIC) System
URL: <<http://www.census.gov/epcd/www/sic.html>>

⁷Web Services Inspection Language (WS-Inspection)
URL: <<http://www.ibm.com/developerworks/library/specification/ws-wsilspec/>>
URL: <<http://msdn2.microsoft.com/en-us/library/ms951237.aspx>>

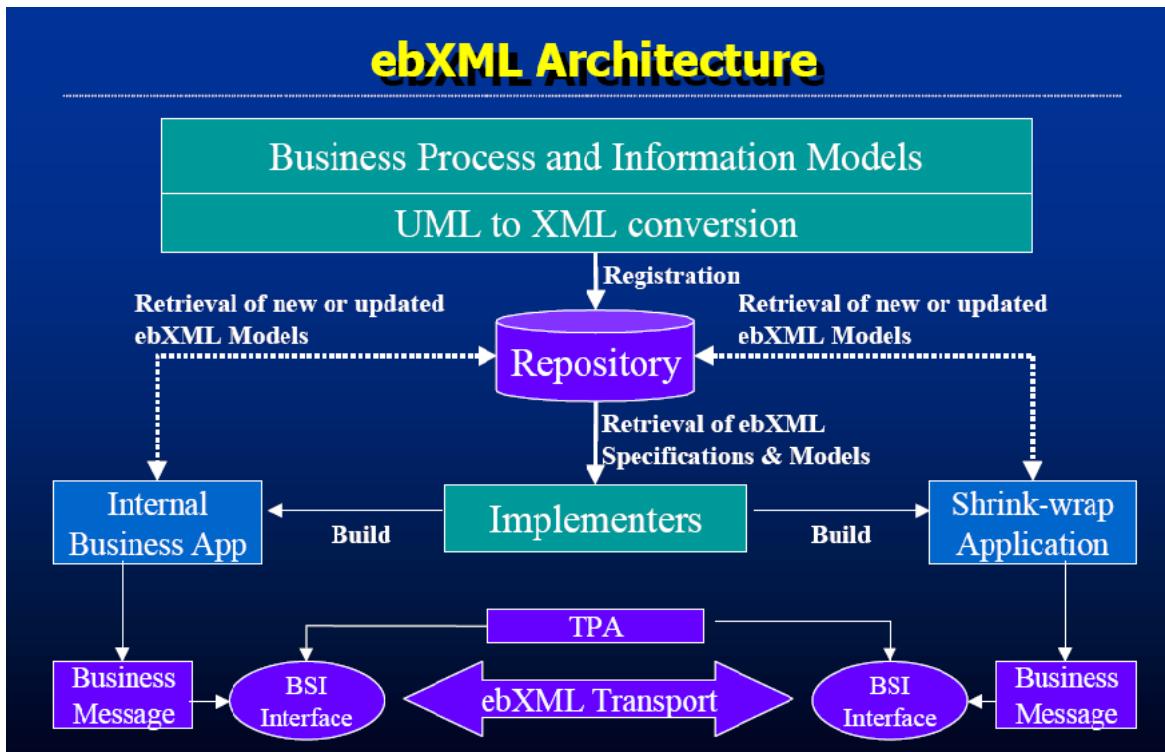
⁸United Nations Body for Trade Facilitation and Electronic Business
URL: <<http://www.unece.org/cefact/>>

⁹The Organization for the Advancement of Structured Information Standards
URL: <<http://www.oasis-open.org/home/index.php>>

¹⁰ebXML – Enabling A Global Electronic Market
URL: <<http://www.ebxml.org/>>

procesy a aktivity, které je možné popsat pomocí syntaxe XML. Toto řešení umožňuje vznik levných a efektivních implementací, které jsou nenáročné na provoz a umožňuje elektronicky obchodovat všem firmám, bez ohledu na velikost či geografické umístění, za pomoci standardizovaných obchodních zpráv založených na XML.

Obecné zásady ebXML jsou veřejné a to umožňuje vývojářům programovat aplikace vytvořené na základě známé a jednotné syntaxe ebXML a snížit tak náklady na výměnu obchodních informací. Implementace architektury ebXML je mnohem levnější než implementace EDI.



Obrázek 4.4: Architektura ebXML (Zdroj [27]).

Mechanismus ebXML tvoří tři základní pilíře - BPM (modelování obchodních procesů), UML (unifikovaný modelovací jazyk) a XML. Standardizační organizace prostřednictvím workshopů a pracovních fór vytvářejí modely, které definují a standardizují běžné obchodní procesy ve firmách. Zájmová sdružení a obchodní skupiny definují obchodní scénáře, popisující vztahy mezi obchodními elementy, informační posloupnosti a potřebné dokumenty, které se v rámci jednotlivých procesů vyměňují. Jednotlivé firmy pak na jejich základě vytvářejí firemní profily, které shrnují informace o nich. To vše se shromažďuje ve veřejně přístupných archivech a registračních knihovnách (repository), se kterými může každý subjekt komunikovat a do nichž může přispívat.

Obrázek 4.4 zobrazuje architekturu ebXML. Podnikové procesní a informační modely popsané (Business process and information models) pomocí UML jsou konvertovány do XML a uloženy do knihovny (Repository). Z repository jsou specifikace a modely vy-

zvedávány a implementovány do podnikových aplikací (Internal business application) nebo do aplikačních balíků (Shrink-wrap application). Tyto aplikace vytváří obchodní zprávy (Business message), které se pomocí rozhraní aplikace ebXML (BSI Interface) rozesílají obchodním partnerům. Vzájemná komunikace je podmíněna uzavřenou obchodní smlouvou (TPA), která určuje podmínky spolupráce.

Kapitola 5

Současný stav využití webových služeb

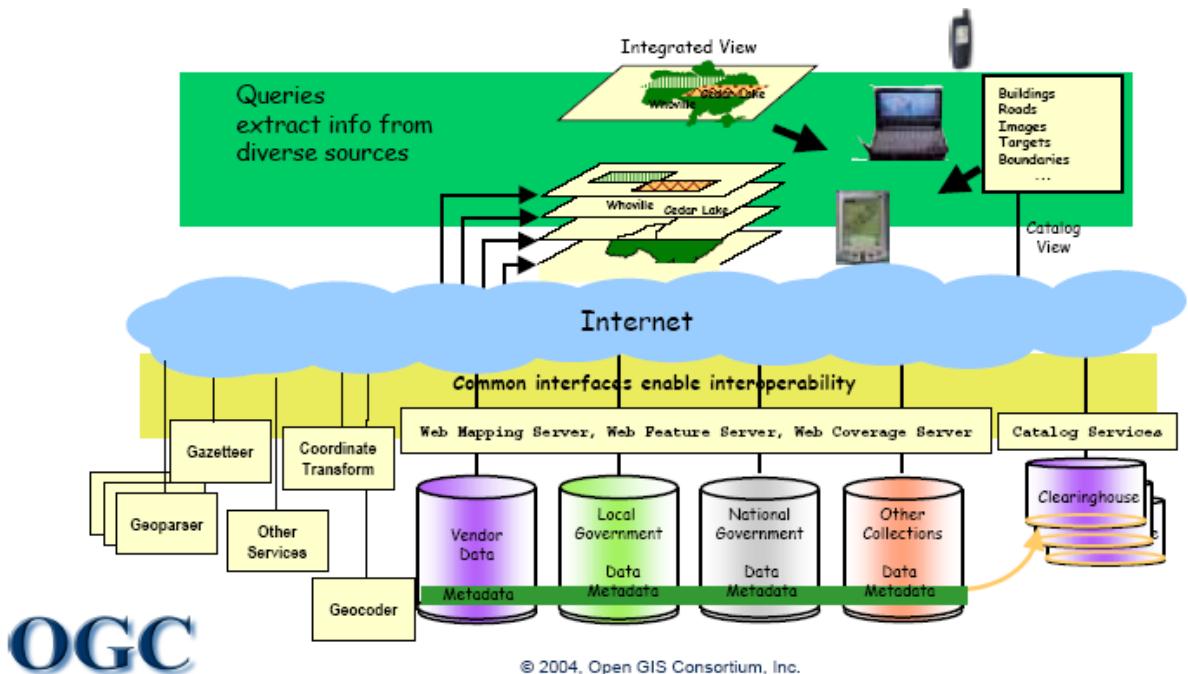
Oblastí, kde se webové služby rychle rozvíjejí, je oblast Geografických informačních systémů, dále jen GIS. Poskytovatelů různých geografických dat existuje mnoho, ale každý z těchto poskytovatelů vlastní data, která jsou důležitá či prospěšná pouze pro něj. Tzn., že pokud hledáme kompletní informaci o nějakém objektu, jsme nuceni prohledávat více úložišť dat. Pro podporu a rozvoj webových služeb v oblasti GIS bylo založeno Open GIS Consortium (OGC) [30] viz kapitola 5.1.

Webové služby se ukazují jako vhodný komunikační prostředek, který dokáže propojit poskytovatele dat a uživateli poskytnout kompletní balík informací z více zdrojů do jednoho místa. Zároveň však není potřeba, aby poskytovatel měnil svoji stávající strukturu dat či zasahoval do fungujících vnitřních procesů práce s daty. Těchto "integračních" principů je s výhodou využito při významných projektech Evropského společenství jako je EULIS [11] viz kapitola 5.2 a INSPIRE [12] viz kapitola 5.3.

5.1 Open GIS Consortium

Open GIS Consortium je mezinárodní neziskové sdružení, jehož hlavní vizí je „svět, ve kterém jsou geoinformace všem ku prospěchu a geo-služby jsou přístupné z kterékoliv sítě, aplikace a platformy“. Cílem OGC je tvorba specifikací pro geoprostorová rozhraní, která jsou otevřeně využitelná na globální úrovni. Základní dokumenty OGC specifikují popis jednoduchých prostorových prvků (simple features), katalogových služeb, webových služeb pro rastrová a vektorová data (web map service, resp. web feature service), deskriptory pro stylové vrstvy (Styled Layer Descriptors), rozšiřující Web Map Service o další operace. Formátem pro výměnu geodat je Geography Markup Language (GML), založený na principech jazyka XML. OpenGIS Architektura je zobrazena na obr. 5.1.

Ve světě nejčastěji používané OGC specifikace jsou Web Map Service (WMS) a Web Feature Service (WFS). WMS je určena pro zobrazování map ve formátech (gif, png, jpeg, atd.). Jednotlivé vrstvy mohou poskytovat různé servery. Služba WMS se může chovat



Obrázek 5.1: OpenGIS Architektura (Zdroj [30]).

i jako klient – může kombinovat více map poskytovaných jinými službami WMS. Specifikace WMS podporuje základní dotazy týkající se obsahu mapy:

- GetCapabilities: vrací XML dokument s metadaty popisujícími službu. Klient tento dokument zpracovává a nabízí uživateli seznam dostupných vrstev, jejich popis, vlastnosti atd.
- GetMap: vrací vlastní mapu v rastrovém tvaru (gif, png, jpeg, atd.),
- GetFeatureInfo: vrací informace o objektu zadaného/vybraného uživatelem.

WFS je určena k přenosu vektorových dat ve formátu Geography Markup Language (GML). Tato služba je vhodná pro další zpracovávání takto získaných dat. WFS podporuje tyto dotazy:

- GetCapabilities: poskytuje informaci o tom, které typy vektorových dat a operace s nimi podporuje,
- DescribeFeatureType: předává informace o službách podporovaných pro daný typ vektorových dat,
- GetFeature: předává informace o daném vektorovém prvku,
- Transaction: zabezpečuje poskytnutí služby na požadavek ze strany uživatele, jedná se o požadavky spojené s modifikací prvku – vložení, odstranění či aktualizaci,

- LockFeature: umožňuje požadavek uzamknout používané prvky po dobu prováděné transakce.

GML specifikuje XML kódování pro přenos a uložení geografických informací modelovaných v souladu s konceptuálním rámcem použitým v sérii standardů ISO 19100 a obsahujících jak prostorové, tak neprostorové vlastnosti geografických prvků. Standard GML definuje syntaxi XML schématu, mechanismy a konvence, které:

- poskytují otevřený rámec pro definování geoprostorových aplikačních schémat a objektů, jenž je nezávislý na poskytovateli,
- povolují profily vyhovující popisným možnostem příslušných podmnožin rámce GML,
- podporují popis geoprostorových aplikačních schémat pro specializované domény a informační společnosti,
- umožňují tvorbu a údržbu propojených geografických aplikačních schémat a datových sad,
- podporují uložení a přenos aplikačních schémat a datových sad.

Formát GML byl v roce 2000 ve Velké Británii zvolen jako národní standard pro výměnu geoprostorových dat.

5.2 EULIS

Projekt EULIS (European Land Information Service) – Evropské pozemkové informační služby byl zahájen v roce 2002 a jeho cílem bylo propojit národní katastrální systémy členů EU. V první vlně se tohoto projektu zúčastnily tyto země: Švédsko, Nizozemí, Anglie, Wales, Skotsko, Rakousko, Finsko, Norsko, Island, viz [32].

Mezi hlavní důvody pro zahájení tohoto projektu, patří zejména problémy s:

- poskytováním hypoték na nemovitosti vlastněné v zahraničí,
- společným evropským trhem s nemovitostmi,
- různými přeshraničními projekty,
- investicemi v EU.

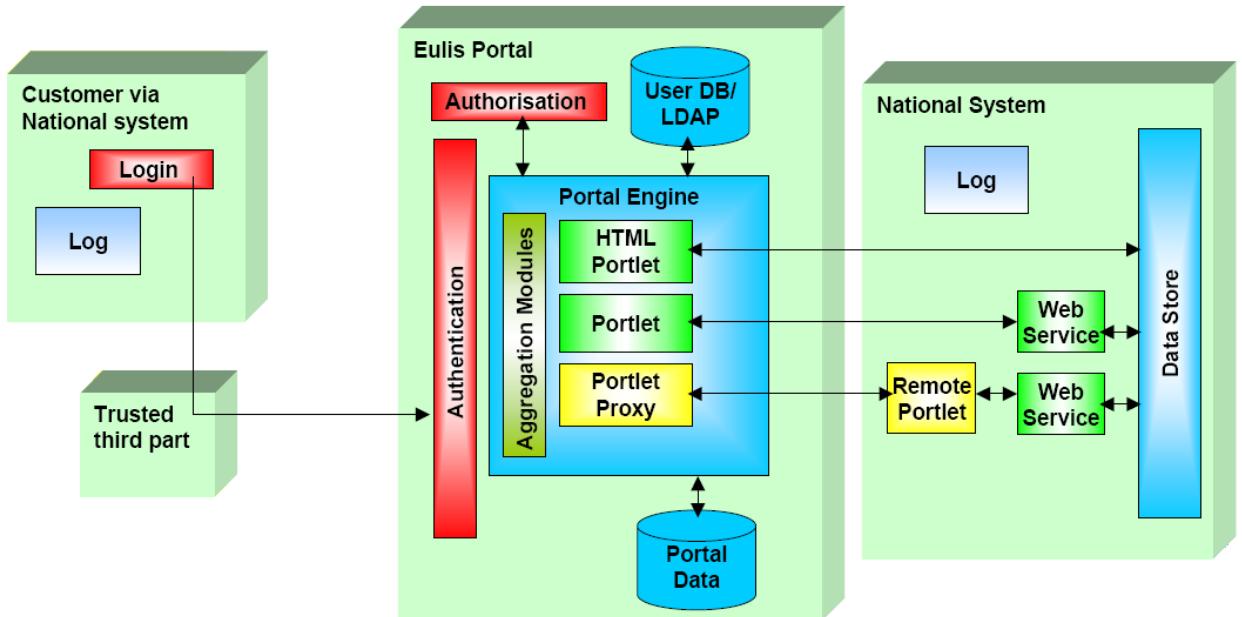
EULIS je určen pouze pro profesionální zákazníky a umožňuje přístup do dalších národních systémů přes jednotný portál. Na portálu jsou k dispozici referenční informace, popisující jednotlivé národní systémy jak z hlediska právního, tak popis a vysvětlení národních informací, které je možno přes portál získat. Součástí je i glosář, který obsahuje

výklad asi 50 pojmu ve všech národních jazycích. Zadávání informací je popsáno v angličtině, výpisu informací se poskytují v národních jazycích (Švédsko a Nizozemí poskytují výpisu i v angličtině).

Funkčnost služby EULIS je možno popsat následovně, viz [32]:

- uživatel se přihlásí do národního systému poskytování informací, kde je autorizován a autentifikován (musí být právoplatným klientem národního systému),
- uživatel zvolí službu EULIS a je přihlášen na portál EULIS, kde je národní systém považován za řádného uživatele (autentifikace uživatele se neopakuje, proběhla u národního systému), portálu jsou poskytnuty informace o uživateli pro fakturační účely, na portálu jsou dostupné informace metadat o registrech, legislativě a způsobech transakcí nemovitostí,
- uživatel zvolí zdroj informací (jiný národní systém), které chce získat. EULIS portál je řádným klientem v národních systémech,
- uživatel získá potřebnou informaci, národní systém poskytuje informace přes portál EULIS a vrací informace pro fakturaci,
- uživateli je zaslána faktura jeho národním systémem.

Technické řešení projektu je podrobněji zobrazeno na obrázku č. 5.2.



Obrázek 5.2: Architektura systému EULIS (Zdroj [11]).

V současné době je připravován projekt EULIS plus s cílem prozkoumat možné zavedení služby do nových členských států EU, včetně ČR.

5.3 INSPIRE

Cílem projektu INSPIRE (Infrastructure for Spatial Information in Europe) [12, 5] je vybudování prostorové informační infrastruktury v rámci Evropského společenství. Projekt je iniciován Evropskou komisí, která v současnosti vytváří konečnou verzi¹ směrnice INSPIRE. Takto vybudovaná infrastruktura bude sloužit ke koordinaci a vzájemnému propojení mezi daty nashromážděnými jednotlivými členskými státy, zejména s ohledem na oblast životního prostředí.

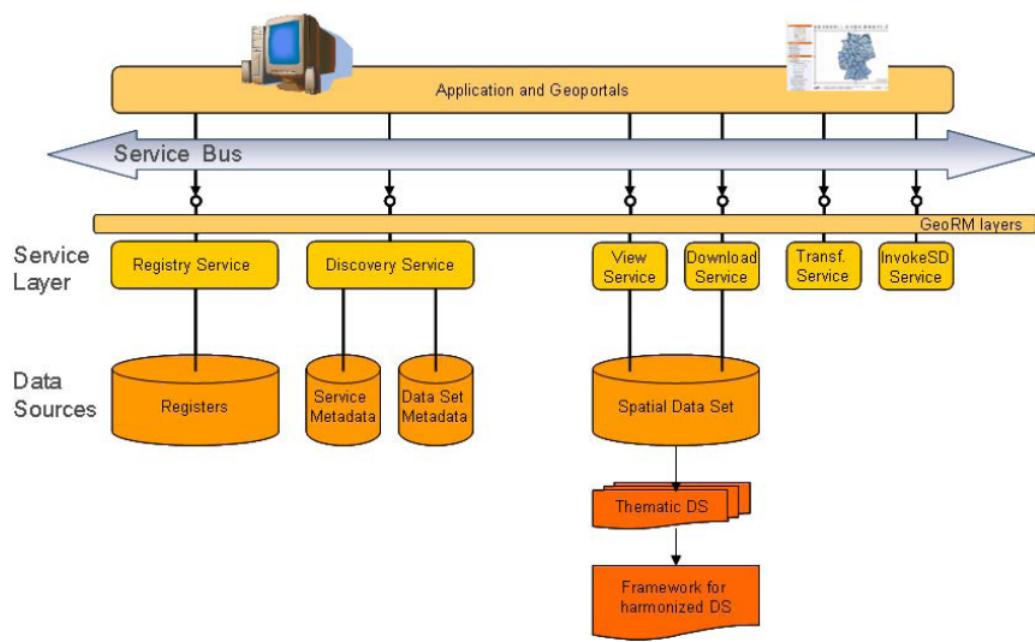
Základní principy INSPIRE:

- data jsou shromažďována jednou, data jsou spravována tam, kde je to nejefektivnější,
- data a služby jsou jednotně popsány stanovenými metadaty,
- jsou stanoveny jednotné podmínky k poskytování a využívání prostorových dat a informací,
- musí být možné data bezešvým způsobem kombinovat z různých zdrojů a sdílet mezi uživateli a aplikacemi,
- data budou poskytována bezplatně.

V ČR vznikla meziresortní komise² pro podporu a rozvoj principů INSPIRE. Jedním z prvních úkolů je definovat metadatový profil napříč jednotlivými resorty. Tento metadatový profil bude důležitý pro bezproblémovou výměnu dat a jejich správné pochopení, zařazení, rozvíjení a udržování v rámci nejen tohoto projektu.

¹směrnice INSPIRE vyšla 25.dubna 2007 a v platnost vstoupila 15. května 2007

²autor byl jejím členem jako zástupce ČÚZK



Obrázek 5.3: Architektura služeb INSPIRE (Zdroj [12]).

Kapitola 6

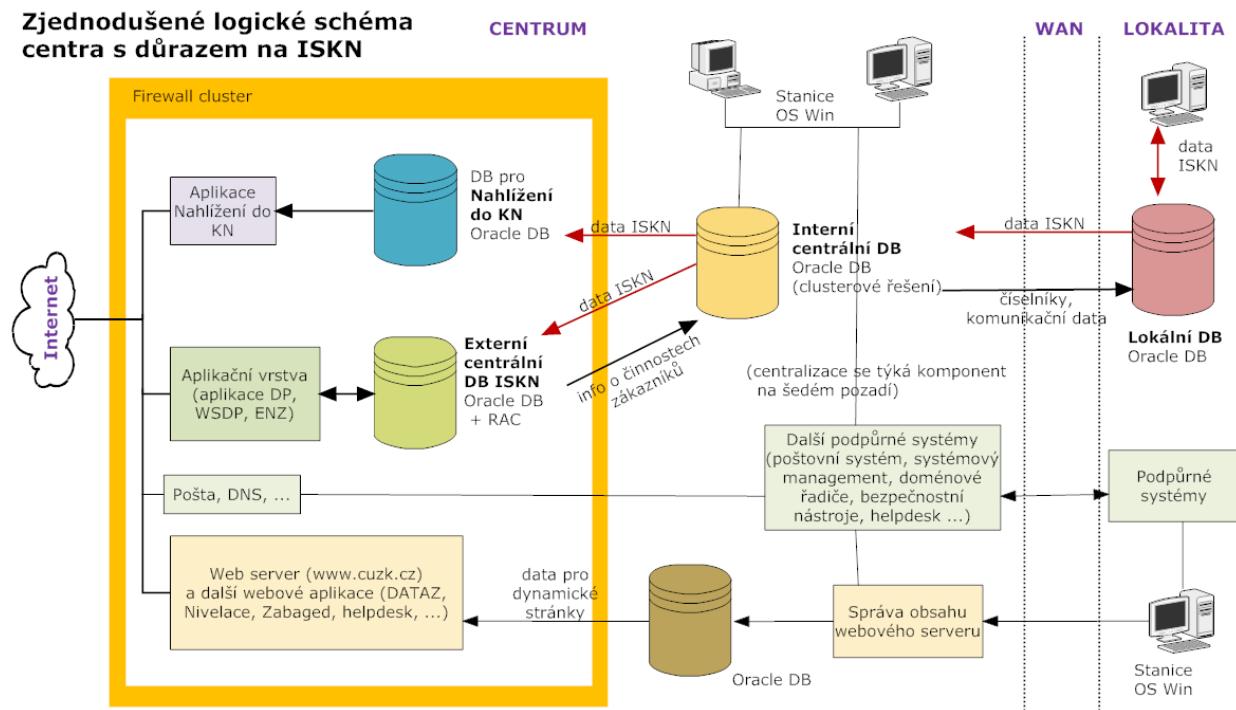
ISKN

Informační systém katastru nemovitostí (ISKN) je informační systém pro podporu výkonu státní správy katastru nemovitostí a pro zajištění uživatelských služeb katastru nemovitostí. Obsahuje zejména prostředky pro vedení popisných a geodetických informací ve společné databázi, dále prostředky pro podporu správních a administrativních činností při vedení katastru nemovitostí a pro správu dokumentačních fondů.

ISKN je jedním z nejrozsáhlejších projektů v rámci budování informační infrastruktury veřejné a státní správy v ČR. Vybudováním nového informačního systému KN byla na základě veřejné obchodní soutěže pověřena firma APP Systems, spol. s r.o. O výstavbě nového systému bylo rozhodnuto vládním nařízením č. 708 ze dne 12.11.1997. V roce 1998 proběhla veřejná obchodní soutěž na výběr software. Byl vybrán databázový systém ORACLE, pro práci s mapami systém BENTLEY a pro řízení systému UNICENTER TNG.

ISKN je tvořen 107 lokálními a jednou centrální databází. Propojení katastrálních úřadů (dále jen KÚ) s centrální databází je provedeno pomocí komunikační sítě WAN. Na lokální úrovni jsou prováděny veškeré změny související s výkonem státní správy katastru nemovitostí. Obsahové změny jsou každé dvě hodiny replikovány z lokálních databází do interní centrální databáze, která obsahuje cca 90% dat lokálních databází. Z interní centrální databáze se data replikují do externí centrální databáze, která slouží k poskytování dat katastru nemovitostí dálkovým přístupem pomocí internetového přístupu. Zjednodušené logické schéma systému ISKN je zobrazeno na obrázku č. 6.1.

Systém ISKN plně zajišťuje vedení jak právních vztahů k nemovitostem, tak i vedení technických údajů o nemovitostech. Jsou integrovány soubory popisných a geodetických informací s dokumentačními fondy. Údaje o fyzických osobách vlastníků a dalších účastníků řízení, tzv. oprávněných subjektů katastru nemovitostí, jsou porovnávány s údaji Informačního systému evidence obyvatel, vedeného Ministerstvem vnitra ČR, také údaje o právnických osobách jsou porovnávány s údaji vedenými o nich v systému Ministerstva vnitra ČR. Výstupy z katastru nemovitostí jsou opatřovány nejen časem vyhotovení, ale i časem platnosti údajů. ISKN umožňuje zobrazení stavu katastru nemovitostí k danému okamžiku v historii.



Obrázek 6.1: Zjednodušené logické schéma ISKN.

6.1 Systém ISKN

ISKN je vytvořen pomocí dvouvrstvé architektury klient/server. Systém je vybudován na těchto technologických základech:

- databázový systém Oracle,
- nástroje pro vizualizaci a údržbu prostorových dat od fy. Bentley (MicroStation a GeoOutlook),
- nástroje systémového managementu CA Unicenter,
- databázové servery tří úrovní podle velikosti pracovišť, servery systémového managementu a pracovní stanice tří typů podle činností v ISKN,
- dva clustery centrálních unixových serverů Alpha výrobce Compaq (HP) a disková pole pro interní a externí centrální databáze ISKN,
- síť WAN, původně založená na technologii Frame Relay, dnes na MPLS.

6.1.1 Rozdělení ISKN

Systém ISKN je navržen jako procedurální informační systém a v rámci vlastního řešení je rozdělen na několik vzájemně velmi úzce propojených částí – aplikací.

Přehled jednotlivých aplikací ISKN:

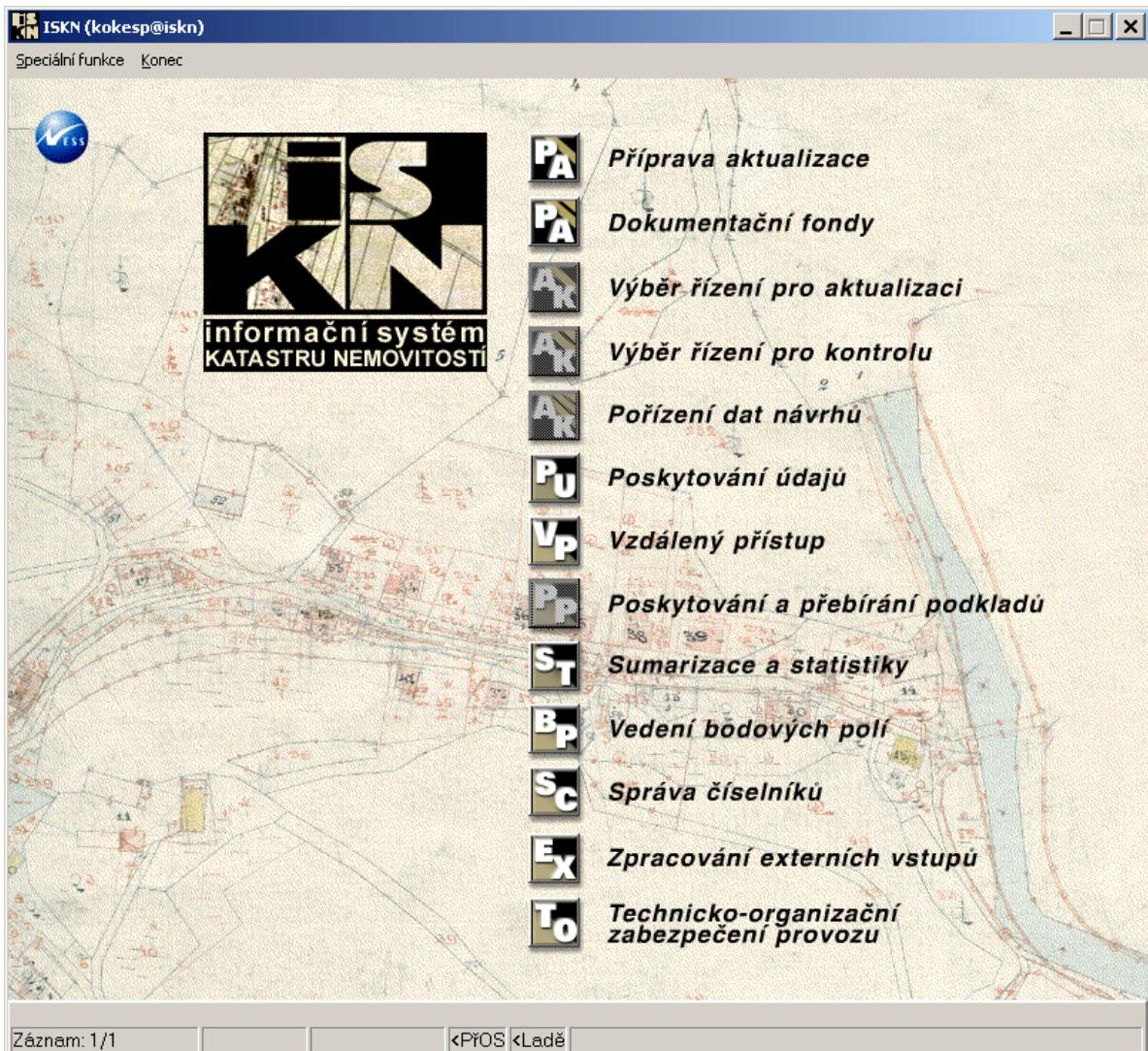
- Aplikace PA — Příprava aktualizace KN
- Aplikace AK — Aktualizace KN
- Aplikace PU — Poskytování údajů
- Aplikace DP — Dálkový přístup
- Aplikace VP — Vzdálený přístup
- Aplikace PP — Poskytování a přebírání podkladů
- Aplikace ST — Sumarizace a statistiky
- Aplikace BP — Bodová pole
- Aplikace SC — Správa číselníků
- Aplikace EX — Vazby na externí systémy
- Aplikace TO — Technicko-organizační zabezpečení provozu ISKN

6.1.1.1 Aplikace PA — Příprava aktualizace KN

Úkolem aplikace PA je vést evidenci průběhu vyřizování požadavků souvisejících s vedením KN, zapsaných v protokolech a knihách. Jedná se zejména o činnosti vyplývající ze zápisu vkladu nebo výmazu práva z katastrálního operátu, potvrzování geometrických plánů (dále jen GP) a poskytování informací. Jejím prostřednictvím se tedy zaznamenává kdo, kdy a jaký úkon provedl v rámci některého z typů řízení. Aplikace PA dále podporuje správu dokumentačních fondů jako jsou: sbírka listin, dokumenty obnovy a údržby katastrálního operátu, dokumenty dřívějších pozemkových evidencí.

6.1.1.2 Aplikace AK — Aktualizace KN

Aplikace AK slouží k aktualizaci KN, kontrole aktualizace zápisu práv vkladem nebo záznamem, ke kontrole GP před jeho převzetím. Obsahuje speciální funkce, kam patří funkce hromadné aktualizace, což je např. přesun katastrálního území (dále jen k.ú.) do jiného, sloučení nebo rozdelení k.ú., změna hranice k.ú., vstup obnoveného katastrálního operátu. Aplikace zahrnuje i funkci pro konstrukci historického stavu KN a funkci pro



Obrázek 6.2: Hlavní menu aplikace ISKN.

tvorbu přehledové mapy. Speciální funkce, které se týkají hromadné aktualizace, jsou speciálním příkladem záznamu. Aktualizace se zúčastní data ve formátu definovaném aplikací PP. Aplikace AK úzce spolupracuje s funkcemi PA v oblasti zápisu práv vkladem nebo záznamem, s aplikací PP v oblasti přebírání výsledků speciálních činností a aplikací PU při poskytování dat pro vytváření GP.

6.1.1.3 Aplikace PU — Poskytování údajů

Cílem aplikace PU je poskytování informací z dat KN. Zejména pak poskytování výpisu z KN a hromadných dat, údajů pro geometrické plány včetně rezervace čísel ZPMZ, nových parcellních čísel a čísel nových bodů PPBP, stanovení, přijetí a zaúčtování poplatků, evidence veškerých smluv sjednaných např. pro poskytování hromadných dat či založení

zákaznického účtu. Mezi funkce aplikace PU patří také zpoplatňování úkonů souvisejících s vedením KN (vklad, potvrzení GP, poskytnutí údajů). Aplikace PU úzce spolupracuje s aplikací PA v oblasti řízení o poskytování informací a řízení o poskytování podkladů pro geometrický plán. Z pohledu aplikace PA je aplikace PU jednou z podfunkcí. Některé funkce aplikace PU umožňují zobrazovat jak aktuální, tak historický i budoucí stav KN.

6.1.1.4 Aplikace DP — Dálkový přístup

Aplikace DP umožňuje přístup k datům ISKN ve formě sestav (ve formátu PDF nebo HTML obsahující textová data) zprostředkovaných uživateli pomocí WWW. Všechny sestavy jsou standardně označovány názvem a zkráceným názvem výstupní sestavy, místem, časem. Při vyhotovení sestavy pomocí dálkového přístupu je tato informace uvedena v sestavě (místo jména uživatele, který sestavu vytvořil). Dále každý z výstupů obsahuje datum a čas, ke kterému se platnost výstupní sestavy vztahuje. Taková informace je potřebná zejména v případě dotazu na historická data katastru nemovitostí, kdy datum platnosti je odlišné od data vytvoření výstupní sestavy.

Databázové tabulky, zachycující právní stav katastru, jsou z katastrálních úřadů a jejich detašovaných pracovišť prostřednictvím resortní sítě WAN replikovány na interní cluster centrální databáze v intervalu 2 hodin. Data na externí cluster v demilitarizované zóně, kam mají přístup zákazníci aplikace DP, se z interního clusteru replikují vždy 15 minut po skončení předchozího replikačního běhu. Celková doba replikace mezi interním a externím clusterem se pohybuje od několika minut do 45 minut ve vyjímečných případech silného zatížení obou databázových serverů a velkého objemu změn (replikovaných dat). Maximální doba dostupnosti informace pro externí uživatele se tedy pohybuje od 135 minut do 3 hodin. Přehledné schéma procesů na úrovni přistupujících uživatelů je zobrazeno na obrázku č. 6.3.

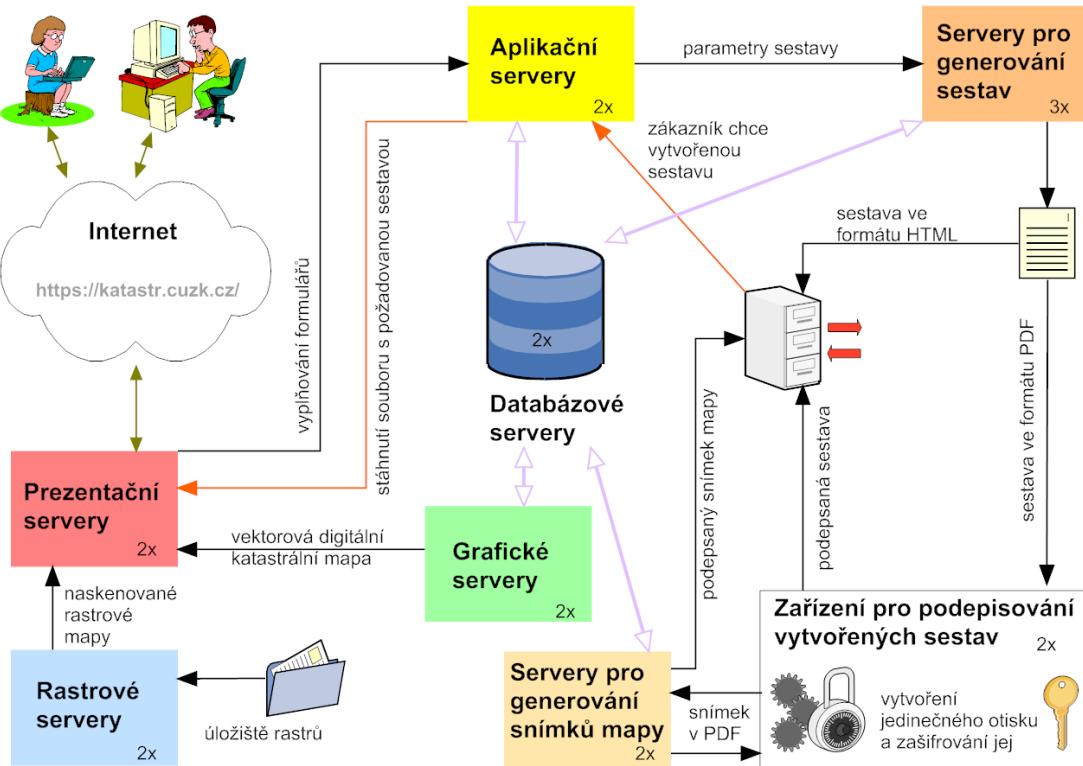
Možnost získat přístup k údajům ISKN prostřednictvím dálkového přístupu je podmíněna uzavřením smlouvy o veřejném přístupu. Smlouvu o dálkovém přístupu lze uzavřít v písemné podobě na kterémkoliv KÚ. Tato smlouva je pak vložena do ISKN na centrálním pracovišti ČÚZK. Na základě uzavřené smlouvy je pro uživatele dálkového přístupu zřízen zákaznický(-é) účet(-y), na který(-é) jsou účtovány prováděné operace. Při zpoplatnění služeb dálkového přístupu se při účtování na odpovídající zákaznický účet zapisuje i identifikace spuštěné funkce. Seznam funkcí u účtu je spolu s ceníkem funkcí využit při fakturaci za dálkový přístup.

6.1.1.5 Aplikace VP — Vzdálený přístup

Aplikace VP (vzdálený přístup) je administrativní částí aplikace DP. Obsahuje správu smluv definujících podmínky DP a správu zákaznických účtů.

6.1.1.6 Aplikace PP — Poskytování a přebírání podkladů

Aplikace PP zajišťuje řízení po obnově katastrálního operátu, včetně přípravy podkladů a převzetí výsledků, řízení o pozemkových úpravách, včetně přípravy podkladů a převzetí



Obrázek 6.3: Zjednodušené schéma Dálkového přístupu.

výsledků, řízení o revizi katastru, včetně přípravy podkladů, přípravu a přebírání podkladů pro zavádění BPEJ do KN, přípravu podkladů pro tvorbu a obnovu státního mapového díla a přebírání přehledové mapy. Tato aplikace neprovádí vlastní změnu dat, jejím úkolem je pouze připravit dodaná data do vhodného formátu pro aplikaci AK. Aplikace PP úzce spolupracuje zejména s aplikací PA a PU v oblasti řízení o obnově katastrálního operátu, řízení o pozemkových úpravách a řízení o revizi katastru. Z pohledu PP je aplikace PU jednou z podfunkcí. Funkce pro přípravu a přebírání podkladů pro zavádění BPEJ do katastru a funkce pro přípravu podkladů pro tvorbu a obnovu státního mapového díla hojně využívají funkce aplikace PU.

6.1.1.7 Aplikace ST — Sumarizace a statistiky

Aplikace ST zajišťuje vytváření statistik a summarizací z dat KN, podporuje přípravy pro řízení resortu, bilance a výkaznictví, podporuje přípravy ekonomických podkladů. Návrh aplikace ST vychází z principů ukládání přepracovaných statistických dat z evidence KN. Tato získaná data jsou podkladem pro tvorbu statistických výstupů. Aplikace ST přebírá podklady pro výkaznictví ze současného programu DOCHÁZKA, podporuje porovnání bilancí a výkaznictví pro vyhodnocení statistických údajů o plnění úkolů KÚ. Zároveň slouží pro summarizaci půdního fondu ČR a pro tvorbu dalších statistických přehledů o počtech objektů či subjektů v KN.

6.1.1.8 Aplikace BP — Bodová pole

Aplikace BP zabezpečuje funkce vedení bodových polí, přípravu a přebírání podkladů pro vedení bodových polí na lokální úrovni, přípravu a přebírání podkladů pro vedení základních bodových polí na centrální úrovni. Aplikace BP spolupracuje zejména s aplikacemi PU a AK. Data aplikace BP pro použití jinými aplikacemi jsou přístupná buď jiným dotazem, nebo prostřednictvím funkcí BP. Body podrobného bodového pole vznikají na lokální úrovni a jsou vedeny v ISKN. Poskytování veškerých údajů o bodovém poli jsou součástí aplikace PU.

6.1.1.9 Aplikace SC — Správa číselníků

Aplikace SC slouží ke správě číselníků, které jsou součástí ISKN. Některé číselníky jsou přebírány z externích zdrojů prostřednictvím aplikace EX. Správou číselníků se rozumí příjem aktualizačního zdroje dat nebo přímá aktualizace číselníků. Číselníky obsažené v ISKN můžeme z hlediska obsahu dělit do čtyř základních skupin:

- Interní číselníky aplikace – obsah těchto číselníků je plně ve správě zhotovitele ISKN a nejsou součástí aplikace SC,
- Resortní číselníky (lokální) – jejich obsah je v působnosti jednotlivých pracovišť resortu, obsahují údaje specifické vždy pro určité pracoviště,
- Resortní číselníky (centrální) – obsah těchto číselníků spadá do působnosti centrálního pracoviště, jejich obsah je definován na centrálním pracovišti a posléze distribuován na jednotlivá pracoviště resortu,
- Externí číselníky – vznikají v působnosti jiných orgánů státní správy případně i jiných organizací a jejich obsah je do ISKN přebíráno od jiných správců.

6.1.1.10 Aplikace EX — Vazby na externí systémy

Cílem aplikace EX je zpracování vstupů a příprava dat pro aktualizaci SC. Při komunikaci s Evidencí občanů zabezpečuje aplikace EX i výstup dat z ISKN pro ověření. Funkce aplikace EX zabezpečují převzetí dat v definovaném výměnném formátu z externího informačního systému do dočasných tabulek aplikace EX a následné zpracování dat do formátu odpovídajícího aplikaci SC.

6.1.1.11 Aplikace TO — Technicko-organizační zabezpečení provozu ISKN

Aplikace TO zabezpečuje veškeré činnosti týkající se technicko-organizačního provozu ISKN. Je určena pro podporu aplikací v ISKN, zejména však jako podpora správy sítě, správy databáze údržby technické infrastruktury, školení uživatelů, sledování provozu, podpora aplikací a v neposlední řadě jako podpora bezpečnostních mechanismů a další speciální funkce.

6.1.2 ISKN v číslech

Systém obsahuje informace¹ o :

- 13 027 katastrálních územích,
- 22 135 574 parcelách,
- 3 790 383 budovách,
- 1 368 660 bytových a nebytových jednotkách,
- 7 831 000 oprávněných osobách,
- 5 403 246 listech vlastnictví.

Za období od 1.1. do 8.8.2007 bylo dálkovým přístupem vydáno celkem 1 290 829 výstupů. Nejzádanějšími výstupy jsou výpisy z katastru nemovitostí – 946 984 (73,4%) a přehledy vlastnictví – 200 021 (15,5%).

6.2 Výměnný formát ISKN – NVF (nový výměnný formát)

Výměnný formát katastru nemovitostí (NVF) [8, 23] slouží k předávání dat mezi systémem ISKN a jinými systémy zpracování dat. Data je možné přebírat ve formě textové nebo ve formátu XML.

6.2.1 Struktura NVF

Export (datový soubor) se skládá ze tří částí:

- hlavičky (&H),
- datových bloků (&B, &D),
- koncového znaku (&K).

Každá věta hlavičky začíná uvozujícím znakem &H, věta popisující definici bloku &B, věta s vlastními daty &D. Každý z datových bloků v sobě obsahuje informaci o atrubutech a jejich formátu následovanou vlastními datovými řádky. Oddělovačem jednotlivých údajů na řádce je středník(;).

¹Stav ke dni 31.12.2006.
Zdroj: „Moderní katastr nemovitostí v srdci evropy“, ČÚZK, 2007

6.2.1.1 Hlavička souboru

Hlavička souboru obsahuje podrobnosti o obsahu a způsobu zadání podmínek pro export dat. Dále obsahuje verzi NVF, datum exportu, původ dat, kódování češtiny, seznam exportovaných datových skupin, časovou podmínu a omezující podmínky pro vytvoření exportu. Příklad 6.1 ukazuje hlavičku exportu dat v klasickém formátu (*.vfk) a příklad 6.2 ve formátu XML. Pro obě hlavičky byly použity stejné zadávací podmínky (omezující podmínkou bylo katastrální území Běleč a byla exportována data z datové skupiny NEMO).

Příklad 6.1: Hlavička NVF

```
&HVERZE;"3.2"
&HVYTVORENO;"11.09.2007 16:34:38"
&HPUVOD;"ISKN"
&HCODEPAGE;"WE8ISO8859P2"
&HSKUPINA;"NEMO"
&HJMENO;"Chromý Radek"
&HPLATNOST;"11.09.2007 16:32:41";"11.09.2007 16:32:41"
&HZMENY;0
&HNAVRHY;0
&HPOLYG;0
&HKATUZE;KOD N6;OBCE_KOD N6;NAZEV T48;PLATNOST_OD D;PLATNOST_DO D
&DKATUZE;601888;535010;"Běleč";"30.09.1991 00:00:00"; ""
&HOPSUB;ID N30;STAV_DAT N2;DATUM_VZNIKU D;DATUM_ZANIKU D;
PRIZNAK_KONTEXTU N1;RIZENI_ID_VZNIKU N30;RIZENI_ID_ZANIKU N30;
ID_JE_1_PARTNER_BSM N30;ID_JE_2_PARTNER_BSM N30;ID_ZDROJ N30;
OPSUB_TYPE T10;CHAROS_KOD N2;ICO N8;DOPLNEK_ICO N3;NAZEV T255;
NAZEV_U T255;RODNE_CISLO T10;TITUL_PRED_JMENEM T35;JMENO T24;
JMENO_U T24;PRIJMENI T35;PRIJMENI_U T35;TITUL_ZA_JMENEM T10;
CISLO_DOMOVNI N4;CISLO_ORIENTACNI T4;NAZEV_ULICE T32;CAST_OBCE T48;
OBEC T48;OKRES T32;STAT T23;PSC N5;MESTSKA_CAST T48;CP_CE N1
&HPAR;ID N30;STAV_DAT N2;DATUM_VZNIKU D;DATUM_ZANIKU D;
PRIZNAK_KONTEXTU N1;RIZENI_ID_VZNIKU N30; RIZENI_ID_ZANIKU N30;
PKN_ID N30;PAR_TYPE T10;KATUZE_KOD N6;KATUZE_KOD_PUV N6;
DRUH_CISLOVANI_PAR N1;KMENOVE_CISLO_PAR N5;ZDPAZE_KOD N1;
PODDELENI_CISLA_PAR N3; DIL_PARCELY N1;MAPLIS_KOD N30;ZPURVY_KOD N1;
DRUPOZ_KOD N2;ZPVYPA_KOD N4; TYP_PARCELY N1;VYMERA_PARCELY N9;
CENA_NEMOVITOSTI N14.2;DEFINICNI_BOD_PAR T100; TEL_ID N30;PAR_ID N30;
BUD_ID N30;IDENT_BUD T1
```

Příklad 6.2: Hlavička XML NVF

```
<?xml version="1.0" encoding="windows-1250"?>
<vfkiskn xmlns="urn:xmlns:iskn.cuzk.cz:vfkiskn">
<hlavicka verze="3.2" vytvorenou="11.09.2007 16:36:07"
vytvoril="Chromý Radek" zmeny="0" navrhy="0">
<platnost od="11.09.2007 16:32:41" do="11.09.2007 16:32:41" />
<datoveskupiny>
  <skupina>NEMO</skupina>
</datoveskupiny>
```

```
</hlavicka>
<vyberovapodminka>
  <katuze>
    <kod>601888</kod>
    <obce_kod>535010</obce_kod>
    <nazev>Běleč</nazev>
    <platnost_od>30.09.1991 00:00:00</platnost_od>
  </katuze>
</vyberovapodminka>
```

6.2.1.2 Datové bloky

Data jsou uspořádána do datových bloků, kterých je dohromady 68. Datový blok obsahuje vždy jeden uvozující rádek (&B), který obsahuje seznam atributů s jejich datovými typy a dále řádky (&D) obsahující vlastní data. Příklad 6.3 ukazuje uvozující rádek pro parcelu, další příklad 6.4 ukazuje naplnění vlastními daty. Na příkladu 6.5 jsou vidět ta samá data (přesněji jedna parcela), ale ve formátu XML.

Příklad 6.3: Uvozující rádek parcely

```
&BPAR; ID N30 ; STAV_DAT N2 ; DATUM_VZNIKU D ; DATUM_ZANIKU D ;
PRIZNAK_KONTEXTU N1 ; RIZENI_ID_VZNIKU N30 ; RIZENI_ID_ZANIKU N30 ;
PKN_ID N30 ; PAR_TYPE T10 ; KATUZE_KOD N6 ; KATUZE_KOD_PUV N6 ;
DRUH_CISLOVANI_PAR N1 ; KMENOVE_CISLO_PAR N5 ; ZDPAZE_KOD N1 ;
PODDELENI_CISLA_PAR N3 ; DIL_PARCELY N1 ; MAPLIS_KOD N30 ; ZPURVY_KOD N1 ;
DRUPOZ_KOD N2 ; ZPVYPKA_KOD N4 ; TYP_PARCELY N1 ; VYMERA_PARCELY N9 ;
CENA_NEMOVITOSTI N14.2 ; DEFINICNI_BOD_PAR T100 ; TEL_ID N30 ; PAR_ID N30 ;
BUD_ID N30 ; IDENT_BUD T1
```

Příklad 6.4: Data o parcele

```
&DPAR ; 734610203 ; 0 ; "25.08.2006 10:06:06" ; " " ; 3 ; 2447011203 ; ; ; "PKN" ;
601888 ; ; 1 ; 1 ; ; 1 ; ; 17380 ; 0 ; 13 ; ; 8695 ; ; " " ; 459011203 ; ; 226910203 ; "a"
```

Příklad 6.5: Datový blok PAR v XML NVF

```
<par>
  <id>734610203</id>
  <stav_dat>0</stav_dat>
  <datum_vzniku>25.08.2006 10:06:06</datum_vzniku>
  <priznak_kontextu>3</priznak_kontextu>
  <rizeni_id_vzniku>2447011203</rizeni_id_vzniku>
  <par_type>PKN</par_type>
  <katuze_kod>601888</katuze_kod>
  <druh_cislovani_par>1</druh_cislovani_par>
  <kmeneove_cislo_par>1</kmeneove_cislo_par>
  <poddeleni_cisla_par>1</poddeleni_cisla_par>
  <maplis_kod>17380</maplis_kod>
```

```

<zpurvy_kod>0</zpurvy_kod>
<drupoz_kod>13</drupoz_kod>
<vymera_parcely>8695</vymera_parcely>
<tel_id>459011203</tel_id>
<bud_id>226910203</bud_id>
<ident_bud>a</ident_bud>
</par>

```

Datové bloky jsou seskupeny podle významu do datových skupin, které tvoří jeden nedělitelný exportovaný logický celek. Datové bloky jsou rozděleny do celkem dvanácti datových skupin, viz tabulka 6.1. Přesné rozdělení datových bloků do skupin lze najít v příloze A.

Kód	Název	Obsah
NEMO	Nemovitosti	Informace o parcelách, budovách, jejich využití, způsoby ochrany a územní identifikaci.
JEDN	Jednotky	Informace o jednotkách, jejich typech a způsobu využití.
BDPA	Bonitní díly parcel	Bonitní díly parcel.
VLST	Vlastnictví	Oprávněné subjekty a jejich vlastnictví.
JPVZ	Jiné právní vztahy	Jiné právní vztahy a číselník typů právních vztahů.
RIZE	Řízení	Informace o řízeních, listiny a přiřazení listin k nemovitostem.
PKMP	Prvky katastrální mapy	Prvky DKM.
BPEJ	BPEJ	Grafické znázornění hranic BPEJ.
GPL	Geometrický plán	Hlavičky geometrických plánů a ZPMZ.
REZE	Rezervovaná čísla	Rezervovaná čísla parcel a rezervovaná čísla PBPP.
DEBO	Definiční body	Definiční body parcel a budov.
ADRM	Adresní místa	Adresní místa budov.

Tabulka 6.1: Datové skupiny (Zdroj [8]).

6.2.1.3 Koncový znak

Koncový znak &K uzavírá datový soubor a má pouze kontrolní úlohu.

6.2.2 Typy exportů

Pro export dat NVF existují dva základní typy exportů:

- stavový export - export obsahuje vždy pouze platná data k určitému datu,
- změnový export - export obsahuje veškeré změny dat za určité časové období.

K posouzení aktuálnosti jednotlivých záznamů slouží atributy „stav dat“ a „příznak kontextu“, případně „datum vzniku“ a „datum zániku“. Tyto atributy obsahují všechny databázové tabulky ISKN podléhající principu historizace. Principem historizace rozumíme stav, kdy se k záznamům udržuje přítomnost i minulost, jedná se například o tabulky evi- dující informace o oprávněných subjektech (dále jen OS), parcelách, budovách, jednotkách, jiných právních vztazích, přiřazených listinách a listech vlastnictví. Tabulky datové skupiny RIZE (výjimkou je datový blok RL) nepodléhají principu historizace, přesto mohou být součástí změnového exportu. Řízení se exportují pouze v případě, kdy datum zplatnění nebo datum uzavření řízení spadá do exportovaného období. Tabulky skupin GMPL a REZE také nepodléhají principu historizace, součástí změnového exportu být ovšem nemohou. Z tabulky 6.2 je patrné, jakých hodnot mohou nabývat atributy „stav dat“ a „příznak kontextu“.

Operace	Stav dat	Příznak kontextu	Význam
UPDATE	-1	1	objekt byl změněn, historizovat platný stav
	-1	3	objekt vznikl a později byl změněn
	0	3	objekt byl změněn, aktuální verze
DELETE	3	1	objekt byl zrušen
INSERT	0	3	objekt vznikl
LOCK	0	2	objekt nebyl změněn

Tabulka 6.2: Hodnoty atributů stav dat a příznak kontextu (Zdroj [8]).

Příklad 6.6 ukazuje změny OS tak, jak jsou exportované pomocí změnových vět NVF vytvořených za období 1.1.2005 až 1.1.2006. První řádek se stavem dat = -1 a příznakem kontextu = 1, nám říká, že je potřeba záznam OS s datumem vzniku 20.7.2004 z přítomnosti posunout do minulosti. Druhý řádek se stavem dat -1 a příznakem kontextu 3 říká: objekt vznikl v exportovaném období a později byl změněn, což znamená vložení řádku a rovnou posun do minulosti. Stav dat = 0 a příznak kontextu = 3 nás upozorňuje, že se jedná o aktuální verzi záznamu, tedy vložení řádku do přítomnosti.

Příklad 6.6: Změnové věty

```
&DOPSUB ;2776293101;-1;"20.07.2004_14:44:21";"07.05.2005_00:03:59";1;...
&DOPSUB ;2776293101;-1;"07.05.2005_00:04:00";"08.11.2005_11:01:59";3;...
&DOPSUB ;2776293101;0;"08.11.2005_11:02:00";"";3;...
```

Atributy „stav dat“ a „příznak kontextu“ u stavových dat nemají z pohledu zpracování dat prakticky žádný význam, protože stavový export obsahuje pouze platná data k určitému datu. Pokud je stavový export prováděn k datu z minulosti, mohou mít některé

věty exportu stav dat roven -1, což znamená, že je záznam vzhledem k systémovému datu v minulosti, ovšem vzhledem k datu platnosti exportu jsou všechny záznamy vždy v přítomnosti. Změnová data mají při pravidelných odběrech mnohonásobně menší obsah než stavová a umožní zákazníkovi vést kompletní historii tabulek předmětu katastru nemovitostí. Jejich zpracování je ovšem náročnější než zpracování stavových dat. Proto je vždy nutné v prvotní analýze návrhu systému pečlivě zvážit, k jakému účelu budou data ISKN přenesená, pomocí NVF, využívána.

Kapitola 7

Webové služby pro přístup k datům KN

Cílem je navržení skupiny webových služeb pro snazší programový přístup k datům katastru nemovitostí, dále jen KN. Tato skupina služeb vytvoří plnohodnotný alternativní programový přístup k aplikaci Nahlížení do KN (viz kapitola 7.1.2). Tyto služby budou sloužit nejen pro podporu externích aplikací, ale i pro podporu interních – resortních aplikací. Použití služeb bude podmíněno bezplatnou registrací.

7.1 Současný stav

Získat data z katastru nemovitostí lze několika základními způsoby. Důležitým faktorem pro zvolení vhodného přístupu k datům KN je další využití takto získaných dat. Základní způsoby pro získání dat z KN lze rozdělit takto:

- fyzická návštěva katastrálního úřadu, či České pošty (pouze výpis z KN a kopie DKM, které Česká pošta poskytuje pomocí aplikace Dálkový přístup),
- internetový přístup k datům KN.

Na katastrálních úřadech lze získat výstupy jak ve formě papírových výstupů, tak ve formě elektronických výstupů. Nejzajímavější z elektronických výstupů je vydávání dat ve formátu VFK (viz kapitola 6.2). VFK slouží k výměně dat mezi různými subjekty, ale hlavně k možnosti automatizovaného zpracování a vyhodnocení jakéhokoliv objemu dat. Tento formát je obecně znám a na jeho zpracování a využívání existuje mnoho různých aplikací. Jistou nevhodou VFK je jeho získání pouze na katastrálních pracovištích. Praktiky již při vyzvednutí exportu jsou data „zastaralá“. V dnešní době je však nutností mít data co nejaktuálnější – nejlépe on-line. Nejaktuálnější data jsou k dipozici pomocí internetových aplikací.

Internetový přístup zajišťují dvě základní aplikace:

- Dálkový přístup (DP),
- Nahlížení do KN.

7.1.1 Dálkový přístup

Aplikace DP (dálkový přístup) umožňuje poskytování údajů z ISKN prostřednictvím dálkového přístupu k centrální databázi ISKN. Dálkovým přístupem se rozumí on-line přístup smluvních partnerů nebo pracovníků katastrálních úřadů k datům centrální databáze ISKN. Výstupy z DP umožňují přístup k platným datům KN s průměrným zpožděním v řádu desítek minut (max. do 120 min.).

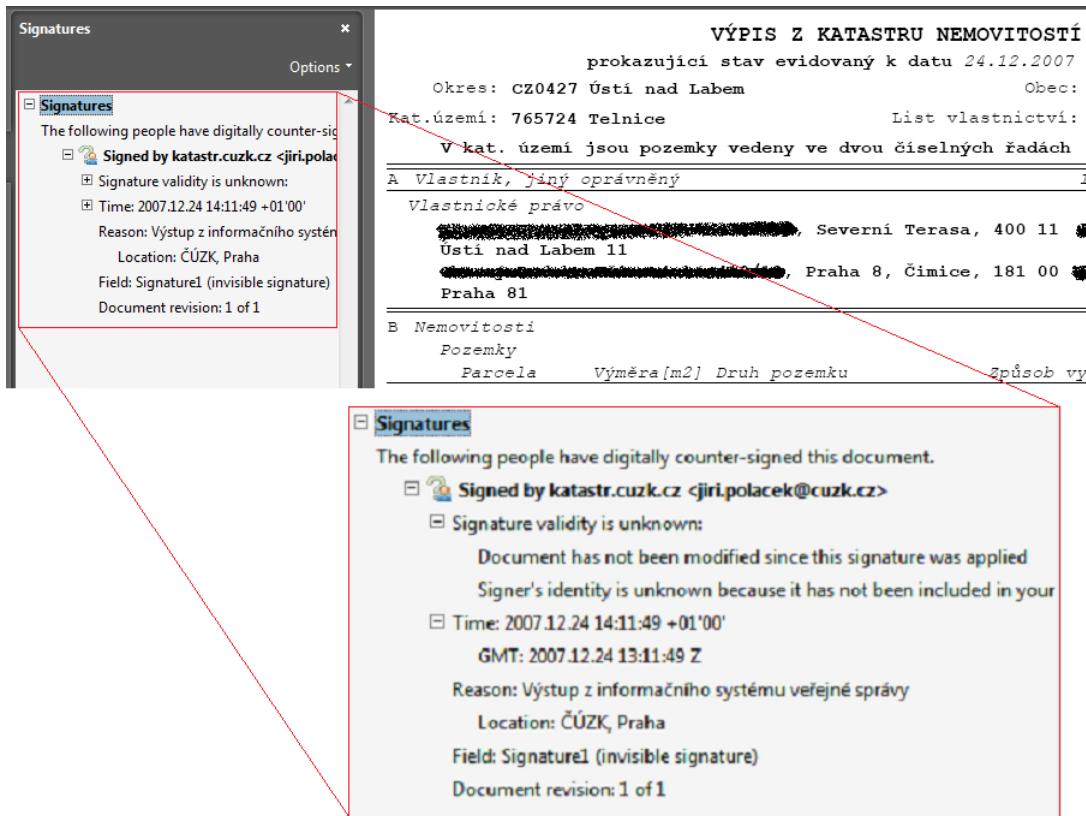
Podrobnější informace o aplikaci lze nalézt v předcházející kapitole 6.1.1.4. Zde bude aplikace podrobněji popsána z hlediska možných přístupů – „oprávnění“ k funkcím a výstupním sestavám DP. Uživatelská oprávnění pro spouštění funkcí DP jsou definována pomocí zákaznických účtů¹

Základní přehled zákaznických účtů aplikace DP:

- interní účet – účet je určen pro zaměstnance rezortu, jsou přístupné všechny funkce a výstupy DP.
- Běžný účet – tento účet disponuje přístupem ke všem funkcím a výstupům DP, účtování je prováděno dle platného ceníku. Tento typ účtu může být založen komukoliv.
- Účet pro bezúplatný dálkový přístup – tento účet je určen pro výkon působnosti orgánů samosprávných územních celků (např. obce, města, kraje, stavební správa, atd.). Nejsou k dispozici všechny výstupy. Zájemce o tento typ účtu se musí nejdříve zaevidovat.
- Účet pro účely vydávání ověřených výstupů z KN – tento typ účtu umožňuje pouze výstupy, které jsou označeny elektronickou značkou (viz obrázek 7.1) mající charakter veřejné elektronické listiny (Výpis z KN a kopie DKM). Účet může být založen pouze subjektům, které mají dle zákona oprávnění ověřovat výstupy z informačních systémů státní správy (např. Notářská komora, Hospodářská komora, Česká pošta, atd.).

Výstupy jsou k dispozici ve formátu pdf nebo html. Výstupy jsou identické s výstupy, které lze získat na katastrálních úřadech. Jak již bylo zmíněno, sestava Výpis KN a Kopie DKM jsou opatřeny elektronickou značkou a mají charakter elektronické veřejné listiny. Vývoj počtu uživatelů, využívajících aplikaci DP, je zobrazen na následujícím obrázku 7.2.

¹ Zákaznické účty a uživatelská oprávnění
URL: <http://www.cuzk.cz/Dokument.aspx?PRARESKOD=998&MENUID=10007&AKCE=DOC:10-ZRIZENI_UCTU_DP>



Obrázek 7.1: Výpis z KN podepsán elektronickou značkou.

7.1.1.1 Webové služby dálkového přístupu

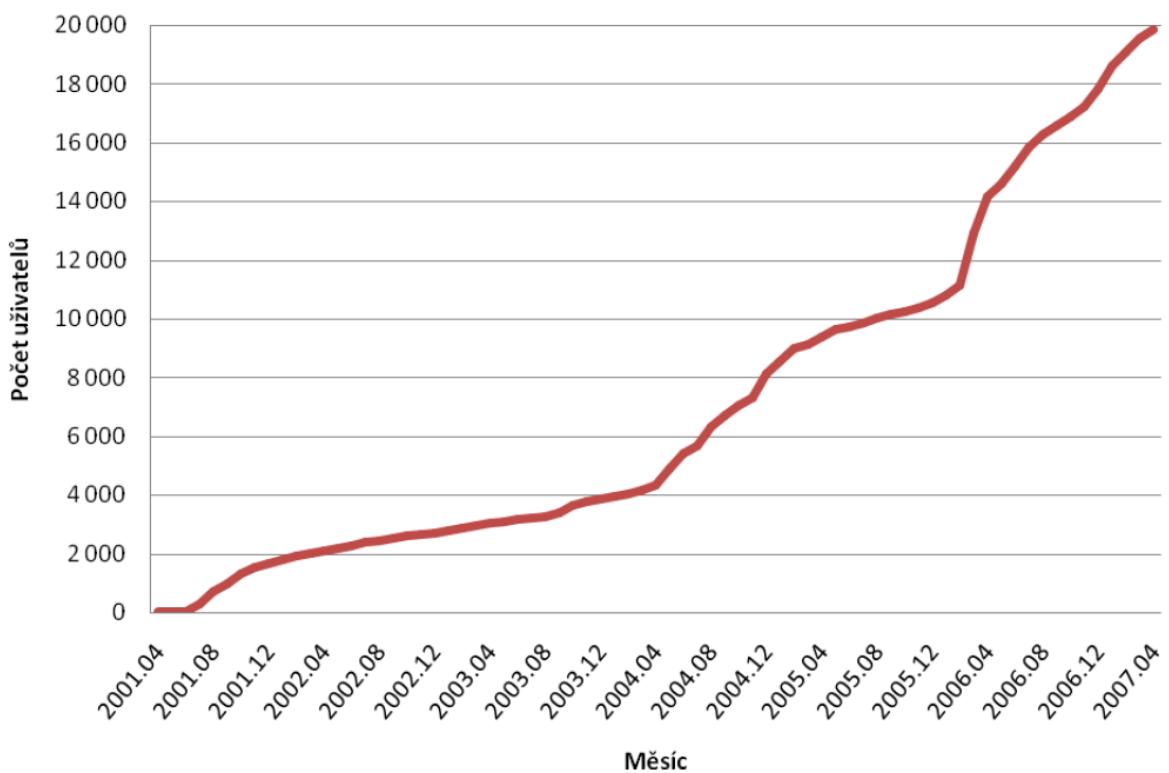
Webové služby, dále jen WSDP, jsou součástí aplikace DP od poloviny roku 2007. WSDP jsou placené a jejich výstupem jsou prozatím sestavy ve formátu pdf (sestavy jsou stejné jako v DP). WSDP mohou používat všechny skupiny uživatelů DP, pouze je potřeba mít založen samostatný účet, který není možný kombinovat se zákaznickým účtem pro další služby (např. v DP).

Služby, které poskytuje aplikace WSDP jsou rozděleny do 4 skupin:

- číselníky – skupina obsahuje funkci na vyhledávání hodnot v číselníku.
- Sestavy – skupina obsahuje funkce pro práci se sestavami.
- Vyhledávání – skupina obsahuje funkce pro vyhledávání oprávněných subjektů, parcel, budov a jednotek.
- Správa účtu – skupina obsahuje funkci na změnu hesla uživatele.

Skupiny obsahují následující služby – funkce:

- číselníky



Obrázek 7.2: Počet uživatelů DP (Zdroj [39]).

- *vratCiselnik* – funkce vrací obsah číselníků.
- Sestavy
 - *seznamSestav* – funkce vrací seznam sestav uživatele,
 - *vratSestav* – funkce vrací vybranou sestavu,
 - *smazSestavu* – funkce smaže sestavu,
 - *generujLV* – funkce vrací sestavu „Výpis z KN“ (parametr – jednoznačná identifikace Listu vlastnictví),
 - *generujLVZjednodusene* – funkce vrací sestavu „Výpis z KN“ (parametry – kód katastrálního území, číslo Listu vlastnictví),
 - *generujLVPresOS* – funkce vrací sestavu „Výpis z KN“ (parametry – kód katastrálního území, jednoznačná identifikace opávněného subjektu),
 - *generujLVPresObjekty* – funkce vrací sestavu „Výpis z KN“ (parametry – typ seznamu objektů (např. seznam parcel), seznam jednoznačných identifikací jednoho typu objektů),

- *generujInfoOParcelach* – funkce vrací sestavu „Informace o parcelách“ (parametry – jednoznačná identifikace parcely nebo kombinace ostatních parametrů o parcele),
- *generujInfoOStavbach* – funkce vrací sestavu „Informace o stavbách“ (parametry – jednoznačná identifikace stavby nebo kombinace ostatních parametrů o stavbě),
- *generujInfoOJednotkach* – funkce vrací sestavu „Informace o jednotkách“ (parametry – jednoznačná identifikace jednotky nebo kombinace ostatních parametrů o jednotce),
- *generujPrehledVlastnictvi* – funkce vrací sestavu „Přehled vlastnictví“ (parametr – jednoznačná identifikace opávněného subjektu),
- *generujEvidenciPravProOsobu* – funkce vrací sestavu „Evidence práv pro osobu“ (parametry – název oprávněného subjektu, rodné číslo/datum narození/IČO oprávněného subjektu),
- *vypisUctu* – funkce vrací sestavu „Výpis stavu účtu“ (parametry – časový rozsah).

- Vyhledávání

- *najdiOS* – funkce vyhledává oprávněný subjekt na základě zadaných parametrů, funkce vrací XML s popisem nalezeného oprávněného subjektu,
- *najdiParcelu* – funkce vyhledává parcelu na základě zadaných parametrů, funkce vrací XML se seznamem nalezených parcel,
- *najdiStavbu* – funkce vyhledává stavbu na základě zadaných parametrů, funkce vrací XML se seznamem nalezených staveb,
- *najdiJednotku* – funkce vyhledává jednotku na základě zadaných parametrů, funkce vrací XML se seznamem nalezených jednotek.

- Správa účtu

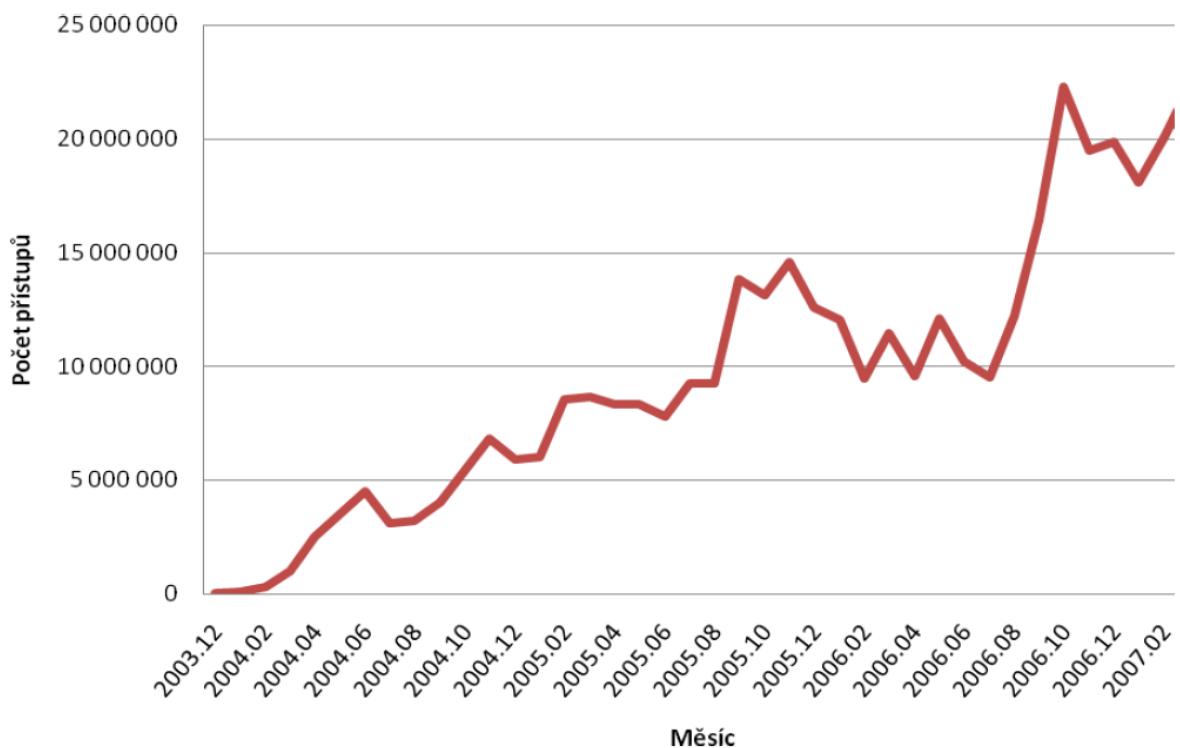
- *zmenHeslo* – funkce umožňuje změnu hesla uživatele.

Základní způsob vracení sestav je asynchronní (uživatel získá pouze jednoznačnou identifikaci sestavy a posléze pomocí funkce *seznamSestav* kontroluje, jestli je již sestava hotová a připavená k vyzvednutí). Možnost vrácení sestavy on-line je možné jen za předpokladu splnění specifických podmínek. Výsledná sestava ve formátu pdf je přiložena k odpovědi jako MIME příloha.

Tyto webové služby jsou prvním krokem k bezproblémové interakci s jinými informačními systémy. Jistou nevýhodou těchto služeb jsou výstupy pouze ve formátu pdf. Takto získaný výstup (data v něm obsažená) se bude jen velmi obtížně dále automatizovaně zpracovávat a využívat.

7.1.2 Nahlížení do KN

Aplikace „Nahlížení do KN“, dále jen Nahlížení, je volně přístupná aplikace, která nevyžaduje žádnou registraci a je zcela zdarma. Aplikace poskytuje některé vybrané údaje z KN, zejména týkající se vlastnictví parcel, budov a jednotek. Dále je možné získat informace o průběhu řízení na katastrálních pracovištích pro účely zápisu vlastnických a jiných práv evidovaných v KN. Výstup dat je prezentován pomocí jazyka HTML, proto automatizované zpracování není jednoduchou záležitostí, avšak je potřeba říci, že aplikace nebyla pro toto zpracování navržena. Přesto se mnoho uživatelů o toto zpracování, s větším či menším úspěchem, pokouší. Rozdíl mezi aplikacemi Dálkový přístup a Nahlížení je v rozsahu poskytovaných dat a služeb. Oblíbenost aplikace dokazuje následující obrázek 7.3 znázorňující vývoj počtu přístupů.



Obrázek 7.3: Vývoj počtu přístupů k aplikaci Nahlížení do KN (Zdroj [39]).

7.2 Analýza

Rozsah služeb, které mají nové webové služby poskytovat, odpovídá rozsahu služeb, které poskytuje aplikace Nahlížení. Jedná se tedy o tyto služby:

- Informace o parcele,

- Informace o budově,
- Informace o jednotce,
- Informace o řízení,
- Nalezení nejbližšího definičního bodu parcely/budovy²,

Zde je nutné připomenout, že tyto služby neposkytují jen elementární informace o nemovitostech či řízení, ale také zobrazují informace, které mají přímou souvislost s těmito objekty. Služby, které poskytují informace o nemovitostech, poskytují také např. informace o vlastnictví, o využití nemovitosti, o katastrálním území, kde nemovitost leží, atd. To má zásadní dopad na celkový návrh webových služeb.

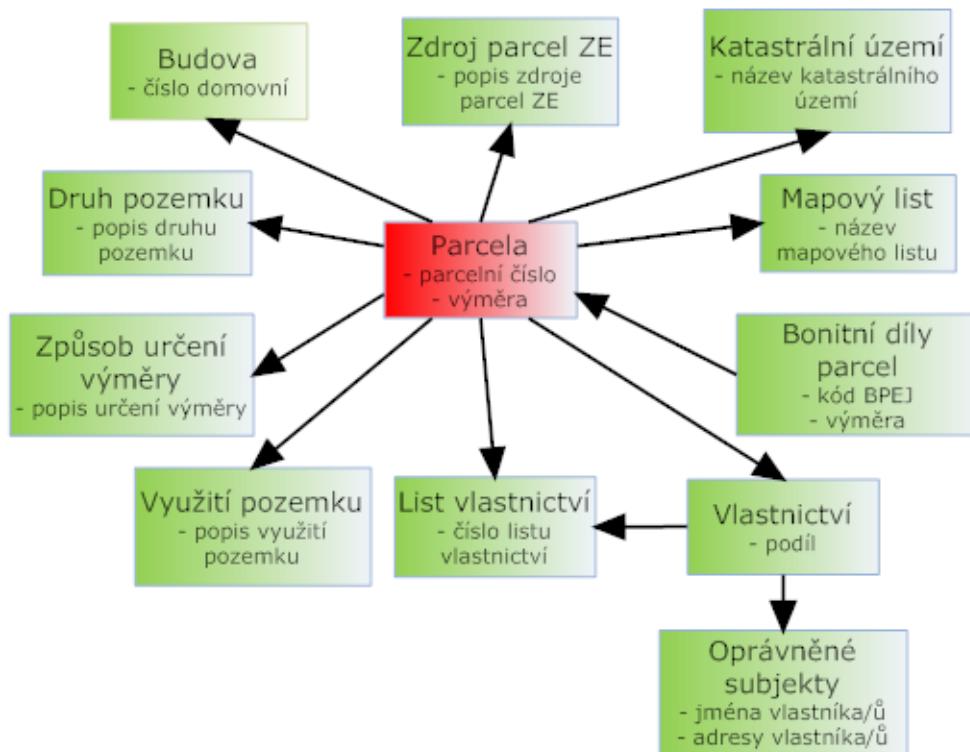
Pro názornou ilustraci o všech informacích, které poskytuje např. služba Informace o parcele, je zobrazen následující podrobný výpis:

- Informace o parcele
 - parcellní číslo,
 - výměra,
 - budova,
 - katastrální území,
 - typ parcely,
 - mapový list,
 - určení výměry,
 - využití pozemku,
 - druh pozemku,
 - číslo listu vlastnictví,
 - jména vlastníka/ů,
 - adresa/y vlastníka/ů,
 - podíl/y vlastníka/ů,
 - BPEJ,
 - výměra.

Na následujícím schématu 7.4 je tento výpis zobrazen po jednotlivých objektech tak, jak jsou navrženy v systému ISKN, včetně jejich vzájemných vazeb.

Na základě předchozích zjištění lze k návrhu webových služeb přistupovat dvěma různými způsoby:

²Tyto služby budou k dispozici v nové verzi aplikace Nahlížení



Obrázek 7.4: Objekt Parcels a její vybrané vazby v systému ISKN.

Varianta A: pro každý objekt, který je nutný pro získání informací, bude vytvořena webová služba, která bude poskytovat informace o tomto objektu včetně vazeb, „odkazů“, na ostatní objekty. Tyto „odkazy“ budou představovat vstupní parametry pro volání webových služeb dalších objektů. Tímto způsobem vznikne několik desítek webových služeb, jejichž vzájemnou kombinací (např. pro získání informace o parcele by webové služby byly vytvořeny podle obrázku 7.4) bude možné získat požadované informace.

Varianta B: druhou variantou je vytvoření webových služeb, které budou informace z různých objektů „skládat“ a poskytovat výstupní informace v obsahu, který uživatelé znají z aplikace Nahlížení. To znamená, že např. informace o parcele bude tvořit jedna webová služba, která bude poskytovat kompletní informace o parcele včetně dalších informací, zjištěných z vazeb na ostatní objekty.

V tabulce 7.1 jsou shrnutý výhody a nevýhody obou variant. Zejména z důvodu získání všech dat na jedno „zavolání“ webové služby, byla zvolena pro další řešení webových služeb **Varianta B**.

	Varianta A	Varianta B
Výhody	Uživatel získá pouze data, která požaduje.	Na jedno „zavolání“ webové služby získá uživatel všechna potřebná data.
	Větší variabilnost pro získání dat.	Snadnější integrovatelnost do vlastních systémů.
Nevýhody	Nutná velmi dobrá znalost vazeb mezi webovými službami (resp. velmi dobrá znalost systému ISKN).	Uživatel získá více dat, která nemusí potřebovat.
	Velké množství služeb – nepřehlednost.	
	Větší zatížení hardwaru pro fungování služeb.	
	Důkladnější zabezpečení služeb.	

Tabulka 7.1: Shrnutí výhod a nevýhod jednotlivých variant pro řešení webových služeb.

7.2.1 Uživatelské scénáře

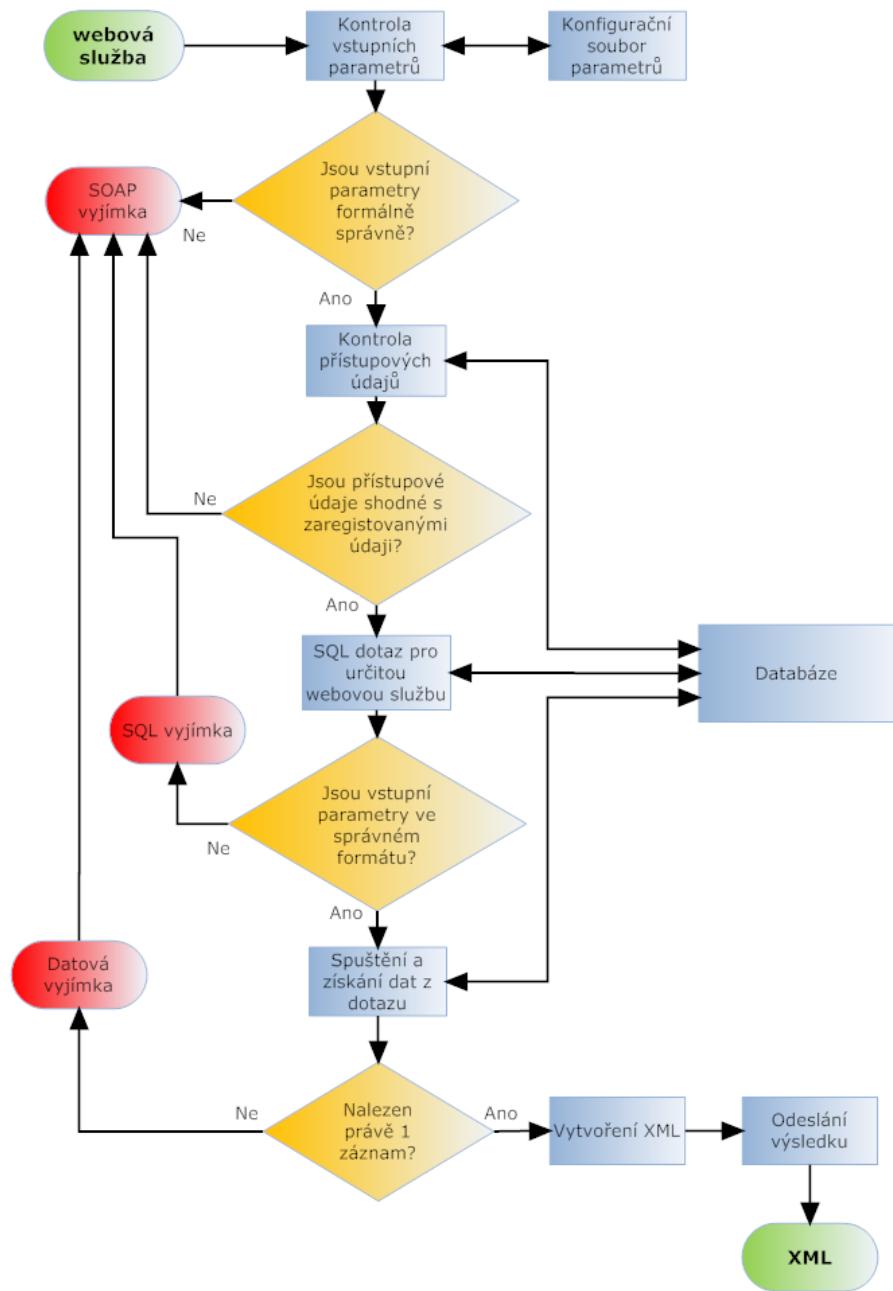
Cílem uživatelských scénářů je popis základní funkcionality navrhovaného systému. Scénáře vychází z definovaných požadavků na budoucí systém a umožňují popsat proces, jak bude systém využíván uživateli či dalšími systémy. Všechny akce v systému jsou vyvolávány Aktéry. Aktér je role, ve které může vystupovat jak člověk tak systém. Účastníci (aktéři) navrhovaného systému webových služeb jsou následující:

- Anonym – budoucí uživatel webových služeb.
- Uživatel – oprávněný uživatel pro využívání webových služeb.
- Administrátor – je uživatel s přidělenými právy pro administraci uživatelských účtů.
- Systém – zastupuje systém jako protějšek uživatelů.

V této kapitole bude uveden pouze základní princip fungování systému webových služeb. Tento princip vysvětluje chování služeb, které budou vracet informace o určitém objektu. Funkce vyhledávací a číselníkové se budou trochu lišit (viz příloha B). Celkový seznam uživatelských scénářů, které definují chování tohoto systému, je uveden v příloze B této práce.

Základní princip (viz obrázek 7.5) lze popsat takto :

1. Uživatel si zvolí webovou službu.
2. Uživatel zadá požadované vstupní parametry.
3. Systém zkонтroluje formální správnost vstupních parametrů.



Obrázek 7.5: Fungování webových služeb.

- (a) Pokud jsou parametry formálně správně, systém pokračuje bodem 4.
 - (b) Pokud parametry nejsou formálně správně, systém oznámí uživateli chybu v zadání parametrů.
4. Systém zkонтroluje platnost přístupových údajů (viz 7.2.4) s údaji, které jsou zaregistrovány v databázi.
- (a) Pokud jsou přístupové údaje v pořádku, systém pokračuje bodem 5.

- (b) Pokud nejsou přístupové údaje v pořádku, systém oznámí uživateli chybu v platnosti přístupových údajů.
5. Systém vyzvedne odpovídající SQL dotaz z databáze a vyplní požadované proměnné, získanými vstupními parametry.
 - (a) Jestliže jsou parametry vloženy do dotazu v pořádku, systém pokračuje bodem 6.
 - (b) Jestliže se nepodaří vstupní údaje vložit do proměnných v SQL dotazu, systém oznámí uživateli chybu ve formátu vstupních údajů.
 6. Systém spustí výsledný dotaz.
 7. Systém získá data.
 - (a) Jestliže je nalezen právě jeden záznam, systém pokračuje bodem 8.
 - (b) Jestliže není nalezen žádný záznam, systém oznámí uživateli neexistenci dat pro zadané vstupní parametry.
 - (c) Jestliže je nalezeno více záznamů, systém oznámí uživateli, že bylo nalezeno více záznamů, aby zpřesnil své zadání.
 8. Systém ze získaných dat vytvoří XML.
 9. Systém odešle výsledné XML.
 10. Uživatel obdrží data v XML (viz 7.2.3.2), která vyhovují zadaným vstupním parametrům.

7.2.2 Návrh webových služeb

Z výsledků úvodní analýzy na rozsah poskytovaných služeb a na základě definovaných uživatelských scénářů lze navrhnout následující sadu webových služeb.

- Získání informací o parcele
 - *infoOParceleID* – vstupním parametrem služby bude jednoznačný identifikátor parcely (ID) v systému ISKN. Tuto službu budou využívat zejména uživatelé, kteří již data ISKN využívají a znají tyto jednoznačné identifikátory. Použití tohoto parametru jim výrazně usnadní práci se získáním požadovaných dat.
 - *infoOParcele* – tato služba bude mít více parametrů a to takových, aby z nich bylo možné jednoznačně identifikovat určitou (jedinou možnou) parcelu.
 - *vyhledaniParcel* – jestliže uživatel nebude přesně znát požadované vstupní parametry u předchozích služeb, může pomocí této služby a jejích parametrů, hledanou parcelu vyhledat.
- Získání informací o budově

- *infoOBudoveID* – dtto jako u služby *infoOParceleID*, jen se jedná o budovu.
- *infoOBudove* – dtto jako u služby *infoOParcele*, jen se jedná o budovu.
- *vyhledaniBudov* – dtto jako u služby *vyhledaniParcel*, jen se jedná o budovu.
- *vyhledaniBudovParID* – tato služba umožní vyhledávat budovy, pomocí jednoznačného identifikátoru parcely (ID), které na této parcele leží.
- Získání informací o jednotce
 - *infoOJednotceID* – dtto jako u služeb *infoOParceleID* a *infoOBudoveID*, jen se jedná o jednotku.
 - *infoOJednotce* – dtto jako u služeb *infoOParcele* a *infoOBudove*, jen se jedná o jednotku.
 - *vyhledaniJednotek* – dtto jako u služeb *vyhledaniParcel* a *vyhledaniBudov*, jen se jedná o jednotku.
 - *vyhledaniJednotekBudID* – služba umožní vyhledat jednotky, které se nalézají v určité budově, opět za pomocí jednoznačného identifikátoru budovy (ID).
- Získání informací o řízení
 - *infoORizeniID* – dtto jako u služeb *infoOParceleID*, *infoOBudoveID* a *infoOJednotceID*, jen se jedná o řízení.
 - *infoORizeni* – dtto jako u služeb *infoOParcele*, *infoOBudove* a *infoOJednotce*, jen se jedná o řízení.
- Nalezení nejbližšího definičního bodu
 - *nejblizsiDefBodPar* – služba bude vracet souřadnice nejbližšího definičního bodu parcely,
 - *nejblizsiDefBodBud* – služba bude vracet souřadnice nejbližšího definičního bodu budovy.
- Získání informací o číselnících
 - *vratSeznamCiselniku* – služba bude vracet seznam dostupných číselníků,
 - *vratCiselnik* – služba bude vracet číselníkové hodnoty vybraného číselníku. Pomocí vstupních parametrů bude možno výběr těchto hodnot ovlivnit.

7.2.3 Výstupní formát

Zvolení jazyka XML jako výstupního formátu byla jasná volba. XML má širokou podporu a dá se poměrně snadno a efektivně zpracovat ve většině programovacích jazyků. Navíc je platformově nezávislé a tak není problém ani s různými operačními systémy.

Dalším faktorem pro zvolení vhodného výstupního formátu byla myšlenka, nevytvářet úplně nový formát, ale pokusit se využít již nějaký existující výstupní formát, který je dostatečně dobře znám a popsán.

Na základě předchozích požadavků je možno říci, že jediný formát, který by vyhovoval, je Výměnný formát katastru nemovitostí – NVF (viz kapitola 6.2). Avšak je potřeba říci, že NVF je velmi komplexní a rozsáhlý formát, který je primárně určen pro export velkých objemů dat.

Pro otestování možností výstupního fomátu v NVF, byl pomocí APV³ vygenerován XML NVF, který měl pouze jediný vstupní parametr – jednoznačnou identifikaci parcely. Tento vygenerový NVF obsahoval více jak 95% informací, které jsou nad rámec informací, které poskytuje aplikace Nahlížení, službou Informace o parcele. Při změně vstupních parameterů (např. při zadání jednoznačné identifikace budovy či jednotky) byly výsledky stejné. Po důkladném prostudování takto získaných exportů XML NVF bylo shledáno, že kompletní formát XML NVF nelze s výhodou použít jako výstupní formát pro webové služby, zejména z důvodu velké složitosti, komplexnosti a zbytečně (pro běžného uživatele) velkého objemu vzájemně provázaných dat.

7.2.3.1 Zajímavé rysy XML NVF

Při analýzách exportů NVF bylo zjištěno několik velmi zajímavých rysů XML NVF. Následující výčet několika zajímavých rysů není rozhodně výčtem úplným a ani to nebylo cílem.

- 1. zajímavý rys:** v formátu XML NVF není zavedena žádná hierarchie elementů. Formát je takřka úplně „plochý“. Existuje zde kořenový element <data>, který obsahuje všechny další elementy. Další zanoření je už jen u jednotlivých datových bloků XML NVF. To znamená, že všechny datové bloky jsou na stejně úrovni.
- 2. zajímavý rys:** ve velmi omezené míře (vlastně pouze v hlavičce) jsou využity možnosti atributů. Jak je patrné na příkladě 7.1, použití jen základních atributů u hlavních elementů, formát zpřehlední a zefektivní a také umožní snazší automatizované zpracování.
- 3. zajímavý rys:** vlastní data jsou promíchána s číselníkovými daty. Je to dáno neexistující hierarchií elementů. Že se jedná o číselník, lze poznat pouze podle jména ohraňujícího elementu.
- 4. zajímavý rys:** k formátu XML NVF neexistuje popis formátu – XML schéma 3.3.1. To vede k zajímavé situaci, kdy NVF v textové formě obsahuje datové typy elementů (viz příklad 6.1), formát XML však nikoliv (viz příklad 6.2).

³APV – aplikační programové vybavení systému ISKN

Příklad 7.1: Příklad XML NVF, v levé části je umístěn původní formát a v pravé části formát s použitím atributů.

```

<?xml version="1.0" encoding="windows-1250"?>
<vfkiskn xmlns="urn:xmlns:iskn.cuzk.cz:vfkiskn">
<hlavicka verze="3.2" vytvoreno="11.09.2007 16:36:07"
vytvoril="Chromý Radek" zmeny="0" navrhy="0">
<platnost od="11.09.2007 16:32:41" do="11.09.2007 16:32:41" />
...
<data>
<par>                                         <par id="235738739" >
  <id>235738739</id>                         ...
  ...
</par>                                         </par>
<bud>                                         <bud id="21842739" >
  <id>21842739</id>                         ...
  ...
</bud>                                         </bud>
  ...
<opsub>                                         <opsub id="43468739" >
  <id>43468739</id>                         ...
  ...
</opsub>                                         </opsub>
<vla>                                         <vla id="99557739" >
  <id>99557739</id>                         ...
  ...
</vla>                                         </vla>
<tel>                                         <tel id="53207739" >
  <id>53207739</id>                         ...
  ...
</tel>                                         </tel>
</data>

```

7.2.3.2 Výstupní formát pro webové služby

NVF je rozděleno do 68 datových bloků⁴. Každý datový blok popisuje nějaký objekt v systému ISKN. Jak bylo řečeno, kompletní NVF se jako exportní formát nehodí, ale proč nepoužít jen ty datové bloky, které budou nutné, pro úplný popis poskytovaných dat webovými službami. Při tomto řešení samozřejmě nelze dále hovořit o klasickém NVF!

Výhodou využití existujících datových bloků je znalost jejich struktury, která je velmi dobře popsána [8] a vyzkoušena. Datové bloky popisují „svěřené“ objekty vyčerpávajícím a dostatečně kvalitním způsobem. Není tedy třeba znova vytvářet a zkoušet nějaký další popis či jinou popisnou strukturu objektů. Navíc datové bloky obsahují informace, na jejichž základě lze datové bloky propojovat a vytvářet tak ucelené bloky souvisejících informací (v NVF jsou tyto ucelené bloky označovány jako skupiny, viz příloha A).

Ze zadání na tvorbu webových služeb víme, jaká data a v jakém rozsahu musí služby poskytovat. Jak by tedy vypadalo použití jen některých datových bloků pro popis

⁴ stav k 1.10.2007

poskytovaných dat?

Tabulka 7.2 přehledně zobrazuje, jaké datové bloky NVF jsou využity pro tvorbu výstupů jednotlivých webových služeb. Popis těchto bloků lze najít v dokumentaci k NVF [8], krátký popis je také k dispozici v příloze A. Výjimku tvoří datové bloky "CISEL", "PRAC" a "OPERACE". Tyto datové bloky obsahují informace, které jsou nepostradatelné pro výstupy z webových služeb a nemají odpovídající ekvivalent v NVF. Struktura všech datových bloků je detailně popsána pomocí XML schéma (XSD), XSD je umístěno v příloze C. Z této struktury je patrné, že pro popis dat nebyly využity všechny informace, které datové bloky v NVF mohou poskytovat. Prostým porovnáním popisu datových bloků z dokumentace NVF [8] a přiloženého XSD (příloha C) lze zjistit, které informace lze získat pouze z NVF a které lze získat pomocí výstupů webových služeb. Poskytovat všechny informace, které jsou dostupné v datových blocích NVF je zbytečné, byly vybrány pouze ty informace, které jsou pro uživatele užitečné a které dostatečně popisují data ve výstupech.

Webové služby	infoOParcelID	infoOParcel	vyhledaniParcel	infoOBudovID	infoOBudove	vyhledaniBudov	vyhledaniBudovParID	infoOJednotceID	infoOJednotce	vyhledaniJednotek	vyhledaniJednotekBudID	infoORizenID	infoORizeni	nejblzsiDefBodPar	nejblzsiDefBodBud	vratSeznamCiselniku	vratCiselnik
datový blok (SKU-PINA.BLOK)																	
NEMO.PAR	#	#	#	%	%	-	-	-	-	-	-	-	-	-	-	-	-
NEMO.BUD	%	%	-	#	#	#	#	#	#	-	-	-	-	-	-	-	-
NEMO.DRUPOZ	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.ZPVYPO	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.ZDPAZE	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.ZPURVY	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.TYPBUD	-	-	-	%	%	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.MAPLIS	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.KATUZE	#	#	-	%	%	-	-	%	%	-	-	-	-	-	-	-	%
NEMO.OBCE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.CASOBC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.NKRAJE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
NEMO.ZPVYBU	-	-	-	%	%	-	-	-	-	-	-	-	-	-	-	-	%
JEDN.JED	-	-	-	-	-	-	-	#	#	#	#	-	-	-	-	-	-
JEDN.TYPJED	-	-	-	-	-	-	-	#	#	-	-	-	-	-	-	-	%
JEDN.ZPVYJE	-	-	-	-	-	-	-	%	%	-	-	-	-	-	-	-	%
BDPA.BDP	%	%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VLST.OPSUB	%	%	-	%	%	-	-	%	%	-	-	-	-	-	-	-	-
VLST.VLA	%	%	-	%	%	-	-	%	%	-	-	-	-	-	-	-	-
VLST.TEL	%	%	-	%	%	-	-	%	%	-	-	-	-	-	-	-	-
RIZE.RIZENI	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.RIZKU	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.PRERIZ	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.UCAST	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.TYPPRE	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	%
RIZE.TYPRIZ	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.TYPUCA	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
RIZE.UCTYP	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-
DEBO.OBDEBO	-	-	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-
CISELNÍK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	#	-	-
PRAC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	%
OPERACE	-	-	-	-	-	-	-	-	-	-	-	#	#	-	-	-	-

Tabulka 7.2: Využití datových bloků pro výstupy webových služeb.

Vysvětlivky k tabulce 7.2:

#	datový blok se musí vyskytovat ve výstupu webové služby
%	datový blok se může vyskytovat ve výstupu webové služby
-	datový blok se nevyskytuje ve výstupu webové služby
CISELNIK	
PRAC	
OPERACE	tyto datové bloky se v NVF nevyskytují

7.2.4 Zabezpečení

Využívání webových služeb bude podmíněno bezplatnou registrací uživatelů. Tato registrace bude prováděna pomocí jednoduchého webového formuláře. Ve formuláři bude několik povinných položek:

- jméno a příjmení,
- kontaktní email,
- doménová adresa⁵ z které bude uživatel webové služby využívat,
- souhlas s podmínkami pro poskytování webových služeb.

Na základě těchto hodnot bude vygenerován přístupový klíč – HASH⁶. Všechny zadané hodnoty a HASH budou uloženy do databáze. Každá z poskytovaných služeb bude obsahovat povinný parametr HASH. Když uživatel zavolá webovou službu, server zkонтroluje příchozí HASH a doménovou adresu požadavku. Pokud bude HASH a doménová adresa souhlasit s zaregistrovanými údaji, budou uživateli zaslána požadovaná data. Pokud ne, bude vrácena chybová zpráva (viz 7.3.3). Pokud bude uživatel porušovat podmínky pro poskytování webových služeb, bude jeho registrace zablokována. Platnost přístupového klíče bude jeden rok.

Zmiňované řešení pro zabezpečení webových služeb ještě není konečné. V půlce ledna byly webové služby nasazeny do zkušebního (vnitrorezortního) provozu. Tento testovací provoz ukáže, jestli navržené řešení plně pokrývá všechny varianty⁷ pro bezpečný provoz služeb a také jestli nebude znepříjemňovat plynulou práci budoucím uživatelům těchto služeb. Aplikace je navržena tak, aby bylo možné v relativně krátké době změnit způsob zabezpečení služeb.

⁵nebo IP adresa

⁶jedinečná hodnota

⁷další možnou variantou pro zabezpečení služeb je využití uživatelských certifikátů

7.3 Webové služby

- Úplný popis webových služeb – WSDL, je umístěn v příloze D a v elektronické formě na přiloženém CD viz příloha F.
- Popis výstupního formátu webových služeb v XML schéma, je umístěn v příloze C a v elektronické formě na přiloženém CD viz příloha F.

7.3.1 Použité technologie

Pro tvorbu webových služeb byly použity tyto technologie a vývojové nástroje:

- Technologie
 - programovací jazyk Java, Enterprise Edition 5 (Java EE 5)⁸
 - * Java API for XML-Based RPC (JAX-RPC)⁹
 - * Java API for XML-Based Web Services (JAX-WS) 2.0¹⁰
 - programovací jazyk Java, Mobile Edition (Java ME)¹¹
 - byla použita databáze Oracle9i Enterprise Edition Release 9.2.0.7.0¹²
 - webové služby budou provozovány na aplikačním serveru GlassFish¹³
- Vývojové nástroje
 - pro vývoj a testování bylo použito prostředí Oracle JDeveloper (10.1.3.3)¹⁴
 - pro testování a tvorbu mobilního klienta bylo použito prostředí NetBeans 6.0¹⁵
 - pro testování bylo použito prostředí Visual Studio 2005¹⁶

⁸Java EE

URL: <<http://java.sun.com/javaee/>>

⁹JAX-RPC Reference Implementation

URL: <<https://jax-rpc.dev.java.net/>>

¹⁰JAX-WS Reference Implementation

URL: <<https://jax-ws.dev.java.net/>>

¹¹Java ME

URL: <<http://java.sun.com/javame/index.jsp>>

¹²Oracle9i Database

URL: <<http://www.oracle.com/technology/products/oracle9i/index.html>>

¹³GlassFish – Open Source Application Server

URL: <<https://glassfish.dev.java.net/>>

¹⁴Oracle JDeveloper

URL: <<http://www.oracle.com/technology/products/jdev/index.html>>

¹⁵NetBeans IDE 6.0

URL: <<http://www.netbeans.org/>>

¹⁶Visual Studio 2005

URL: <[http://msdn2.microsoft.com/cs-cz/vs2005/default\(en-us\).aspx](http://msdn2.microsoft.com/cs-cz/vs2005/default(en-us).aspx)>

7.3.2 Vstupní parametry

Všechny vstupní parametry, které jsou použity v poskytovaných službách, jsou znázorněny v tabulce 7.3. Z tabulky lze zjistit název parametru, jednoduchý popis parametru, jeho datový typ a formát parametru. Vstupní parametry jsou typu RPC style, formát parametrů je přímo definován v popisu webových služeb – WSDL (viz příloha D).

Název parametru	Popis parametru	Datový typ	Formát parametru
parID, budID, jedID, rizID	Jednoznačný identifikátor v systému ISKN	long	celočíselná hodnota max. délka 19
katuzeKod	Katastrální území kód	int	celočíselná hodnota délky 6
typParcely	Typ parcely (PKN, PZE)	String	znaková hodnota o délce 3 znaků
druhCislovani	Druh číslování	String	celočíselná hodnota délky 1
kmenoveCislo	Kmenové číslo	int	celočíselná hodnota max. délky 5
poddeleni	Poddělení	String	celočíselná hodnota max. délky 3
zdrojZE	Zdroj parcel ZE	String	celočíselná hodnota délky 1
castObcKod	Kód části obce	int	celočíselná hodnota max. délky 6
typBudKod	Typ budovy	String	celočíselná hodnota délky 1
cisloBud	Číslo budovy	int	celočíselná hodnota max. délky 4
cisloJed	Číslo jednotky	String	celočíselná hodnota max. délky 10
krajKod	Kód kraje	int	celočíselná hodnota délky 2
pracKod	Kód pracoviště	int	celočíselná hodnota délky 3
typRizKod	Kód typu řízení	String	znaková hodnota o délce 1 znaku
poradoveCislo	Pořadové číslo řízení	int	celočíselná hodnota max. délky 8
rokRiz	Rok řízení	int	znaková hodnota o délce 4 znaků
y	Souřadnice Y	double	číselná hodnota max. délky 8
x	Souřadnice X	double	číselná hodnota max. délky 9
nazevCiselniku	Název číselníku	String	znaková hodnota max. délky 255
kodHodnotyCiselniku	Kód hledané hodnoty v číselníku	String	číselná hodnota max. délky 6
datum	Datum změny	String	znaková hodnota max. délky 10
nazevPolozky	Název hledané položky	String	znaková hodnota max. délky 30
hash	Vygenerovaný HASH	String	celočíselná hodnota délky 28

Tabulka 7.3: Vstupní parametry webových služeb.

7.3.3 Chybová hlášení

Aplikace bude vracet chybová hlášení přímo v návratové zprávě ve formátu SOAP Fault (4.1.3). Výhodou tohoto řešení je celkem snadné zpracování této chybové zprávy na úrovni klientské aplikace. Aby bylo možné tyto zprávy zpracovávat, musí mít klientská aplikace podporu¹⁷ pro zachytávání SOAP výjimek – `SOAPFaultException`.

Pokud budou webové služby nedostupné, bude návratová zpráva vypadat následovně – příklad 7.2.

Příklad 7.2: Návratová zpráva

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:Fault xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/">
      <ns3:ns3="http://www.w3.org/2003/05/soap-envelope">
        <faultcode>ns2:Server</faultcode>
        <faultstring>Služba je dočasně nedostupná.</faultstring>
      </ns3:ns3>
    </ns2:Fault>
  </S:Body>
</S:Envelope>
```

Při dalších typech chybových zpráv zůstane struktura zprávy stejná, pouze se změní hodnota chybového hlášení – `<faultstring>`. V příkladě 7.3 je zobrazeno několik chybových zpráv, které mohou být webovými službami vygenerovány.

Příklad 7.3: Chybové hlášení

```
...
<faultstring>Pro zadané parametry nebyla vybrána žádná data.
</faultstring>
...
<faultstring>Hash nemá správnou hodnotu.</faultstring>
...
<faultstring>Parametr Typ budovy není zadán správně.
</faultstring>
...
<faultstring>Parametr Katastrální území není zadán správně.
</faultstring>
...
<faultstring>Parametr datum nemá správnou hodnotu. Správný formát
je DD.MM.RRRR (např. 01.01.2001).</faultstring>
...
```

Všechna chybová hlášení jsou soustředěna do jednoho XML konfiguračního souboru, který je součástí projektu (`Messages.xml`).

¹⁷ Java API for XML-based RPC (JAX-RPC)

7.3.4 Výstupní formát

Jako návratový typ webových služeb byl zvolen primitivní datový typ `java.lang.String`. Struktura výsledného výstupního formátu v XML je následující¹⁸:

- kořenový element `<wsvfkiskn>`

```
<wsvfkiskn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns="urn:xmlns:nahlichenidokn.cuzk.cz:wsvfkiskn">
```

- hlavička `<hlavicka>`

```
<hlavicka verze="" vytvoreno="2008-01-25T10:06:14"
            vytvoril="WSNahlicheni" zmeny="0" navrhy="0">
  <platnost od="2008-01-25T10:06:14" do="2008-01-25T10:06:14">
    </platnost>
  <datoveskupiny>
    <skupina>NEMO</skupina>
    <skupina>VLST</skupina>
  </datoveskupiny>
</hlavicka>
```

- data `<data>`, element obsahuje datové bloky

```
<data>
  <par>
    ...
  </par>
  <bud>
    <id>21917739</id>
    <typbud_kod>1</typbud_kod>
    <caobce_kod>6483</caobce_kod>
    <cislo_domovni>10</cislo_domovni>
    <zpybu_kod>7</zpybu_kod>
    <tel_id>53335739</tel_id>
  </bud>
  <typbud>
    ...
  </typbud>
  <katuze>
    ...
  </katuze>
  ...
</data>
```

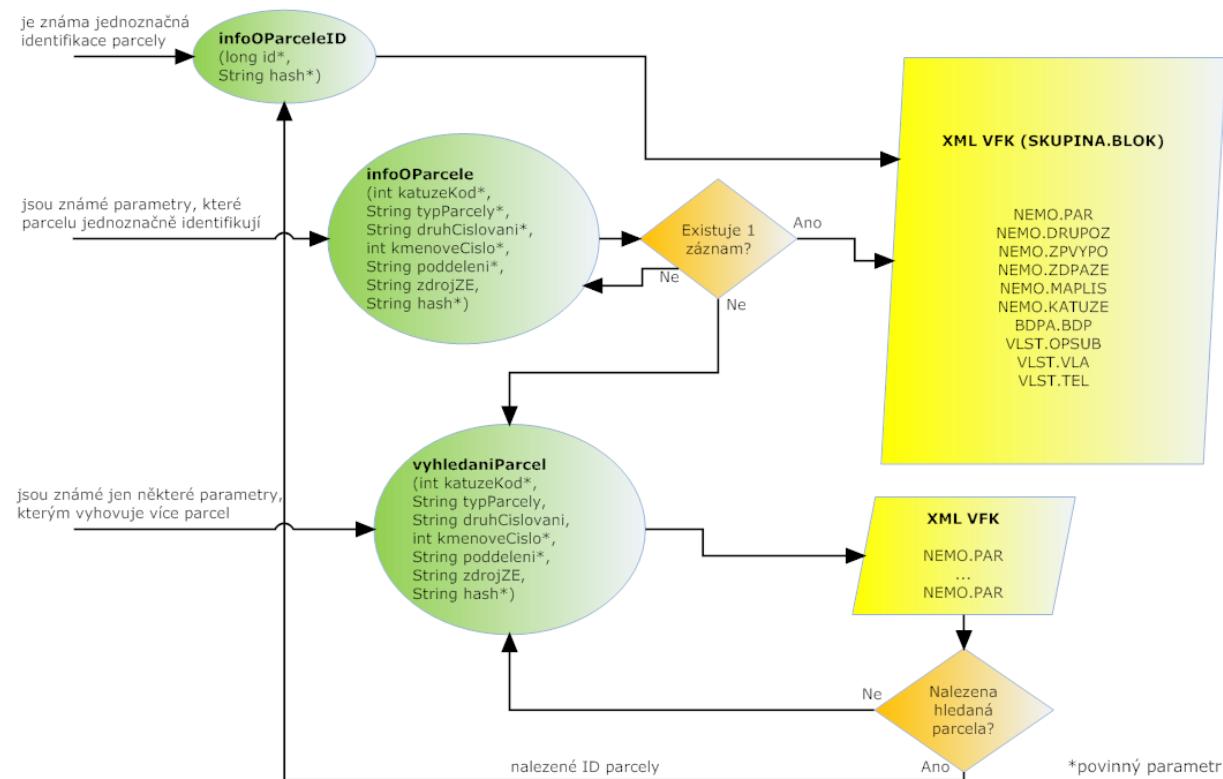
Kódování výsledného XML dokumentu je v UTF-8.

¹⁸struktura je téměř totožná s XML NVF

7.3.5 Získání informací o parcele

Získat informace o parcele lze třemi způsoby (viz obrázek 7.6):

1. Je známa jednoznačná identifikace parcely (ID), lze použít službu *infoOParceleID*.
2. Jsou známé parametry, které parcelu jednoznačně identifikují, lze použít službu *infoOParcele*.
3. Jsou známé jen některé parametry, které vyhovují více parcelám, lze použít službu *vyhledaniParcel*.



Obrázek 7.6: Webové služby pro parcely.

7.3.5.1 Informace o parcele – jednoznačný identifikátor

Název funkce: *infoOParceleID*

Vstupní parametry jsou zobrazeny v tabulce 7.4.

Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.7.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace parcely	parID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.4: Parametry webové služby *infoOParcelID*.

Příklady volání:

```
infoOParcelID(333189739, "12Xt11JHU5ekUH13fFshxdEH1M="))  
infoOParcelID(332072739, "12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.5.2 Informace o parcele

Název funkce: *infoOParcel*

Vstupní parametry jsou zobrazeny v tabulce 7.5.

Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.7.

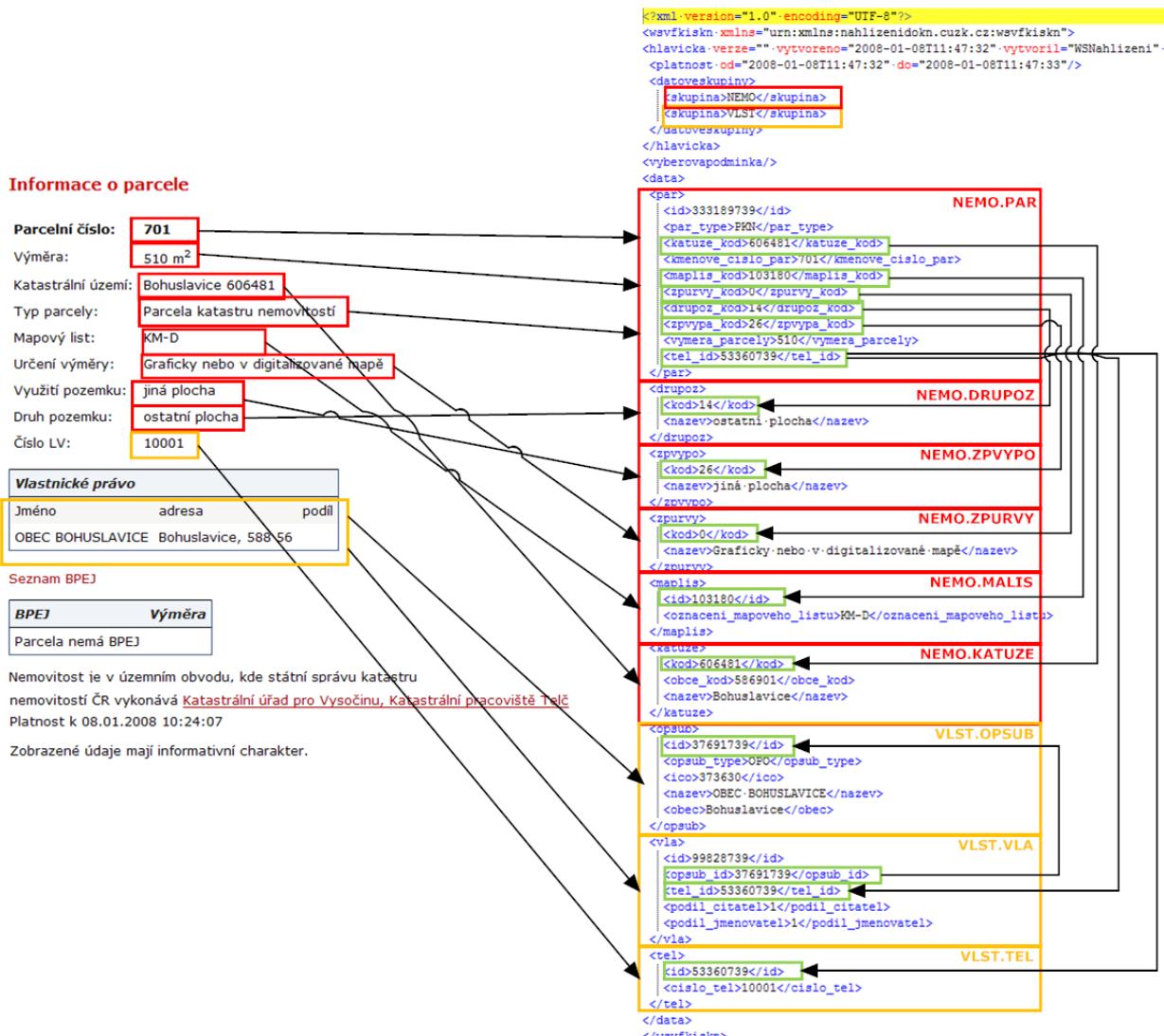
Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Katastrální území	katuzeKod	ano	int
Typ parcely (PKN, PZE)	typParcely	ano	String
Druh číslování	druhCislovaní	ano	String
Kmenové číslo	kmenoveCislo	ano	int
Poddělení	poddelení	ano ¹⁹	String
Zdroj parcel ZE	zdrojZE	ne	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.5: Parametry webové služby *infoOParcel*.

Příklady volání:

```
infoOParcel(606481, "PKN", "2", 685, "2", null, "12Xt11JHU5ekUH13fFshxdEH1M="))  
infoOParcel(705268, "PZE", "2", 307, null, null, "12Xt11JHU5ekUH13fFshxdEH1M="))
```

¹⁹jestliže parcela nemá poddělení, tento parametr nebude vyplněn



Obrázek 7.7: Vyhodnocení výstupního formátu z webových služeb *infoOParcelID infoOParcel*. Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.

7.3.5.3 Vyhledání parcel

Název funkce: *vyhledaniParcel*

Vstupní parametry jsou zobrazeny v tabulce 7.6.

Funkce vrací pouze jeden opakující se datový blok – NEMO.PAR (viz tabulka 7.2). Datový blok se opakuje podle počtu nalezených parcel, které vyhovují zadání. Cílem je nalézt hledanou parcelu a získat její jednoznačný identifikátor (ID). Pro získání podrobných informací o parcele pak lze použít služba *infoOParcelID*.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Katastrální území	katuzeKod	ano	int
Typ parcely (PKN, PZE)	typParcely	ne	String
Druh číslování	druhCisloVani	ne	String
Kmenové číslo	kmenoveCislo	ano	int
Poddelení	poddeleni	ano ²⁰	String
Zdroj parcel ZE	zdrojZE	ne	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.6: Parametry webové služby *vyhledaniParcel*.

Příklady volání:

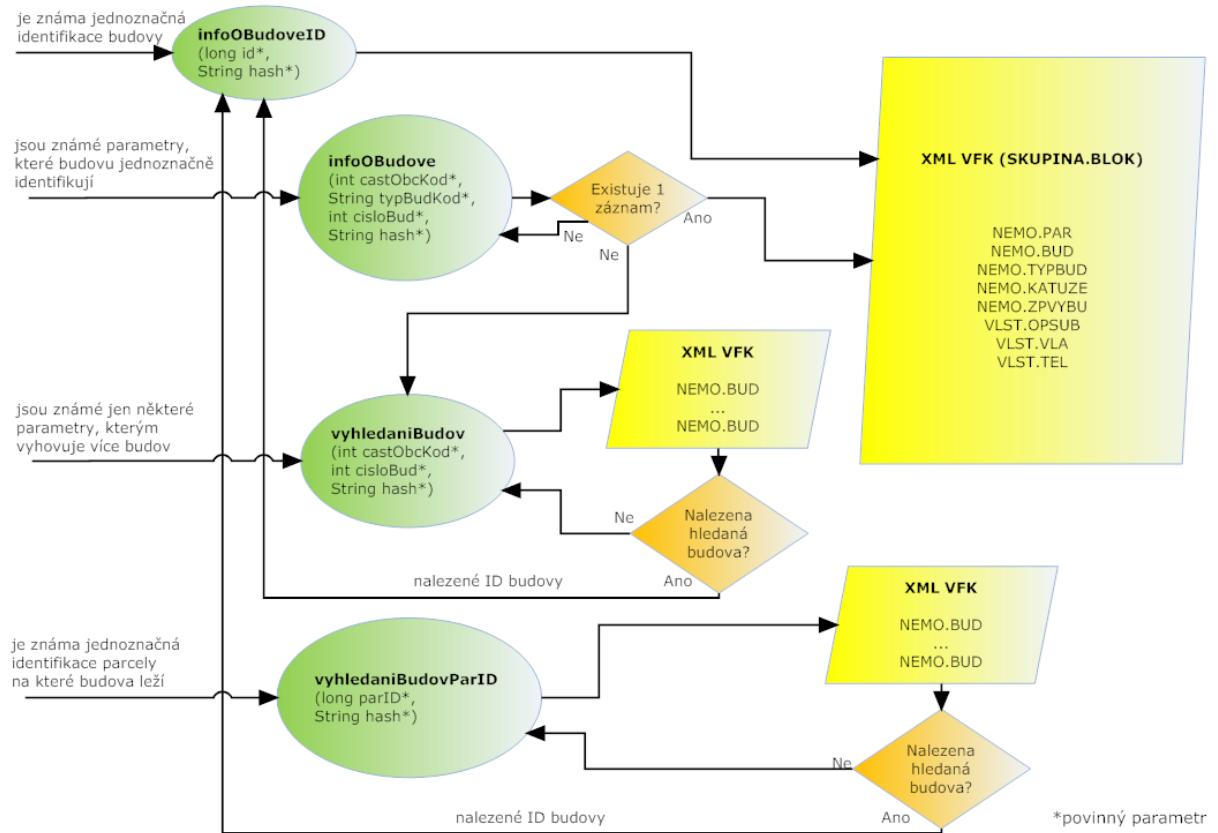
```
vyhledaniParcel(606481, "PKN", null, 470, null, null, "12Xt11JHU5ekUH13fFshxdEH1M="))  
vyhledaniParcel(705268, "PZE", null, 307, null, null, "12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.6 Získání informací o budově

Získat informace o budově lze čtyřmi způsoby (viz obrázek 7.8):

1. Je známa jednoznačná identifikace budovy (ID), lze použít službu *infoOBudoveID*.
2. Jsou známé parametry, které budovu jednoznačně identifikují, lze použít službu *infoOBudove*.
3. Jsou známé jen některé parametry, které vyhovují více budovám, lze použít službu *vyhledaniBudov*.
4. Je známa jednoznačná identifikace parcely (ID) na které budova leží, lze použít službu *vyhledaniBudovParID*.

²⁰jestliže parcela nemá poddelení, tento parametr nebude vyplněn



Obrázek 7.8: Webové služby pro budovy.

7.3.6.1 Informace o budově – jednoznačný identifikátor

Název funkce: *infoOBudoveID*

Vstupní parametry jsou zobrazeny v tabulce 7.7.

Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.9.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace budovy	budID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.7: Parametry webové služby *infoOBudoveID*.

Příklady volání:

```
infoOBudoveID(21842739, "12Xt11JHU5ekUH13fFshxdEH1M="))
infoOBudoveID(21917739, "12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.6.2 Informace o budově

Název funkce: *infoOBudove*

Vstupní parametry jsou zobrazeny v tabulce 7.8.

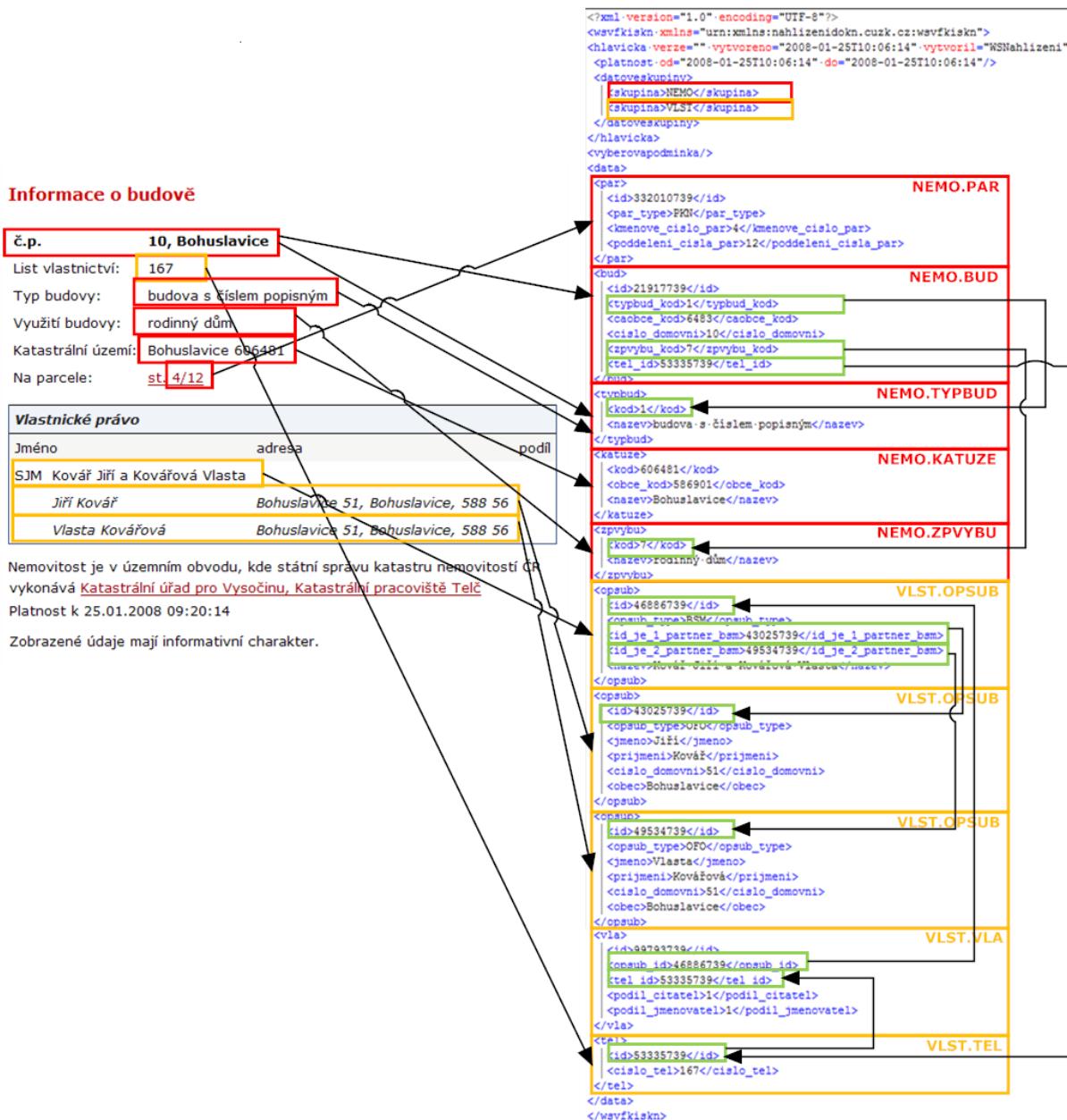
Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.9.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Kód části obce	castObcKod	ano	int
Typ budovy	typBudKod	ano	String
Číslo budovy	cisloBud	ano	int
Vygenerovaný HASH	hash	ano	String

Tabulka 7.8: Parametry webové služby *infoOBudove*.

Příklady volání:

```
infoOBudove(6483,"1",10,"12Xt11JHU5ekUH13fFshxdEH1M="))
infoOBudove(7790,"1",27,"12Xt11JHU5ekUH13fFshxdEH1M="))
```



Obrázek 7.9: Vyhodnocení výstupního formátu z webových služeb *infoOBudoveID* a *infoOBudove*. Vlevo jsou informace získané z aplikace Nahlízení a vpravo informace získané z webových služeb.

7.3.6.3 Vyhledání budov

Název funkce: *vyhledaniBudov*

Vstupní parametry jsou zobrazeny v tabulce 7.9.

Funkce vrací pouze jeden opakující se datový blok – NEMO.BUD (viz tabulka 7.2). Datový blok se opakuje podle počtu nalezených budov, které vyhovují zadání. Cílem je nalézt

hledanou budovu a získat její jednoznačný identifikátor (ID). Pro získání podrobných informací o budově pak lze použít služba *infoOBudoveID*.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Kód části obce	castObcKod	ano	int
Typ budovy	typBudKod	ano	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.9: Parametry webové služby *vyhledaniBudov*.

Příklady volání:

```
vyhledaniBudov(6483,10,"12Xt11JHU5ekUH13fFshxdEH1M="))
vyhledaniBudov(45713,24,"12Xt11JHU5ekUH13fFshxdEH1M=")
```

7.3.6.4 Vyhledání budov – jednoznačný identifikátor parcely

Název funkce: *vyhledaniBudovParID*

Vstupní parametry jsou zobrazeny v tabulce 7.10.

Funkce vrací pouze jeden opakující se datový blok – NEMO.BUD (viz tabulka 7.2). Datový blok se opakuje podle počtu nalezených budov, které vyhovují zadání. Cílem je nalézt hledanou budovu (která leží na zadané parcele) a získat její jednoznačný identifikátor (ID). Pro získání podrobných informací o budově pak lze použít služba *infoOBudoveID*.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace parcely na které budova leží	parID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.10: Parametry webové služby *vyhledaniBudovParID*.

Příklady volání:

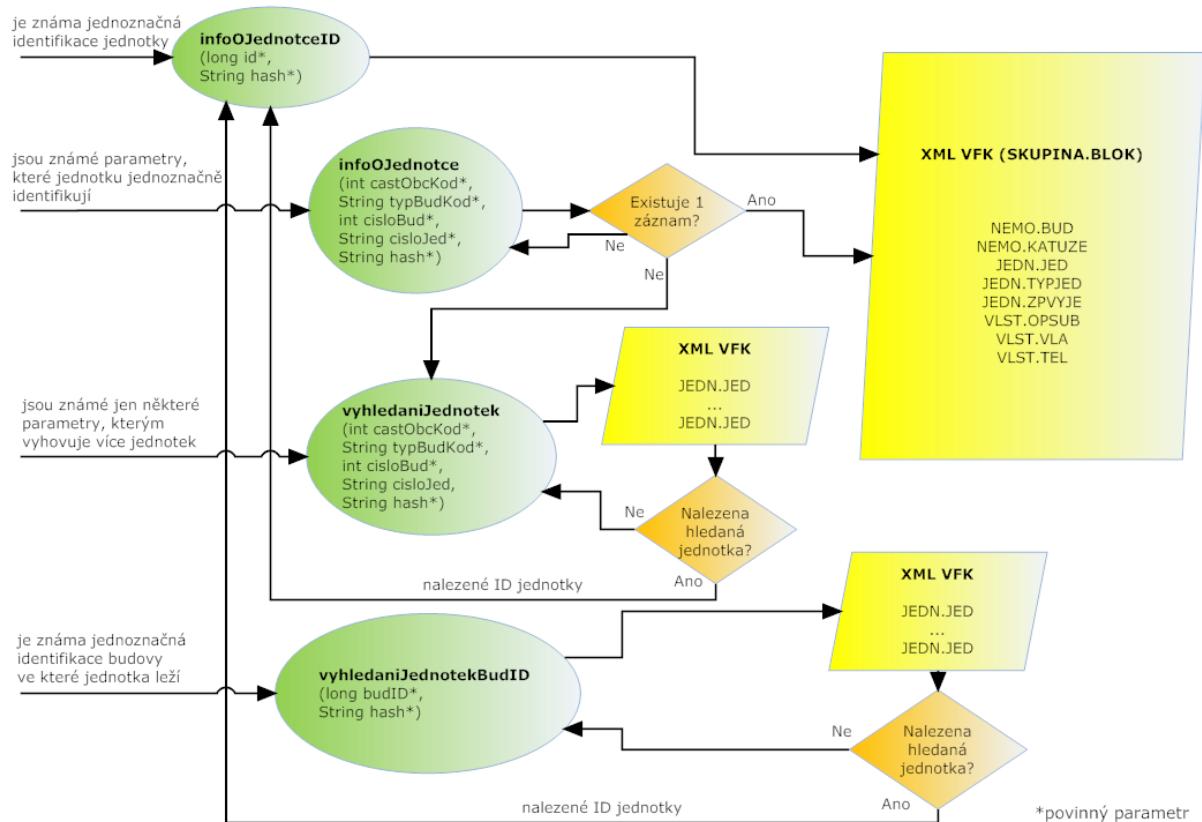
```
vyhledaniBudovParID(3320227367,"12Xt11JHU5ekUH13fFshxdEH1M=")
vyhledaniBudovParID(3123227324,"12Xt11JHU5ekUH13fFshxdEH1M=")
```

7.3.7 Získání informací o jednotce

Získat informace o jednotce lze čtyřmi způsoby (viz obrázek 7.10):

1. Je známa jednoznačná identifikace jednotky (ID), lze použít službu *infoOJednotceID*.

2. Jsou známé parametry, které jednotku jednoznačně identifikují, lze použít službu *infoOJednotce*.
3. Jsou známé jen některé parametry, které vychovují více jednotkám, lze použít službu *vyhledaniJednotek*.
4. Je známa jednoznačná identifikace budovy (ID) ve které jednotka leží, lze použít službu *vyhledaniJednotekBudID*.



Obrázek 7.10: Webové služby pro jednotky.

7.3.7.1 Informace o jednotce – jednoznačný identifikátor

Název funkce: *infoOJednotceID*

Vstupní parametry jsou zobrazeny v tabulce 7.11.

Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.11.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace jednotky	jedID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.11: Parametry webové služby *infoOJednotceID*.

Příklady volání:

```
infoOJednotceID(2068739,"12Xt11JHU5ekUH13fFshxdEH1M="))
infoOJednotceID(2643739,"12Xt11JHU5ekUH13fFshxdEH1M=")
```

7.3.7.2 Informace o jednotce

Název funkce: *infoOJednotce*

Vstupní parametry jsou zobrazeny v tabulce 7.12.

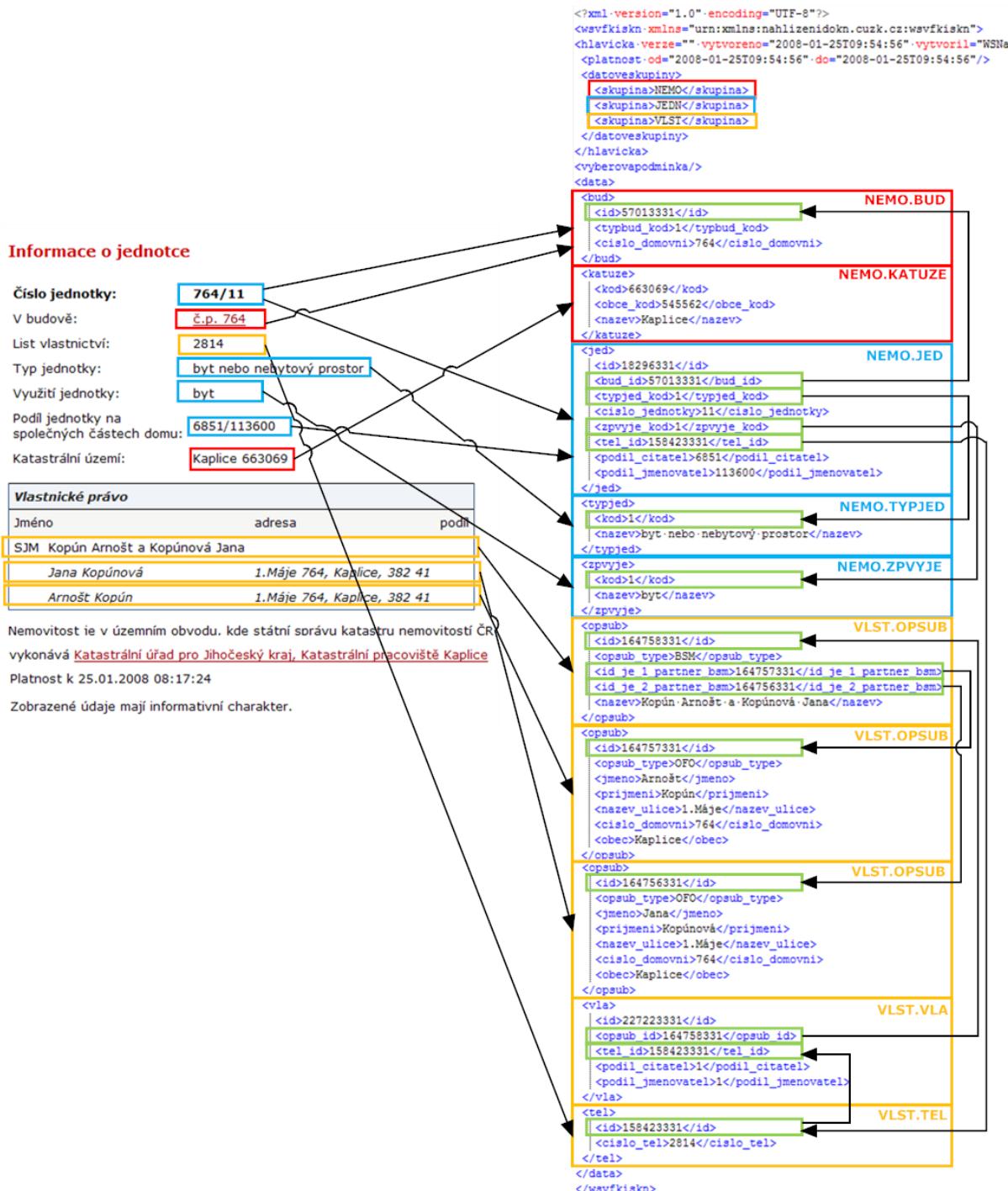
Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.11.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Kód části obce	castObcKod	ano	int
Typ budovy	typBudKod	ano	String
Číslo budovy	cisloBud	ano	int
Číslo jednotky	cisloJed	ano	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.12: Parametry webové služby *infoOJednotce*.

Příklady volání:

```
infoOJednotce(72206,"1",93,"93001","12Xt11JHU5ekUH13fFshxdEH1M=")
infoOJednotce(403989,"1",764,"11","12Xt11JHU5ekUH13fFshxdEH1M=")
```



Obrázek 7.11: Vyhodnocení výstupního formátu z webových služeb *infoOJednotceID* a *infoOJednotce*. Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.

7.3.7.3 Vyhledání jednotek

Název funkce: *vyhledaniJednotek*

Vstupní parametry jsou zobrazeny v tabulce 7.13.

Funkce vrací pouze jeden opakující se datový blok – NEMO.JED (viz tabulka 7.2). Datový blok se opakuje podle počtu nalezených jednotek, které vyhovují zadání. Cílem je nalézt hledanou jednotku a získat její jednoznačný identifikátor (ID). Pro získání podrobných informací o jednotce pak lze použít služba *infoOJednotceID*.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Kód části obce	castObcKod	ano	int
Typ budovy	typBudKod	ano	String
Číslo jednotky	cisloJed	ne	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.13: Parametry webové služby *vyhledaniJednotek*.

Příklady volání:

```
vyhledaniJednotek(72206,"1",93,null,"12Xt11JHU5ekUH13fFshxdEH1M="))
vyhledaniJednotek(100056,"1",242,null,"12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.7.4 Vyhledání jednotek – jednoznačný identifikátor budovy

Název funkce: *vyhledaniJednotekBudID*

Vstupní parametry jsou zobrazeny v tabulce 7.14.

Funkce vrací pouze jeden opakující se datový blok – NEMO.JED (viz tabulka 7.2). Datový blok se opakuje podle počtu nalezených jednotek, které vyhovují zadání. Cílem je nalézt hledanou jednotku (která se vyskytuje v zadané budově) a získat její jednoznačný identifikátor (ID). Pro získání podrobných informací o jednotce pak lze použít služba *infoOJednotceID*.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace budovy ve které jednotka existuje	budID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.14: Parametry webové služby *vyhledaniJednotekBudID*.

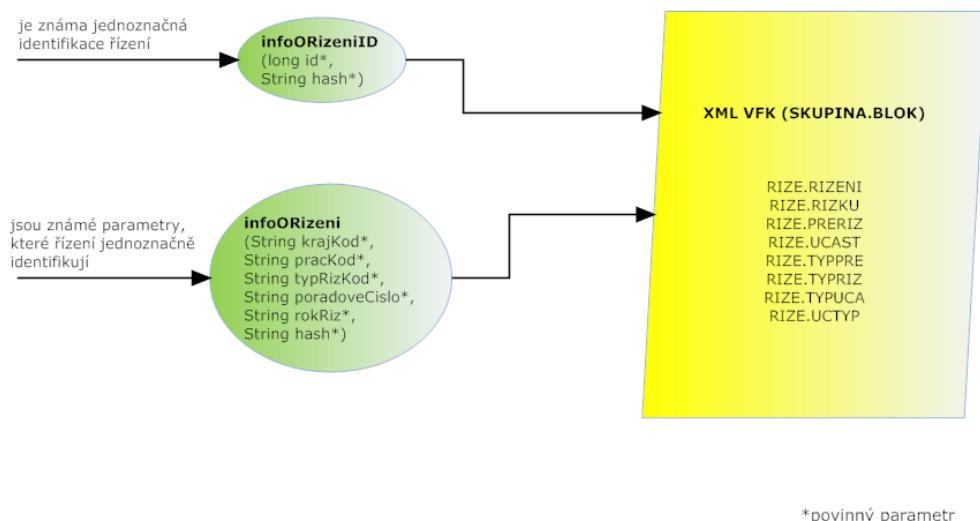
Příklady volání:

```
vyhledaniJednotekBudID(22913739,"12Xt11JHU5ekUH13fFshxdEH1M="))
vyhledaniJednotekBudID(23766739,"12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.8 Získání informací o řízení

Získat informace o řízení lze dvěma způsoby (viz obrázek 7.12):

1. Je známa jednoznačná identifikace řízení (ID), lze použít službu *infoORizeniID*.
2. Jsou známé parametry, které řízení jednoznačně identifikují, lze použít službu *infoORizeni*.



*povinný parametr

Obrázek 7.12: Webové služby pro řízení.

7.3.8.1 Informace o řízení – jednoznačný identifikátor

Název funkce: *infoORizeniID*

Vstupní parametry jsou zobrazeny v tabulce 7.15.

Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.13.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Jednoznačná identifikace řízení	rizID	ano	long
Vygenerovaný HASH	hash	ano	String

Tabulka 7.15: Parametry webové služby *infoORizeniID*.

Příklady volání:

`infoORizeniID(172571739, "12Xt11JHU5ekUH13fFshxdEH1M=")`

`infoORizeniID(172481739, "12Xt11JHU5ekUH13fFshxdEH1M=")`

7.3.8.2 Informace o řízení

Název funkce: *infoORizeni*

Vstupní parametry jsou zobrazeny v tabulce 7.16.

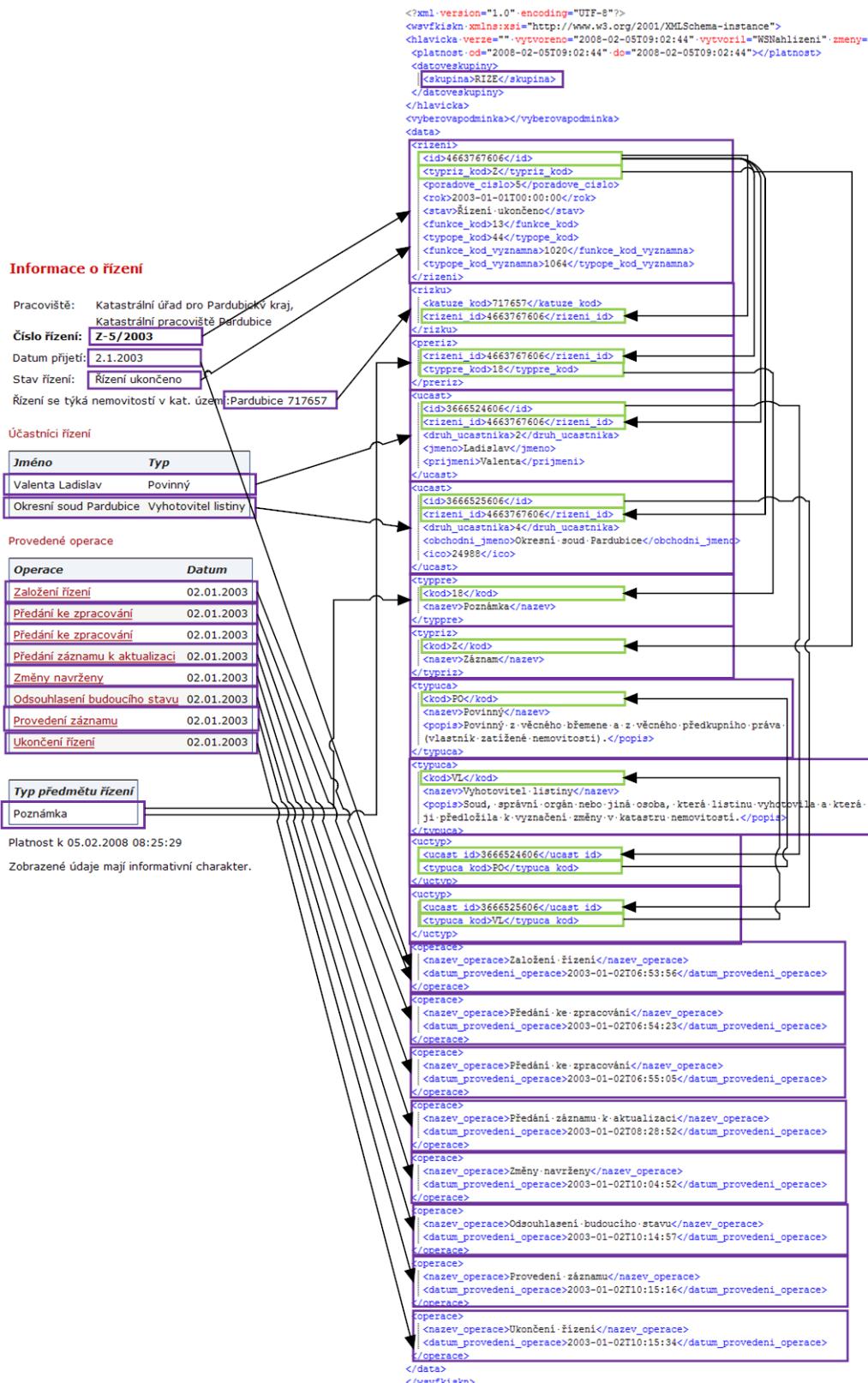
Funkce vrací seznam datových bloků podle tabulky 7.2. Vyhodnocení získaných dat je zobrazeno na obrázku 7.13.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Kód kraje	krajKod	ano	int
Kód pracoviště	pracKod	ano	int
Kód typu řízení	typRizKod	ano	String
Pořadové číslo řízení	poradoveCislo	ano	int
Rok řízení	rokRiz	ano	int
Vygenerovaný HASH	hash	ano	String

Tabulka 7.16: Parametry webové služby *infoORizeni*.

Příklady volání:

```
infoORizeni(108,739,"V",176,1997,"12Xt11JHU5ekUH13fFshxdEH1M="))  
infoORizeni(94,606,"Z",5,2003,"12Xt11JHU5ekUH13fFshxdEH1M="))
```



Obrázek 7.13: Vyhodnocení výstupního formátu z webových služeb *infoORizeniID* a *infoORizeni*. Vlevo jsou informace získané z aplikace Nahlížení a vpravo informace získané z webových služeb.

7.3.9 Nalezení nejbližšího definičního bodu

Nalezení nejbližšího definičního bodu je možné pro:

1. parcely, lze použít službu *nejblizsiDefBodPar*,
2. budovy, lze použít službu *nejblizsiDefBodBud*.

7.3.9.1 Nalezení nejbližšího definičního bodu parcely

Název funkce: *nejblizsiDefBodPar*

Vstupní parametry jsou zobrazeny v tabulce 7.17.

Funkce vrací jeden datový blok DEBO.OBDEBO (viz tabulka 7.2).

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Souřadnice Y	y	ano	double
Souřadnice X	x	ano	double
Vygenerovaný HASH	hash	ano	String

Tabulka 7.17: Parametry webové služby *nejblizsiDefBodPar*.

Příklad volání:

```
nejblizsiDefBodPar(678350.00,1157540.00,"12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.9.2 Nalezení nejbližšího definičního bodu budovy

Název funkce: *nejblizsiDefBodBud*

Vstupní parametry jsou zobrazeny v tabulce 7.18.

Funkce vrací jeden datový blok DEBO.OBDEBO (viz tabulka 7.2).

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Souřadnice Y	y	ano	double
Souřadnice X	x	ano	double
Vygenerovaný HASH	hash	ano	String

Tabulka 7.18: Parametry webové služby *nejblizsiDefBodBud*.

Příklad volání:

```
nejblizsiDefBodBud(678350.00,1157540.00,"12Xt11JHU5ekUH13fFshxdEH1M="))
```

7.3.10 Získání informací o číselnících

Získat informace o číselnících, lze dvěma způsoby:

1. je znám název číselníku, lze použít službu *vratCiselnik*.
2. Není znám název číselníku či je potřeba získat seznam dostupných číselníků, lze použít službu *vratSeznamCiselniku*.

7.3.10.1 Informace o číselníku

Název funkce: *vratCiselnik*

Vstupní parametry jsou zobrazeny v tabulce 7.19.

Funkce vrací datový blok, který odpovídá vybranému číselníku (viz tabulka 7.2). Výběr hodnot lze ovlivnit podle kombinace nepovinných parametrů funkce.

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Název číselníku	nazevCiselniku	ano	String
Kód hledané hodnoty v číselníku	kodHodnotyCiselniku	ne	String
Datum změny v číselníku	datum	ne	String
Název hledané položky v číselníku	nazevPolozky	ne	String
Vygenerovaný HASH	hash	ano	String

Tabulka 7.19: Parametry webové služby *vratCiselnik*.

Příklady volání:

```
vratCiselnik("DRUPOZ",null,null,null,"12Xt11JHU5ekUH13fFshxdEH1M=")
vratCiselnik("DRUPOZ","8",null,null,"12Xt11JHU5ekUH13fFshxdEH1M=")
vratCiselnik("DRUPOZ",null,null,"plocha","12Xt11JHU5ekUH13fFshxdEH1M=")
vratCiselnik("DRUPOZ",null,"1.1.2000",null,"12Xt11JHU5ekUH13fFshxdEH1M=")
vratCiselnik("DRUPOZ","11","1.1.2000","plocha","12Xt11JHU5ekUH13fFshxdEH1M=")
```

7.3.10.2 Seznam číselníků

Název funkce: *vratSeznamCiselniku*

Vstupní parametr je zobrazen v tabulce 7.20.

Funkce vrací opakující se datový blok CISELNIK, funkce vrací seznam všech dostupných číselníků

Popis parametru	Název parametru	Povinnost vyplnění	Datový typ
Vygenerovaný HASH	hash	ano	String

Tabulka 7.20: Parametry webové služby *vratSeznamCiselniku*.

Příklad volání:

```
vratSeznamCiselniku("12Xt11JHU5ekUH13fFshxdEH1M="))
```

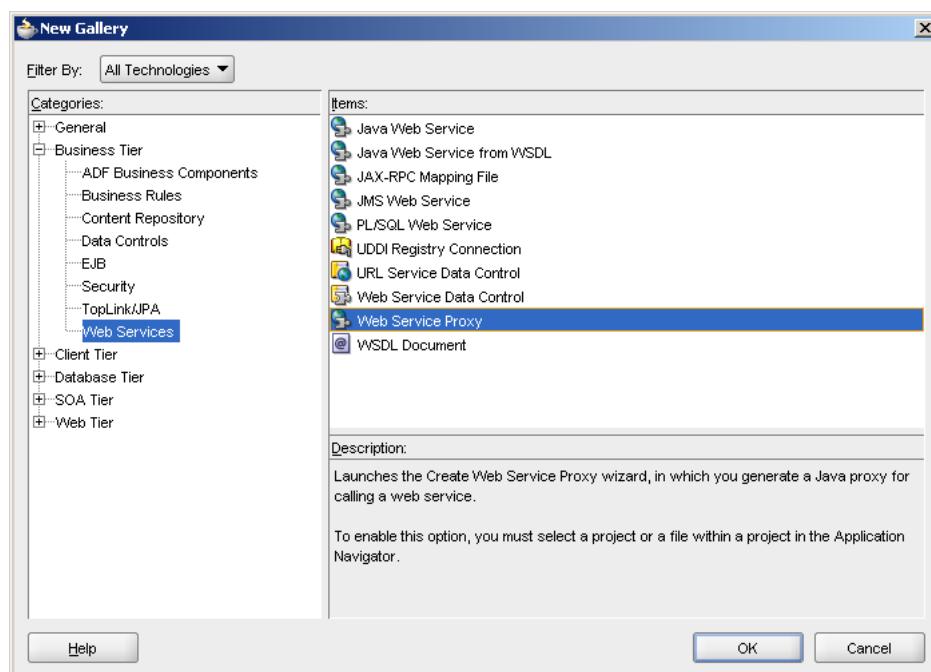
7.3.11 Tvorba klientské aplikace

Vytvořit klientskou část k webové službě je dnes pro průměrně zdatného programátora snadnou záležitostí. Většina dnes zdarma dostupných vývojových nástrojů, totiž umožňuje klientskou část vygenerovat na základě WSDL.

7.3.11.1 Klasický klient

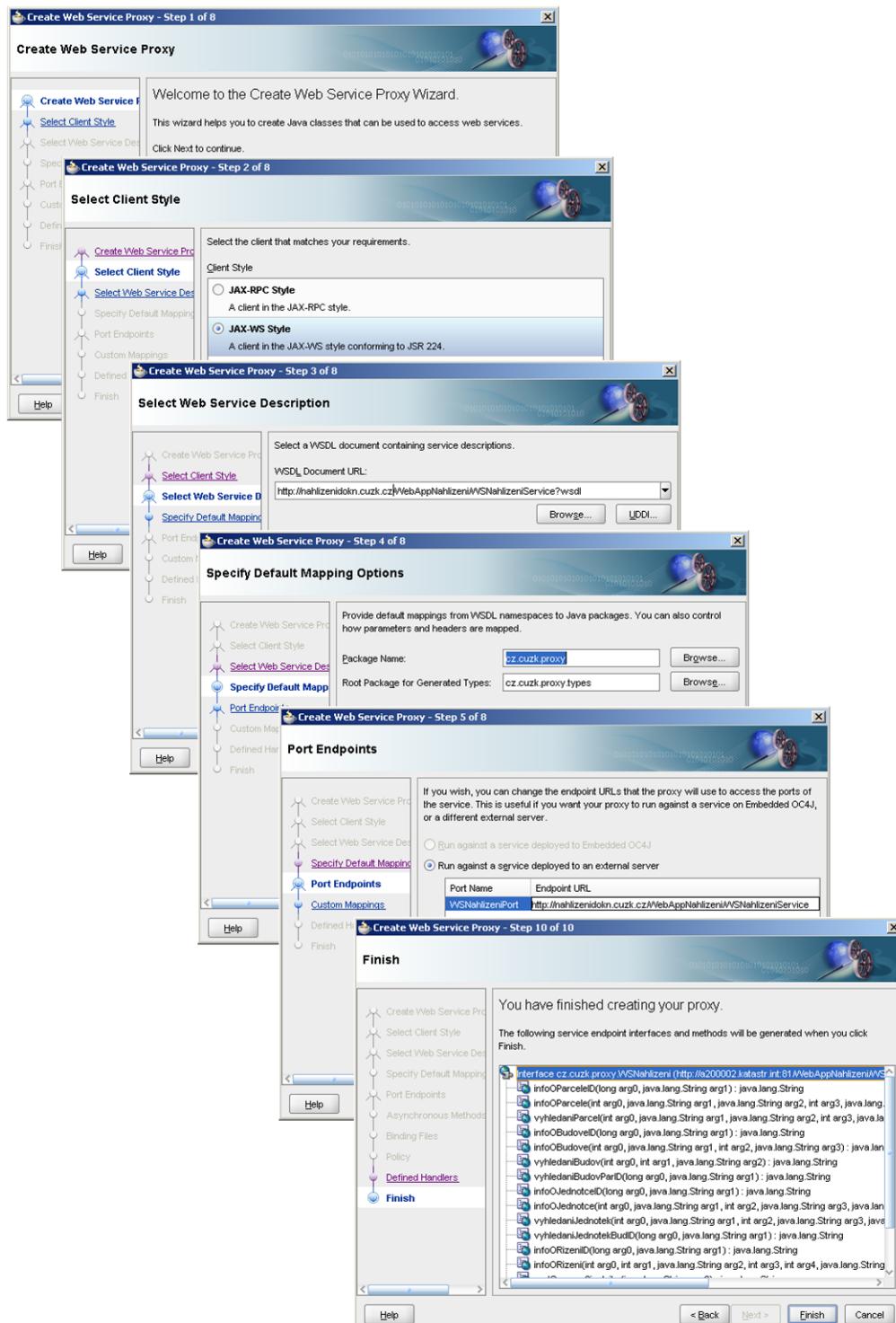
Postup tvorby klientské části v JDeveloperu (v 11.1.1.0):

- Do existujícího projektu přidáme ”zástupce webové služby” – Web Service Proxy (viz obrázek 7.14).



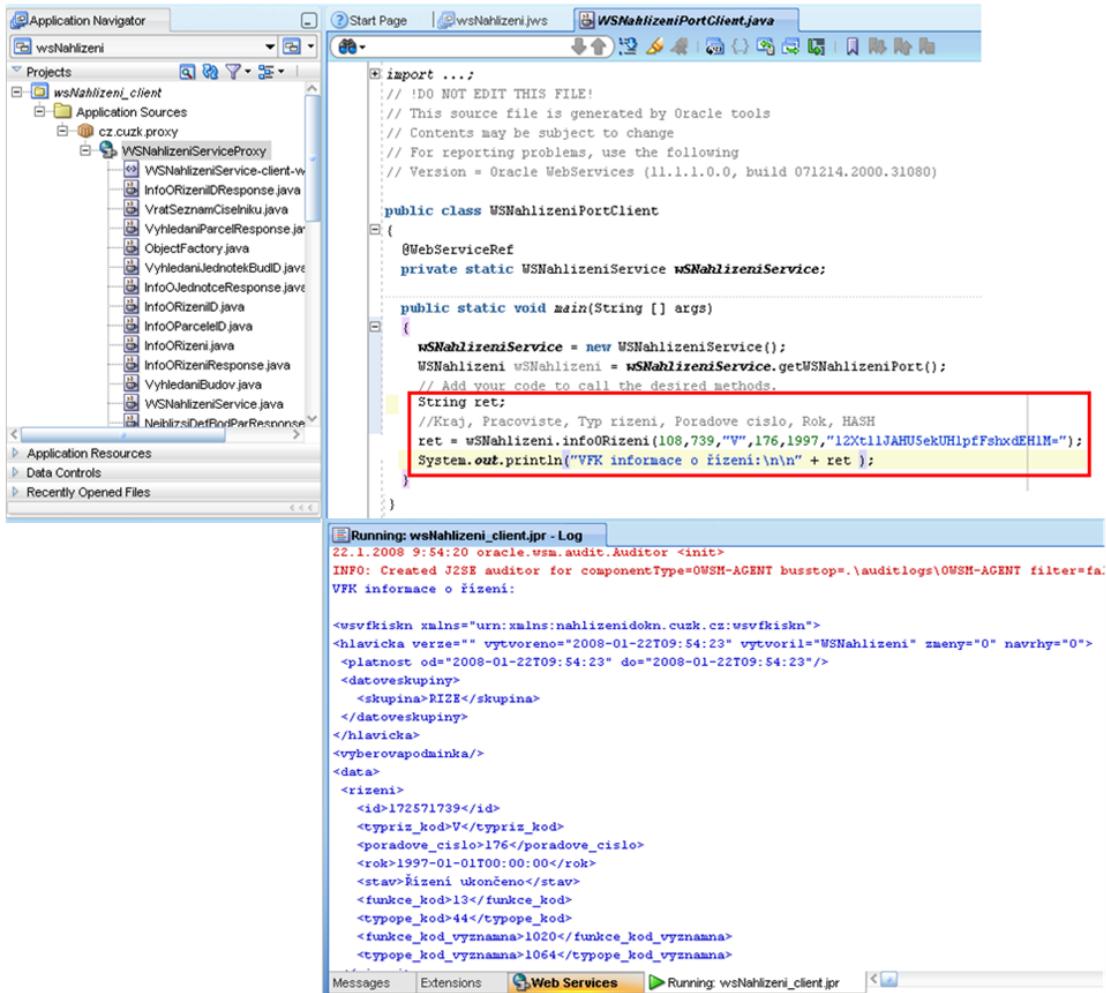
Obrázek 7.14: Web Service Proxy

2. Poté se otevře průvodce, který v několika krocích provede uživatele vytvořením Web Service Proxy (viz obrázek 7.15).



Obrázek 7.15: Tvorba klientské části pomocí průvodce v JDeveloperu (v 11.1.1.0)

3. Do vygenerovaných zdrojových kódů je potřeba doplnit volání webové služby. V našem případě jde o volání webové služby "infoORizeni". V konzoli je zobrazen výsledek, který vrátila webová služba (viz obrázek 7.16).



```

import ...
// DO NOT EDIT THIS FILE!
// This source file is generated by Oracle tools
// Contents may be subject to change
// For reporting problems, use the following
// Version = Oracle WebServices (11.1.1.0.0, build 071214.2000.31080)

public class WSNahlizeniPortClient
{
    @WebServiceRef
    private static WSNahlizeniService wSNahlizeniService;

    public static void main(String [] args)
    {
        wSNahlizeniService = new WSNahlizeniService();
        WSNahlizeni wSNahlizeni = wSNahlizeniService.getWSNahlizeniPort();
        // Add your code to call the desired methods.

        String ret;
        //Kraj, Pracoviste, Typ rizeni, Poradove cislo, Rok, HASH
        ret = wSNahlizeni.infoORizeni(108,739,"V",176,1997,"12Xt11JAHU5ekUHlpfFshxdEH1M");
        System.out.println("VFK informace o rizeni:\n" + ret );
    }
}

Running: wsNahlizeni_client.jpr - Log
22.1.2008 9:54:20 oracle.usm.audit.Auditor <init>
INFO: Created J2EE auditor for componentType=OWSM-AGENT busstop=.auditlogs\OWSM-AGENT filter=fa:
VFK informace o rizeni:

<usvfkiskn xmlns="urn:xalns:nahlichenidokn.czuk.cz:usvfkiskn">
<hlavicka verze="" vytvoreno="2008-01-22T09:54:23" vytvoril="WSNahlizeni" zmeny="0" navrh="0">
<platnost od="2008-01-22T09:54:23" do="2008-01-22T09:54:23"/>
<datoveskupiny>
    <skupina>RIZE</skupina>
</datoveskupiny>
</hlavicka>
<vyberovapodminka>
<data>
<rizeni>
    <id>172571739</id>
    <typriz_kod>V</typriz_kod>
    <poradove_cislo>176</poradove_cislo>
    <rok>1997-01-01T00:00</rok>
    <stav>Rizeni ukonceno</stav>
    <funkce_kod>13</funkce_kod>
    <typope_kod>44</typope_kod>
    <funkce_kod_vyznamna>1020</funkce_kod_vyznamna>
    <typope_kod_vyznamna>1064</typope_kod_vyznamna>

```

Obrázek 7.16: Vytvořené zdrojové kódy klientské části webové služby

Zdrojové kódy jsou k dispozici na přiloženém CD. Struktura CD je popsána v příloze F.

7.3.11.2 Mobilní klient

Vytvoření klientské aplikace pro mobilní telefon již není tak snadnou záležitostí jako bylo vytvoření klasického řádkového klienta. Opět není problém ve vytvoření "zástupce webové služby", ale spíše znalost specifikace J2ME (Java Platform, Micro Edition (Java ME))²¹.

²¹ Java ME Technology API Documentation
URL: <<http://java.sun.com/javame/reference/apis.jsp>>

V následné ukázce je vytvořená klientská aplikace pro mobilní telefony NOKIA²². V této ukázkové aplikaci byly implementovány čtyři základní webové služby: *infoOParcele*, *infoOBudove*, *infoOJednotce*, *infoORizeni*.

Ukázková klientská aplikace pro mobilní telefony:

- Úvodní menu



Obrázek 7.17: Úvodní menu aplikace.

- Služba *infoOParcele*.

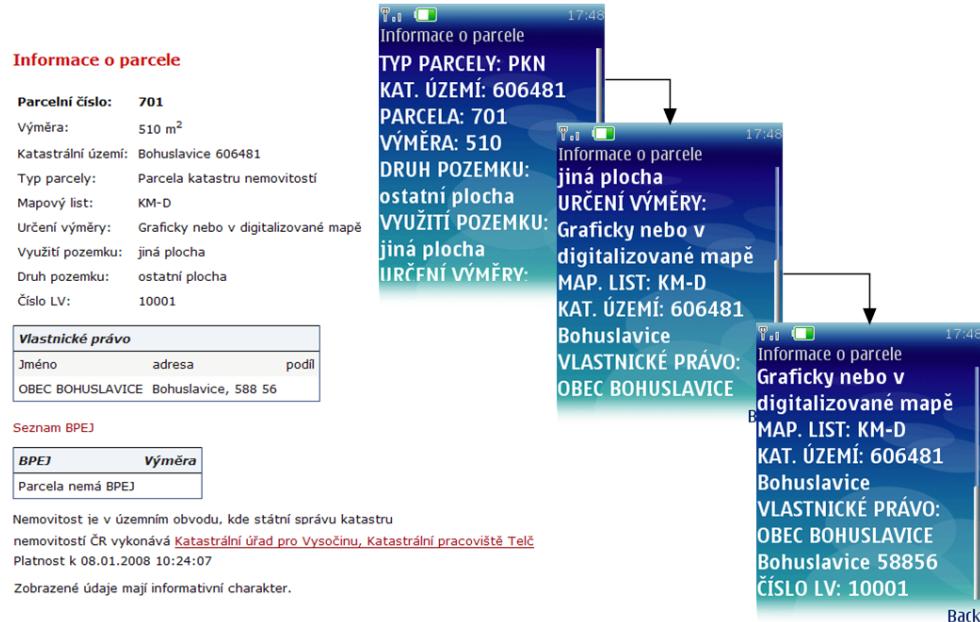
Zadání pro webovou službu – viz obrázek 7.18:

- Katastrální území = 606481 (Bohuslavice)
- Typ parcely = PKN
- Druh číslování = 2
- Kmenové číslo parcely = 701
- Poddělení čísla parcely = není
- Zdroj parcel ZE = není

²²pro vývoj na určité typy mobilních telefonů je nutné mít SDK od výrobce těchto telefonů

Obrázek 7.18: Zadání pro službu *infoOParcele*.

Výsledek webové služby – viz obrázek 7.19.



Obrázek 7.19: Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.

- Služba *infoOBudove*

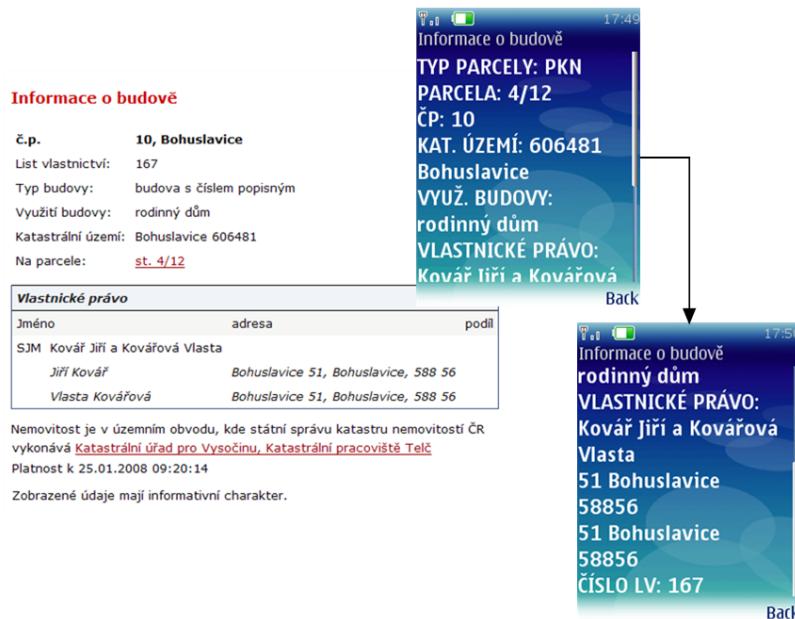
Zadání pro webovou službu – viz obrázek 7.18:

- Část obce kód = 6483 (Bohuslavice)
- Typ budovy = 1 (budova s č.p.)
- Číslo budovy = 10



Obrázek 7.20: Zadání pro službu *infoOBudove*.

Výsledek webové služby – viz obrázek 7.21.



Obrázek 7.21: Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.

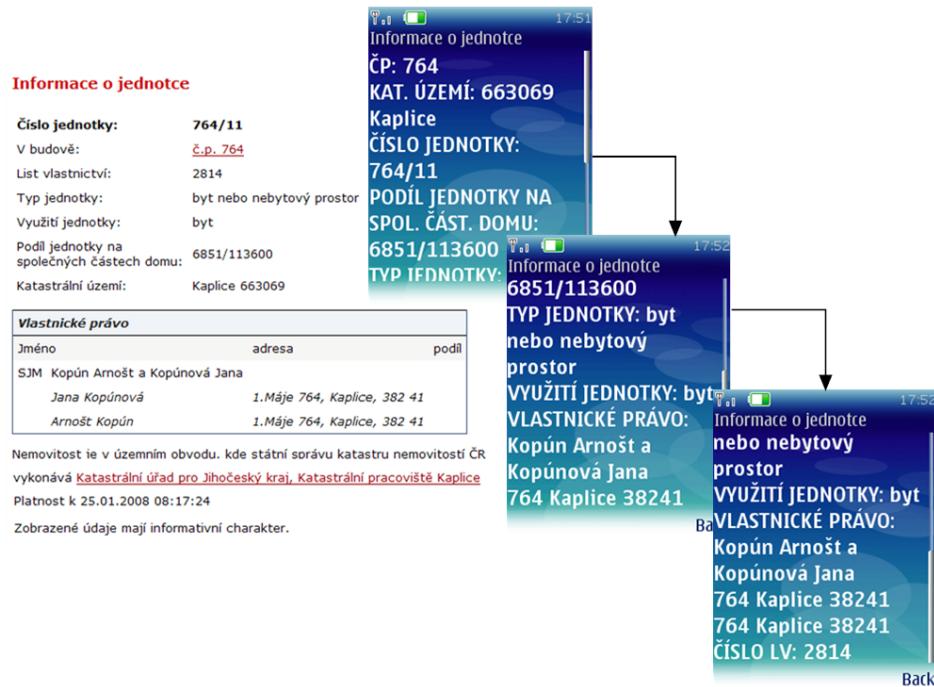
- Služba *infoOJednotce*

Zadání pro webovou službu – viz obrázek 7.22:

- Část obce kód = 403989 (Kaplice)
- Typ budovy = 1 (budova s č.p.)
- Číslo budovy = 764
- Číslo jednotky = 11

Obrázek 7.22: Zadání pro službu *infoOJednotce*.

Výsledek webové služby – viz obrázek 7.23.



Obrázek 7.23: Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.

- Služba *infoORizeni*

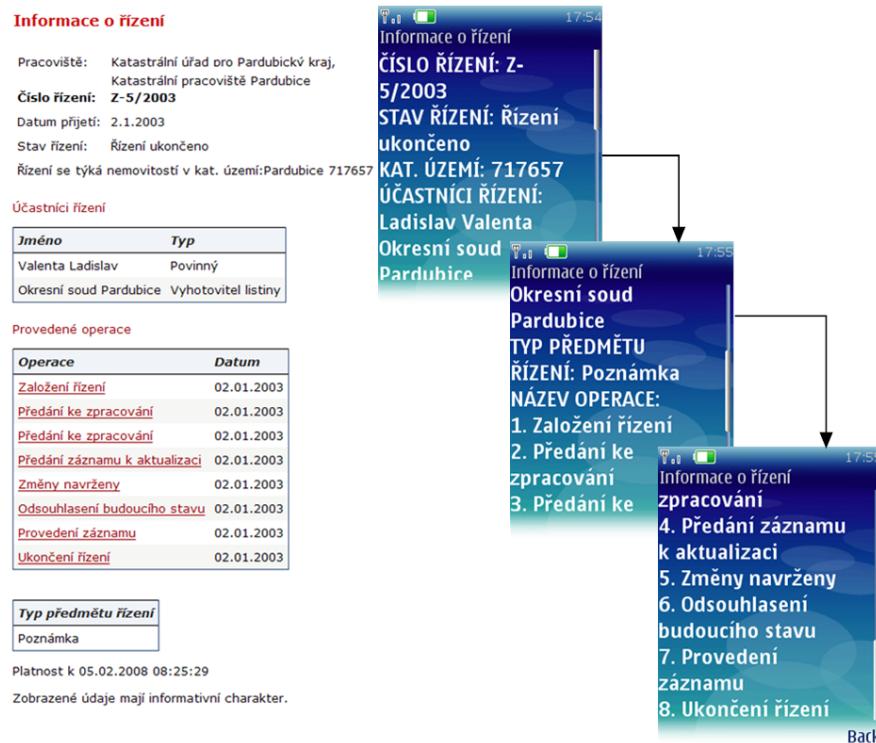
Zadání pro webovou službu – viz obrázek 7.24:

- Kraj = 94 (Pardubický)
- Pracoviště = 606 (Katastrální pracovište Pardubice)
- Typ řízení = Z (Záznam)
- Pořadové číslo = 5
- Rok = 2003



Obrázek 7.24: Zadání pro službu *infoORizeni*.

Výsledek webové služby – viz obrázek 7.25.



Obrázek 7.25: Porovnání výsledků mezi aplikací Nahlížení a ukázkové aplikace.

Zdrojové kódy této aplikace jsou k dispozici na přiloženém CD. Struktura CD je popsána v příloze F.

Kapitola 8

Závěr

V současné době zaznamenávají webové služby velký rozmach, všude se setkáváme s požadavky na jejich zavedení. Mohlo by se zdát, že tato technologie je úplnou novinkou. Ale opak je pravdou. Technologie webových služeb je známa nejméně 5 let. Co se tedy stalo, že najednou zaznamenávají takový zájem?

Především tato technologie „dozrála“. Mnohé technologie, na kterých jsou webové služby založeny, byly od té doby standardizovány. Byly zjištěny jejich silné i slabé stránky a pryč je i doba, kdy byly webové služby bezhlavě používány na všechno možné. Je potřeba vědět, že webové služby se nehodí na vše a nelze je vždy s výhodou použít. Dalším impulsem, který stál za renesancí webových služeb, je nebývalý rozmach kvalitních vývojových nástrojů, které umožňují jejich daleko snazší a efektivnější vývoj. Nejenom že oddělují vývojáře od nízkoúrovňového programování, ale umožňují i velice snadno vytvářet klientské aplikace.

Ukazuje se, že je výhodné poskytovat rozhraní pro webové služby, které bude zprostředkovávat přístup k informacím či různým datům. Potenciální uživatel či zákazník, ale také např. jiný informační systém, může takto definované služby integrovat do vlastních systémů a posléze takto získané informace či data dále libovolně zpracovávat. To má velkou výhodu pro poskytovatele těchto služeb, protože se nemusí starat o velké množství typů výstupních formátů, tak aby vyhověl všem požadavkům klientů, ale může se soustředit na poskytování informací pouze v jednom snadno zpravovatelném formátu. Tato situace je pak výhodná pro obě komunikující strany.

8.1 Vlastní řešení

Cílem této práce je demonstrovat využití webových služeb v katastru nemovitostí. Webové služby v této oblasti nejsou žádnou novinkou. Aplikace „Dálkový přístup“¹ má implementováno rozhraní pro poskytování webových služeb. Tento přístup je zpoplatněn a po-

¹Dálkový přístup
URL: <<https://katastr.cuzk.cz/>>

skytuje výstupy pouze ve formátu PDF². Poskytovaný výstup v PDF je elektronicky podpsán. Tento formát poskytovaných výstupů je pro některé uživatele výhodný, dokonce nezbytný, ale existuje i velká část uživatelů pro které je tento typ formátu nevhodný. Jedná se zejména o uživatele, kteří chtejí získané výstupy (data v něm obsažená) dále automatizovaně zpracovávat. Formát PDF lze velmi těžko automatizovaně zpracovat či analyzovat.

V rámci této práce byly navrženy a realizovány webové služby, které poskytují výstupní formát v jazyce XML. To umožňuje jeho další bezproblémové zpracování. Nově vytvořené webové služby poskytují data a informace v rozsahu, jaký nyní poskytuje aplikace „Nahlížení do KN“³. Poskytování služeb bude zdarma, pouze je potřeba se bezplatně zaregistrovat k odběru služeb.

Výstupní formát v XML je vytvořen z datových bloků Výměnného formátu ISKN (NVF). Použití datových bloků pro popis poskytovaných dat je velmi vhodné, protože tyto datové bloky vyčerpávajícím a dobře definovaným způsobem, popisují objekty (data) v systému ISKN. Pro výstupy z nově vytvářených webových služeb jsou použity jen ty datové bloky, které přímo popisují vydávaná data. Tímto způsobem se však naruší vazby mezi jednotlivými bloky a nedá se dále hovořit o klasickém NVF. Tím tedy vznikne nový formát, který popisuje data pomocí datových bloků NVF, ale neposkytuje všechny vazby, které naopak poskytuje NVF. Kromě výhod uvedených výše, je další výhoda v již existující dokumentaci těchto datových bloků. Uživatel, který již data v NVF odebírá, může velmi snadným porovnáním zjistit (mezi dokumentací k NVF a mezi XML schématem, které popisuje XML výstupy), jaká data může získat pomocí webových služeb zdarma. A tedy jestli potřebuje všechna data, která jsou obsažena v NVF. To samé platí i opačně.

Webové služby jsou roděny do šesti základních skupin, podle typu dat, které budou poskytovat.

1. Skupina služeb poskytuje informace o parcele a tvoří jí následující služby:
 1. *infoOParceleID*,
 2. *infoOParcele*,
 3. *vyhledaniParcel*.
2. Skupina služeb poskytuje informace o budově a tvoří jí následující služby:
 1. *infoOBudoveID*,
 2. *infoOBudove*,
 3. *vyhledaniBudov*,
 4. *vyhledaniBudovParID*.
3. Skupina služeb poskytuje informace o jednotce a tvoří jí následující služby:

²Portable Document Format

URL: <http://en.wikipedia.org/wiki/Portable_Document_Format>

³Nahlížení do KN

URL: <<http://nahlizenidokn.cuzk.cz/>>

1. *infoOJednotceID*,
 2. *infoOJednotce*,
 3. *vyhledaniJednotek*,
 4. *vyhledaniJednotekBudID*.
4. Skupina služeb poskytuje informace o řízení a tvoří jí následující služby:
 1. *infoORizeniID*,
 2. *infoORizeni*.
 5. Skupina služeb umožňuje nalezení neblížšího definičního bodu parcely nebo budovy a tvoří jí následující služby:
 1. *nejblzsiDefBodPar*,
 2. *nejblzsiDefBodBud*.
 6. Skupina služeb poskytuje informace o číselnících a tvoří jí následující služby:
 1. *vratCiselnik*,
 2. *vratSeznamCiselniku*.

Návratový typ všech webových služeb je primitivní datový typ **String**. V současné době (leden 2008) jsou webové služby v testovacím provozu v rámci ČÚZK. Je zejména testován obsah výstupních dat, který je porovnáván s daty získanými z aplikace „Nahlížení do KN“. Dále je systém webových služeb podroben zátěžovému testování, protože je očekáván velmi silný zájem veřejnosti o tyto služby. Služby jsou provozovány na aplikačním serveru GlassFish⁴ od fy. Sun Microsystems⁵.

8.2 Další vývoj

Oznámení, že ČÚZK chystá zprovoznění alternativního přístupu k datům KN pomocí webových služeb, v rozsahu aplikace „Nahlížení do KN“, vzbudilo velký zájem veřejnosti. Ozvalo se několik soukromých firem, které ČÚZK nabízeli pomocnou ruku při závěrečném testování. Velký zájem o tyto služby projevil také systém Czech POINT⁶. Po koordinační schůzce mezi ČÚZK a zástupci Czech POINTu, Czech POINT projevil zájem o vytvoření ještě jedné speciální služby, která bude přístupná jen tomuto systému. Jedná se o funkci,

⁴GlassFish – Open Source Application Server
URL: <<https://glassfish.dev.java.net/>>

⁵Sun Microsystems
URL: <<http://www.sun.com/>>

⁶Czech POINT – Český podací ověřovací informační národní terminál
URL: <<http://www.czechpoint.cz/>>

která bude poskytovat náhled listu vlastnictví (LV) část B⁷. Systém Czech POINT očekává od výstupů z této služby, získání přesnějších informací pro zadání vstupních parametrů pro webové služby aplikace „Dálkový přístup“ (WSDP). Výsledkem služeb WSDP je podepsaný výpis z KN v elektornické formě ve formátu PDF.

Navržený systém webových služeb nemusí poskytovat výstupy jen externím aplikacím, ale může poskytovat tyto služby i interním aplikacím. I o tento přístup byl projeven velký zájem, např. při obnově operátu, kdy je potřeba zjistit informace o parcele ze sousedního katastrálního území než je obnovou operátu dotčené katastrální území.

Systém webových služeb navržených a realizovaných v této práci je navržen tak, aby bylo do budoucna možné a to bez výrazných zásahů, poskytované služby a výstupy dále rozšiřovat podle aktuálních požadavků.

⁷část B–LV obsahuje údaje o nemovitostech
URL:<http://www.cuzk.cz/Dokument.aspx?PRARESKOD=998&MENUID=10007&AKCE=DOC:10-VYSTUPY_Z_KN.PROSTRDP>

Literatura

- [1] BRAY, T., ET AL.: *Extensible Markup Language (XML) 1.0 (Second Edition)* [online] W3C consortium, 2000. URL: <<http://www.w3.org/TR/REC-xml/>>
- [2] BRAY, T., ET AL.: *Namespaces in XML 1.0 (Second Edition)* [online] W3C consortium. URL: <<http://www.w3.org/TR/REC-xml-names/>>
- [3] CHINNICI, R., ET AL.: *WSDL (Web Services Description Language)* [online] W3C consortium, 2007. URL: <<http://www.w3.org/TR/wsdl20/>>
- [4] CLARK, J., MAKOTO, M.: *RELAX NG Specification* [online] OASIS, 2001. URL: <<http://relaxng.org/spec-20011203.html>>
- [5] ČESKÁ INFORMAČNÍ AGENTURA ŽIVOTNÍHO PROSTŘEDÍ (CENIA): *Projekt INSPIRE* [online] CENIA. URL: <<http://www.cenia.cz/inspire/>>
- [6] ČESKÝ ÚŘAD ZEMĚMĚŘICKÝ A KATASTRÁLNÍ (ČÚZK): *Dálkový přístup do KN* [online] ČÚZK. URL: <<https://katastr.cuzk.cz/>>
- [7] ČESKÝ ÚŘAD ZEMĚMĚŘICKÝ A KATASTRÁLNÍ (ČÚZK): *Nahlížení do katastru nemovitostí* [online] ČÚZK. URL: <<http://nahlizenidokn.cuzk.cz/>>
- [8] ČESKÝ ÚŘAD ZEMĚMĚŘICKÝ A KATASTRÁLNÍ (ČÚZK): *Výměnný formát ISKN* [online] ČÚZK. URL: <http://www.cuzk.cz/Dokument.aspx?PRARESKOD=10&MENUID=10015&AKCE=DOC:10-VF_ISKNTEXT>
- [9] CÖMERT, C., AKINCI, H.: *Web Services: An e-Goverment Perspective* [online] URL: <<http://www.fig.net>>.
- [10] DOOB, M.: *Jemný úvod do TeXu* Český překlad - Daneš J., Veselý J., Československé sdružení uživatelů TeXu (CSTUG), Praha, 1993.
- [11] EULIS (European Land Information Service) [online] URL: <<http://www.eulis.org/>>.

- [12] EVROPSKÁ KOMISE: *Směrnice INSPIRE* [online]
URL: <<http://inspire.jrc.it>>
- [13] FONTANA J.: *Vzkříšení modelu distribuovaných aplikací* [seriál online]
týdeník Computerworld, č.5, IDG Czech, a.s., Praha, 2004. ISSN 1210-9924.
URL: <<http://www.computerworld.cz/>>.
- [14] GRAHAM S., ET AL.: *Building Web Services with Java*.
Sams Publishing, Indiana, USA, 2005. ISBN 0-672-32641-8.
- [15] GRANGARD, A., EISENBERG, B., NICKULL, D., ET AL.: *ebXML Technical Architecture Specification v1.0.4* [online]
OASIS, UN/CEFACT, 2001. URL: <<http://www.ebxml.org/specs/ebTA.pdf>>.
- [16] GUDGIN, M., ET AL.: *SOAP (Simple Object Access Protocol)* [online]
W3C consortium. URL: <<http://www.w3.org/TR/soap12-part1/>>.
- [17] GUNZER, H.: *Introduction to Web Services* [online]
white paper, Borland, CA, USA, 2002.
URL: <<http://www.borland.com/>>.
- [18] HAROLD, E.R., MEANS, W.S.: *XML v kostce*.
Computer Press, Praha, 2002. ISBN 80-7226-712-4
- [19] HNOJIL, J.: *Výhody OGC webových služeb pro veřejnou správu*.
Konference Internet ve státní správě a samosprávě[CD-ROM], Hradec Králové, 24.-25.4.2003.
- [20] CHROMÝ, R.: *Úvod do webových služeb*
Juniorstav 2005 – 7. Odborná konference doktorského studia s mezinárodní účastí[CD-ROM]. Vysoké učení technické v Brně, Fakulta stavební, Ústav geodezie, 1. - 2. únor 2005.
- [21] JIROUŠ, V.: *Webové služby v knihovnictví*
Automatizace knihovnických procesů 2005(AKP 2005), 10.ročník semináře, Liberec, 2005.
- [22] KOKEŠ, P.: *Datový model systému ISKN*
Diplomová práce, ČVUT, fa Stavební, 2002.
- [23] KOKEŠ, P.: *Obsah a struktura výměnného formátu ISKN, možnosti využití* [online]
42. Geodetické informační dny, Spolek zeměměřičů Brno, 2006.
URL: <<http://csgk.fce.vutbr.cz/Oakce/A21/>>
- [24] KOSEK, J.: *XML schémata* [online]
URL: <<http://www.kosek.cz/xml/schema/index.html>>

- [25] KUBA, M.: *Web Services* [online]
Ústav výpočetní techniky, Masarykova univerzita.
URL:
<http://www.ics.muni.cz/~makub/soap/MartinKuba_WebServices_Datakon2006_clanek.pdf>
- [26] LE HORS, A., ET AL.: *Document Object Model (DOM) Level 3 Core Specification* [online]
W3C consortium, URL: <<http://www.w3.org/TR/DOM-Level-3-Core/>>
- [27] McGOVERN J., TYAGI S., ET AL.: *Java Web Services Architecture*
Morgan Kaufmann Publishers, San Francisco, 2003. ISBN 1-55860-900-8
- [28] MEGGINSON, D.: *Simple API for XML (SAX)* [online]
URL: <<http://www.saxproject.org/>>
- [29] OLŠÁK, P.: *TeXbook naruby*
Konvoj, Brno, 2001.
- [30] OPEN GIS CONSORCIUM: *Open GIS Consortium* [online]
URL: <<http://www.opengis.org/>>
- [31] ORACLE: *Oracle* [online]
URL: <<http://www.oracle.com/>>
- [32] SUCHÁNEK, V.: *Cestovní zpráva: Setkání účastníků projektu EULIS* [online]
URL:
<http://www.vugtk.cz/odis/sborniky/cest_zpravy/05/31-EULIS_plus_Brusel.pdf>
- [33] SYSTINET CORP.: *Introduction to Web Services Architecture* [online]
Systinet, URL: <<http://www.systinet.org/>>
- [34] VLIST VAN DER, E.: *Relax NG* [online]
O'Reilly & Associates, 2003. ISBN: 0596004214
URL: <<http://books.xmlschemata.org/relaxng/>>
- [35] OASIS STANDARD: *UDDI—Online community for the Universal Description, Discovery, and Integration OASIS Standard* [online]
URL: <<http://uddi.xml.org/>>
- [36] URI PLANNING INTEREST GROUP, W3C-IETF: *URIs, URLs, and URNs: Clarifications and Recommendations 1.0* [online]
URL: <<http://www.w3.org/TR/uri-clarification/>>
- [37] W3C CONSORTIUM: *W3C consortium* [online]
URL: <<http://www.w3.org/>>
- [38] W3C CONSORTIUM: *XML Schema* [online]
URL: <<http://www.w3.org/XML/Schema>>
- [39] ZMEŠKAL, K.: *Systém pro podporu externích uživatelů aplikací ČÚZK*
Diplomová práce, Univerzita Tomáše Bati ve Zlíně, 2007.

Příloha A

Seznam skupin datových bloků VF ISKN

Následující tabulka obsahuje seznam skupin datových bloků výměnného formátu ISKN a datových bloků v jednotlivých skupinách

Skupina		Datový blok			
Jméno	Kód	Kód	Zdrojové tabulky v ISKN	Popis	Aplikace
Nemovitosti	NEMO	PAR	PARCELY PARCELY_M PARCELY_B	Parcely	AK
		BUD	BUDOZY BUDOZY_M BUDOZY_B	Budovy	AK
		CABU	CASTI BUDOV	Části budov	AK
		ZPOCHN*	ZP_OCHRANY_NEM	Číselník způsobů ochrany nemovitosti	SC
		DRUPOZ*	D_POZEMKU	Číselník druhů pozemku	SC
		ZPVYPO*	ZP_VYUZITI_POZ	Číselník způsobů využití pozemku	SC
		ZDPAZE*	ZDROJE_PARCEL_ZE	Číselník zdrojů parcel ZE	SC
		ZPURVY*	ZP_URCENI_VYMERY	Číselník způsobů určení výměry	SC
		TYPBUD*	T_BUDOV	Číselník typů budov	SC
		MAPLIS*	MAPOVE_LISTY	Číselník mapových listů	SC

Příloha A. Seznam skupin datových bloků VF ISKN

Skupina		Datový blok			
Jméno	Kód	Kód	Zdrojové tabulky v ISKN	Popis	Aplikace
	KATUZE*	KATASTR_UZEMI		Číselník katastrálních území	SC
	OBCE*	OBCE		Číselník obcí – vázaně	SC
	CASOBC*	CASTI_OBCI		Číselník částí obce – vázaně	SC
	OKRESY*	OKRESY		Číselník okresů – vázaně	SC
	KRAJE*	KRAJE		Číselník krajů - vázaně	SC
	NKRAJE*	NOVÉ KRAJE		Číselník nových krajů - vázaně	SC
	RZO	R_ZPOCHR R_ZPOCHR_M R_ZPOCHR_B		Přiřazení způsobu ochrany k nemovitostem.	AK
	ZPVYBU*	ZP_VYUZITI_BUD		Způsob využití budov	SC
Jednotky	JEDN	JED	JEDNOTKY JEDNOTKY_M JEDNOTKY_B	Jednotky	AK
	TYPJED*	T_JEDNOTEK		Číselník typů jednotek	SC
	ZPVYJE*	ZP_VYUZITI_JED		Způsob využití jednotek	SC
Bonitní díly parcel	BDPA	BDP	BONIT_DILY_PARC BONIT_DILY_PARC_M BONIT_DILY_PARC_B	Bonitní díly parcel	AK
Vlastnictví	VLST	OPSUB	OPRAV_SUBJEKTY OPRAV_SUBJEKTY_M OPRAV_SUBJEKTY_B	Oprávněné subjekty	AK
	VLA		VLASTNICTVI VLASTNICTVI_M VLASTNICTVI_B	Vlastnictví	AK
	CHAROS*	CHAR_OS		Číselník charakteristik oprávněných subjektů	SC
	TEL		TELESA TELESA_M TELESA_B	Katastrální tělesa	AK

Příloha A. Seznam skupin datových bloků VF ISKN

Skupina		Datový blok			
Jméno	Kód	Kód	Zdrojové tabulky v ISKN	Popis	Aplikace
Jiné právní vztahy	JPVZ	JPV	JINE_PRAV_VZTAHY JINE_PRAV_VZTAHY_M JINE_PRAV_VZTAHY_B	Jiné právní vztahy	AK
		TYPRAV*	T_PRAVNICH_VZT	Číselník typů právních vztahů	SC
Řízení	RIZE	RIZENI*	RIZENI	Řízení (vklad, zá-znam)	PA
		RIZKU*	RIZENI_KU	Vazba Řízení – Katastrální území	PA
		OBJRIZ*	OBJEKTY_RIZENI	Objekty řízení (parcely, budovy,..)	PA
		PRERIZ*	PREDMETY_RIZENI	Předměty řízení	PA
		UCAST*	UCASTNICI	Účastníci řízení	PA
		ADRUC*	ADRESY	Adresy účastníků řízení	PA
		LISTIN*	LISTINY	Listiny	PA
		DUL*	DALSI_UDAJE_LISTINY	Další údaje listin	SC
		LDU*	LISTINY_DALSI_UDAJE	Vazba Listiny – Další údaje listin	PA
		TYPLIS*	T_LISTIN	Číselník typů listin	SC
		TYPPRE*	T_PREDMETU_R	Číselník typů předmětu řízení	SC
		TYPRIZ*	TYPY_RIZENI	Typy řízení	PA
		TYPUCA*	TYPY_UCASTNIKU	Typy účastníků řízení	PA
		UCTYP*	UCASTNICI_TYP	Vazba Účastnící – Typy účastníků řízení	PA
		RL	R_LIST	Přiřazení listin k nemovitostem, vlastnictví a jiným právním vztahům	AK
		OBESMF*	OBESLANI_MF	Obeslání účastníků řízení	PA

Příloha A. Seznam skupin datových bloků VF ISKN

Skupina		Datový blok			
Jméno	Kód	Kód	Zdrojové tabulky v ISKN	Popis	Aplikace
Prvky katastrální mapy	PKMP	SOBR*	BODY_POLOHOPISU SPB_SDOGEOM – sloupec PLATNA = „a“	Souřadnice obrazu bodů polohopisu v mapě	AK
		SBP	SPOJENI_B_POLOH SPOJENI_B_POLOH_M	Spojení bodů polohopisu – definuje polohopisné liniové prvky	AK
		SBM	DPM_SDOGEOM OP_SDOGEOM HBPEJ_SDOGEOM DPM_M_SDOGEOM OP_M_SDOGEOM HBPEJ_M_SDOGEOM	Spojení bodů mapy – definuje nepolohopisné liniové prvky	AK
		KODCHB*	KODY_CHAR_Q_BODU	Číselník kódů charakteristiky kvality bodu	SC
		TYPSOS*	T_SOURAD_SYS	Číselník typů souřadnicových systémů	SC
		HP (PL)	HRANICE_PARCEL HRANICE_PARCEL_M	Hranice parcel	AK
		OP (NL NB)	OBRAZY_PARCEL OP_SDOGEOM OBRAZY_PARCEL_M OP_M_SDOGEOM	Obrazy parcel (parcelní číslo, značka druhu pozemku,...)	AK
		OB (PL NB)	OBRAZY_BUDOV OB_SDOGEOM OBRAZY_BUDOV_M OB_M_SDOGEOM	Obrazy budov (obvod budovy, značka druhu budovy)	AK
		DPM (NB PB NL PL)	DALSI_PRVKY_MAPY DPM_SDOGEOM DALSI_PRVKY_MAPY_M DPM_M_SDOGEOM	Další prvky mapy	AK
		OBBP (PB NB)	OBRAZY_BODU_BP OBBP_SDOGEOM OBRAZY_BODU_BP_M OBBP_M_SDOGEOM	Obrazy bodů BP	AK
		TYPPPD*	T_PRVKU_P_DAT	Číselník typů prvků prostorových dat	SC
		ZVB	ZOBRAZENI_VB	Zobrazení věcných břemen	AK

Příloha A. Seznam skupin datových bloků VF ISKN

Skupina		Datový blok			
Jméno	Kód	Kód	Zdrojové tabulky v ISKN	Popis	Aplikace
		POM	PRVKY_O_MAPY	Prvky orientační mapy	AK
		SPOM	SPOJENI_PO_MAPY	Spojení prvků orientační mapy – definuje liniové prvky.	AK
BPEJ	BPEJ	HBPEJ (NL)	HRANICE_BPEJ HRANICE_BPEJ_M	Hranice BPEJ	AK
		OBPEJ (NB)	OZNACENI_BPEJ OBPEJ_SDOGEOM OZNACENI_BPEJ_M OBPEJ_M_SDOGEOM	Označení BPEJ	AK
Geometrický plán	GMPL* NZ		NAVRHY_ZMEN_KM	Hlavičky geometrických plánů a ostatních změn KM.	AK
		ZPMZ	ZPMZ	Hlavičky ZPMZ	AK
		NZZP	NZ_ZPMZ	Vazební tabulka návrhy změn KM – ZPMZ	AK
		SPOL	BODY_POLOHOPISU SPB_SDOGEOM – sloupec PLATNA = „n“	Souřadnice polohy bodů polohopisu (měřené)	AK
Rezervovaná čísla	REZE*	RECI	REZ_PARCELNI_CISLA	Rezervovaná parcelní čísla	PU
		DOCI	PARCELY	Dotčená parcelní čísla	AK
		DOHICI	ARRPA	Dotčená historická parcelní čísla	AK
		REZBP	REZ_CISLA_PBPP	Rezervovaná čísla bodu PBPP	PU
Definiční body	DEBO	OBDEBO	OBRAZY_DEF_BODU	Obrazy definičních bodů	AK
Adresní místa	ADRM*BUDOBJ		BUD_OBJ	Odkazy objektů na adresy	AK
		ADROBJ	ADRESA	Adresy	SC

NB – nepolohopisný bodový prvek

PB – polohopisný bodový prvek

NL – nepolohopisný liniový prvek

PL – polohopisný liniový prvek

Příloha B

Uživatelské scénáře

B.1 Uživatelé

B.1.1 Registrace

Uživatel (anonym) není registrován a nemůže využívat poskytované služby. Pro registraci systém zobrazí formulář s registračními údaji. Po vyplnění registračních údajů uživatel (anonym) stiskne tlačítko "Registrovat".

Vstupní podmínky: existuje anonym

Aktér: anonym

Spouštěč: anonym zvolí registraci

Cíl: registrovaný uživatel

Hlavní úspěšný scenář:

1. Anonym zvolí registraci
2. Systém zobrazí formulář s registračními údaji
3. Anonym vyplní povinné položky
4. Anonym stiskne tlačítko "Registrovat"
5. Systém zaregistruje registrační údaje, vytvoří uživatelský účet a vygeneruje přístupový klíč – HASH
6. Systém oznámí uživateli, že je zaregistrován a zašle na zaregistrovaný email přístupový klíč

Alternativní scenáře:

*. 1. Anonym může registraci kdykoli před potvrzením přerušit

5.1. 1. Uživatelský účet již existuje

5.1. 2. Systém oznámí anonymovi, že registrace nelze provést

B.1.2 Zrušení registrace

Uživatelský účet může být z různých důvodů zrušen (např. dlouhodobé nevyužívání přístupového klíče, porušení podmínek pro poskytování služeb, atd.).

Vstupní podmínky: existuje uživatelský účet

Aktér: administrátor

Spouštěč: administrátor zvolí zrušení registraci

Cíl: zrušení registrace, uživatel je anonym

Hlavní úspěšný scenář:

1. Administrátor zvolí zrušení registrace
2. Systém zobrazí formulář pro zrušení registrace
3. Administrátor stiskne tlačítko "Zrušit registraci"
4. Systém zruší registraci
5. Systém oznámí uživateli, že jeho účet byl zrušen

Alternativní scenáře:

*. 1. Administrátor může zrušení registrace kdykoli před potvrzením přerušit

B.2 Získání informací o parcele

B.2.1 Informace o parcele – jednoznačná identifikace

Uživatel potřebuje získat informace o parcele a zná její jednoznačnou identifikaci v systému ISKN. Vstupní parametry jsou celkem dva: jednoznačná identifikace parcely a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o parcele

Aktér: uživatel

Spouštěč: uživatel zvolí informace o parcele

Cíl: uživatel obdrží informace o parcele

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o parcele
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o parcele

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
- 4.1. 1. vstupní parametry nejsou správně
- 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
- 5.1. 1. systém nenalezl žádnou parcelu odpovídající zádání
- 5.1. 2. systém oznámí uživateli neexistenci parcely
- 5.1. 3. *Informace o parcele* s jinými parametry

B.2.2 Informace o parcele

Uživatel potřebuje získat informace o parcele a zná skupinu vstupních parametrů, které parcelu v systému ISKN jednoznačně identifikují. Existuje několik vstupních parametrů, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o parcele

Aktér: uživatel

Spouštěč: uživatel zvolí informace o parcele

Cíl: uživatel obdrží informace o parcele

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o parcele
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkontroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o parcele

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 4.1. 1. vstupní parametry nejsou správně
 - 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 5.1. 1. systém nenašel žádnou parcelu odpovídající zádání
 - 5.1. 2. systém oznámí uživateli neexistenci parcely
 - 5.1. 3. *Vyhledání parcel*
 - 5.2. 1. systém nalezl více než jednu parcelu odpovídající zádání
 - 5.2. 2. systém oznámí uživateli, že bylo nalezeno více parcel
 - 5.2. 3. *Informace o parcele* s jinými parametry
 - 5.2. 4. *Vyhledání parcel*

B.2.3 Vyhledání parcel

Uživatel pořebuje získat přesnější identifikační údaje o parcele, ale zná jen některé vstupní parametry a těmto parametrym vyhovuje více parcel v systému ISKN. Uživatel zadá požadované vstupní parametry, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat přesnější identifikační údaje o parcele

Aktér: uživatel

Spouštěč: uživatel zvolí vyhledání parcel

Cíl: uživatel obdrží seznam přesných identifikačních údajů parcel, které vychovují zadaným vstupním parametry

Hlavní úspěšný scenář:

1. Uživatel zvolí vyhledání parcel
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odesle uživateli výsledný XML dokument
7. Uživatel obdrží seznam parcel s přesnými identifikačními údaji

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
- 4.1. 1. vstupní parametry nejsou správně
- 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
- 5.1. 1. systém nenalezl žádné parcely odpovídající zádání
- 5.1. 2. systém oznámí uživateli neexistenci parcel
- 5.1. 3. *Vyhledání parcel* s jinými parametry
- 5.2. 1. systém nenalezl více než deset parcel, které odpovídající zádání
- 5.2. 2. systém oznámí uživateli, že existuje více než deset nalezených parcel
- 5.2. 3. *Vyhledání parcel* s jinými parametry

B.3 Získání informací o budově

B.3.1 Informace o budově – jednoznačná identifikace

Uživatel potřebuje získat informace o budově a zná její jednoznačnou identifikaci v systému ISKN. Vstupní parametry jsou celkem dva: jednoznačná identifikace budovy a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o budově

Aktér: uživatel

Spouštěč: uživatel zvolí informace o budově

Cíl: uživatel obdrží informace o budově

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o budově
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkontroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o budově

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádnou budovu odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci budovy

5.1. 3. *Informace o budově* s jinými parametry

B.3.2 Informace o budově

Uživatel potřebuje získat informace o budově a zná skupinu vstupních parametrů, které budovu v systému ISKN jednoznačně identifikují. Existuje několik vstupních parametrů, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o budově

Aktér: uživatel

Spouštěč: uživatel zvolí informace o budově

Cíl: uživatel obdrží informace o budově

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o budově
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o budově

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 4.1. 1. vstupní parametry nejsou správně
 - 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 5.1. 1. systém nenalezl žádnou budovu odpovídající zádání
 - 5.1. 2. systém oznámí uživateli neexistenci budovy
 - 5.1. 3. *Vyhledání budov*
 - 5.2. 1. systém nalezl více než jednu budovu odpovídající zádání
 - 5.2. 2. systém oznámí uživateli, že bylo nalezeno více budov
 - 5.2. 3. *Informace o budově* s jinými parametry
 - 5.2. 4. *Vyhledání budov*

B.3.3 Vyhledání budov

Uživatel pořebuje získat přesnější identifikační údaje o budově, ale zná jen některé vstupní parametry a těmto parametry vyhovuje více budov v systému ISKN. Uživatel zadá požadované vstupní parametry, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat přesnější identifikační údaje o budově

Aktér: uživatel

Spouštěč: uživatel zvolí vyhledání budov

Cíl: uživatel obdrží seznam přesných identifikačních údajů budov, které vyhovují zadaným vstupním parametry

Hlavní úspěšný scenář:

1. Uživatel zvolí vyhledání budov
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkontroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží seznam budov s přesnými identifikačními údaji

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 4.1. 1. vstupní parametry nejsou správně
 - 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 5.1. 1. systém nenalezl žádné budovy odpovídající zádání
 - 5.1. 2. systém oznámí uživateli neexistenci budov
 - 5.1. 3. *Vyhledání budov* s jinými parametry
 - 5.2. 1. systém nenalezl více než deset budov, které odpovídající zádání
 - 5.2. 2. systém oznámí uživateli, že existuje více než deset nalezených budov
 - 5.2. 3. *Vyhledání budov* s jinými parametry

B.3.4 Vyhledání budov – jedoznačná identifikace parcely

Uživatel pořebuje získat přesnější identifikační údaje o budově, ale zná jen jednoznačnou identifikaci parcely, na které budova leží, v systému ISKN. Uživatel zadá požadovaný vstupní parametr a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat přesnější identifikační údaje o budově

Aktér: uživatel

Spouštěč: uživatel zvolí vyhledání budov

Cíl: uživatel obdrží identifikačních údaje budvy, která vyhovuje zadanému vstupnímu parametru

Hlavní úspěšný scenář:

1. Uživatel zvolí vyhledání budov
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží budovu s přesnými identifikačními údaji

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádnou budovu odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci budovy

5.1. 3. *Vyhledání budov s jinými parametry*

5.1. 4. *Informace o parcele*

B.4 Získání informací o jednotce

B.4.1 Informace o jednotce – jednoznačná identifikace

Uživatel potřebuje získat informace o jednotce a zná její jednoznačnou identifikaci v systému ISKN. Vstupní parametry jsou celkem dva: jednoznačná identifikace jednotky a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o jednotce

Aktér: uživatel

Spouštěč: uživatel zvolí informace o jednotce

Cíl: uživatel obdrží informace o jednotce

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o jednotce
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o jednotce

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádnou jednotku odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci jednotky

5.1. 3. *Informace o jednotce* s jinými parametry

B.4.2 Informace o jednotce

Uživatel potřebuje získat informace o jednotce a zná skupinu vstupních parametrů, které jednotku v systému ISKN jednoznačně identifikují. Existuje několik vstupních parametrů, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o jednotce

Aktér: uživatel

Spouštěc: uživatel zvolí informace o jednotce

Cíl: uživatel obdrží informace o jednotce

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o jednotce
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o jednotce

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 4.1. 1. vstupní parametry nejsou správně
 - 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 5.1. 1. systém nenalezl žádnou jednotku odpovídající zádání
 - 5.1. 2. systém oznámí uživateli neexistenci jednotky
 - 5.1. 3. *Vyhledání jednotek*
 - 5.2. 1. systém nalezl více než jednu jednotku odpovídající zádání
 - 5.2. 2. systém oznámí uživateli, že bylo nalezeno více jednotek
 - 5.2. 3. *Informace o jednotce* s jinými parametry
 - 5.2. 4. *Vyhledání jednotek*

B.4.3 Vyhledání jednotek

Uživatel pořebuje získat přesnější identifikační údaje o jednotce, ale zná jen některé vstupní parametry a těmto parametrym vyhovuje více jednotek v systému ISKN. Uživatel zadá požadované vstupní parametry, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat přesnější identifikační údaje o jednotce

Aktér: uživatel

Spouštěč: uživatel zvolí vyhledání jednotek

Cíl: uživatel obdrží seznam přesných identifikačních údajů jednotek, které vyhovují zadáným vstupním parametrym

Hlavní úspěšný scenář:

1. Uživatel zvolí vyhledání jednotek
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkontroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží seznam jednotek s přesnými identifikačními údaji

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

- 4.1. 1. vstupní parametry nejsou správně
- 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

- 5.1. 1. systém nenalezl žádné jednotky odpovídající zádání
- 5.1. 2. systém oznámí uživateli neexistenci jednotek
- 5.1. 3. *Vyhledání jednotek* s jinými parametry

- 5.2. 1. systém nalezl více než deset jednotek, které odpovídající zádání
- 5.2. 2. systém oznámí uživateli, že existuje více než deset nalezených jednotek
- 5.2. 3. *Vyhledání jednotek* s jinými parametry

B.4.4 Vyhledání jednotek – jednoznačná identifikace budovy

Uživatel pořebuje získat přesnější identifikační údaje o jednotce, ale zná jen jednoznačnou identifikaci budovy, ve které jednotka leží, v systému ISKN. Uživatel zadá požadovaný vstupní parametr a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat přesnější identifikační údaje o jednotce

Aktér: uživatel

Spouštěč: uživatel zvolí vyhledání jednotek

Cíl: uživatel obdrží seznam přesných identifikačních údajů jednotek, které vyhovují zadánému vstupnímu parametru

Hlavní úspěšný scenář:

1. Uživatel zvolí vyhledání jednotek
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží seznam jednotek s přesnými identifikačními údaji

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenašel žádné jednotky odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci jednotek

5.1. 3. *Vyhledání jednotek* s jinými parametry

5.1. 4. *Informace o budově*

5.2. 1. systém nalezl více než deset jednotek, které odpovídající zádání

5.2. 2. systém oznámí uživateli, že existuje více než deset nalezených jednotek

5.2. 3. *Vyhledání jednotek* s jinými parametry

B.5 Získání informací o řízení

B.5.1 Informace o řízení – jednoznačná identifikace

Uživatel potřebuje získat informace o řízení a zná jeho jednoznačnou identifikaci v systému ISKN. Vstupní parametry jsou celkem dva: jednoznačná identifikace řízení a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o řízení

Aktér: uživatel

Spouštěc: uživatel zvolí informace o řízení

Cíl: uživatel obdrží informace o řízení

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o řízení
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkontroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o řízení

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádné řízení odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci řízení

5.1. 3. *Informace o řízení s jinými parametry*

B.5.2 Informace o řízení

Uživatel potřebuje získat informace o řízení a zná skupinu vstupních parametrů, které řízení v systému ISKN jednoznačně identifikují. Existuje několik vstupních parametrů, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o řízení

Aktér: uživatel

Spouštěč: uživatel zvolí informace o řízení

Cíl: uživatel obdrží informace o řízení

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o řízení
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o řízení

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádné řízení odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci řízení

5.1. 3. *Informace o řízení s jinými parametry*

5.2. 1. systém nalezl více než jedno řízení odpovídající zádání

5.2. 2. *Informace o řízení s jinými parametry*

B.6 Nalezení nejbližšího definičního bodu

B.6.1 Nalezení nejbližšího definičního bodu parcely

Uživatel potřebuje získat informace o nejbližším definičním bodu parcely. Všechny vstupní parametry jsou povinné, jedná se o souřadnice výchozího bodu pro hledání, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o nejbližším definičním bodu parcely

Aktér: uživatel

Spouštěč: uživatel zvolí nalezení nejbližšího definičního bodu parcely

Cíl: uživatel obdrží informace o nejbližším definičním bodu parcely

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o nejbližším definičním bodu parcely
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o řízení

Alternativní scenáře:

- 3.1. 1. vstupní parametry nejsou formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 4.1. 1. vstupní parametry nejsou správně
 - 4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů
 - 5.1. 1. systém nenalezl žádný definiční bod parcely v systémem daném rozsahu
 - 5.1. 2. systém oznámí uživateli neexistenci definičního bodu v určitém rozsahu od zadaného výchozího bodu
 - 5.1. 3. *Nalezení nejbližšího definičního bodu parcely s jinými parametry*

B.6.2 Nalezení nejbližšího definičního bodu budovy

Uživatel potřebuje získat informace o nejbližším definičním bodu budovy. Všechny vstupní parametry jsou povinné, jedná se o souřadnice výchozího bodu pro hledání, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace o nejbližším definičním bodu budovy

Aktér: uživatel

Spouštěč: uživatel zvolí nalezení nejbližšího definičního bodu budovy

Cíl: uživatel obdrží informace o nejbližším definičním bodu budovy

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o nejbližším definičním bodu budovy
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o nejbližším definičním bodu budovy

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádný definiční bod budovy v systémem daném rozsahu

5.1. 2. systém oznámí uživateli neexistenci definičního bodu v určitém rozsahu od zadaného výchozího bodu

5.1. 3. *Nalezení nejbližšího definičního bodu budovy s jinými parametry*

B.7 Získání informací o číselnících

B.7.1 Seznam číselníků

Uživatel potřebuje získat seznam dostupných číselníků. Vstupní parametrem bude pouze platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat seznam dostupných číselníků

Aktér: uživatel

Spouštěč: uživatel zvolí seznam číselníků

Cíl: uživatel obdrží seznam dostupných číselníků

Hlavní úspěšný scenář:

1. Uživatel zvolí seznam číselníků
2. Uživatel vyplní povinný vstupní parametr
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží seznam číselníků

Alternativní scenáře:

- 3.1. 1. vstupní parametr není formálně správně
- 3.1. 2. systém oznámí uživateli chybu v zadání vstupního parametru
- 4.1. 1. vstupní parametr není správně
- 4.1. 2. systém oznámí uživateli chybu v zadání vstupního parametru

B.7.2 Informace o číselníku

Uživatel potřebuje získat informace o číselníku (hodnoty číselníku) a zná skupinu vstupních parametrů, které číselník v systému ISKN jednoznačně identifikuje. Dále vstupní parametry fungují jako filtr pro výběr hodnot z číselníku. Existuje několik vstupních parametrů, některé jsou nepovinné, a platný přístupový klíč.

Vstupní podmínky: uživatel potřebuje získat informace z číselníku

Aktér: uživatel

Spouštěč: uživatel zvolí informace o číselníku

Cíl: uživatel obdrží informace o číselníku

Hlavní úspěšný scenář:

1. Uživatel zvolí informace o číselníku
2. Uživatel vyplní povinné vstupní parametry
3. Systém zkонтroluje formální správnost parametrů
4. Systém porovná uložené a získané registrační údaje
5. Systém spustí výsledný dotaz (uložený dotaz + vstupní parametry)
6. Systém odešle uživateli výsledný XML dokument
7. Uživatel obdrží informaci o číselníku (hodnoty číselníku)

Alternativní scenáře:

3.1. 1. vstupní parametry nejsou formálně správně

3.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

4.1. 1. vstupní parametry nejsou správně

4.1. 2. systém oznámí uživateli chybu v zadání vstupních parametrů

5.1. 1. systém nenalezl žádný číselník odpovídající zádání

5.1. 2. systém oznámí uživateli neexistenci číselníku

5.1. 3. *Informace o číselníku* s jinými parametry

5.1. 4. *Seznam číselníků*

5.2. 1. systém nenalezl žádné hodnoty v číselníku, které odpovídají zádání

5.2. 2. systém oznámí uživateli nenalezení hodnot v číselníku

5.2. 3. *Informace o číselníku* s jinými parametry

Příloha C

XML schéma výstupního formátu webových služeb

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:xmlns:nahlichenidokn.cuzk.cz:wsvkiskn"
            xmlns="urn:xmlns:nahlichenidokn.cuzk.cz:wsvkiskn"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            attributeFormDefault="qualified"
            elementFormDefault="qualified">
    <xs:element name="wsvkiskn">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="hlavicka" minOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="platnost" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:attribute name="od" form="unqualified" type="xs:dateTime" use="optional"/>
                                    <xs:attribute name="do" form="unqualified" type="xs:dateTime" use="optional" />
                            </xs:complexType>
                        </xs:element>
                    <xs:element name="datoveskupiny" minOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="skupina" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="verze" form="unqualified" type="xs:string" use="optional"/>
                <xs:attribute name="vytvoreno" form="unqualified" type="xs:string" use="optional"/>
                <xs:attribute name="vytvoril" form="unqualified" type="xs:
```

```
        string" use="optional"/>
    <xs:attribute name="zmeny" form="unqualified" type="xs:string"
      " use="optional"/>
    <xs:attribute name="navrhy" form="unqualified" type="xs:
      string" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="vyberovapodminka" type="xs:string" minOccurs="0
  "/>
<xs:element name="data" minOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="par" minOccurs="0" maxOccurs="unbounded"
        >
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1"
              />
            <xs:element name="par_type" type="xs:string"
              minOccurs="0" />
            <xs:element name="katuze_kod" type="xs:int"
              minOccurs="0" />
            <xs:element name="kmenove_cislo_par" type="xs:int"
              minOccurs="1" />
            <xs:element name="zdpaze_kod" type="xs:string"
              minOccurs="0" />
            <xs:element name="poddeleni_cisla_par" type="xs:
              string" minOccurs="0" />
            <xs:element name="maplis_kod" type="xs:string"
              minOccurs="0" />
            <xs:element name="zpurvy_kod" type="xs:string"
              minOccurs="0" />
            <xs:element name="drupoz_kod" type="xs:string"
              minOccurs="0" />
            <xs:element name="zpvypa_kod" type="xs:string"
              minOccurs="0" />
            <xs:element name="vymera_parcely" type="xs:string"
              minOccurs="0" />
            <xs:element name="tel_id" type="xs:long" minOccurs=
              "0" />
            <xs:element name="bud_id" type="xs:long" minOccurs=
              "0" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    <xs:element name="bud" minOccurs="0" maxOccurs="unbounded"
      ">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="id" type="xs:long" minOccurs="1"
            />
          <xs:element name="typbud_kod" type="xs:int"
            minOccurs="0" />
          <xs:element name="caobce_kod" type="xs:int"
```

```
        minOccurs="0" />
    <xs:element name="cislo_domovni" type="xs:int"
        minOccurs="0" />
    <xs:element name="zpvbybu_kod" type="xs:string"
        minOccurs="0" />
    <xs:element name="tel_id" type="xs:long" minOccurs=
        "0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="drupoz" minOccurs="0" maxOccurs="
    unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:string" minOccurs="
                1" />
            <xs:element name="nazev" type="xs:string" minOccurs
                ="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="zpvypo" minOccurs="0" maxOccurs="
    unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
                />
            <xs:element name="nazev" type="xs:string" minOccurs
                ="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="zdpaze" minOccurs="0" maxOccurs="
    unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
                />
            <xs:element name="nazev" type="xs:string" minOccurs
                ="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="zpurvy" minOccurs="0" maxOccurs="
    unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
                />
            <xs:element name="nazev" type="xs:string" minOccurs
                ="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<xs:element name="typbud" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:int" minOccurs="1" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="maplis" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="id" type="xs:long" minOccurs="1" />
<xs:element name="oznaceni_mapoveho_listu" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="katuze" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:int" minOccurs="1" />
<xs:element name="obce_kod" type="xs:int" minOccurs="0" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="obce" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:int" minOccurs="1" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="casobc" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:int" minOccurs="1" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
```

```
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="zpvybu" minOccurs="0" maxOccurs=""
    unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
/>
            <xs:element name="nazev" type="xs:string" minOccurs
= "0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="jed" minOccurs="0" maxOccurs="unbounded
">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1"
/>
            <xs:element name="bud_id" type="xs:long" minOccurs=
"0" />
            <xs:element name="typjed_kod" type="xs:string"
minOccurs="0" />
            <xs:element name="cislo_jednotky" type="xs:string"
minOccurs="0" />
            <xs:element name="zpvyje_kod" type="xs:string"
minOccurs="0" />
            <xs:element name="tel_id" type="xs:long" minOccurs=
"0" />
            <xs:element name="podil_citatel" type="xs:string"
minOccurs="0" />
            <xs:element name="podil_jmenovatel" type="xs:string
" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="typjed" minOccurs="0" maxOccurs="unbounded
">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
/>
            <xs:element name="nazev" type="xs:string" minOccurs
= "0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="zpvyje" minOccurs="0" maxOccurs="unbounded
">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kod" type="xs:int" minOccurs="1"
/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<xs:element name="nazev" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="bdp" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="par_id" type="xs:long" minOccurs="0" />
<xs:element name="bpej_kod" type="xs:string" minOccurs="0" />
<xs:element name="vymera" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="opsub" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="id" type="xs:long" minOccurs="1" />
<xs:element name="opsub_type" type="xs:string" minOccurs="0" />
<xs:element name="id_je_1_partner_bsm" type="xs:string" minOccurs="0" />
<xs:element name="id_je_2_partner_bsm" type="xs:string" minOccurs="0" />
<xs:element name="ico" type="xs:long" minOccurs="0" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
<xs:element name="jmeno" type="xs:string" minOccurs="0" />
<xs:element name="prijmeni" type="xs:string" minOccurs="0" />
<xs:element name="nazev_ulice" type="xs:string" minOccurs="0" />
<xs:element name="cislo_domovni" type="xs:string" minOccurs="0" />
<xs:element name="cislo_orientacni" type="xs:string" minOccurs="0" />
<xs:element name="obec" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="vla" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="id" type="xs:long" minOccurs="1" />
```

```
        />
    <xs:element name="opsub_id" type="xs:long"
        minOccurs="0" />
    <xs:element name="tel_id" type="xs:long" minOccurs=
        "0" />
    <xs:element name="podil_citatel" type="xs:string"
        minOccurs="0" />
    <xs:element name="podil_jmenovatel" type="xs:string"
        " minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="tel" minOccurs="0" maxOccurs="unbounded"
    ">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1"
                />
            <xs:element name="cislo_tel" type="xs:string"
                minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="rizeni" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1"
                />
            <xs:element name="typriz_kod" type="xs:string"
                minOccurs="0" />
            <xs:element name="poradove_cislo" type="xs:string"
                minOccurs="0" />
            <xs:element name="rok" type="xs:dateTime" minOccurs
                ="0" />
            <xs:element name="stav" type="xs:string" minOccurs=
                "0" />
            <xs:element name="funkce_kod" type="xs:string"
                minOccurs="0" />
            <xs:element name="typope_kod" type="xs:string"
                minOccurs="0" />
            <xs:element name="funkce_kod_vyznamna" type="xs:
                string" minOccurs="0" />
            <xs:element name="typope_kod_vyznamna" type="xs:
                string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="rizku" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="katuze_kod" type="xs:int"
                minOccurs="1" />
```

```
<xs:element name="rizeni_id" type="xs:long"
    minOccurs="1" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="preriz" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="rizeni_id" type="xs:long"
    minOccurs="1" />
<xs:element name="typpre_kod" type="xs:string"
    minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ucast" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="id" type="xs:long" minOccurs="1"
    />
<xs:element name="rizeni_id" type="xs:long"
    minOccurs="0" />
<xs:element name="druh_ucastnika" type="xs:string"
    minOccurs="0" />
<xs:element name="obchodni_jmeno" type="xs:string"
    minOccurs="0" />
<xs:element name="ico" type="xs:string" minOccurs="0"
    />
<xs:element name="jmeno" type="xs:string" minOccurs="0"
    />
<xs:element name="prijmeni" type="xs:string"
    minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="typpre" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:int" minOccurs="1"
    />
<xs:element name="nazev" type="xs:string" minOccurs="0"
    />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="typriz" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:string" minOccurs="1"
    />
```

```
<xs:element name="nazev" type="xs:string" minOccurs="0" />
<xs:element name="popis" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="typuca" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="kod" type="xs:string" minOccurs="1" />
<xs:element name="nazev" type="xs:string" minOccurs="0" />
<xs:element name="popis" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="uctyp" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="ucast_id" type="xs:long" minOccurs="1" />
<xs:element name="typuca_kod" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="operace" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="nazev_operace" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="datum_provedeni_operace" type="xs:dateTime" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="obdebo" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="datum_vzniku" type="xs:dateTime" minOccurs="0" />
<xs:element name="par_id" type="xs:long" minOccurs="0" />
<xs:element name="bud_id" type="xs:long" minOccurs="0" />
<xs:element name="souradnice_y" type="xs:string"
```

```
        minOccurs="0" />
    <xs:element name="souradnice_x" type="xs:string"
        minOccurs="0" />
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Příloha D

WSDL – popis webových služeb

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://wsnahlizeni.cuzk.cz/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://wsnahlizeni.cuzk.cz/"
    name="WSNahlizeniService">

<types>
    <xs:schema xmlns:tns="http://wsnahlizeni.cuzk.cz/" xmlns:xs="http://
        www.w3.org/2001/XMLSchema" version="1.1" targetNamespace="http://
        wsnahlizeni.cuzk.cz/">
        <xs:element name="infoOBudove" type="tns:infoOBudove"/>
        <xs:element name="infoOBudoveResponse" type="tns:
            infoOBudoveResponse"/>
        <xs:element name="infoOBudoveID" type="tns:infoOBudoveID"/>
        <xs:element name="infoOBudoveIDResponse" type="tns:
            infoOBudoveIDResponse"/>
        <xs:element name="info0Jednotce" type="tns:info0Jednotce"/>
        <xs:element name="info0JednotceResponse" type="tns:
            info0JednotceResponse"/>
        <xs:element name="info0JednotceID" type="tns:info0JednotceID"/>
        <xs:element name="info0JednotceIDResponse" type="tns:
            info0JednotceIDResponse"/>
        <xs:element name="info0Parcele" type="tns:info0Parcele"/>
        <xs:element name="info0ParceleResponse" type="tns:
            info0ParceleResponse"/>
        <xs:element name="info0ParceleID" type="tns:info0ParceleID"/>
        <xs:element name="info0ParceleIDResponse" type="tns:
            info0ParceleIDResponse"/>
        <xs:element name="info0Rizeni" type="tns:info0Rizeni"/>
        <xs:element name="info0RizeniResponse" type="tns:
            info0RizeniResponse"/>
        <xs:element name="info0RizeniID" type="tns:info0RizeniID"/>
        <xs:element name="info0RizeniIDResponse" type="tns:
            info0RizeniIDResponse"/>

```

```
    infoORizeniIDResponse" />
<xs:element name="nejblizsiDefBodBud" type="tns:nejblizsiDefBodBud"
/>
<xs:element name="nejblizsiDefBodBudResponse" type="tns:
nejblizsiDefBodBudResponse" />
<xs:element name="nejblizsiDefBodPar" type="tns:nejblizsiDefBodPar"
/>
<xs:element name="nejblizsiDefBodParResponse" type="tns:
nejblizsiDefBodParResponse" />
<xs:element name="vratCiselnik" type="tns:vratCiselnik" />
<xs:element name="vratCiselnikResponse" type="tns:
vratCiselnikResponse" />
<xs:element name="vratSeznamCiselniku" type="tns:
vratSeznamCiselniku" />
<xs:element name="vratSeznamCiselnikuResponse" type="tns:
vratSeznamCiselnikuResponse" />
<xs:element name="vyhledaniBudov" type="tns:vyhledaniBudov" />
<xs:element name="vyhledaniBudovResponse" type="tns:
vyhledaniBudovResponse" />
<xs:element name="vyhledaniBudovParID" type="tns:
vyhledaniBudovParID" />
<xs:element name="vyhledaniBudovParIDResponse" type="tns:
vyhledaniBudovParIDResponse" />
<xs:element name="vyhledaniJednotek" type="tns:vyhledaniJednotek" />
<xs:element name="vyhledaniJednotekResponse" type="tns:
vyhledaniJednotekResponse" />
<xs:element name="vyhledaniJednotekBudID" type="tns:
vyhledaniJednotekBudID" />
<xs:element name="vyhledaniJednotekBudIDResponse" type="tns:
vyhledaniJednotekBudIDResponse" />
<xs:element name="vyhledaniParcel" type="tns:vyhledaniParcel" />
<xs:element name="vyhledaniParcelResponse" type="tns:
vyhledaniParcelResponse" />
<xs:complexType name="nejblizsiDefBodPar">
<xs:sequence>
<xs:element name="y" type="xs:double" />
<xs:element name="x" type="xs:double" />
<xs:element name="hash" type="xs:string" minOccurs="1" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="nejblizsiDefBodParResponse">
<xs:sequence>
<xs:element name="return" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="infoORizeniID">
<xs:sequence>
<xs:element name="rizID" type="xs:long" />
<xs:element name="hash" type="xs:string" minOccurs="1" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="infoORizeniIDResponse">
<xs:sequence>
<xs:element name="return" type="xs:string" minOccurs="0" />
```

```
</xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniParcel">
  <xs:sequence>
    <xs:element name="katuzeKod" type="xs:int"/>
    <xs:element name="typParcely" type="xs:string" minOccurs="0"/>
    <xs:element name="druhCislovani" type="xs:string" minOccurs="0"
      />
    <xs:element name="kmenoveCislo" type="xs:int"/>
    <xs:element name="poddeleni" type="xs:string" minOccurs="0"/>
    <xs:element name="zdrojZE" type="xs:string" minOccurs="0"/>
    <xs:element name="hash" type="xs:string" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniParcelResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="vratSeznamCiselniku">
  <xs:sequence>
    <xs:element name="hash" type="xs:string" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="vratSeznamCiselnikuResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="infoORizeni">
  <xs:sequence>
    <xs:element name="krajKod" type="xs:int"/>
    <xs:element name="pracKod" type="xs:int"/>
    <xs:element name="typRizKod" type="xs:string" minOccurs="0"/>
    <xs:element name="poradoveCislo" type="xs:int"/>
    <xs:element name="rokRiz" type="xs:int"/>
    <xs:element name="hash" type="xs:string" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="infoORizeniResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="infoOBudove">
  <xs:sequence>
    <xs:element name="castObcKod" type="xs:int"/>
    <xs:element name="typBudKod" type="xs:string" minOccurs="0"/>
    <xs:element name="cisloBud" type="xs:int"/>
    <xs:element name="hash" type="xs:string" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="infoOBudoveResponse">
  <xs:sequence>
```

```
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniJednotek">
    <xs:sequence>
        <xs:element name="castObcKod" type="xs:int"/>
        <xs:element name="typBudKod" type="xs:string" minOccurs="0"/>
        <xs:element name="cisloBud" type="xs:int"/>
        <xs:element name="cisloJed" type="xs:string" minOccurs="0"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniJednotekResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniBudovParID">
    <xs:sequence>
        <xs:element name="parID" type="xs:long"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniBudovParIDResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="nejblizsiDefBodBud">
    <xs:sequence>
        <xs:element name="y" type="xs:double"/>
        <xs:element name="x" type="xs:double"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="nejblizsiDefBodBudResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniBudov">
    <xs:sequence>
        <xs:element name="castObcKod" type="xs:int"/>
        <xs:element name="cisloBud" type="xs:int"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniBudovResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="infoOParcele">
    <xs:sequence>
```

```
<xs:element name="katuzeKod" type="xs:int"/>
<xs:element name="typParcely" type="xs:string" minOccurs="0"/>
<xs:element name="druhCislovani" type="xs:string" minOccurs="0"
    />
<xs:element name="kmenoveCislo" type="xs:int"/>
<xs:element name="poddeleni" type="xs:string" minOccurs="0"/>
<xs:element name="zdrojZE" type="xs:string" minOccurs="0"/>
<xs:element name="hash" type="xs:string" minOccurs="1"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="info0ParceleResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0Jednotce">
    <xs:sequence>
        <xs:element name="castObcKod" type="xs:int"/>
        <xs:element name="typBudKod" type="xs:string" minOccurs="0"/>
        <xs:element name="cisloBud" type="xs:int"/>
        <xs:element name="cisloJed" type="xs:string" minOccurs="0"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0JednotceResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniJednotekBudID">
    <xs:sequence>
        <xs:element name="budID" type="xs:long"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vyhledaniJednotekBudIDResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="vratCiselnik">
    <xs:sequence>
        <xs:element name="nazevCiselniku" type="xs:string" minOccurs="1"
            "/>
        <xs:element name="kodHodnotyCiselniku" type="xs:string"
            minOccurs="0"/>
        <xs:element name="datum" type="xs:string" minOccurs="0"/>
        <xs:element name="nazevPolozky" type="xs:string" minOccurs="0"/
            >
            <xs:element name="hash" type="xs:string" minOccurs="1"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="vratCiselnikResponse">
    <xs:sequence>
```

```
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0BudoveID">
    <xs:sequence>
        <xs:element name="budID" type="xs:long"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0BudoveIDResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0JednotceID">
    <xs:sequence>
        <xs:element name="jedID" type="xs:long"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0JednotceIDResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0ParceleID">
    <xs:sequence>
        <xs:element name="parID" type="xs:long"/>
        <xs:element name="hash" type="xs:string" minOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="info0ParceleIDResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
</types>
<message name="info0ParceleID">
    <part name="parameters" element="tns:info0ParceleID"/>
</message>
<message name="info0ParceleIDResponse">
    <part name="parameters" element="tns:info0ParceleIDResponse"/>
</message>
<message name="info0Parcele">
    <part name="parameters" element="tns:info0Parcele"/>
</message>
<message name="info0ParceleResponse">
    <part name="parameters" element="tns:info0ParceleResponse"/>
</message>
<message name="vyhledaniParcel">
    <part name="parameters" element="tns:vyhledaniParcel"/>
</message>
<message name="vyhledaniParcelResponse">
```

```
<part name="parameters" element="tns:vyhledaniParcelResponse"/>
</message>
<message name="info0BudoveID">
    <part name="parameters" element="tns:info0BudoveID"/>
</message>
<message name="info0BudoveIDResponse">
    <part name="parameters" element="tns:info0BudoveIDResponse"/>
</message>
<message name="info0Budove">
    <part name="parameters" element="tns:info0Budove"/>
</message>
<message name="info0BudoveResponse">
    <part name="parameters" element="tns:info0BudoveResponse"/>
</message>
<message name="vyhledaniBudov">
    <part name="parameters" element="tns:vyhledaniBudov"/>
</message>
<message name="vyhledaniBudovResponse">
    <part name="parameters" element="tns:vyhledaniBudovResponse"/>
</message>
<message name="vyhledaniBudovParID">
    <part name="parameters" element="tns:vyhledaniBudovParID"/>
</message>
<message name="vyhledaniBudovParIDResponse">
    <part name="parameters" element="tns:vyhledaniBudovParIDResponse"/>
</message>
<message name="info0JednotceID">
    <part name="parameters" element="tns:info0JednotceID"/>
</message>
<message name="info0JednotceIDResponse">
    <part name="parameters" element="tns:info0JednotceIDResponse"/>
</message>
<message name="info0Jednotce">
    <part name="parameters" element="tns:info0Jednotce"/>
</message>
<message name="info0JednotceResponse">
    <part name="parameters" element="tns:info0JednotceResponse"/>
</message>
<message name="vyhledaniJednotek">
    <part name="parameters" element="tns:vyhledaniJednotek"/>
</message>
<message name="vyhledaniJednotekResponse">
    <part name="parameters" element="tns:vyhledaniJednotekResponse"/>
</message>
<message name="vyhledaniJednotekBudID">
    <part name="parameters" element="tns:vyhledaniJednotekBudID"/>
</message>
<message name="vyhledaniJednotekBudIDResponse">
    <part name="parameters" element="tns:vyhledaniJednotekBudIDResponse"/>
</message>
<message name="info0RizeniID">
    <part name="parameters" element="tns:info0RizeniID"/>
</message>
```

```
<message name="infoORizeniIDResponse">
    <part name="parameters" element="tns:infoORizeniIDResponse"/>
</message>
<message name="infoORizeni">
    <part name="parameters" element="tns:infoORizeni"/>
</message>
<message name="infoORizeniResponse">
    <part name="parameters" element="tns:infoORizeniResponse"/>
</message>
<message name="vratSeznamCiselniku">
    <part name="parameters" element="tns:vratSeznamCiselniku"/>
</message>
<message name="vratSeznamCiselnikuResponse">
    <part name="parameters" element="tns:vratSeznamCiselnikuResponse"/>
</message>
<message name="vratCiselnik">
    <part name="parameters" element="tns:vratCiselnik"/>
</message>
<message name="vratCiselnikResponse">
    <part name="parameters" element="tns:vratCiselnikResponse"/>
</message>
<message name="nejblizsiDefBodPar">
    <part name="parameters" element="tns:nejblizsiDefBodPar"/>
</message>
<message name="nejblizsiDefBodParResponse">
    <part name="parameters" element="tns:nejblizsiDefBodParResponse"/>
</message>
<message name="nejblizsiDefBodBud">
    <part name="parameters" element="tns:nejblizsiDefBodBud"/>
</message>
<message name="nejblizsiDefBodBudResponse">
    <part name="parameters" element="tns:nejblizsiDefBodBudResponse"/>
</message>
<portType name="WSNahizeniService">
    <operation name="infoOParceleID">
        <input message="tns:infoOParceleID"/>
        <output message="tns:infoOParceleIDResponse"/>
    </operation>
    <operation name="infoOParcele">
        <input message="tns:infoOParcele"/>
        <output message="tns:infoOParceleResponse"/>
    </operation>
    <operation name="vyhledaniParcel">
        <input message="tns:vyhledaniParcel"/>
        <output message="tns:vyhledaniParcelResponse"/>
    </operation>
    <operation name="infoOBudoveID">
        <input message="tns:infoOBudoveID"/>
        <output message="tns:infoOBudoveIDResponse"/>
    </operation>
    <operation name="infoOBudove">
        <input message="tns:infoOBudove"/>
        <output message="tns:infoOBudoveResponse"/>
    </operation>
```

```
<operation name="vyhledaniBudov">
    <input message="tns:vyhledaniBudov"/>
    <output message="tns:vyhledaniBudovResponse"/>
</operation>
<operation name="vyhledaniBudovParID">
    <input message="tns:vyhledaniBudovParID"/>
    <output message="tns:vyhledaniBudovParIDResponse"/>
</operation>
<operation name="info0JednotceID">
    <input message="tns:info0JednotceID"/>
    <output message="tns:info0JednotceIDResponse"/>
</operation>
<operation name="info0Jednotce">
    <input message="tns:info0Jednotce"/>
    <output message="tns:info0JednotceResponse"/>
</operation>
<operation name="vyhledaniJednotek">
    <input message="tns:vyhledaniJednotek"/>
    <output message="tns:vyhledaniJednotekResponse"/>
</operation>
<operation name="vyhledaniJednotekBudID">
    <input message="tns:vyhledaniJednotekBudID"/>
    <output message="tns:vyhledaniJednotekBudIDResponse"/>
</operation>
<operation name="info0RizeniID">
    <input message="tns:info0RizeniID"/>
    <output message="tns:info0RizeniIDResponse"/>
</operation>
<operation name="info0Rizeni">
    <input message="tns:info0Rizeni"/>
    <output message="tns:info0RizeniResponse"/>
</operation>
<operation name="vratSeznamCiselniku">
    <input message="tns:vratSeznamCiselniku"/>
    <output message="tns:vratSeznamCiselnikuResponse"/>
</operation>
<operation name="vratCiselnik">
    <input message="tns:vratCiselnik"/>
    <output message="tns:vratCiselnikResponse"/>
</operation>
<operation name="nejblizsiDefBodPar">
    <input message="tns:nejblizsiDefBodPar"/>
    <output message="tns:nejblizsiDefBodParResponse"/>
</operation>
<operation name="nejblizsiDefBodBud">
    <input message="tns:nejblizsiDefBodBud"/>
    <output message="tns:nejblizsiDefBodBudResponse"/>
</operation>
</portType>
<binding name="WSNahlicheniServicePortBinding" type="tns:
    WSNahlicheniService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style=
        "document"/>
    <operation name="info0ParceleID">
```

```
<soap:operation soapAction="" />
<input>
    <soap:body use="literal" />
</input>
<output>
    <soap:body use="literal" />
</output>
</operation>
<operation name="infoOParcele">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vyhledaniParcel">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="info0BudoveID">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="info0Budove">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vyhledaniBudov">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vyhledaniBudovParID">
```

```
<soap:operation soapAction="" />
<input>
    <soap:body use="literal" />
</input>
<output>
    <soap:body use="literal" />
</output>
</operation>
<operation name="info0JednotceID">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="info0Jednotce">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vyhledaniJednotek">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vyhledaniJednotekBudID">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="info0RizeniID">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="info0Rizeni">
```

```
<soap:operation soapAction="" />
<input>
    <soap:body use="literal" />
</input>
<output>
    <soap:body use="literal" />
</output>
</operation>
<operation name="vratSeznamCiselniku">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="vratCiselnik">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="nejblizsiDefBodPar">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
<operation name="nejblizsiDefBodBud">
    <soap:operation soapAction="" />
    <input>
        <soap:body use="literal" />
    </input>
    <output>
        <soap:body use="literal" />
    </output>
</operation>
</binding>
<service name="WSNahizeniService">
    <port name="WSNahizeniServicePort" binding="tns:
        WSNahizeniServicePortBinding">
        <soap:address location="http://a200002:81/WebAppNahizeni/
            WSNahizeniService" />
    </port>
</service>
</definitions>
```

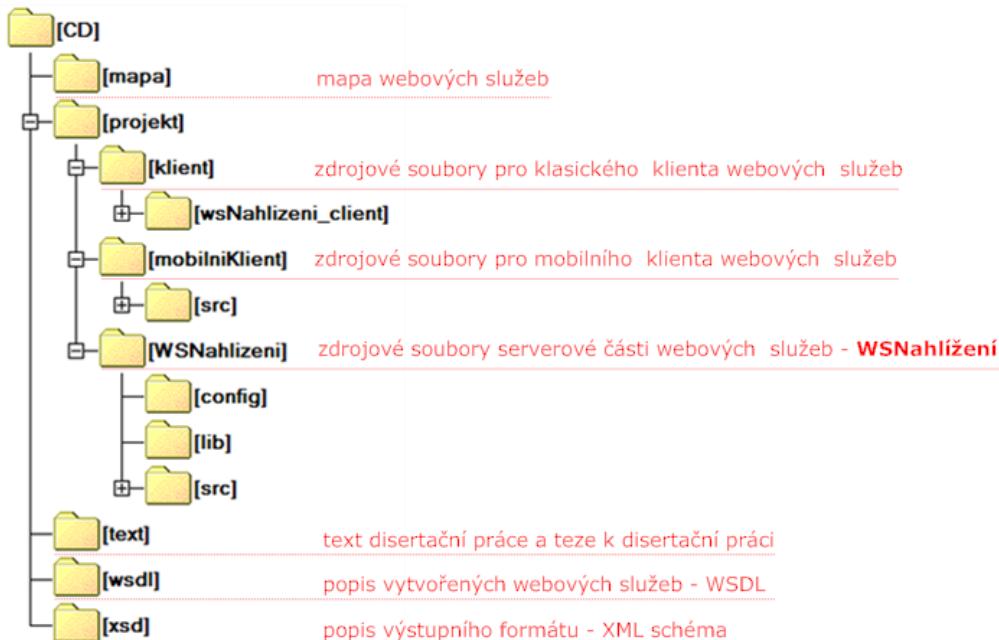
Příloha E

Mapa webových služeb

Příloha F

Obsah přiloženého CD

Struktura adresářů je následující:



V adresáři projekt jsou uloženy všechny zdrojové kódy, které byly během tohoto projektu vytvořeny. WSDL je vytvořeno pro testovací prostředí (ale plně funkční), v době psaní této práce ještě nebylo rozhodnuto o konečném umístění webových služeb.