

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program : Aplikovaná informatika

Obor: Informační systémy a technologie

ANALÝZA A ŘEŠENÍ SYSTÉMU PRO MONITORING ISIR A VYBRANÝCH REGISTRŮ ARES

DIPLOMOVÁ PRÁCE

Student : Jan Faltis

Vedoucí : Doc. Ing. Jan Pour, CSc.

Oponent : Ing. Lucie Poludvorná

2012

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 2. května 2012

.....

Jméno a příjmení studenta

Poděkování

Rád bych poděkoval Doc. Ing. Janu Pourovi, CSc. za cenné rady, připomínky, trpělivost a za odborné vedení při psaní mé diplomové práce.

Abstrakt

Diplomová práce se zabývá analýzou požadavků, návrhem a implementací Systému pro monitoring insolvenčního rejstříku a obchodního rejstříku ze systému administrativního registru ekonomických subjektů. Systém je navržen tak, aby bylo možné přidávat následně podle poptávky další zdroje pro monitoring.

Práce definuje vlastní metodiku pro tvorbu Systému pro malou Společnost s důrazem na pochopitelnost navrženého řešení zákazníkem.

V práci nejprve vždy analyzuji požadavky na Systém, následně formuluji několik modelů vždy s nižší úrovní abstrakce a na základě těchto modelů vytvořím implementaci Systému pomocí několika funkčních přírůstků.

Klíčová slova

ISIR, ARES, databáze, monitoring, analýza, implementace, model, metodika, .NET, proces, webová služba

Abstract

This diploma thesis deals with the analysis of the requirements, design and implementation of the System for monitoring of insolvency register and commercial register from administrative register of economical subjects.

The system has an extendable design which allows further sources to be added at a later date to suit the client's needs.

The design process is oriented at small business customers and emphasises the need for simplicity and understandability of the marketed solution.

System requirements are analysed first, subsequently, several models with decreasing levels of abstraction are formulated and the System is then implemented, according to these models, using several functional increments.

Keywords

ISIR, ARES, database, monitoring, analysis, implementation, model, method, .NET, process, web service

Obsah

1. Úvod.....	7
1.1. Vymezení tématu práce a důvod výběru tématu	7
1.2. Cíle práce a způsob jejich dosažení.....	7
1.3. Způsob/metoda dosažení cíle	7
1.4. Předpoklady a omezení práce	7
1.5. Struktura práce	7
1.6. Výstupy práce a očekávané přínosy	8
2. Charakteristika oblasti.....	8
2.1.1. Charakteristika oblasti správy pohledávek	8
2.1.2. Charakteristika oblasti správy pohledávek ve Společnosti.....	8
3. Komentovaná řešená prací	9
3.1. Principy a modely řízení podnikové informatiky [6]	9
3.2. Extrémní programování [7]	9
3.3. Metodiky budování informačních systémů [8].....	10
3.4. Podniková informatika [9].....	10
3.5. Návrhové vzory [10]	10
3.6. Metodika vývoje IS s pomocí nástroje Power Designer [11].....	10
4. Metodika práce.....	12
4.1. Analýza a návrh.....	13
4.1.1. Analýza/model obsahový	14
4.1.2. Analýza/model technologický	15
4.1.3. Analýza/model implementační	16
4.2. Implementace	16
4.3. Zavedení do provozu	16
4.4. Provoz a užití.....	16
5. Analýza/model obsahový ISIR	16
5.1. Analýza Základních požadavků Společnosti.....	16
5.2. Analýza ISIR	17

5.2.1.	Definice ISIR a jeho obsahu.....	17
5.2.2.	Popis možností přístupu do ISIR.....	18
5.3.	Analýza požadavků společnosti na navržené řešení.....	19
5.3.1.	Datové úložiště.....	20
5.3.2.	Aktualizační program.....	20
5.3.3.	Monitorovací program.....	20
5.3.4.	Webová služba prohledávání.....	21
5.4.	Tvorba Demo verzí.....	21
5.4.1.	Aktualizační program.....	21
5.4.2.	Monitorovací program.....	22
5.5.	Specifikace entit.....	23
5.6.	Analýza možností ISIR.....	24
5.6.1.	Základní popis.....	25
5.6.2.	Věc.....	25
5.6.3.	Událost a akce.....	25
5.6.4.	Poznámka.....	25
5.6.5.	Poznámky a události/akce.....	25
5.6.6.	Typické akce.....	26
5.6.7.	Datový model.....	27
5.6.8.	Volání webové služby.....	27
5.6.9.	Rozhraní webové služby.....	27
5.6.10.	Poznámka.....	27
5.6.11.	Událost.....	28
5.6.12.	Věc.....	29
5.6.13.	Osoba.....	30
5.6.14.	Adresa.....	32
5.7.	Návrh datové základny.....	32
6.	Analýza/model technologický ISIR.....	34
6.1.	Architektura Systému.....	34

6.1.1.	Lehký klient a tlustý server	34
6.1.2.	Tlustý klient a lehký server	35
6.2.	Úložiště dat.....	36
6.2.1.	Relační databáze.....	36
6.2.2.	Objektová databáze	37
6.2.3.	Relačně-objektová databáze	37
6.2.4.	Volba technologie.....	37
6.3.	Webová služba prohledávání.....	37
6.3.1.	Volba technologie.....	38
6.3.2.	Detailní popis webové služby.....	38
6.4.	Aktualizační program	42
6.4.1.	Platforma .NET	42
6.4.2.	Platforma Java	43
6.4.3.	Volba technologie.....	43
6.4.4.	Detailní popis aktualizačního programu.....	43
6.5.	Monitorovací program.....	50
6.5.1.	Volba technologie.....	51
6.5.2.	Detailní popis monitorovacího programu.....	51
7.	Analýza/model implementační ISIR	55
7.1.	Úložiště dat.....	56
7.1.1.	Kritéria rozhodování.....	56
7.1.2.	Microsoft SQL Server	56
7.1.3.	MySQL	56
7.1.4.	Volba úložiště.....	56
7.2.	Fyzický datový model	57
7.2.1.	Obhajoba řešení fyzického modelu	57
7.3.	Aktualizační a monitorovací program	58
7.4.	Volba konkrétního programovacího jazyka	59
8.	Implementace ISIR.....	59

8.1.	Přírůstek 1	59
8.1.1.	Tvorba databáze.....	59
8.1.2.	Tvorba aktualizacího programu.....	60
8.1.3.	Testování prvního přírůstku	61
8.1.4.	Nasazení prvního přírůstku.....	62
8.1.5.	Výstupy prvního přírůstku.....	63
8.2.	Přírůstek 2.....	63
8.2.1.	Tvorba webové služby vyhledávání	63
8.2.2.	Testování druhého přírůstku.....	65
8.2.3.	Výstupy druhého přírůstku	65
8.3.	Přírůstek 3.....	65
8.3.1.	Tvorba monitorovacího programu.....	65
8.3.2.	Interní testování třetího přírůstku	66
8.3.3.	Akceptační testování přírůstku u třetí strany.....	67
8.3.4.	Nasazení programu monitoringu	68
8.3.5.	Výstupy třetího přírůstku.....	68
9.	Provoz a užití ISIR	68
10.	Analýza/model obsahový ARES	68
10.1.	Analýza základních požadavků Společnosti.....	68
10.2.	Analýza ARES	69
10.2.1.	Definice ARES.....	69
10.2.2.	Přístup ke zdrojům ARES	69
10.2.3.	Definice rozhraní ARES.....	70
10.3.	Návrh řešení monitoringu ARES.....	70
10.3.1.	Využití databáze	71
10.3.2.	Návrh úpravy monitorovacího programu	72
11.	Analýza/model technologický ARES.....	73
11.1.	Architektura Systému	73
11.2.	Monitorovací program.....	74

11.2.1.	Komunikace se službou ARES.....	74
11.2.2.	Procedurální popis funkcionality.....	74
11.2.3.	Model tříd monitorovacího programu	75
12.	Analýza/model implementační ARES.....	77
13.	Implementace ARES	77
13.1.	Přírůstek 4.....	77
13.1.1.	Úprava databáze	78
13.1.2.	Úprava monitorovacího programu.....	78
13.1.3.	Interní testování čtvrtého přírůstku	78
13.1.4.	Akceptační testování čtvrtého přírůstku u třetí strany	79
14.	Provoz a užití upraveného Systému	79
15.	Závěr.....	80
16.	Terminologický slovník	80
17.	Seznam Literatury	81
18.	Seznam obrázků	86
19.	Seznam tabulek.....	87

1. ÚVOD

1.1. Vymezení tématu práce a důvod výběru tématu

Tématem diplomové práce je analýza požadavků, návrh a implementace systému pro monitoring Insolvenčního rejstříku, dále jen „ISIR“, a vybraných částí Administrativního registru ekonomických subjektu, dále jen „ARES“. Konkrétní vyvíjený informační systém bude dále označován jako „Systém“, jeho komponenty potom podle svého účelu. Práce je prováděna pro malou společnost, která vyvíjí informační systémy pro správu pohledávek a tyto potom prodává dalším společnostem. Specifikace Systému budou určeny zástupcem Společnosti v průběhu tvorby Systému. Pro téma práce jsem se rozhodl z několika důvodů. Prvním je potřeba daného Systému ve Společnosti a zároveň i možnost prodeje Systému třetím stranám, tedy užitečnost provedené práce a zisk. Druhým je možnost vypracování celého Systému, který spolupracuje se systémy státní správy a zabývá se správou ve finančním sektoru, což je obor, kterému se již od počátku svého studia na VŠE věnuji.

1.2. Cíle práce a způsob jejich dosažení

Cílem práce je analyzovat potřeby Společnosti a získat specifikaci funkcionality od Systému požadované. Poté podle specifikace vytvořit návrh a následně implementaci Systému. Mým přínosem při řešení těchto cílů je analýza situace, kdy v závislosti na konzultacích se zástupcem Společnosti vytvořím funkcionální model a jeho popis, následné zvolení vhodného způsobu implementace a samotná implementace Systému.

1.3. Způsob/metoda dosažení cíle

Metodiku zpracování podrobně rozebírám v kapitole Metodika práce.

1.4. Předpoklady a omezení práce

Systém je omezen především po stránce finanční, jeho provoz nesmí být finančně náročný, a tedy není možné využívat drahé technologické, nebo implementační řešení, kromě řešení odpovídajících standardům Společnosti. Systém je dále omezen dobou odezvy při prohledávání, která nesmí být významně dlouhá (určeno Společností).

1.5. Struktura práce

Práce je strukturována tak, že nejprve se věnuji metodice použité pro vývoj Systému a následně podle této metodiky postupuji. Tedy práce začíná definicí metodiky, následně analýze požadavků na monitoring ISIR, jejich implementací, analýze požadavků na monitoring ARES a končí opět implementací.

1.6. Výstupy práce a očekávané přínosy

Výstupem práce bude především Systém pro monitoring ISIR/ARES, tyto výstupy potom budou určeny k užívání třetími stranami, které mají zájem na monitoringu těchto zdrojů, tedy většinou Společnosti zaměřené na správu pohledávek a případně Společnosti se zájmem na sledování změn různých subjektů v ARES.

Očekávaným přínosem je finanční přínos z prodeje Systému dalším společnostem. Vedlejším přínosem potom bude možná použitelnost práce jako referenčního materiálu při vývoji Systémů pro malé Společnosti.

2. CHARAKTERISTIKA OBLASTI

Charakteristiku jsem rozdělil na dvě části. V první se věnuji popisu stavu oblasti správy pohledávek, pro kterou je řešení vyvíjeno. Ve druhé části popisují stav panující se Společnosti a případně v dalších relevantních společnostech.

2.1.1. Charakteristika oblasti správy pohledávek

Společnost vyvíjí a prodává informační systém pro správu pohledávek. Systém je vyvíjen pro další společnosti, které se zabývají správou a vymáháním pohledávek. Pohledávky jsou právem věřitele na plnění od dlužníka [1]. Pohledávky v rámci systému procházejí několika stavy, které vycházejí ze zákona určujícího postup řešení pohledávek [2]. V průběhu správy pohledávek se může stát, že je na stejného dlužníka vyhlášeno insolvenční řízení, což ovlivňuje průběh správy takové pohledávky. Z toho důvodu společnosti zabývající se správou pohledávek chtějí znát, zdali již dlužník není veden v jiném insolvenčním řízení. Společnost tak chce rozvinout své stávající řešení o monitoring ISIR a zároveň poskytnout i jen monitoring rejstříku pro společnosti s vlastním systémem správy pohledávek. V průběhu analýzy potřeby a plánování systému pro monitoring ISIR vyvstala potřeba pro monitoring vybraných rejstříků ARES (především obchodní rejstřík). Potřeba vyvstala od jiné společnosti, pro kterou Společnost systém správy pohledávek vyvíjí a která by ráda vyhledávala údaje dodatečné údaje o ekonomických subjektech přímo v prostředí správy pohledávek. Společnost se pak rozhodla zařadit i monitoring ARES.

2.1.2. Charakteristika oblasti správy pohledávek ve Společnosti

V současné době Společnost nabízí vlastní systém pro správu pohledávek, který je využíván dalšími společnostmi. Systém spravuje pohledávky dlužníků a umožňuje jejich výpisy, elektronickou komunikaci s vybranými exekutorskými úřady a elektronickou komunikaci se soudy pomocí služeb Justice [3], především elektronický platební rozkaz. Společnost neumí monitorovat existenci dlužníka v ISIR a ráda by tuto funkcionalitu získala, protože tato znalost (tedy insolvenční řízení) může silně ovlivnit celou správu pohledávky. Vzhledem k tomu, že je tedy nutno monitorovat ISIR, by Společnost ráda umožnila i monitorování vybraných registrů ARES, aby zvýšila svou znalost

případného dlužníka pohledávky. Řešení pomocí jiného systému, který by monitoring umožňoval, Společnost zavrhl z důvodů bezpečnosti, nákladnosti a záměru nabídnout vlastní komplexní řešení oblasti.

3. KOMENTOVANÁ REŠERŠE PRACÍ

Rešerše prací takto specifického tématu je poměrně náročná z toho důvodu, že v České Republice sice existuje několik řešení pro monitoring ISIR, ale tato řešení jsou dodávána jako aplikace, bez práce dokumentující jejich vývoj, navíc velmi často neumožňují automatické zpracování velkého množství dotazů. Při pokusech o nalezení takovýchto prací jsem neuspěl ani s poměrně jednoduchým dotazem "aplikace ISIR ARES", nebo "vývoj ISIR ARES" ve vyhledávací službě Google Scholar [4], nebo jen Google [5]. Jediným použitelným odkazem mimo povětšinou reklamních nabídek je paradoxně reference mé vlastní práce v systému ISIS. V zahraničí se bohužel tématu také nikdo nevěnuje.

Pokud bych tedy změnil zadání na obecnější vývoj aplikace, lze najít enormně velké množství prací, které se mu věnují. Prakticky každá učebnice programování se věnuje vývoji aplikací. Existuje nepřeberné množství metodik a postupů, které by se daly zmínit. Proto jsem se rozhodl použít především práce, se kterými jsem se setkal v průběhu studia na VŠE a práce, které jsem používal ve své vlastní praxi. Práce pokrývají velkou část vývoje aplikací a to jak z popisu aplikací, metodik jejich návrhu i samotné tvorby, nasazení a použití. Tyto práce pak také používám ve vlastní diplomové práci.

3.1. Principy a modely řízení podnikové informatiky [6]

Publikace se zabývá velmi zešíroka podnikovou informatikou. Pro mě důležité jsou především principy návrhu a tvorby analytických modelů, ze kterých čerpám při formulaci metody, kterou dospěji k cíli diplomové práce. Publikace je dále skvělým zdrojem termínů a definicí.

3.2. Extrémní programování [7]

Publikace se zabývá praktikami extrémního programování, tedy vývoje aplikace ve velmi malém týmu. Rozhodl jsem se použít některé praktiky z toho důvodu, že vývoj mé práce probíhá prakticky jen v mé vlastní režii s konzultacemi a zpětnou vazbou od Společnosti, případně třetích stran. Při tomto způsobu vývoje jsem se snažil maximalizovat míru kontrol nad aplikací (především pomocí jednotkových testů), abych minimalizoval šanci na výskyt chyb v aplikaci způsobených úpravami. Pro další snížení chybné funkcionality využívám i akceptační testování zákazníkem (který je v extrémním programování prakticky dalším členem týmu) a pro případ že by nějaká chyba přeci jen zůstala nepovšimnuta, tak využívám velmi krátké iterace, přírůstky, které dovolují rychlou reakci a opravu funkcionality.

3.3. Metodiky budování informačních systémů [8]

Práce rozebírá různé metodiky pro správu a vývoj informačních systémů a následně formuluje metodický rámec pro výběr vhodné metodiky pro realizovaný projekt. Pro mou diplomovou práci je zdroj důležitý především proto, že velmi přehledně uvádí základní seznam metodik i s jejich výhodami a nevýhodami a poskytuje vhodný zdroj definic informatických termínů.

3.4. Podniková informatika [9]

Tato publikace se zabývá obecně prakticky celou problematikou podnikové informatiky. Pro mou práci nejdůležitější částí je potom část "Životní cyklus aplikací podnikové informatiky", která definuje obecný proces, metodu, kterou se vyvíjejí aplikace. Tuto metodu jsem částečně přebíral (viz. kapitola 4) a samozřejmě upravil pro své účely. Druhou kapitolou, ze které jsem se rozhodl čerpat je potom "Datové modelování a návrh databází". Publikaci jsem použil především proto, že problematiku definuje na velmi vhodné úrovni, dostatečně obecně a zároveň vyčerpávajícím způsobem, který se snaží postihnout všechny varianty. Tedy není problém si upravit potřebné části pro své použití (případně vynechat), ale nehrozí opomenutí kritických faktorů. Mnou vyvíjená aplikace potom spadá do podnikové informatiky a proto jsou pro mě metody v publikaci popsány použitelné. Dle mého názoru se pak jedná o prakticky nejlepší popis podnikové informatiky jako celku, se kterým se setkal.

3.5. Návrhové vzory [10]

Práce se zabývá rozбором technik objektového programování, především pak návrhovými vzory, které definuje jako "Návrhové vzory mají v programování podobný význam jako vzorce v matematice: zrychlují řešení a zestručňují a zkvalitňují komunikaci", nebo "vzorečky, které použiješ při návrhu architektury budoucí aplikace". Vedle návrhových vzorů potom rozebírá a zdůvodňuje použití zásad objektového programování. Použití jak vzorů, tak dodržování zásad se mi při tvorbě aplikací již mnohokrát velmi vyplatilo, především pak při změnách aplikací, kdy vhodná architektura a řešení aplikace umožňuje změnit, či rozšířit jejich funkcionalitu bez velkých obtíží, které nastávaly bez jejich použití. Dále seznam těchto vzorů umožňuje jejich velmi rychlé použití, bez nutnosti delší analýzy a rozboru různých návrhů na jedné straně a vyšší míru standardizace na straně druhé, kdy většina programátorů dokáže identifikovat návrhový vzor rychleji, než alternativní řešení. Práci jsem se potom rozhodl použít také z důvodu její formy rozhovoru studenta s pedagogem, která poměrně přístupnou formou vysvětluje i náročné abstraktní problémy. Ačkoliv práce používá pro příklady jazyk JAVA, její principy jsou použitelné pro všechny objektově orientované jazyky.

3.6. Metodika vývoje IS s pomocí nástroje Power Designer [11]

Práci jsem použil především proto, že popisuje podobně jako [9] proces vývoje aplikací. To, že práce využívá nástroj Power Designer nebrání použití i za využití jiných nástrojů podpory návrhu. Navíc se práce zabývá i teoretickými aspekty vývoje a tak v ní popsané části metodiky používám i ve

své práci, především metodu tří architektur, která vede ke snížení nákladů na změnu funkcionality v průběhu životního cyklu aplikace. Používám i některé prostředky pro popis funkcionality aplikace, ale u těchto se potom snažím spíše o pochopitelnost neinformatickými zákazníky.

4. METODIKA PRÁCE

V práci budu postupovat podle vlastní metodiky, kterou jsem odvodil především z těchto publikací [9], [11], [8], [6]. Metodiku v této práci chápu jako „jasný způsob uspořádání myšlenek a činů“ [12] a dále jako „doporučený souhrn etap, přístupů, zásad, postupů, pravidel, dokumentu, řízení, metod, technik a nástrojů pro tvůrce informačních systému, který pokrývá celý životní cyklus informačních systému. Určuje kdo, kdy, co a proč má dělat během vývoje a provozu IS“ [11].

Důvod, proč specifikuji metodiku vlastní je ten, že se snažím nalézt ideální stav popisu a řízení vývoje aplikace v prostředí zvolené Společnosti. Použití některých z klasických metodik jako RUP, EUP, UP, MSF, OPEN/EML jsem vyloučil pro nevhodnost použití „těžké“ metodiky v malém vývojovém týmu, viz [8] „Obecně je použití klasických metodik vhodné spíše pro střední, a velké projekty, které jsou náročné po technické anebo organizační (manažerské) stránce. Naopak tyto metodiky nebývají vhodné pro použití u malých vývojových týmů.“ V [8] je potom dále stručný popis agilních metodik vývoje následovaný principy a hodnotami definovanými v Manifestu agilního vývoje software převzatými z [13], které dávají přednost:

- individualitám a komunikaci před procesy a nástroji
- fungujícímu softwaru před podrobnou dokumentací
- spolupráci se zákazníkem před sjednáváním kontraktu
- reakci na změnu před plněním plánu

Rozhodl jsem se tedy pro principy agilního vývoje, ale i metodiky agilní předpokládají určitou minimální velikost týmu, kterou já nenaplňuji a tak jejich zásady např. párové programování, nebo společné vlastnictví kódu [7] nejsou použitelné, navíc tyto definují povětšinou jen přístup k tvorbě aplikace, kdežto já bych rád definoval etapy a činnosti. Přebírám tak spíše jejich zásady, které se snažím maximálně naplnit. Navíc v [11] je uvedeno „žádnou metodiku nelze nikdy použít tak jak je, tedy jako kuchařku“ a dále „Samotné použití metodiky pak vyžaduje její implementaci v konkrétních znalostních podmínkách dané organizace, dotvoření do konkrétní podoby vnitřního předpisu a stylu práce“. V této části tedy definuji postup, metody a přístupy průběhu tvorby aplikace, podle které se následně budu řídit.

V publikacích [9], [11] si můžeme všimnout, že metodika tvorby IS je zde odvozena od životního cyklu aplikace. V [9] je to zdůvodněno odvozením od ITIL (Information Technology Infrastructure Library) [ITIL], který chápe proces řízení rozvoje aplikace jako její životní cyklus. Fáze životního cyklu aplikace se pak v [9] dělí takto:

- Plánování a příprava
- Analýza a návrh
- Implementace

- Zavedení do provozu
- Provoz a užití
- Rozvoj a optimalizace

Každou fázi potom detailněji popisuje a specifikuje konkrétní činnosti ve fázi obvykle prováděné. Já se budu věnovat jen některým fázím, protože pohled v [9] i [11] se zabývá řízením a rozvojem v rámci celého podniku, zatímco já řeším jen jednu relativně izolovanou aplikaci, kterou Společnost hodlá nabízet v rámci svých služeb. Metodiku potom také musím uzpůsobit proto, že sice budu v častém kontaktu se zástupcem Společnosti, ale nebudu od něj získávat žádné dokumenty, nebo přesné specifikace. Zástupce silně preferuje systém, kdy v rámci častých schůzek já předkládám možná řešení a on jen vyjadřuje názor na tato řešení, tedy detailní specifikace budu dělat já a jen je zástupci předkládat a případně upravovat. Pro tvorbu aplikace jsme tak spolu se zástupcem Společnosti zvolili iterační model [8] tvorby systému. Dále tato práce není vedena čistě z pohledu Společnosti, ale spíše z pohledu dodavatele, který pro Společnost aplikaci vyvíjí. Tedy některé části v práci nemají smysl. Fáze, kterými se chci v práci zabývat, jsou:

- Analýza a návrh
- Implementace
- Zavedení do provozu
- Provoz a užití

V otázce vlastního přístupu k tvorbě Systému potom budu vycházet z některých principů extrémního programování [7], [8]. Metodiku, ze které chci některé přístupy vybrat, jsem zvolil především na základě [8]. Nejprve jsem se rozhodl pro agilní metodiky, ale jelikož i tyto „lehčí“ metodiky obvykle předpokládají více pracovníků vývoje a tedy neodpovídají mému stavu, rozhodl jsem se jen přebrat určité dle mého názoru důležité principy. Samotný výběr metodiky by mohl být tématem na delší rozsah, ale rozhodl jsem se mu zde moc nevěnovat, protože se chci zaměřit spíše na vlastní téma práce. Pro více informací a de-facto zdůvodnění výběru doporučuji výše uvedené publikace.

Dále uvádím detailní popis všech fází, činnosti, ze kterých se budou fáze skládat, jejich smysl a výstupy.

4.1. Analýza a návrh

Jelikož fáze pokrývá poměrně velkou část práce, předpokládám její další dělení, které uvádím níže. Výstupem fáze by měl být návrh řešení, který odpovídá jak možnostem systémů ISIR a ARES na jedné straně, tak požadavkům Společnosti na druhé. Ve fázi bych rád vyšel především z modelu tří architektur, tak jak je popsán v [11]. Tedy nejprve vytvořím obsahový popis/analýzu toho, **co** by měl systém umět, následně popis/analýzu technologickou která uvádí, **jak** bude systém vytvořen a na závěr pak část implementační která specifikuje, **čím** bude dosaženo cíle. Postup je na stejném principu jako

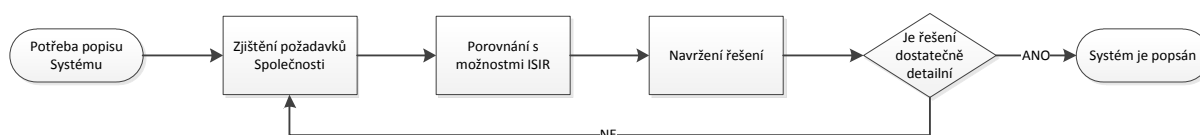
datové modelování uvedené v [9], jen jsem se ho rozhodl rozšířit na celou aplikaci. Navíc podobný přístup můžeme dále najít v obvyklých návrhových vzorech pro tvorbu objektově orientovaných aplikací např. [10], nebo [15]. Důvody pro toto rozhodnutí jsou popsány ve výše uvedených publikacích a zejména to jsou:

"Každá vyšší úroveň je zcela nezávislá na úrovni nižší, tedy popis obsahu systému je nezávislý jak na technologické úrovni, tak implementační. Technologické řešení je pak nezávislé na implementačním. To zaručuje volbu nejvhodnějšího řešení na každé úrovni. Stejný popis obsahu lze případně řešit jinými technologickými, ale i implementačními metodami, tedy v případě nutného předělání systému zůstává vždy maximum vykonané práce. A zároveň nutná změna na nižší úrovni (např. implementační) neovlivní popis obsahu aplikace."

V [11] je pak uvedeno „potřeba takového rozlišení modelu není dána jenom potřebou rozdělit specifikaci systému na etapy, ale také a především "technologickou" potřebou nezávislosti specifikace systému na použité koncepci organizace dat a implementačním prostředí, která umožňuje klasifikovat činnosti údržby a rozvoje systému na úpravy vyplývající ze změn v reálném světě (okolí systému) a změn v realizačním prostředí.“

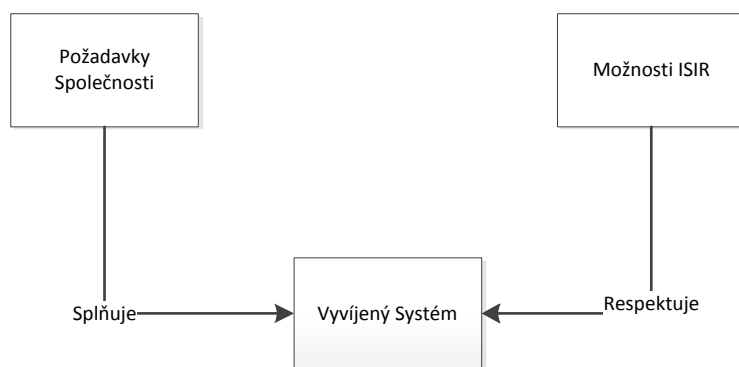
4.1.1. Analýza/model obsahový

Smyslem analýzy je popsat po obsahové stránce funkčnost Systému. Jejím výstupem bude detailní popis toho, co by měl Systém dělat. Vzhledem k použití iteračního modelu tak vlastně vyplývá postup od obecných požadavků Společnosti k detailní specifikaci respektující možnosti ISIR. Postup práce vysvětluje obrázek 1.



Obrázek 1 získání popisu Systému (Zdroj: Autor)

Samotný Systém pak musí splňovat požadavky Společnosti na jedné straně a možnosti ISIR na straně druhé, z toho vyplývá možné omezení požadavků Společnosti a zároveň také specifikaci obecného architektonického uspořádání Systému už v části obsahové analýzy. Situace je znázorněna na obrázku 2.



Obrázek 2 omezení Systému (Zdroj: Autor)

V rámci analýzy nejprve zjistím základní požadavky Společnosti na aplikaci. Následovat bude analýza možností systému ISIR, po kterém bude předloženo řešení ke schválení Společnosti, pokud nebude řešení schváleno, ať z důvodu nedostatečného popisu, nebo odmítnutí způsobu řešení zástupcem Společnosti, bude následovat další iterace. Výstupem analýzy by měla být analýza možností ISIR, z ní vycházející popis entit vyskytujících se v Systému, příklady použití, významné procesy, návrhy uživatelského rozhraní a případně fundamentální uspořádání řešení vyplývající z možností ISIR. Chtěl bych zdůraznit význam návrhů uživatelského rozhraní, tvorbu prototypů zdůrazňuje mnoho publikací a metodik, např. [9], nebo RAD (Rapid Application Development) viz [16] a uživatelské rozhraní jsem si vybral z toho důvodu, že jeho tvorba je minimálně náročná na čas a zároveň poskytuje poměrně přesnou představu funkcionality systému. I při prezentaci zákazníkům, kteří nemají vysoké informační povědomí a vlastně ani nevědí, co přesně by měl tvořený systém dělat, návrh uživatelského rozhraní poskytuje formu, které porozumí. Pokud potom návrh doplníme popisem akcí, které nastávají při různých událostech, dostává zákazník velmi přesnou představu a může rovnou reagovat a určit, které funkce potřebují dále, které nejsou navrženy vhodně a které jsou zcela zbytečné. Samozřejmě metoda předpokládá poměrně malý systém, u velkých ERP by asi nebyla praktická, ale u malých projektů bych ze své zkušenosti řekl, že může nahradit většinu obsahové analýzy a poskytnout přesnou specifikaci, co zákazník chce.

4.1.2. Analýza/model technologický

Smyslem analýzy je zvolit a popsat po technologické stránce funkčnost Systému. Jejím výstupem bude detailní popis toho, jak bude Systém dosahovat svého cíle, popsání v obsahové analýze. Dále bude výstupem analýzy architektonické řešení Systému a jeho komponent, volba prostředků pro tvorbu Systému, tedy např. typ datového úložiště, programovací jazyk n-té generace, nebo daná platforma a konceptuální datový model úložiště. Všechny tyto výstupy musí být schváleny zástupcem Společnosti.

4.1.3. Analýza/model implementační

Smyslem analýzy je popsat konkrétní prostředky, kterými Systém dosahuje cíle a navazuje tak na analýzu technologickou. Výstupem analýzy bude volba konkrétního programovacího jazyka a s ním svázané metodiky/pravidla tvorby, volba konkrétního úložiště dat a jeho fyzický datový model.

4.2. Implementace

Fáze se zabývá samotnou tvorbou řešení Systému, což je jejím výstupem. V této fázi proběhne jeho vývoj, testování a úpravy podle požadavků Společnosti. Vývoj bude probíhat podle modelu „programuj a opravuj“ [17], tedy několika přírůstky, kdy bude nejprve vytvořen přírůstek (část řešení systému), ten bude následně testován jak interně, tak v některých případech externě a v případě schválení bude nasazen do provozu. Použitý model je poměrně neformální, ale odpovídá vztahu se Společností při řešení Systému.

4.3. Zavedení do provozu

Tato fáze byla spojena s fází implementace. Jednotlivé přírůstky budou vždy při svém dokončení nasazeny do provozu.

4.4. Provoz a užití

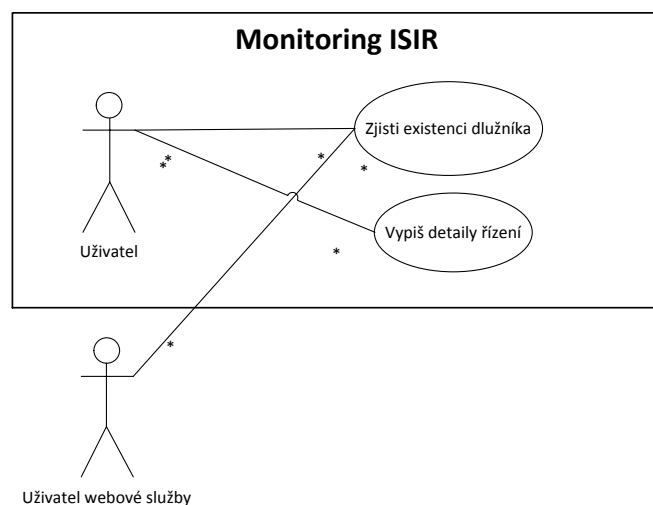
Vzhledem k poměrně velkému časovému rozpětí mezi začátkem práce na Systému a odevzdáním této diplomové práce jsem zahrnul i fázi provoz a užití, ve které očekávám především monitorování provozu aplikace a případné návrhy na její změnu. Fáze také může sloužit jako ukazatel kvality provedení předchozích fází.

5. ANALÝZA/MODEL OBSAHOVÝ ISIR

5.1. Analýza Základních požadavků Společnosti

Zástupce společnosti na první schůzce formuloval požadavek na monitorování rejstříku ISIR. Monitorování má být dostupné jako modul do informačního systému společnosti a vedle toho jako samostatný program, který přijímá strukturované dotazy a na jejich základě vypisuje existenci/neexistenci dlužníka v ISIR. Dále by měla být dostupná webová služba, která na základě jednoduchého dotazu vrátí příznak existence dlužníka v rejstříku. U služby se počítá s rozšířením její funkcionality i na výpis detailů o řízení, ale to není předmětem této práce. V práci se zabývám pouze samostatným programem pro monitoring ISIR, modul do informačního systému bude řešen podobně, ale zástupce si nepřál uvedení řešení v práci. Řešení programu pro monitoring je ale v některých místech ovlivněno stávajícím informačním systémem. Požadovaný stav zachycuje obrázek 1, příklad použití monitoringu ISIR. Uživatelem je míněn uživatel, který používá program monitoringu, uživatel webové služby je potom uživatel, který jen využívá poskytovanou webovou službu. Zjišťování existence musí probíhat jak na požádání u jednoho subjektu, stejně jako to umožňují stránky justice,

ale i dávkově na základě vhodně strukturovaných vstupních dat. Webová služba má být maximálně jednoduchá a tedy vracet příznak jen na základě jednoduchého dotazu na jednoho dlužníka.



Obrázek 3 příklad použití monitoringu ISIR (Zdroj: Autor)

5.2. Analýza ISIR

V této části uvádím, co to je ISIR, jaká data obsahuje a jakým způsobem je možno k nim přistupovat. Na závěr je pak uveden model, který ukazuje daný stav řešení vzhledem k možnostem ISIR.

5.2.1. Definice ISIR a jeho obsahu

Rejstřík ISIR vychází ze zákona č. 182/2006 Sb., o úpadku a způsobech jeho řešení [18], který nabyl účinnosti dne 1. ledna 2008. Jako zdroj používám v této části především portál Business center [18], což je portál zabývající se financemi, právem a podnikáním, alternativním zdrojem je potom Portál veřejné správy České republiky [19]. Portál Business center jsem zvolil z důvodu vyšší přehlednosti zobrazení zákona.

Část třetí insolvenčního zákona obsahuje ustanovení ohledně ISIR, jejichž důležité části zde uvádím.

Obecné informace - § 419

ISIR je informačním systémem veřejné správy, jehož správcem je Ministerstvo spravedlnosti. Obsahuje seznam insolvenčních správců, seznam dlužníků a insolvenční spisy. Pro každého dlužníka se vede jeden insolvenční spis. ISIR je veřejně přístupný, každý má právo do něj nahlížet a pořizovat si z něj kopie a výpisy.

Informace o dlužnících v ISIR - § 420

Údaje o dlužnících v ISIR záleží na typu osoby, kterou dlužník je.

- Fyzická osoba - zapisuje se do seznamu dlužníků její jméno, příjmení, bydliště, rodné číslo, a nemá-li rodné číslo, datum narození.
- Fyzická osoba podnikatel - zapisuje se do seznamu dlužníků vedle údajů fyzické osoby i dodatek odlišující její firmu, používá-li jej při svém podnikání, dále místo podnikání, jestliže se liší od bydliště, a identifikační číslo.
- Právnícká osoba - zapisuje se do seznamu dlužníků její obchodní firma nebo název, sídlo a identifikační číslo.

Informace o zapisování obsahu do ISIR - § 421

Výše specifikované údaje zapíše insolvenční soud do seznamu dlužníků, jakmile nastanou účinky spojené se zahájením insolvenčního řízení, nejpozději však do 7 dnů po tomto okamžiku. Není-li mu některý z těchto údajů v uvedené době znám, zapíše jej insolvenční soud do seznamu dlužníků, jakmile v insolvenčním řízení vyjde najevo.

Neprodleně po ustanovení insolvenčního správce zapíše insolvenční soud údaj o tom do seznamu dlužníků. Je-li insolvenční správce fyzickou osobou, zapisuje se do seznamu dlužníků jeho jméno, příjmení, sídlo, rodné číslo, a nemá-li rodné číslo, datum narození; je-li insolvenční správce právnickou osobou, zapisuje se do seznamu dlužníků jeho obchodní firma nebo název, sídlo a identifikační číslo.

Informace o uvedených dokumentech a událostech v ISIR - § 422

V insolvenčním rejstříku insolvenční soud zveřejňuje chronologicky s uvedením okamžiku vložení tyto písemnosti a informace:

- rozhodnutí insolvenčního soudu vydaná v insolvenčním řízení a v incidenčních sporech.
- veškerá podání, která se vkládají do soudního spisu vedeného insolvenčním soudem ohledně dlužníka, nestanoví-li zákon jinak.
- další informace, stanovené zákonem.

Podání došla insolvenčnímu soudu v elektronické podobě a písemnosti pořizované insolvenčním soudem se do insolvenčního rejstříku vkládají pomocí elektronického přenosu dat. Ostatní písemnosti a podání se vkládají přenesením jejich obrazové podoby do insolvenčního rejstříku.

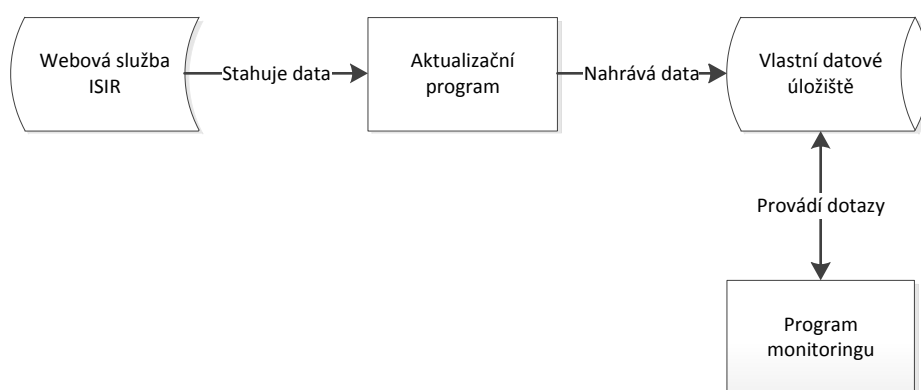
5.2.2. Popis možností přístupu do ISIR

Do ISIR lze nahlížet pomocí webového formuláře, ale touto cestou lze prohledávat jen po jednom dlužníkovi, což je pro zpracování desítek pohledávek nepraktické. Insolvenční rejstřík proto nabízí i webovou službu ISIR jejíž popis je následující: „Ministerstvo spravedlnosti ČR poskytuje od 1.1.2008 také webovou službu, která umožňuje automaticky komunikovat a vyměňovat si údaje z

aplikace insolvenčního rejstříku s jinými aplikacemi přes Internet. Poskytuje shodné údaje, které jsou dostupné v interaktivním prohlížení insolvenčního rejstříku. Tyto údaje jsou určeny pro strojové zpracování. Služba je koncipovaná jako veřejná a je zdarma. Je poskytována na adrese: https://isir.justice.cz:8443/isir_ws/services/IsirPub001 [20]. Popis veřejného rozhraní je uveden na adrese https://isir.justice.cz:8443/isir_ws/services/IsirPub001?wsdl a dokumentace webové služby je na adrese https://isir.justice.cz/isir/help/Popis_WS.pdf . Služba je nabízena pomocí technologie webové služby [web services], tedy pomocí přenosu XML [21] souborů přes internet.

Služba nabízí jen replikaci dat [22]. Tedy webová služba neumožňuje samotné prohledávání rejstříků, ale nabízí výpis událostí insolvenčního rejstříku v chronologickém pořadí, jak byly v rejstříku zapsány „Služba je navržena tak, aby poskytovala maximální množství informací při zachování vysoké dostupnosti. Informace o insolvenčním řízení jsou v databázi ISIR uložena formou událostí/akcí, ke kterým v průběhu řízení dochází. Každá akce má své unikátní číslo generované sekvenčně při zápisu události do systému ISIR. „ [23]. Služba tedy předpokládá vytvoření vlastních kopií datového úložiště u uživatelů webové služby „Služba předpokládá vytvoření vlastních kopií databází na straně uživatelů. Důvodem je požadavek vysoké dostupnosti.“ [23]. Jakkoliv by toto provedení mohlo být diskutabilní [24], není předmětem této práce.

Z provedení webové služby ISIR tedy vychází, že je nutno vytvořit vlastní datové úložiště záznamů ISIR a to pravidelně (ideálně neustále) aktualizovat v případě existence nových záznamů v ISIR. Vedle monitorovacího programu tak musí být vytvořen i program aktualizací. Situace je zobrazena na obrázku 4.



Obrázek 4 řešení monitoringu ISIR (Zdroj: Autor)

5.3. Analýza požadavků společnosti na navržené řešení

Na základě předchozích analýz jsem navrhl řešení monitoringu ISIR, které je na obrázku 2 a s navrhovaným řešením seznámil zástupce Společnosti. Monitoring tedy předpokládá tvorbu vlastního datového úložiště pro uložení záznamů, tvorbu aktualizací programu, který bude neustále zkoušet naplnit nové záznamy z webové služby ISIR do vlastní databáze a nakonec vlastní monitorovací

program pro prohledávání datového úložiště. Zástupce společnosti navrhované řešení schválil a uvedl další požadavky k jednotlivým částem.

5.3.1. Datové úložiště

Zástupce společnosti formuloval požadavky na úložiště, které společnost bude muset provozovat:

- Datové úložiště by mělo být realizováno formou databáze.
- Databáze by měla být dostupná pomocí SQL dotazů [24].
- Databáze by měla být dostupná zdarma, nebo velmi levně (posoudí zástupce společnosti při předložení návrhů). Zástupce dále navrhl, že by databáze mohla být mysql [25] databáze, již umístěná na serveru Společnosti, ale vyjádřil přání analyzovat další možnosti.
- Databáze by měla umožňovat jednoduché prohledávání i plnění.
- Databáze by měla odpovídat standardům Společnosti pro tvorbu databází, tedy každá tabulka by měla mít vlastní umělý primární klíč typu integer.
- Databáze by měla umožňovat připojení z více programů a jejich obsluhu.

5.3.2. Aktualizační program

Zástupce společnosti formuloval požadavky na aktualizací program, který společnost bude muset provozovat:

- Program by měl být velmi jednoduchý, jen s textovým výpisem, co program zrovna dělá.
- Program by měl umožňovat zapnout a vypnout plnění.
- Program by měl zkoušet, zdali existují nové záznamy na webové službě ISIR a v případě jejich existence je zkopírovat do vlastní databáze.
- Program by měl v případě neexistence nových záznamů počkat daný čas a následně znovu zkusit plnit.

5.3.3. Monitorovací program

Zástupce společnosti formuloval požadavky na monitorovací program pro prohledávání databáze:

- Program by měl umožňovat prohledání databáze na základě zadaných specifikací dlužníka.
- Program by měl umožňovat dávkové prohledání databáze na základě vstupního souboru.
- Program by měl zobrazovat, zdali je dlužník v databázi, změnu stavu dlužníka (například další dokument, případně jakákoliv změna relevantní k řízení) od posledního prohledání.
- Program by měl v budoucích verzích umožňovat zobrazit celý výpis detailů řízení dlužníka, kterého našel v databázi. Ale tato funkcionality má nízkou prioritu a Společnost preferuje rychlejší realizaci ostatních požadavků. Systém by tak měl být navržen, aby výpis detailů bylo možno doplnit.

Vzhledem k datům, které společnost o pohledávkách a dlužnících provozuje ve svém systému, by měl program umět vyhledávat především podle následujících údajů:

- Jméno
- Příjmení
- Název
- Rodné číslo
- Identifikační číslo
- Datum narození

Vedle výše specifikovaných dat by program měl umožňovat zobrazení vlastního identifikačního znaku, jaký je používán v systému společnosti, druhu osoby (fyzická, fyzická podnikatel, právnická) a dále spisové značky insolvenčního řízení v případě nalezení dlužníka v databázi.

5.3.4. Webová služba prohledávání

Zástupce společnosti formuloval požadavky na webovou službu pro prohledávání databáze:

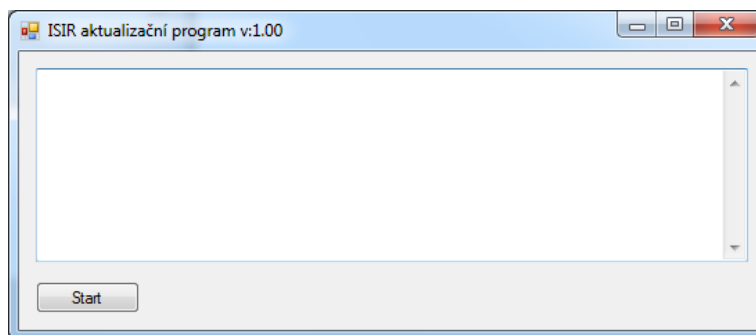
- Služba přijímá požadavky na prohledávání formou parametrů v URL adrese, tedy například
www.web_spolecnosti.cz/prohledej_jmeno=karel,prijmeni=novak,rc=1111111111.
- Služba by měla vypsát, zdali je subjekt v databázi, tedy vrátit jako odpověď jednoduchou stránku, která obsahuje buď text „ANO“ v případě nalezení, nebo „NE“ v případě nenalezení. Předpokládá se využití služby třetí stranou a toto je dohodnuté rozhraní, na kterém se Společnost shodla.
- Služba by měla na požádání umět vypsát datum poslední aktualizace databáze.
- Služba musí umět vyhledávat fyzické osoby podle jména, příjmení a data narození, nebo rodného čísla a právnické osoby buď podle identifikačního čísla, nebo názvu.

5.4. Tvorba Demo verzí

Na základě požadavků Společnosti na programy navrhovaného řešení jsem vytvořil demonstrační ukázky uživatelského rozhraní navrhovaných programů. Navrhovaná uživatelská rozhraní jsou pouze ilustrační a nijak nepodmiňují finálně použitou technologii, která je specifikována níže.

5.4.1. Aktualizační program

Návrh uživatelské rozhraní aktualizací programu je na obrázku 5. Návrh předpokládá výpis činnosti aktualizací programu do textového pole a možnost zapnutí/vypnutí programu pomocí tlačítka. Návrh byl schválen zástupcem Společnosti.



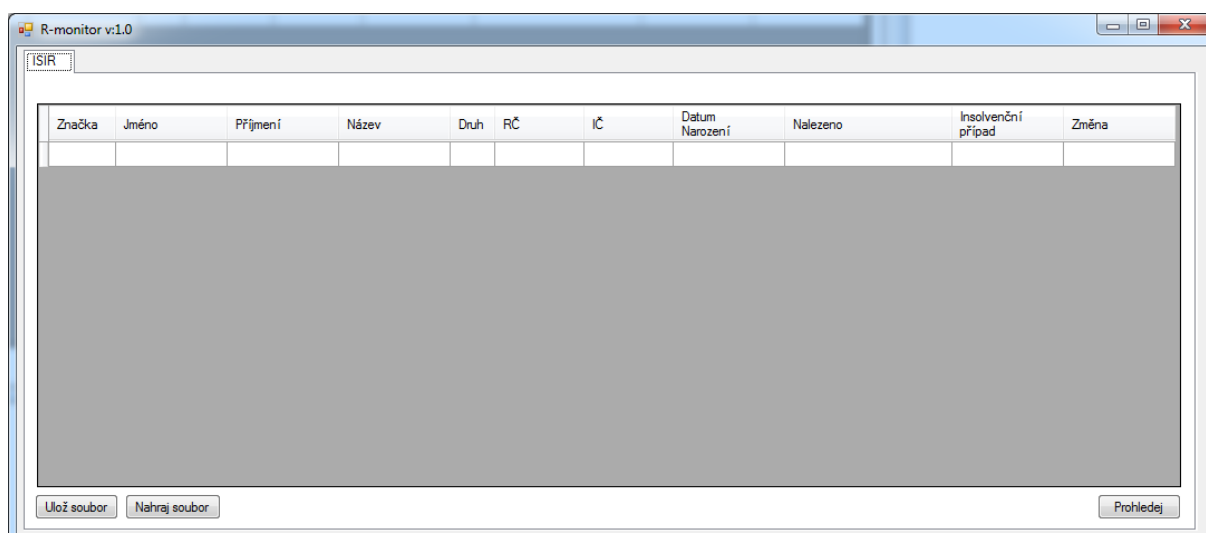
Obrázek 5 návrh uživatelského rozhraní aktualizacího programu (Zdroj: Autor)

5.4.2. Monitorovací program

Návrh uživatelské rozhraní monitorovacího programu je na obrázku 6. Návrh předpokládá práci nad dávkou dlužníků, kteří jsou vyhledáváni v databázi ISIR. Seznam dlužníků je uveden v tabulce. Použité atributy dlužníků vycházejí z informací ukládaných v ISIR a specifikovaných výše a atributů významných pro systém Společnosti, jejich seznam i s komentářem uvádím níže:

- Identifikace – Identifikátor dlužníka v dalším informačním systému. Tento atribut byl zvolen z toho důvodu, že systém pro správu pohledávek Společnosti obsahuje jednoznačný identifikátor pohledávky, který je často používán, je tedy praktické ho použít i ve vyhledávání v ISIR. Stejně tak předpokládám existenci podobných identifikátorů u případných dalších třetích stran.
- Druh osoby – Atribut určuje, o jakou osobu dlužníka se jedná, fyzickou, podnikatele, nebo právnickou. Atribut může ovlivňovat způsob vyhledávání.
- Spisová značka – Atribut převzatý z ISIR, pokud je dlužník v databázi ISIR nalezen, je k němu přiřazena spisová značka insolvenčního řízení.
- Jméno – Atribut určující jméno dlužníka, je použito u fyzických osob a podnikatelů.
- Příjmení – Atribut určující příjmení dlužníka, je použito u fyzických osob a podnikatelů.
- RČ – Atribut rodné číslo, je použito u fyzických osob a může být u podnikatelů.
- Datum narození – Atribut určující datum narození fyzické osoby, nebo podnikatele.
- Název – Atribut určující obchodní název dlužníka, použito u právnických osob.
- IČ – Atribut určující Identifikační číslo dlužníka, použito u právnických osob, může být u podnikatelů.
- Nalezeno – Atribut určuje, zdali byl dlužník nalezen v databázi ISIR.
- Změna – Atribut určuje, zdali se stav řízení dlužníka od posledního vyhledávání změnil.

Návrh byl schválen zástupcem Společnosti.



Obrázek 6 návrh uživatelského rozhraní monitorovacího programu (Zdroj: Autor)

5.5. Specifikace entit

Vedle atributů vyjmenovaných výše, které vycházejí z prvotních požadavků Společnosti, bude potřeba ukládat mnohé další informace. Tato nutnost je podložena požadavkem Společnosti na možnost tisku detailů dlužníka a to způsobem podobným, jakým detaily poskytuje portál ISIR. Pro ilustraci uvádím na obrázku 7 výpis náhodně vybraného dlužníka z webu portálu ISIR.

Detail insolvenčního řízení Filip Adamec

<

Obrázek 7 výpis detailů insolvenčního řízení (Zdroj: [20])

Z obrázku tedy vyplývá nutnost ukládat další informace jak o samotné osobě dlužníka, insolvenčním správci, tak událostech, které nastaly v insolvenčním řízení. Zde bych mohl udělat poměrně vyčerpávající analýzu, které informace jsou pro Společnost skutečně nutné, ale zástupce Společnosti navrhl pro jistotu ukládat veškeré informace, které rejstřík poskytuje. V budoucnu by tedy při potřebě dodatečných dat nebylo nutné provádět jejich komplikované doplnění. Svůj návrh doprovodil podmínkou, že se nesmí jednat o příliš velké množství dat např. více než deset GB. Při specifikaci všech ukládaných entit budu vycházet z [20]. Protože manuál je poměrně vyčerpávající, popisuje způsob poskytování i samotná data a protože si myslím, že analýza služby ISIR je velmi významným krokem v navržení řešení Systému a také pro větší přehlednost jsem se rozhodl analýzu ISIR a z ní vycházející kompletní specifikaci entit zařadit jako samostatnou kapitolu mezi obsahovou a technologickou analýzu, ač de-facto spadá do obsahové analýzy.

5.6. Analýza možností ISIR

V této části se zabývám popisem jakým způsobem a jaká data poskytuje webová služba ISIR. Vycházím především z [23], což je manuál popisující chování služby a je volně stažitelný na portálu justice. Celá kapitola 5.6 je přebrána ze zdroje, ale považuji ji za tak důležitou, že ji s úpravami

uvádím. V práci vycházím z verze 1.5 a v popisu postupuji v pořadí, jak je uveden ve zdroji. Nechci zde rozebírat celý manuál, ale pouze části podstatné pro tvorbu Systému. Základní popis služby byl již uveden výše.

5.6.1. Základní popis

5.6.2. Věc

Věcí se rozumí jedno insolvenční řízení proti konkrétní osobě. Věc popisují její atributy a události, které ve věci nastaly.

5.6.3. Událost a akce

Událost je chápána jako vlastnost věci, tedy jedná se jakousi skutečnost, která nastala v insolvenčním řízení. Akce je potom zavedena jako přenosová jednotka webové služby. Pro srovnání, událost zadává nějaký uživatel a může být zrušena jako mylný zápis. Je identifikována spisovou značkou věci, druhem oddílu a číslem v tomto oddílu. Akci potom generuje systém ISIR jako prvek přenosu dat. Je identifikována svým číslem.

5.6.4. Poznámka

Z důvodu úspory přenášených dat a také možnosti upravení struktury přenášených dat obsahuje volání události jen základní údaje její identifikace. Samotná další data potom obsahuje poznámka, která je vložena jako xml dokument v události. Poznámka je definována svých XSD (Xml Schema Definition), což je schéma definující strukturu poznámky. Každá poznámka má v sobě vložen odkaz na své schéma, pokud se tedy změní struktura poznámek, změní se i element odkazující na verzi schématu použitou.

5.6.5. Poznámky a události/akce

Přenos byl navržen tak, že většina akcí/událostí obsahuje poznámku prázdnou, nebo jen s minimem informací.

Číselník akcí/událostí lze potom dělit takto:

- Události/akce podle číselníku
 - Obsahují informace o zveřejňovaných událostech, vznikají na základě zápisu uživatelem.
 - Poznámka je prázdná, nebo jen s minimem informací o události.
 - Neobsahují informace o osobách, nebo věci jako takové.
- Servisní události/akce
 - Servisní události/akce generuje automaticky systém.
 - Servisní události/akce obsahují informace o řízení – data o osobách a věci.
 - Servisní události mohou měnit jiné události – ukončit jejich platnost.

- Servisní události mohou měnit číselník událostí.

5.6.6. Typické akce

V následující sekci uvádím typické akce, které probíhají na webových službách ISIR. U akcí vynechávám samotný příklad xml dat z důvodu ušetření místa.

Založení spisu

Spis je založen první běžnou událostí. Hned po této akci následuje servisní akce s osobou dlužníka.

Založení osoby v řízení

Osoby jsou zakládány servisními akcemi po akci, se kterou souvisejí. Podobným způsobem jsou zakládány adresy u osob.

Zpožděné doplnění dokumentu

Protože pro zveřejnění insolvenčního řízení jsou krátké lhůty a některé dokumenty ještě nemusí být připraveny, případně anonymizovány, lze publikovat událost bez dokumentu, který bude doplněn později. Dokument je doplněn událostí se stejným typem a identifikací, jen vyplněným dokumentem.

Změna osoby/adresy

Změny osob a adres jsou generovány jednou za den pomocí stejných servisních událostí, které tyto osoby a adresy založily.

Rušení záznamů o událostech, osobách, věcech

Typicky v případě mylného zápisu je občas potřeba událost, osobu, adresu nebo věc zrušit. Událost ruší akce se stejným typem, která ji založila. Akce obsahuje datum zrušení a identifikaci události (spisová značka, druh oddílu, pořadí v oddílu).

Osoba je rušena servisní událostí pro změnu osoby s vyplněným datem zrušení osoby.

Změna číselníku událostí

Servisní událost bez spisové značky, bude obsahovat:

- číslo postižené události podle číselníku
- popis postižené události podle číselníku
- popis změny
- datum platnosti postižené události od
- datum platnosti postižené události do

- příznak, zda se jedná o servisní událost (servisní se nezobrazují na webu)

Kompletní číselník událostí je uveden na adrese https://isir.justice.cz/isir/help/Cis_udalosti.xls

Změna poznámky

Pro změnu formátu poznámky je speciální servisní událost „Změna poznámky“ s typem 330. Událost obsahuje datum, od kdy změna platí, nové URI souboru se schématem poznámky a popis změny.

5.6.7. Datový model

5.6.8. Volání webové služby

Webová služba podporuje dva typy volání, se zadaným posledním ID akce nebo se zadaným datem akce.

- **getIsirPub001** – Volání webové služby s parametrem typu datum. Vrací všechny akce s časem $\geq$ zadanému datu.
- **getIsirPub0012** – Volání webové služby s parametrem typu long. Vrací všechny akce s id $>$ než je zadané číslo.

5.6.9. Rozhraní webové služby

Tabulka 1 specifikuje strukturu události/akce, tak jak ji webová služba poskytuje. Webová služba vrací pole těchto akcí.

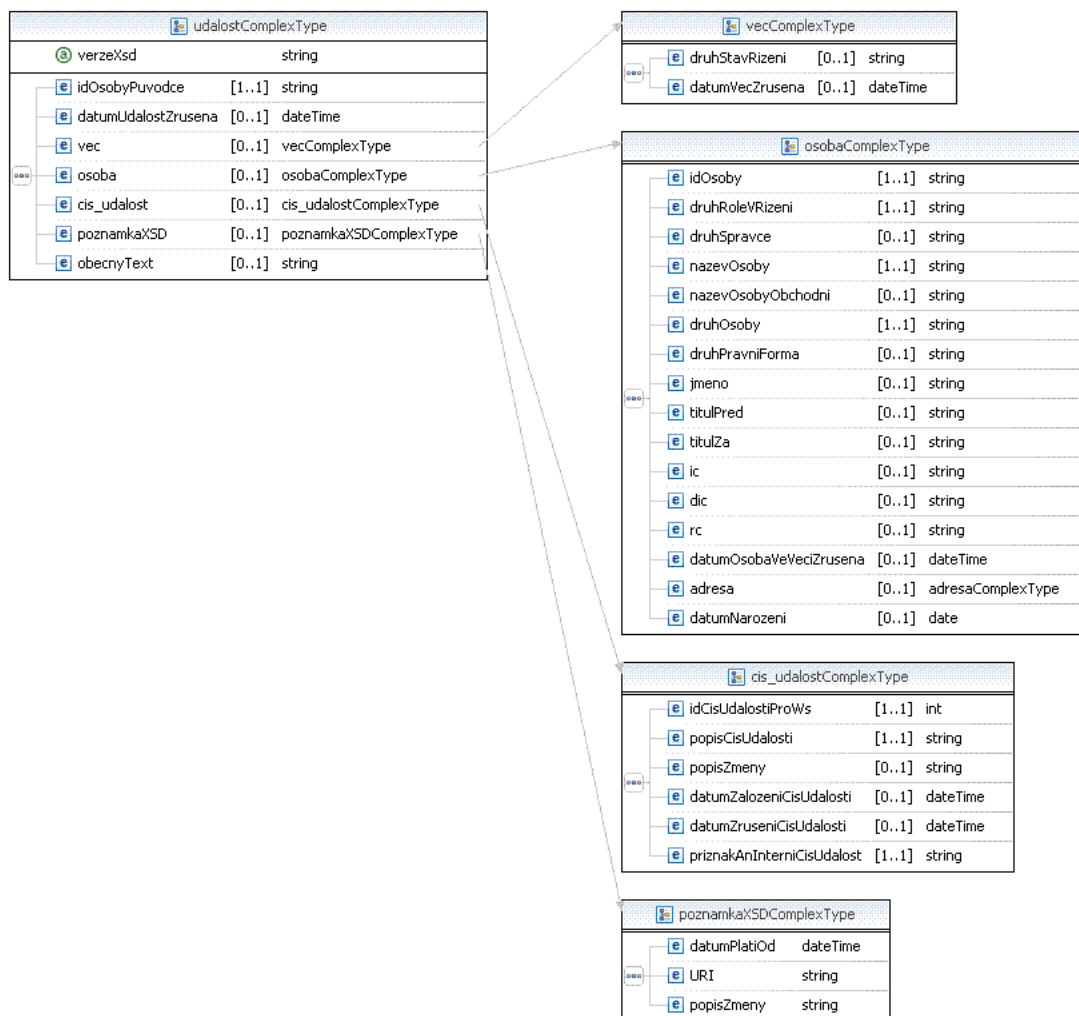
Tabulka 1 specifikace struktury akce (Zdroj: [23])

Název	Popis
Cas	Datum a čas vzniku události.
Id	Unikátní sekvenční číslo události v ISIRu.
IdDokument	URI adresa dokumentu.
Poznamka	Poznámka obsahuje strukturované údaje k události, pro vybrané události zobrazuje formou XML dokumentu údaje např. o dlužníkovi.
spisZnacka	Spisová značka řízení, ke kterému událost patří.
Typ	Typ události podle číselníku událostí.
TypText	Popis události podle číselníku událostí.
Oddil	Oddíl, do kterého událost patří.
PoradiVOddilu	Pořadové číslo v rámci oddílu, spolu se spisovou značkou a oddílem identifikuje událost.

5.6.10. Poznámka

Na obrázku 8 je zobrazena struktura poznámky a dalších elementů, které může obsahovat. Obrázek je již poměrně silně technologicky zatížen, ale dal jsem přednost autenticitě před formální správností

vlastního zobrazení. Z obrázku můžeme vidět, že jedna akce s poznámkou může obsahovat vždy jen maximálně jednu věc, jednu osobu, jednu položku číselníku a je popsána jedním XSD poznámkou.



Obrázek 8 struktura poznámky (Zdroj: [23])

5.6.11. Událost

V tabulce 2 je uvedena struktura události poskytovaná webovou službou. Událost je zde složena jak ze samotných návratových hodnot webové služby, tak atributů, které se nacházejí až v poznámce. V tabulce nejsou uvedeny některé další atributy, které bych události přiřadil podle předchozího popisu, nemá popsané sloupce a komentáře jsou částečně nevyplněny. Toto považuji za chybu zpracování manuálu.

Tabulka 2 struktura událostí (Zdroj: [23])

Klíč	SpisZnacka	String	50	
	Oddil	String	10	
	PoradiVOddilu	Int		
Atributy	IdOsobyPuvodce	String	20	Soud, kde událost vznikla

	datumUdalostZrusena	Datetime		Datum zrušení události
--	---------------------	----------	--	------------------------

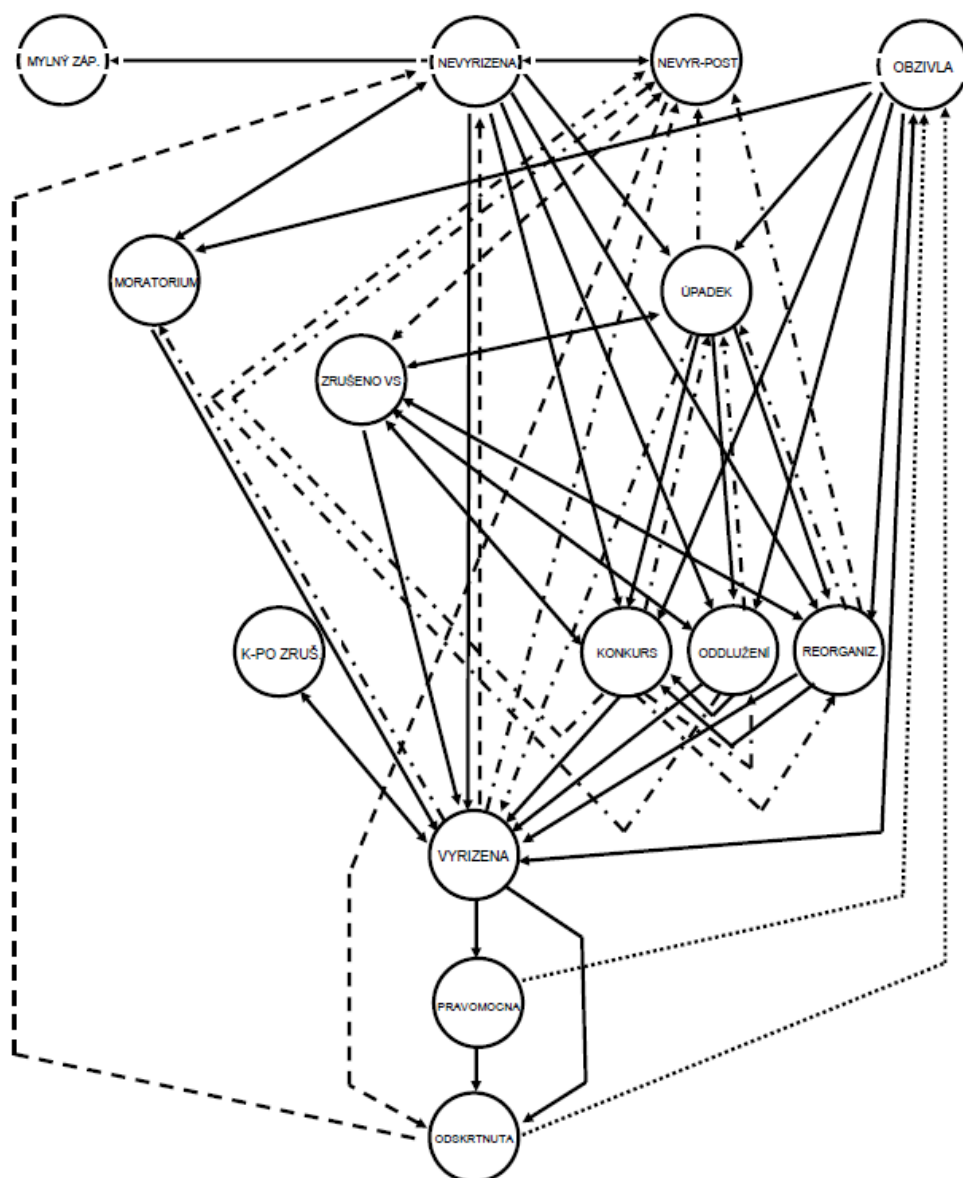
5.6.12. Věc

V tabulce 3 je uvedena struktura věci (řízení) poskytovaná webovou službou. Oproti specifikaci události je zde již popis kompletní.

Tabulka 3 struktura věci/řízení (Zdroj: [23])

Klíč	SpisZnacka	String	50	
Atributy	DruhStavRizeni	String	10	Stav řízení
	datumVecZrusena	Datetime		Datum zrušení věci (mylný zápis)

K věci je potom v manuálu uveden ještě diagram možných přechodů mezi stavy. Diagram je na obrázku 9. K diagramu se váže ještě legenda uvedená na obrázku 10.



Obrázek 9 diagram možných přechodů mezi stavy věci (Zdroj: [23])

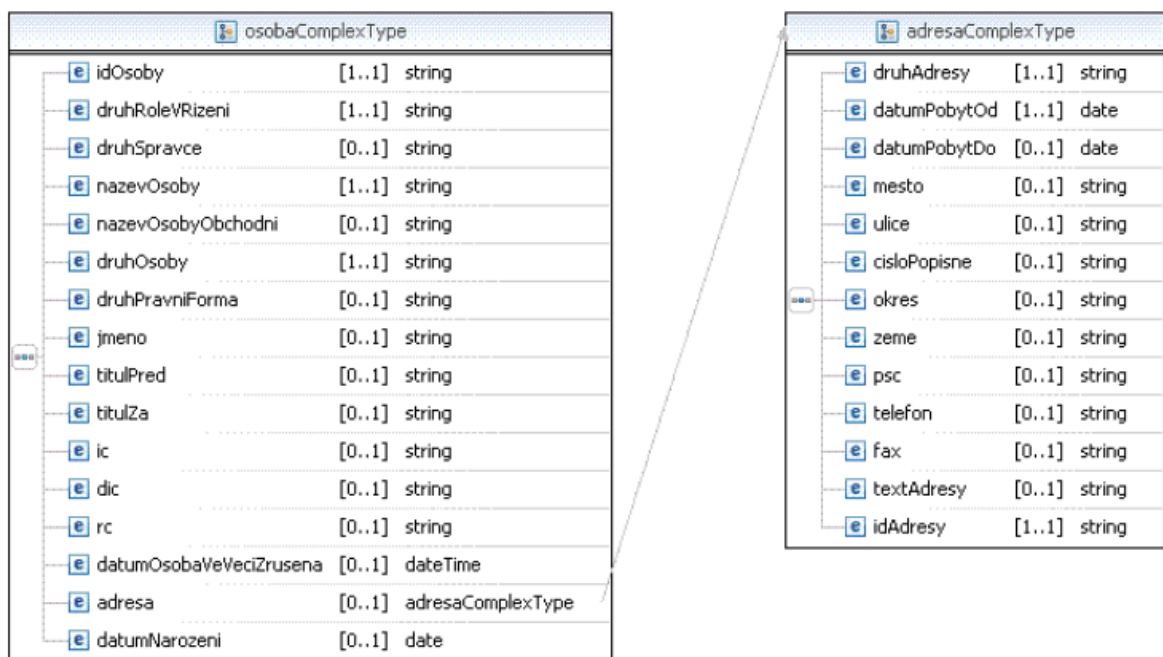
Legenda:

- 1. plně – obecně povolené přechody mezi stavy
- - - - -→ 2. přerušované – obecně nepovolené přechody mezi stavy, jen na určenou událost (postoupení, přikázání, pokračování v řízení)
- . - . - .→ 3. čerchované – přechod na základě rozhodnutí VS/NS, kdy se nejedná o obecně povolený přechod mezi stavy
-→ 4. tečkované – přechod na základě dovolání

Obrázek 10 legenda přechodů mezi stavy věci (Zdroj: [23])

5.6.13. Osoba

Součástí přenášených dat může být i element osoby. Na obrázku 11 je ukázána struktura tohoto elementu a případné adresy osoby.



Obrázek 11 struktura elementu osoby (Zdroj: [23])

Tabulka 4 potom upřesňuje specifikaci osoby.

Tabulka 4 specifikace osoby (Zdroj: [23])

Klíč	SpisZnacka	String	50	
	IdOsoby	String	20	
Atributy	DruhRoleVRizeni	String	10	Role v jaké osoba ve věci vystupuje
	DruhSpravce	String	10	Druh správce
	NazevOsoby	String	255	Přesný název osoby
	NazevOsobyObchodni	String	255	Obchodní název osoby
	DruhOsoby	String	10	Druh osoby (fyzická / právnická / organizace...)
	DruhPravniForma	String	10	Kód právní formy osoby
	Jmeno	String	40	Křestní jméno osoby
	TitulPred	String	50	Tituly osoby před jménem
	TitulZa	String	50	Tituly osoby za jménem
	Ic	String	9	Identifikační číslo
	Dic	String	14	Daňové identifikační číslo
	Rc	String	11	Rodné číslo
	DatumOsobaVeVeciZrusena	Datetime		Datum zrušení osoby ve věci
	DatumNarozeni	Datetime		Datum narození osoby

5.6.14. Adresa

Tabulka 5 ukazuje přesnou specifikaci elementu adresy osoby.

Tabulka 5 specifikace adresy (Zdroj: [23])

Klíč	SpisZnacka	String	50	
	IdAdresy	String	20	
Atributy	DruhAdresy	String	10	Kód druhu adresy
	DatumPobytOd	Date		Datum začátku platnosti adresy
	DatumPobytDo	Date		Datum konce platnosti adresy
	Město	String	255	Město/obec
	Ulice	String	255	Ulice
	CisloPopisne	String	10	Číslo popisné
	Okres	String	30	Okres
	Zeme	String	255	Země
	PSC	String	6	PSČ
	Telefon	String	30	Telefon
	fax	String	30	Fax
	TextAdresy	String	255	Text adresy e-mailu, URL apod.

5.7. Návrh datové základny

Na základě předchozího požadavku Společnosti uchování všech dat poskytovaných webovou službou ISIR a na základě analýzy dat ISIR jsem vytvořil model datové základny, kterou by Systém měl uchovávat. V rámci obsahové analýzy jsem vytvořil zatím technologicky nezávislý model, který může být implementován více způsoby. Při návrhu jsem se řídil metodikou návrhu datového modelu popsaného v [27]. Dále jsem respektoval požadavek Společnosti, aby každá entita měla zcela umělý číselný identifikátor. Jakkoliv by toto rozhodnutí mohlo být sporné vzhledem k typickému použití relačních databází a jejich normalizačních metod pro ukládání dat, dodržení konvencí Společnosti pro návrh je dle mého názoru přednější. Navíc tyto výjimky v použití normalizačních pravidel relačních databází můžeme vidět i v trendu objektově relačních, nebo objektových databází např. v [28], [29], nebo případně v postupech pro optimalizaci výkonu relačních databází, např. zde [30].

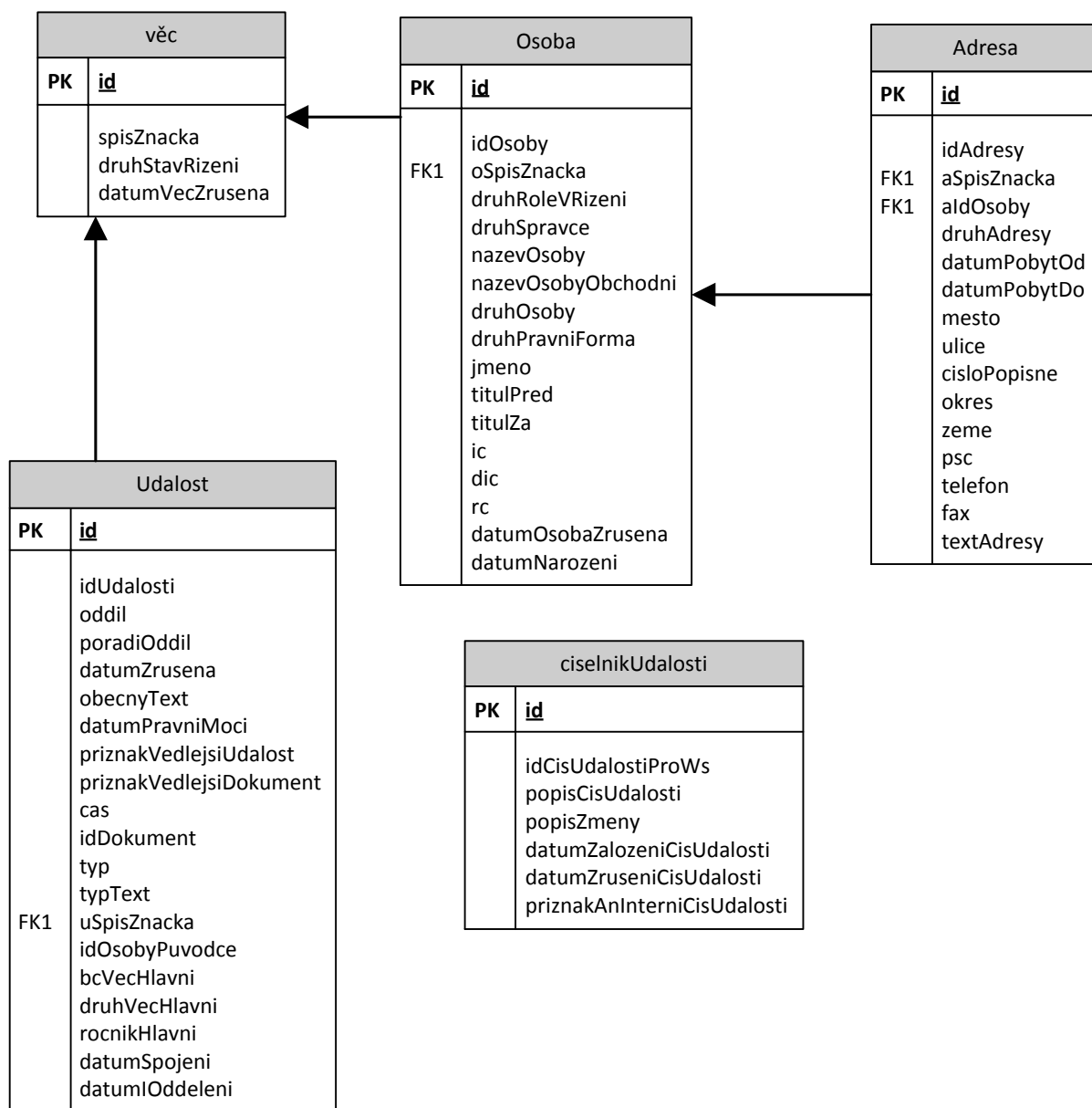
Dále jsem zjistil, že manuál ISIR, ze kterého jsem dělal analýzu, neobsahuje kompletní popis všech atributů zmíněných entit. Rozhodl jsem se přiřadit i atributy, které služba poskytuje, ale manuál nezmiňuje.

Návrhů uchovávajících vhodným způsobem data poskytovaná službou ISIR by samozřejmě mohlo být více, ale rozhodl jsem se model navrhnout co nejvíce tak, aby odpovídal datové struktuře samotné služby ISIR. Ačkoliv služba ISIR slouží k replikaci a neuvádí samotné schéma replikované databáze

(což chápu jako poměrně velký nedostatek), lze z výstupů prohledávání insolvenčního rejstříku a z manuálu služby ISIR odvodit některé fundamentální skutečnosti:

- Hlavní entitou, která spojuje ostatní je věc, tedy samotné insolvenční řízení.
- Věc může mít přiřazené osoby v různých rolích, dlužník, správce atd.
- Osoby mohou mít přiřazenou adresu.
- Další skutečnosti věci jsou vyjádřeny pomocí událostí, které ve věci nastaly.

Podle výše zmíněných jsem stvořil první návrh datového úložiště a ten dal zástupci Společnosti ke schválení. Návrh datového úložiště je na obrázku 12.



Obrázek 12 návrh úložiště dat (Zdroj: Autor)

Vzhledem k tomu, že se zatím jedná jen o obsahový návrh úložiště dat, uvedu zdůvodnění některých sporných řešení až u fyzického návrhu úložiště.

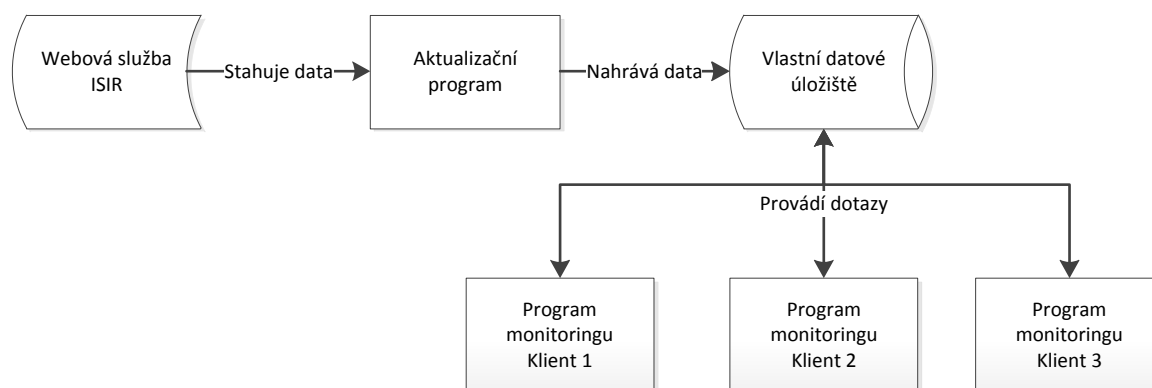
Návrh byl schválen zástupcem společnosti.

6. ANALÝZA/MODEL TECHNOLOGICKÝ ISIR

V této části provádím výběr technologií, které budou použity pro tvorbu Systému. Nejprve uvádím zvolenou architekturu Systému po které následují kapitoly, podle jednotlivých modulů Systému. Pro každý funkční modul Systému tak uvádím nejdříve požadavky a možnosti, kterými by šlo řešení dosáhnout. Po zvolení technologie pak uvádím detailní popis modulu na technologické úrovni.

6.1. Architektura Systému

Ještě je nutné podotknout, že i řešení popsané v části analýzy obsahové by šlo realizovat několika způsoby a proto zde specifikuji konkrétní. Mezi fundamentální volby patří například architektura systému. Vycházím z návrhu systému na obrázku 4 v kapitole 5.2.2. Vzhledem k tomu, že Společnost předpokládá využití Systému více subjekty, rozhodl jsem se zvolit architekturu klient - server (viz. [31]). Situaci ilustruje obrázek 13.



Obrázek 13 využití Systému více klienty (Zdroj: Autor)

Ale i při zvolené architektuře existují dvě hlavní varianty a velké množství případných modifikací.

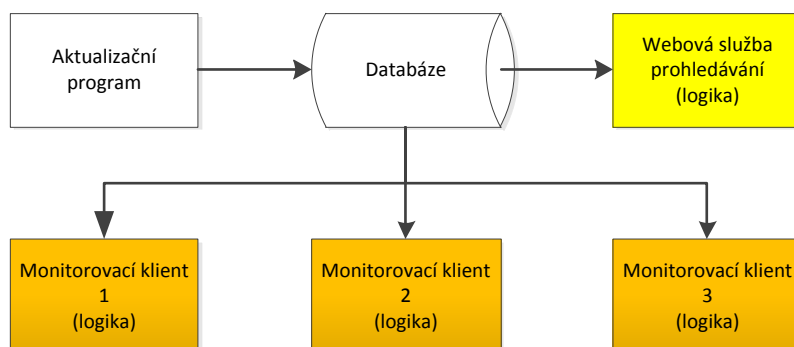
6.1.1. Lehký klient a tlustý server

Řešení se snaží realizovat většinu logiky aplikace na straně serveru a stranu klienta využívat pokud možno jen k zobrazení uživatelského rozhraní. Výhodou je větší zabezpečení vlastního zpracování, které se děje na straně serveru (tedy chráněnější), dále přenositelnost aplikací, kdy lze jednoduše vytvořit klientský program pro více platform, protože slouží jen k zobrazení dat a není potřeba přenášet logiku. Od této vlastnosti se pak odvíjí i snadnější úprava logiky, kdy stačí upravit serverovou částí a všichni klienti tak získají novou upravenou funkcionalitu bez nutnosti úpravy klientského programu. Architektura je podobná služebně orientované architektuře, protože funkcionalita serveru lze jednoduše dále nabídnout jako služba.

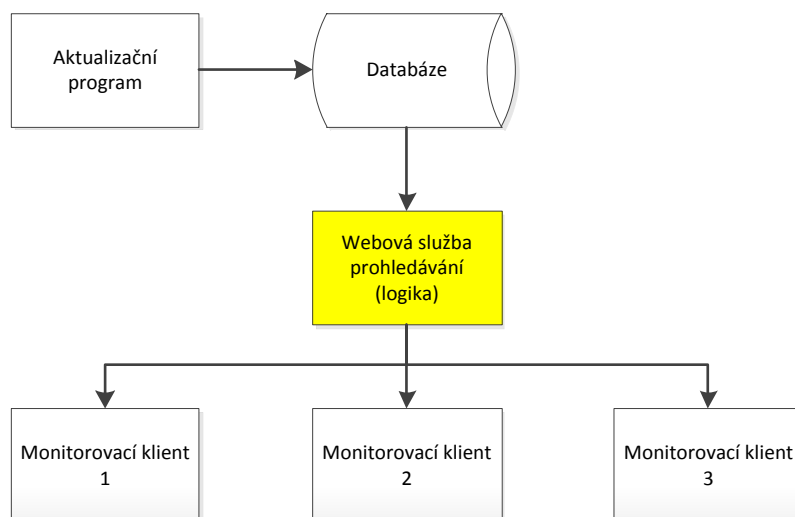
6.1.2. Tlustý klient a lehký server

Řešení předpokládá umístění logiky aplikace na straně klienta a server využívá spíše jen jako úložiště dat. Klienti sami formulují dotazy a řídí si operace, vyžadují spíše čistá data, nad kterými pak realizují funkcionalitu. Řešení snižuje nároky na server a přenáší zátěž na klientské programy, na druhou stranu ale může být náročnější na přenos dat a může přenášet zbytečná data. Dále je řešení méně bezpečné, pokud je klientský program dekompilován, odhaluje strukturu databáze. Stejně tak v případě odposlechu komunikace mohou být odchytávány dotazy a z nich odvozeny vlastnosti databáze. Řešení pak také předpokládá úpravu klientského programu a nutnost jeho přeinstalování v případě větší změny (neparametrizované) funkcionality.

Rozhodl jsem se použít řešení lehký klient a tlustý server a to především z toho důvodu, že součástí požadavků Společnosti bylo i vytvoření webové služby prohledávání a tato služba musí umět vyhledávat dlužníky podle stejných kritérií jako monitorovací program, je jednodušší vytvořit webovou službu tak, aby vracela vedle indikace existence dlužníka i data vyžadovaná monitorovacím programem. Systém tak získá všechny výhody řešení lehkého klienta a zároveň bude zaručena konzistence mezi logikou prohledávání. Řešení dále umožňuje i další funkcionalitu jako tvorbu statistik využití služeb podle IP adres a od toho případně odvozené vylepšení Systému. Obrázek 14 ukazuje variantu využití tlustého klienta a případné rozmístění logiky do více částí (ještě mohou existovat různé verze klientského programu s různou logikou), zatímco obrázek 15 ukazuje využití lehkého klienta, kdy je veškerá logika umístěna na straně serveru u databáze.



Obrázek 14 varianta tlustého klienta (Zdroj: Autor)



Obrázek 15 varianta lehkého klienta (Zdroj: Autor)

6.2. Úložiště dat

Úložiště dat musí splňovat požadavky Společnosti (viz kapitola 5.3.1) a musí být schopno uchovávat data specifikovaná v návrhu úložiště (viz kapitola 5.7). Tedy musí se především jednat o databázi, která je schopna komunikovat dotazovacím jazykem SQL [25]. Druhá podmínka vyřazuje z voleb historické technologie databází, např. hierarchické, nebo síťové, kterými se nebudu zabývat. Při těchto základních požadavcích se z hlediska databázové technologie [32] nabízí tato řešení:

- Relační databáze [33]
- Objektová databáze [34]
- Relačně-objektová databáze [35]

6.2.1. Relační databáze

Relační databáze splňuje veškeré požadavky Společnosti, naopak Společnost by použití relační databáze preferovala. Technologie vychází z matematických teorií E.F.Codda [36] a poskytuje tak na rozdíl od ostatních i pevné teoretické základy. Z praktického hlediska je potom využití relační databáze bezproblémové, Společnost má zkušenosti s použitím relačních databází, jedná se o technologii, která umožňuje komunikaci pomocí SQL a případně dalších procedurálních jazyků v závislosti na konkrétní databázi, většina nástrojů a programovacích jazyků nabízí podporu komunikace s relačními databázemi a databázi je možno provozovat jak vlastními prostředky, pomocí externího provozovatele webhostingu [37], nebo pomocí Iaas/Paas [38], jako např. Amazon [39], Microsoft [40] a další. Nevýhody použití relační databáze v tomto případě nemá, požadavky na Systém neobsahují žádné problémové situace, jako ukládání velkých grafických souborů, nebo nutnost uložení strukturovaných XML dat.

6.2.2. Objektová databáze

Použití objektové databáze by sice umožnilo ukládat specifikovaná data a navíc by nabídlo lepší provázanost s typickým objektovým jazykem, ale vzhledem k tomu, že ukládaná data jsou poměrně velmi jednoduchá a nejedná se o komplikované objekty (kdy použití objektové databáze šetří čas i prostředky), databáze by ve výsledku byla spíše pomalejší. Dále by nemuselo být dostupné typické SQL, ale spíše OQL (Object Query Language), což by komplikovalo tvorbu jednoduchého webového rozhraní pro dotazování. K objektovým databázím existuje méně nástrojů, celková podpora je nižší a i poskytovatelé databázových služeb spíše nabízejí klasické relační databáze. Použití objektové databáze bych tedy v tomto případě jak kvůli jednoduchému návrhu, tak kvůli nákladům nedoporučil.

6.2.3. Relačně-objektová databáze

Relačně objektové databáze kombinují přístup obou výše jmenovaných, snaží se spojit výhody relačních databází, především SQL, ale zároveň nabízejí některé možnosti objektových, jako např. jednoznačné databázové identifikátory, ukládání polí v tabulkách, vlastní typy, funkcionalitu objektů, nebo dědičnost [29]. Bohužel stále přetrvávají některé problémy, jako nižší podpora a obvykle vyšší náklady při správě a pořizování. Společnost opět nemá s relačně objektovými databázemi zkušenosti a potenciální výhody objektového přístupu by v tomto případě nevyužila. Použití objektově relační databáze bych tak nedoporučil.

6.2.4. Volba technologie

Úložiště jsem se rozhodl realizovat pomocí relační databáze a to především z toho důvodu, že se jedná o velmi rozšířenou technologii. Existuje k nim velká podpora a velká uživatelská komunita, tedy lze předpokládat, že nenastane neznámá kritická chyba a i v případě že by nastala, bude pravděpodobně brzy odstraněna. Navíc Společnost má s technologií zkušenosti a vlastní i server provozující relační databáze.

6.3. Webová služba prohledávání

Webová služba musí splňovat požadavky specifikované v kapitole 5.3.4. Vzhledem k tomu, že služba má vracet standardní html stránku, která má obsahovat jednoduchou odpověď ANO/NE a případně i identifikaci insolvenčního řízení a jeho poslední události, předpokládám řešení pomocí skriptovacího jazyka na straně serveru, konkrétně PHP. Funkcionality by šlo samozřejmě dosáhnout i mnoha jinými prostředky, například JSP (Java Server Pages), ASP a podobné, ale tyto technologie jsou výrazně náročnější na zdroje na straně serveru (například instalace balíčku s ASP a podobné) a ne všichni poskytovatelé toto vybavení nabízejí. Na druhou stranu PHP se v průběhu let stalo prakticky všudypřítomné (viz [41]) a naprostá většina poskytovatelů hostingu nabízí i PHP interpret, navíc Společnost má s vývojem řešení v PHP zkušenosti, čímž se usnadňuje i případný další vývoj řešení.

6.3.1. Volba technologie

Na základě předchozího zdůvodnění a po konzultaci se zástupcem Společnosti jsem se rozhodl webovou službu prohledávání realizovat pomocí řešení jazyka PHP.

6.3.2. Detailní popis webové služby

Níže uvádím detailní popis funkcionality webové služby z procedurálního pohledu a následně popis architektonický z objektového pohledu, které dle mého názoru zajistí dostatečný a vyčerpávající definici požadavků na webovou službu.

6.3.2.1. Popis funkcionality

Požadovaná funkcionality webové služby byla popsána v kapitole 5.3.4. Vstupní data jsou v tomto případě parametry předávané metodou GET v URL. Webová služba má vrátit html formulář s odpovědí ANO v případě existence a NE v případě neexistence vyhledávaného subjektu v databázi. V části 3.1 jsem potom tyto návratové hodnoty rozšířil i o insolvenční značku a datum poslední události k hledanému insolvenčnímu řízení. Podle požadavků Společnosti má služba vyhledávat subjekty podle těchto kritérií:

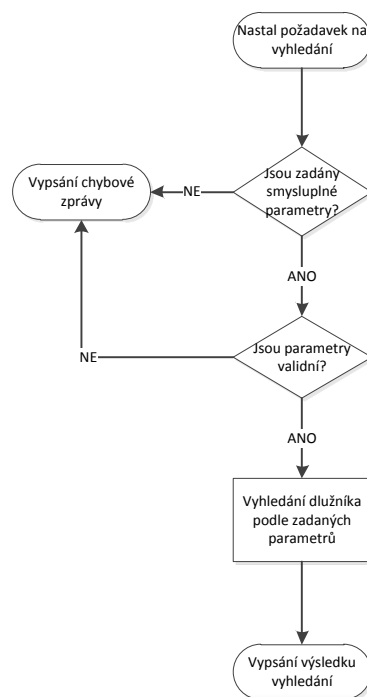
- Fyzická osoba
 - Rodné číslo
 - Jméno, příjmení a rodné číslo
 - Jméno, příjmení a datum narození
- Právnícká osoba
 - Identifikační číslo
 - Název

Vzhledem k tomu, že data poskytovaná webovou službou ISIR umožňují i výskyt identifikačního čísla u fyzických osob (podnikatelé), po konzultaci se Společností jsem rozšířil vyhledávání u fyzické osoby o identifikační číslo.

Před formulací samotného procesu vyhledávání nejdříve popíši funkcionalitu celé webové služby vyhledávání:

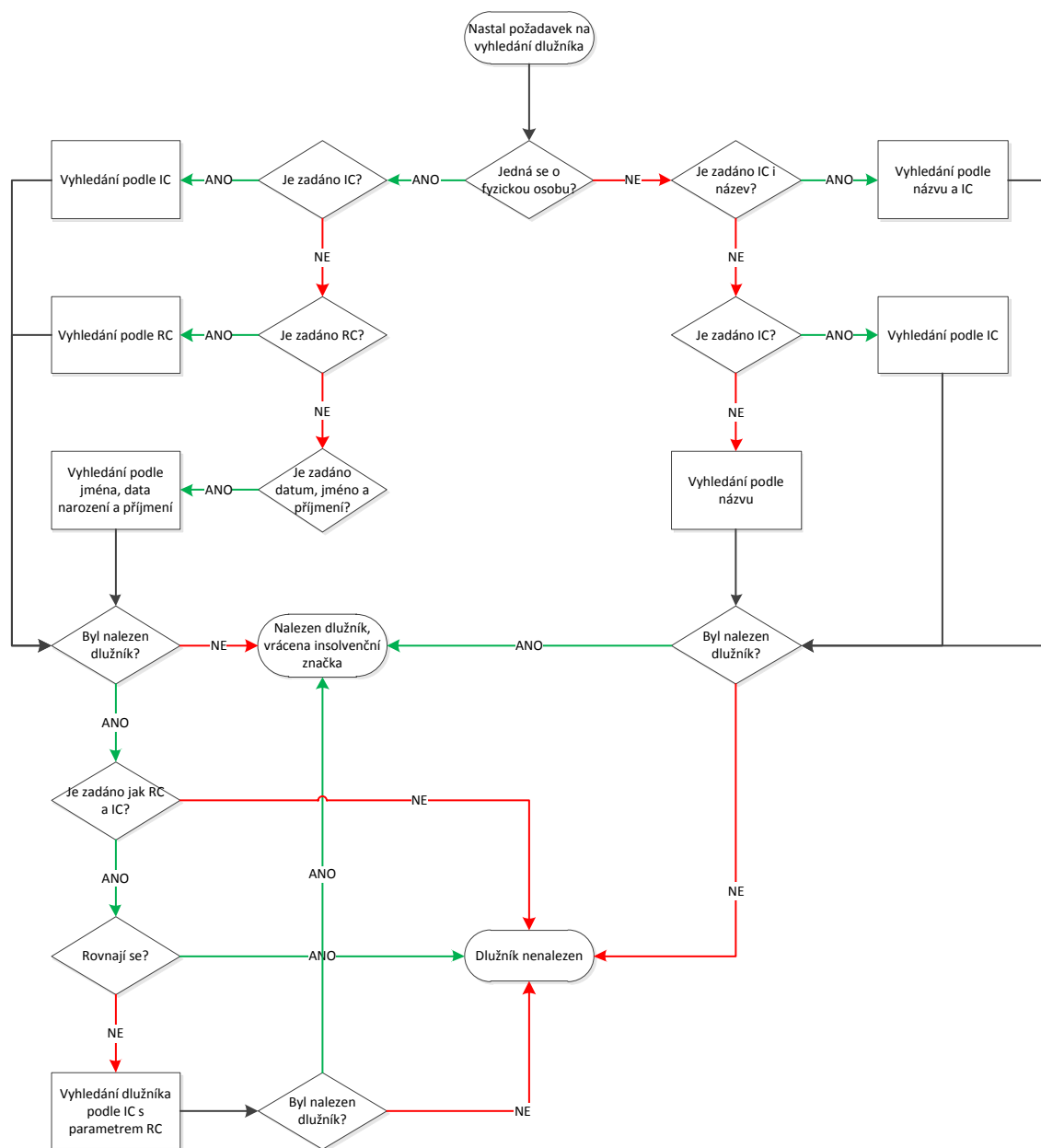
- Služba přijme data předaná parametry metodou GET v url.
- Služba vyhodnotí zda-li je zadána smysluplná kombinace parametrů.
- Služba validuje přijaté parametry (formát data, validita RČ/IČ).
- Služba provede vyhledávání a zobrazí výsledek.

Popis chování služby je na obrázku 16.



Obrázek 16 popis chování webové služby vyhledání (Zdroj: Autor)

Samotný proces vyhledávání má probíhat tak, že se snaží co nejdříve nalézt nejjistější shodu a tedy má vyhledávat od jednoznačných identifikátorů po nejednoznačné. Zároveň ale v případě nenalezení dlužníka podle jednoznačných, již nemá vyhledávat podle nejednoznačných. Tedy u fyzické osoby je nejvhodnější vyhledávat podle identifikačního čísla a pokud není nalezena shoda, vyhledávání ukončit, protože součástí vyhledávání sice mohl být dotaz se zadaným jménem, příjmením, datem narození a identifikačním číslem, ale osoba s daným číslem v databázi neexistuje. U názvů, jmen a příjmení by vyhledávání mělo být tolerantní a tedy vyhledávat i subjekty, jejichž atributy obsahují vyhledávaná data, tedy např. při vyhledávání jména "arel" by měl být nalezen dlužník se jménem "Karel". Dále byl přidán požadavek na záměnu identifikačního a rodného čísla v případě jejich neshody a nenalezení podle předchozího vyhledávání u fyzických osob. Samozřejmě počet možných výsledků musí být omezen pro zamezení posílání příliš obecných dotazů. Na základě těchto požadavků jsem formuloval proces vyhledávání, tak jak ho má webová služba prohledávání provozovat a který je na obrázku 17. Pro vyšší přehlednost jsem varianty rozhodnutí "ANO" označil zeleně a varianty "NE" červeně. Proces předpokládá smysluplná vstupní data, tedy alespoň jedno zadané kritérium.



Obrázek 17 vyhledání dlužníka ve webové službě prohledávání (Zdroj: Autor)

6.3.2.2. Konceptuální model webové služby

Po formulování chování webové služby jsem se rozhodl definovat architektonický pohled na webovou službu vyhledávání pomocí objektového přístupu. Jelikož služba bude vytvořena v programu PHP, který od verze 5 umožňuje objektový přístup (viz. [42]), a věřím, že objektový přístup k definování služby může zvýšit její přehlednost a případně usnadnit další úpravy. Model je na obrázku 18.

Prohledej
-RC -IC -druhOsoby -nazev -prijmeni -datumNarozeni
+prohledej() -kontrolaFyzicka() -kontrolaPravnicka() -vytvorQueryFyzicka() -vytvorQueryPravnicka() -hledej() -pripojit() -vypisChybu() -vypisVysledek() -upravRC()

Obrázek 18 objektový model webové služby (Zdroj: Autor)

Služba je na modelu definována jako jeden objekt se vstupními parametry, které již byly popsány výše. Jedinou úpravou je použití parametru `nazev` jako `nazev` pro právnické subjekty a jako `jméno` pro fyzické. Její metody pak umožňují funkcionalitu definovanou v předchozí kapitole a jsou vytvořeny tak, aby měly jednoznačný účel. Níže uvádím popis a účel metod:

- "prohledej" je hlavní metodou třídy, která spouští vyhledávání a používá ostatní metody. Nejprve zkontroluje zadané parametry, vytvoří dotaz který následně vyhledá a zobrazí výsledek.
- "kontrolaFyzická" je metoda, která kontroluje validitu a smysluplnost zadaných dat pro fyzické osoby.
- "kontrolaPravnicka" je obdobou předchozí metody pro právnické osoby.
- "vytvorQueryFyzicka" slouží k vytvoření dotazu na databázi se zadanými parametry pro fyzické osoby.
- "vytvorQueryPravnicka" je obdobou předchozí metody pro právnické osoby.
- "hledej" slouží k samotnému hledání subjektu se zadaným dotazem.
- "pripojit" slouží k připojení (vytvoření objektu připojení) na databázi.
- "vypisChybu" je metoda pro vypsání chyby z jakékoliv jiné metody.
- "vypisVysledek" slouží ke správnému naformátování a vypsání výsledku.
- "upravRC" je metoda pro úpravu formátu rodného čísla v dotazu (metodu jsem přidal na základě vlastních zkušeností, kdy většina údajů o subjektech se nedrží jednotné konvence v případě rodných čísel a balíky dat tak obsahují údaje jak s lomítkem, tak bez. Tedy obvykle ve všech svých programech zajišťuji převod na jednotný formát pokud možno u všech vstupů).

6.3.2.3. Úprava výstupních dat webové služby

V návaznosti na kapitolu 6.1, kde jsem se rozhodl používat webovou službu i pro dotazy monitorovacího programu, bude nutné upravit i požadovaná výstupní data webové služby. Zatímco

původní požadavek Společnosti bylo zobrazovat prosté ANO/NE (viz. kapitola 5.3.4) monitorovací program bude potřebovat i zobrazení případného insolvenčního řízení a poslední události která v daném řízení nastala. Protože Společnost již zveřejnila rozhraní ANO/NE jednomu externímu subjektu, rozhodl jsem se provozovat obě varianty, pomocí dvou verzí webové služby, jen s pozměněným URL. Úprava bude spočívat v rozdílných dotazech na databázi a rozdílném zobrazení výsledku vyhledávání.

6.4. Aktualizační program

Aktualizační program by měl splňovat požadavky popsané v kapitole 5.3.2 a zároveň poskytovat základní grafické rozhraní určené k jeho obsluze, popřípadě zjištění chyby, tak jak je navrženo v kapitole 5.4.1. K dosažení těchto cílů by šlo použít různé programovací jazyky [43]. Abych se vyvaroval velkého množství posuzovaných možností, uvedu základní kritéria, která by technologie měla splňovat. Tato základní kritéria vychází z požadavků Společnosti. Na výběr uvádím jen technologie s objektovými jazyky z důvodu jejich rozšířenosti, naopak nevěnuji se exotickým řešením, protože by pravděpodobně předpokládaly určitou vstupní investici a navíc by jejich další rozvoj mohl být problematický.

- Mělo by se jednat o technologii široce používanou, s velkou podporou.
- Technologie by měla být dostupná buď zdarma, nebo se jednat o technologii, kterou má Společnost licencovanou.
- Technologie musí umožňovat spolupráci s relační databází.
- Technologie nemusí být optimalizována pro nejvyšší výkon, ale spíše pro jednoduchost správy, vývoje a případných dalších úprav.

Uvedená kritéria splnily tři platformy:

- Platforma .NET, Společnost vlastní licence pro použití jejích vývojových prostředků a používá je již ve svých jiných projektech.
- Platforma Java, v požadovaných verzích je dostupná zdarma.

6.4.1. Platforma .NET

Platforma .NET je vyvíjena společností Microsoft [44]. Jedná se o rozsáhlou platformu, která obsahuje mnoho programovacích jazyků pro různá řešení. Všechny jazyky jsou vždy kompilovány do kódu platformy a díky tomu je možné používat na jedné stanici s podporou platformy jakákoliv její řešení (samotný přeložený kód je spouštěn na virtuálním stroji). Vývojové prostředky platformy jsou sice placené, ale Společnost již vlastní licence a zabývá se tvorbou řešení postavených na platformě. K platformě je široká podpora společnosti Microsoft a existuje k ní mnoho komunitních stránek, potencionální problémy tak pravděpodobně budou mít jednoduše dostupné řešení. Stejně tak k platformě díky jejímu použití existuje široká nabídka frameworků a metodik vývoje. Platforma

obsahuje i prostředky pro spolupráci s databázemi, její použití je podmíněno buď instalací operačního programu MS Windows, popřípadě instalací rozšiřujícího balíčku na starším operačních programech Windows.

6.4.2. Platforma Java

Platforma Java je vyvíjena společností Sun a oproti .NETu nabízí naprostou nezávislost na operačním systému. Opět se jedná o jazyk překládaný do kódu platformy a spustitelný na virtuálním stroji, tedy jazyk také vyžaduje určitou instalaci podpůrného prostředí u klienta. Platforma je v požadovaném rozsahu dostupná zdarma, existuje k ní několik vývojových prostředí, je s ní svázána široká komunita a tedy nabízí podporu. Jazyk velmi dobře spolupracuje s databázemi a umožnil by vývoj řešení. Bohužel jazyk neobsahuje některé komponenty a předpokládá buď jejich vlastní vývoj, nebo nákup (a tedy i dodatečnou investici) od dodavatele. Dále jazyk díky tomu, že je dostupný i zdarma nemá tak kvalitní podporu jako platforma .NET. Navíc Společnost již ukončila vývoj svých aplikací na platformě Java a jeho volba by tedy byla nekonzistentní vzhledem ke strategii Společnosti.

6.4.3. Volba technologie

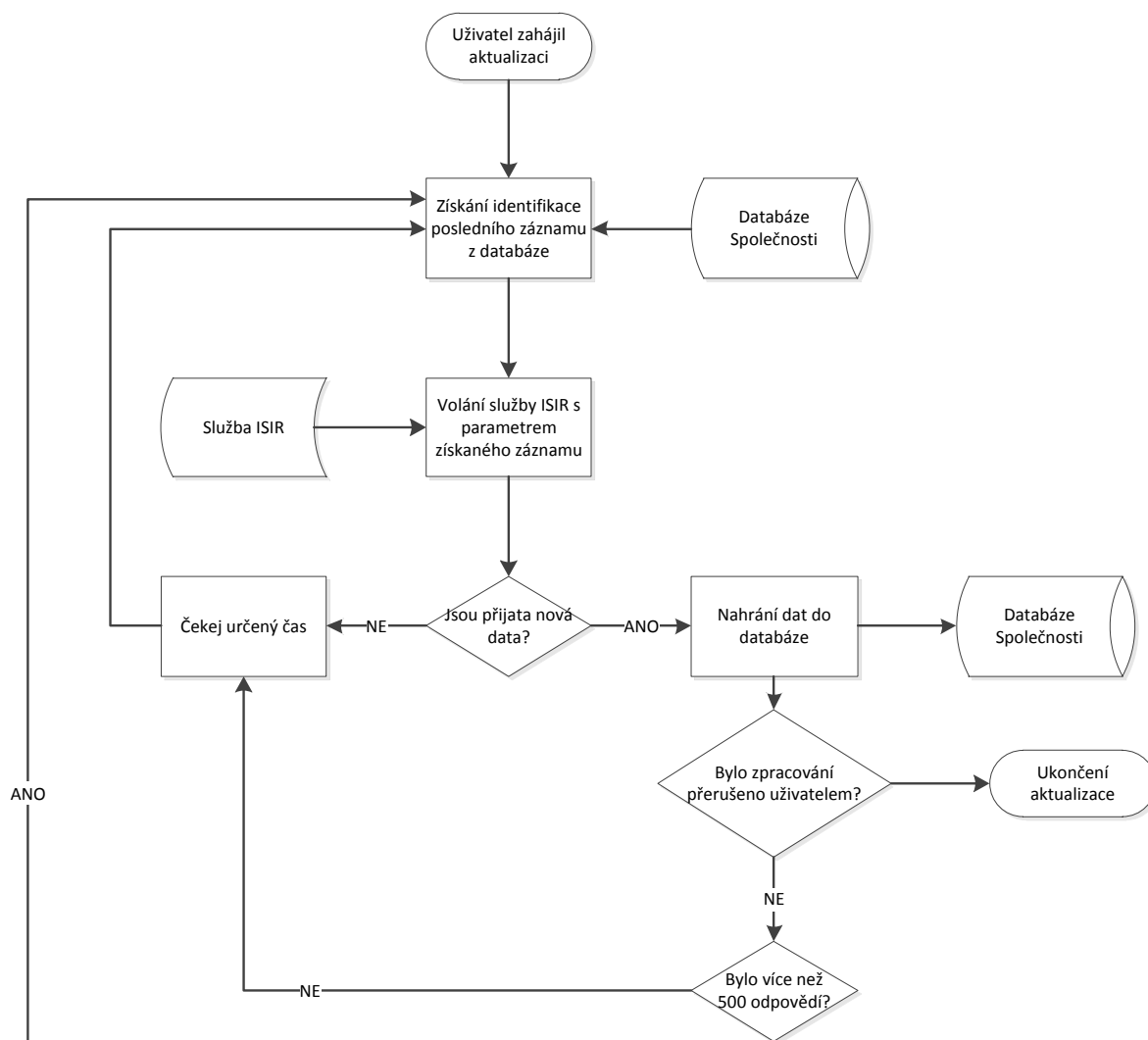
Na základě předchozích bodů a po konzultaci se zástupcem Společnosti jsem se rozhodl vytvořit aktualizací program pomocí platformy .NET. Ačkoliv základní požadavky splnily obě zmíněné možnosti, rozhodl jsem se především kvůli zkušenostem s vývojem na platformě, jak mých, tak Společnosti. Dále proto, že větší část aplikačního portfolia Společnosti funguje na aplikacích od společnosti Microsoft a jedná se tak o konzistentní řešení.

6.4.4. Detailní popis aktualizací programu

Po volbě technologie použité k realizaci aktualizací programu již mohu přistoupit i k detailnímu popisu jeho chování a architektury. Nejdříve uvádím popis chování aktualizací programu z procedurálního pohledu u kterého vycházím z požadavků Společnosti, jak byly formulovány v kapitole 5.3.2. Následuje model datových tříd poskytovaných službou ISIR, proces nahrání dat do databáze a celý popis je ukončen modelem tříd aktualizací programu, což odpovídá architektuře programu. Tímto tedy definuji chování programu, data na vstupu, metodu pro zpracování dat a uspořádání programu.

6.4.4.1. Proces Aktualizace

Proces aktualizace je na obrázku 19 a je formulován za předpokladu ideálního chodu, tedy v případě, že nenastanou komplikace například s připojením k databázi, nebo službě ISIR. V případě výskytu komplikací program vypíše chybovou zprávu a přeruší zpracování podle typu komplikace. Při problémech s připojením na určený čas, při ostatních (např. nevalidní data) zcela a neumožní další aktualizaci.



Obrázek 19 proces nahrávání dat aktualizacním programem (Zdroj: Autor)

Z popisu požadavků Společnosti i z procesu aktualizace je vidět, že proces aktualizace vždy ukončen jen v případě akce uživatele a pokud tato nenastane, aktualizace probíhá nekonečně dlouho.

6.4.4.2. Model tříd poskytovaných ISIR

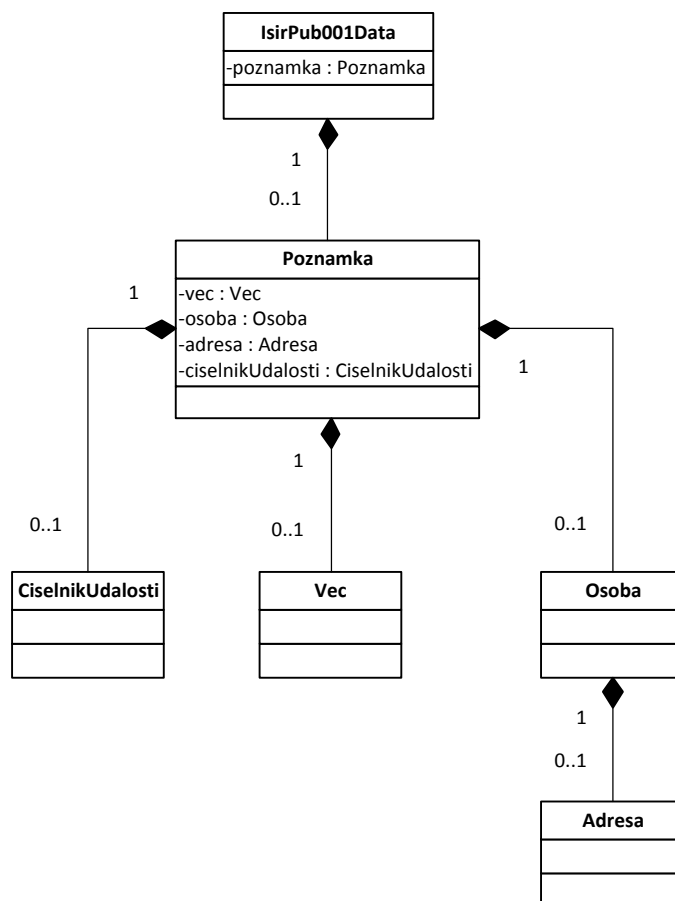
Webová služba ISIR poskytuje data ve formátu XML. Aktualizační program musí nejprve službu zavolat, specifikovat požadavek na data (tedy buď id, nebo datum), data přijmout, transformovat do objektové podoby a následně uložit do relační databáze.

Transformaci lze za předpokladu, že webová služba nabízí svůj interface pomocí formátu WSDL, přenáší data protokolem SOAP a data jsou přenášena v XML struktuře viz [45], provést několika způsoby, kdy nejvhodnějšími je buď vytvořit vlastní mechanismus, který přesně prochází XML strukturu a vytváří programátorem definované objekty, nebo automatické převedení, kdy v se ve vývojovém nástroji referencuje webová služba a vývojový nástroj vytvoří třídy pro práci jak se službou, tak s daty, které poskytuje. Metoda je popsána např. v [46] , nebo [47]. Největší rozdíl mezi

metodami je přenesení odpovědnosti za transformaci a vhodně vytvořené datové objekty s funkcionalitou na stranu tvůrců webové služby a tvůrců vývojového prostředí při automatickém převedení. Pro obě metody uvádím výhody a nevýhody.

- Vlastní tvorba transformace přináší hlavní výhodu v tom, že programátor přesně ví, které části navrácených dat jsou pro něj důležité a již na vstupu do systému převede data do nejvhodnější možné podoby. Omezuje využití paměti, protože nevytváří pro systém nepodstatné části a zároveň může již k datovým třídám připojit poměrně velkou částí funkcionality (především univerzální) bez obav z toho, že nová verze webové služby jeho funkcionalitu nebude podporovat. Dále také snižuje šanci, že kvůli nevhodně definované webové službě se ve vygenerovaných objektech objeví chyby. Hlavní nevýhoda leží v tom, že řešení je zdoluhavější, vyžaduje dobré porozumění poskytovaným datům, způsob získávání dat nemusí být optimalizovaný a zároveň v případě větší změny webové služby může dojít k tomu, že bude nutno předělat buď proces transformace, případně i datové objekty, protože již nebudou odpovídat datům služby.
- Automatické generování objektů podle reference webové služby je extrémně rychlé, stačí službu referencovat a vývojové prostředí díky poskytnutému popisu služby i dat vytvoří jak prostředky ke komunikaci se službou, tak objekty k práci s jejími daty. Zároveň vytvoří i transformační mechanismus mezi XML a objekty dat. Další výhodou je spolehlivost tohoto řešení, vycházející ze standardního použití mnoha subjektů, ovšem za předpokladu že tvůrce webové služby poskytne validní specifikaci (příklad nevalidního je vidět v [24] v části "Generování klienta služby podle WSDL"). Mezi nevýhody pak patří šance, že tvůrce používá nevhodné názvy (opět vidět v předchozím zdroji), šance že vygenerované objekty nebudou vůbec schopny s webovou službou komunikovat a díky velkému rozsahu generovaného kódu bude velmi obtížné najít a odstranit chybu, větší využití paměti při použití velkých objektů, které nejsou využívány a nutnost transformovat získané objekty dat v některých případech do dalších objektů, vhodných pro systém.

Rozhodl jsem se použít variantu automatického generování prostředků pro komunikaci se službou a to především z důvody úspory času a optimálního převodu dat služby do objektové podoby. Variantu jsem vyzkoušel na testovacím programu a vygenerované prostředky vracely validní odpovědi bez jakýchkoliv problémů. Po tomto rozhodnutí tedy mohu uvést model tříd poskytovaných službou ISIR. Model bych mohl opět jednoduše vygenerovat z vývojového prostředí, ale takto vytvořený přesný model je nepřehledný, proto uvádím vlastní zjednodušený. V modelu jsem vynechal většinu atributů, které jsou již popsány v kapitole 6 a přejmenoval jsem objekty. Model odpovídá vygenerovaným objektům webové služby a je na obrázku 20.



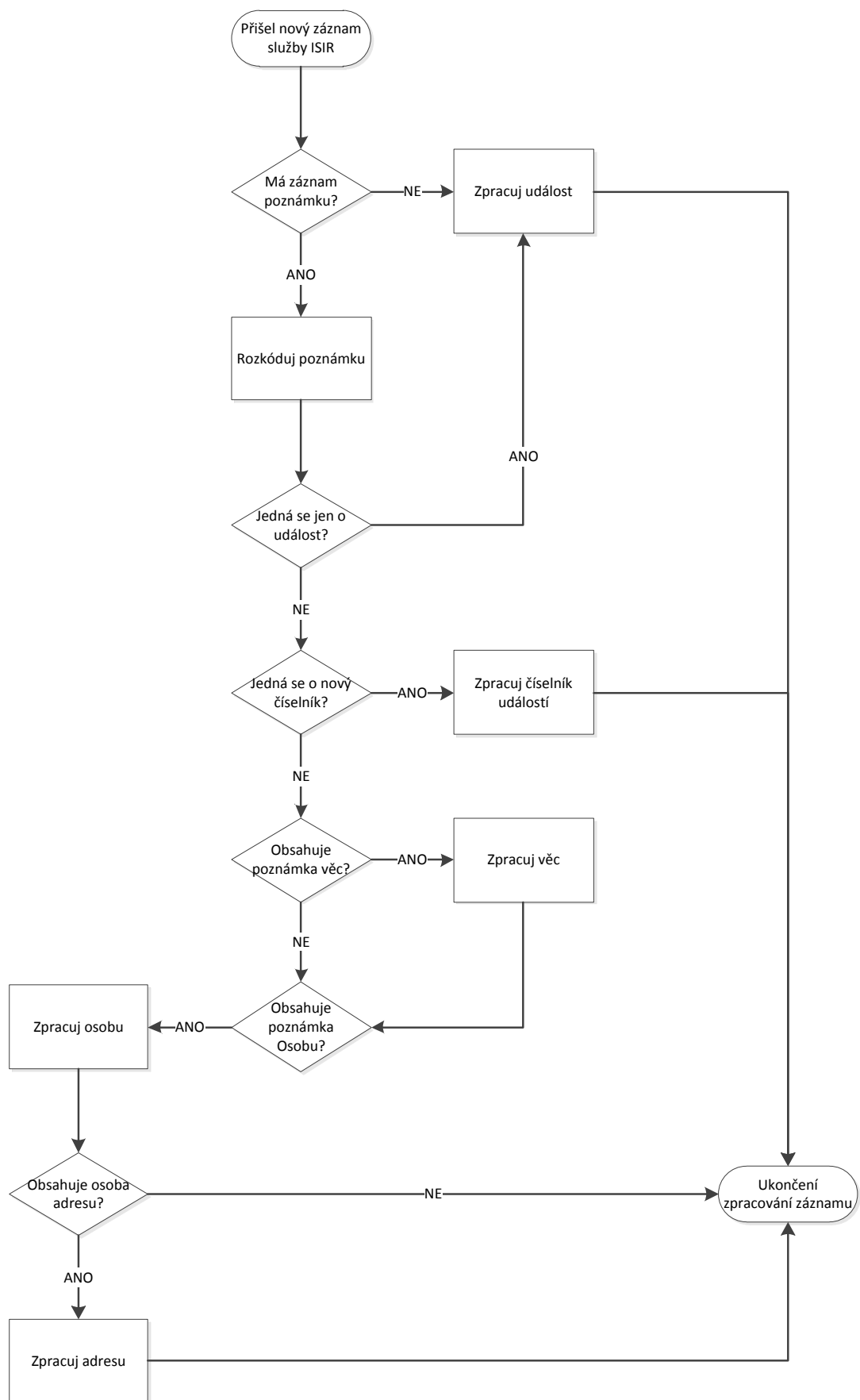
Obrázek 20 model objektů poskytovaných službou (Zdroj: Autor)

6.4.4.3. Proces nahrání dat do databáze

Proces "Nahrání dat do databáze" je podprocesem procesu aktualizace na obrázku 19. Nahrání dat musí správně převést službou poskytovaná data v objektech (viz obrázek X) na data vhodná k uložení do databáze a tato uložit. Tento proces je tedy hlavní částí aplikační logiky aktualizací programu. Při jeho formulaci je důležité si uvědomit, že model databáze a model dat poskytovaných službou nejsou zcela totožné, toto vychází z podstaty služby, která je replikací databáze. Chování procesu potom vychází především z kapitoly 2.6 a vyjádřil jsem ho takto:

- Pokud záznam nemá poznámku, jedná se o jednoduchou událost, program zpracuje událost a dále nepokračuje.
- Pokud záznam obsahuje poznámku, program ji rozkóduje. Pokud poznámka obsahuje číselník událostí jedná se o změnu/přidání číselníku událostí, program ho zpracuje a dále nepokračuje.
- Pokud poznámka neobsahuje číselník událostí, ale neobsahuje ani věc, nebo osobu, jedná se o událost, program zpracuje událost a dále nepokračuje.
- Pokud poznámka obsahuje věc, program ji zpracuje a pokračuje dále.
- Pokud poznámka obsahuje osobu, program ji zpracuje a pokračuje dále.
- Pokud osoba obsahuje adresu, program ji zpracuje a pokračuje dále.

Proces nahrání dat je na obrázku 21.



Obrázek 21 proces nahrání dat do databáze (Zdroj: Autor)

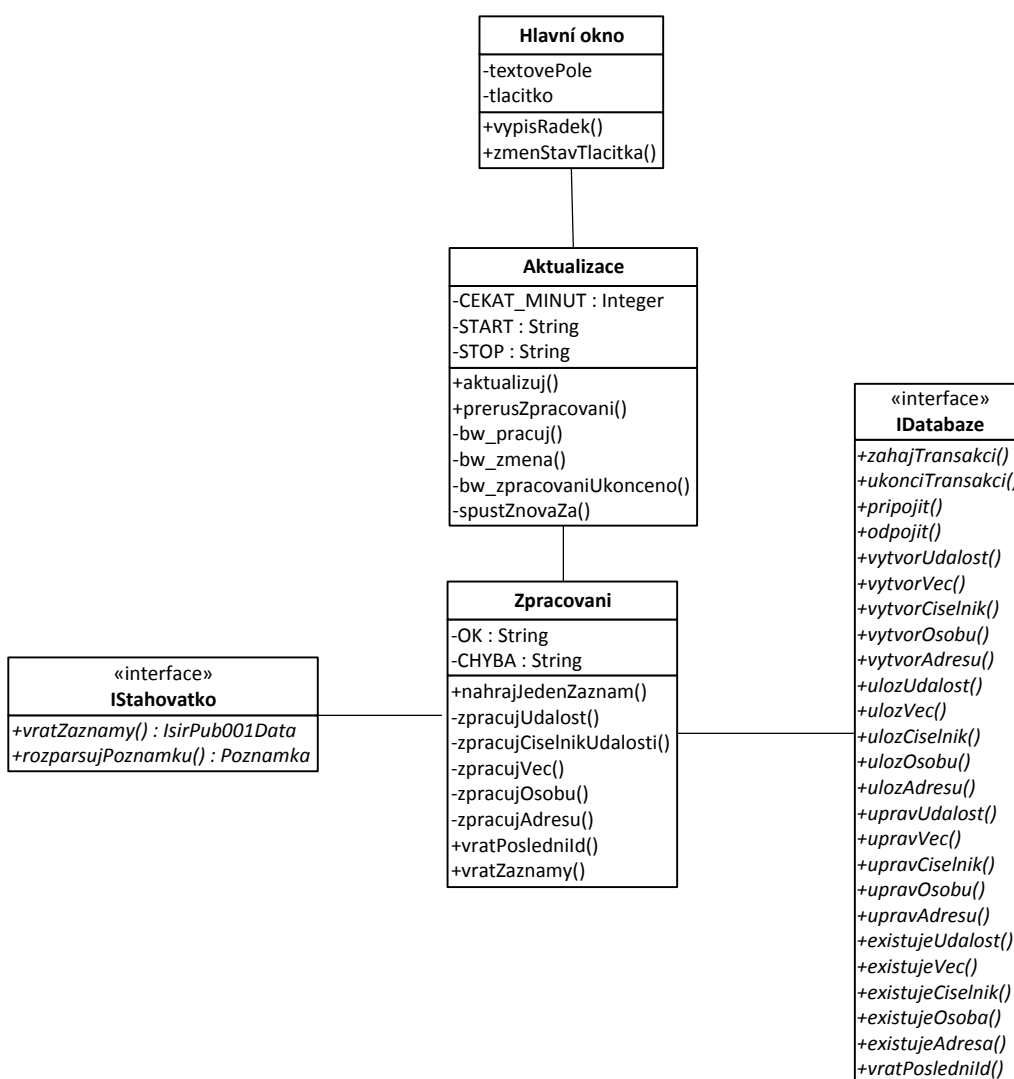
6.4.4.4. Model tříd aktualizačního programu

V návaznosti na předchozí kapitoly, které definovali jak data poskytovaná službou, tak úložiště dat, tedy cíl dat a způsob chování aktualizačního programu jsem vytvořil model tříd, tedy architekturu aktualizačního programu. Při jeho tvorbě jsem se řídil především zásadami popsány v [10]. Architekturu programu zde chápu podle definice z [10] a to jako "to co se nemění". Cílem architektury je vytvořit dostatečně flexibilní řešení, které umožňuje jak změny služby ISIR na straně jedné, tak případnou změnu databáze na straně druhé. Zároveň by neměla existovat silná vazba mezi uživatelským rozhraním a výkonnou částí programu. Při tvorbě modelu jsem postupoval takto:

- Vytvořil jsem třídu "Hlavní okno" reprezentující uživatelské rozhraní. Třída musí umět zobrazovat zprávy a proto jsem jí přiřadil metodu "vypisRadek". Dále musí umět měnit popis tlačítka mezi stavy "start" a "přerušit", proto jsem přidal metodu "zmenStavTlacitka". Dále jsem třídě přiřadil atributy symbolizující textové pole a tlačítko.
- Vytvořil jsem třídu "Aktualizace" reprezentující hlavní logiku programu. Protože předpokládám, že Společnost by nebyla spokojena s aplikací, jejíž uživatelské rozhraní při výkonu zpracování nereaguje, bude potřeba aplikaci vytvořit za využití více vláknového zpracování viz např. [47], tomu odpovídají tři soukromé metody "bw_pracuj", "bw_zmena" a "bw_zpracovaniUkonceno". První metoda zahajuje práci vlákna, druhá slouží k hlášení změny ve zpracování metodou a třetí činnost, která proběhne při ukončení vlákna. Další soukromou metodou je potom metoda "spustZnovaZa", která slouží k přerušení činnosti aktualizačního programu na definovaný čas. Veřejné metody potom slouží k ovládání celého programu a tedy metoda "aktualizuj" zahajuje činnost programu a metoda "prerusZpracovani" k přerušení zpracování, metody budou používané uživatelským rozhraním. Třída obsahuje i tři konstanty, CEKAT_MINUT specifikuje čas, který má aktualizační program čekat mezi jednotlivými pokusy o nahrávání, START a STOP potom obsahují texty pro tlačítko uživatelského rozhraní.
- Třída Zpracovani obsahuje dvě konstanty OK a CHYBA, používané k indikaci stavu zpracování, metodu pro získání identifikace posledního záznamu z databáze "vratPosledniId", metodu pro získání seznamu odpovědí webové služby "vratZaznamy", metodu pro zpracování jednoho záznamu "nahrajJedenZaznam" a následně soukromé metody třídy pro zpracování jednotlivých entit odpovídající procesu nahrání dat do databáze. Třída Zpracování bude statická, tedy při běhu programu se nevytváří žádná instance této třídy a neuchovává si žádná data mimo jednotlivé metody a konstanty. Stejně tak její metody jsou všechny statické.
- Interface IStahovatko slouží k definování nutného rozhraní, které musí splňovat třída, aby bylo možné ji používat ke komunikaci s webovou službou. Rozhraní definuje dvě metody, které případné třídy jej implementující musí obsahovat, metoda "vratZaznamy" slouží k získání seznamu záznamů typu IsirPub001Data od webové služby. Metoda

"rozparsujPoznamku" potom k převodu poznámky na datové typy, které může obsahovat (událost, věc atd.). Rozhodl jsem se definovat rozhraní, aby bylo případně možné používat různé varianty třídy komunikující s webovou službou, například v případě její změny.

- Interface IDatabase definuje nutné rozhraní, které musí obsahovat třída sloužící ke komunikaci s databází. Zvolil jsem rozhraní z toho důvodu, že v případě změny databáze stačí změnit jen třídu implementující rozhraní a zbytek programu může zůstat beze změny. Interface definuje metody ovládající transakce, připojení a odpojení od databáze a dále metody pro vytvoření objektů pro uložení do databáze, metody pro uložení dat do databáze, metody pro úpravu dat v databázi a metody pro zjištění existence v databázi.



Obrázek 22 model tříd aktualizací programu (Zdroj: Autor)

6.5. Monitorovací program

Monitorovací program by měl splňovat požadavky popsané v kapitole 2.1.3.3 a zároveň poskytovat základní grafické rozhraní určené k jeho obsluze, tak jak je navrženo v kapitole 2.1.4.2. K dosažení

těchto cílů by opět šlo použít různé programovací jazyky, jako v předchozí kapitole. Ale z důvodu úspory místa vynechám delší rozbor a jen uvedu další výhody použití platformy .NET.

- Platforma poskytuje excelentní zabudovanou podporu spolupráce s kancelářskými aplikacemi MS Office. Navíc tato podpora je kontinuální, tedy s další verzí MS Office jsou upraveny i nástroje platformy pro spolupráci. Vzhledem k tomu, že všichni klienti Společnosti používají dokumenty aplikací MS Office, je tato podpora velmi důležitá.
- Vzhled programů platformy je prakticky stejný, jako vzhled operačního programu Windows a programů od společnosti Microsoft. Toto opět usnadňuje použití aplikací klienty.
- Jedná se o konzistentní technologii používanou Společností, v tomto případě by bylo zbytečné používat jiné prostředky řešení.

6.5.1. Volba technologie

Na základě předchozích bodů a po konzultaci se zástupcem Společnosti jsem se rozhodl vytvořit monitorovací program pomocí platformy .NET.

6.5.2. Detailní popis monitorovacího programu

Po volbě technologie opět můžeme přistoupit k detailnímu popisu monitorovacího programu. Vzhledem k jeho odlišné funkcionalitě od aktualizací se budou lišit i prostředky pro jeho dostatečný popis. Vstupní data jsou v tomto případě dána kapitolou 2.4.2 a váže se k nim jen případná validace některých polí (například kontrola validity zadaného RČ/IČ), případná další funkcionalita, jako import z tabulek MS Excel a podobné bude předmětem až dalších rozšíření programu. Nejprve se tedy věnuji uložení dat, popisu funkcionality monitorovacího programu, popisu metody vyhledávání a kapitolu opět uzavírám modelem tříd programu.

6.5.2.1. Volba ukládání dat

Co se týče uchovávání dat o dlužnících a jejich vyhledávání, požadavek Společnosti byl ukládat tato data do souboru (viz. kapitola 2.3.3). Jako formát ukládaných dat jsem zvolil XML soubor a to především z těchto důvodů:

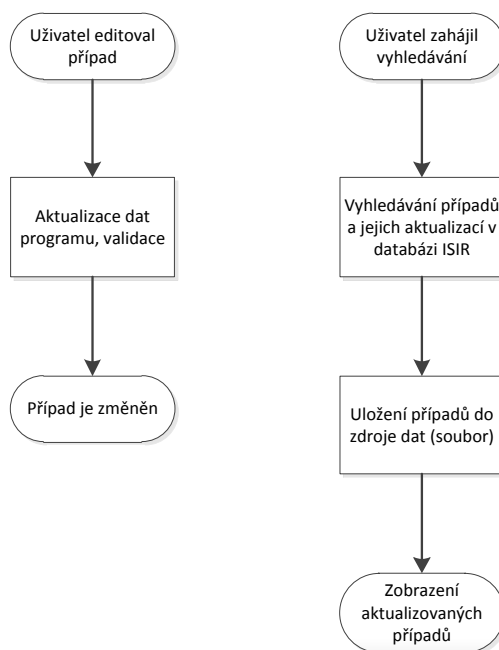
- XML soubor lze jednoduše nahrávat a ukládat pomocí prostředků .NET.
- Prostředí .NET umožňuje uložení dat v objektech do XML bez silné vazby na definici dat, tedy při změně definice objektů dat se automaticky při jejich použití změní i XML soubor. Stejně tak při načítání XML souboru jsou ignorována přebytečná data v něm.
- XML soubor je poměrně dobře čitelný i za použití textových editorů.

6.5.2.2. Procedurální popis funkcionality

Vzhledem k popisu požadavků Společnosti na funkcionalitu monitorovacího programu jsem definoval hlavní procesy, které musí program umožňovat:

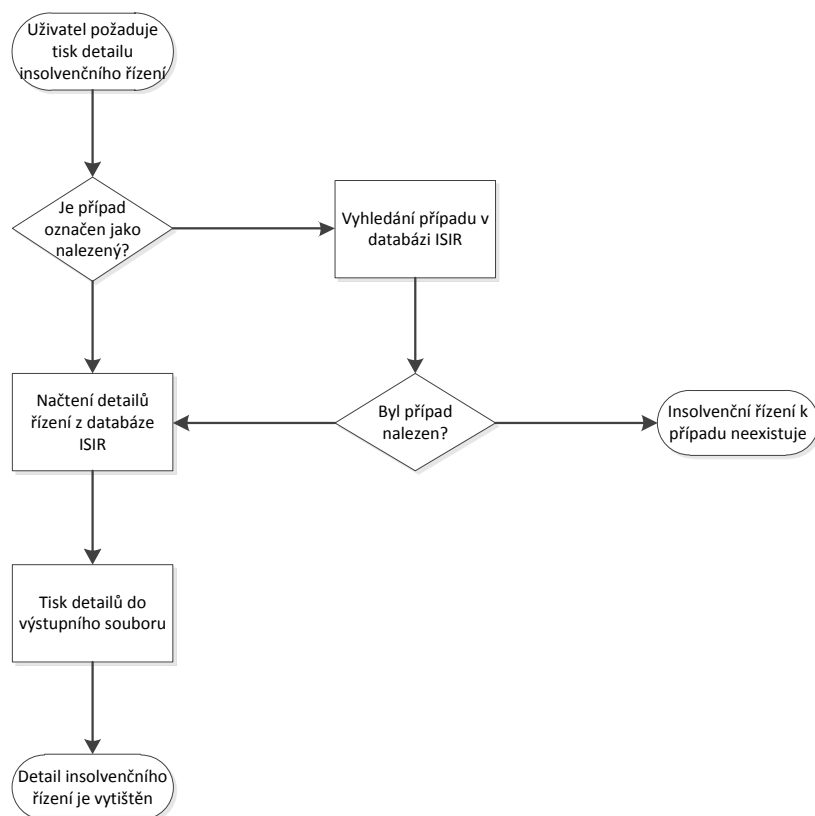
- Editace (přidání, smazání) případů/osob k vyhledání.
- Vyhledání případů v databázi ISIR.
- Tisk detailu nalezeného insolvenčního řízení.

První dva procesy jsem z důvodu úspory místa a jejich jednoduchosti umístil na jednom obrázku, procesy jsou na obrázku 23.



Obrázek 23 procesy editace a vyhledání monitorovacího programu (Zdroj: Autor)

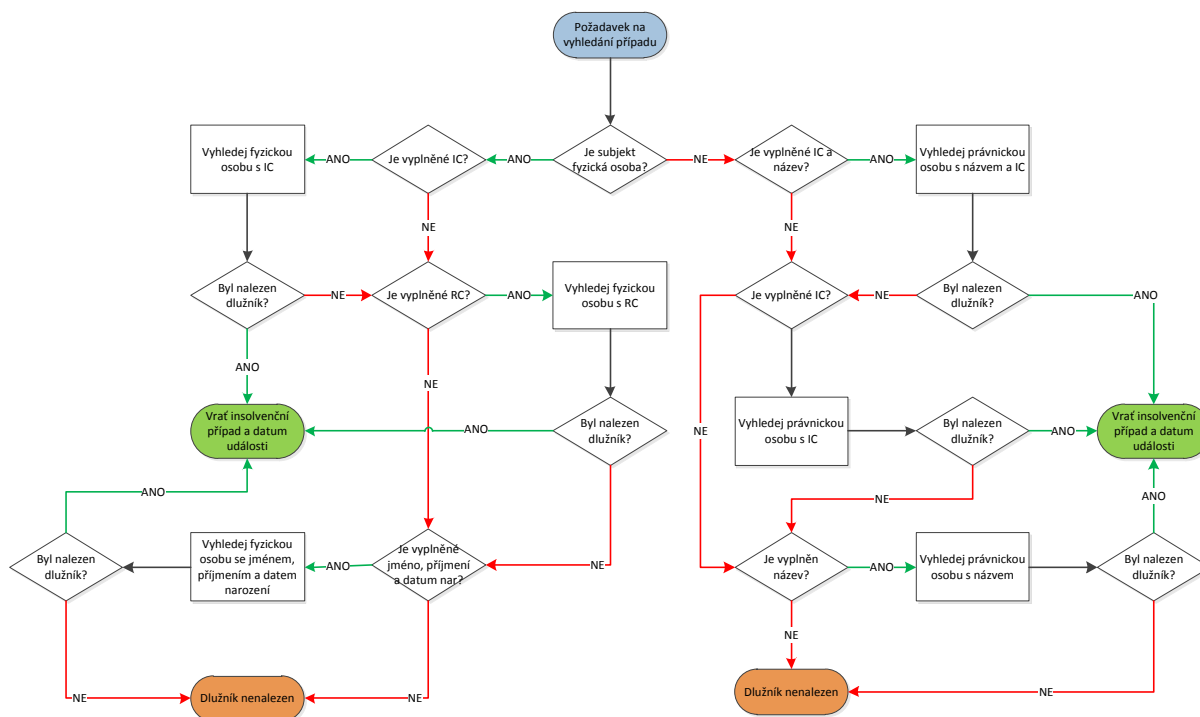
Třetím a komplikovanějším procesem je potom tisk detailu případu. Ten by měl sloužit k zobrazení vyčerpávajícího výpisu celé historie insolvenčního řízení a měl by se podobat výpisu z ISIR v kapitole 2.6. Proces se v případě akce uživatele pokusí vyhledat daný případ a pokud je úspěšný, vypíše detail řízení do souboru. Proces je na obrázku 24.



Obrázek 24 proces tisku detailu insolvenčního řízení (Zdroj: Autor)

6.5.2.3. Popis metody vyhledávání

Vzhledem k přesunutí samotného vyhledávání z monitorovacího programu na webovou službu prohledávání zbývá monitorovacímu programu pouze určit, jakým způsobem webovou službu zavolat. Protože obecně je preferovaný jistý nález (tedy jistotu že se jedná o vyhledávaný subjekt a ne jen o shodu jmen), monitorovací program musí postupovat od Identifikačního čísla a rodného čísla ke jménu, příjmení a datu narození, nebo názvu subjektu, ale na rozdíl od webové služby monitorovací program by se měl postupně pokusit nalézt výskyt podle všech validních možností vyhledávání. Tento postup vyplynul z konzultací se zástupcem Společnosti a zároveň vede k nižšímu zatížení webové služby, kdy pravděpodobnější nález sníží šanci na potřebu dalšího dotazu. Proces, metoda vyhledání je na obrázku 25. Opět jsem pro vyšší přehlednost obarvil určité varianty rozhodovacích částí procesu a dále také úspěšné a neúspěšné konce.



Obrázek 25 metoda vyhledávání monitorovacího programu (Zdroj: Autor)

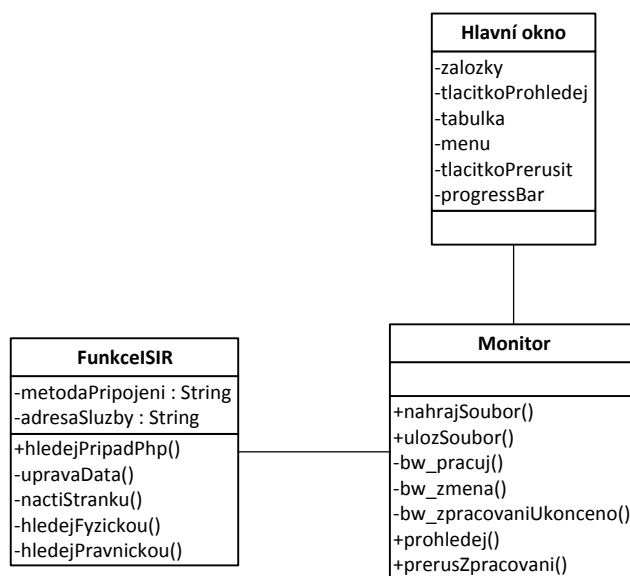
6.5.2.4. Model tříd monitorovacího programu

Stejně jako v případě aktualizací programu jsem vytvořil model tříd, tedy architektonický pohled monitorovacího programu. Řešení tentokrát není tak univerzální co se týče definování rozhraní a jeho hlavním cílem je oddělit funkcionalitu tříd podle jejich zaměření. Ve výsledku je jedna třída zodpovědná za uživatelské rozhraní, jedna za správu dat programu a volání vyhledávání a jedna starající se o samotné vyhledávání v databázi ISIR skrze webovou službu vyhledávání. Tento model umožňuje implementaci jak původních požadavků Společnosti, tak funkcionality popsané výše. Model je na obrázku 26 a při jeho tvorbě jsem postupoval takto:

- Vytvořil jsem třídu "Hlavní okno" zodpovědnou za uživatelské rozhraní. Třída obsahuje především grafické objekty, jak byly zobrazeny v kapitole 2.4.2 a to "TabControl" (záložky), tlačítka pro zahájení a přerušení prohledávání, tabulku pro zobrazení případů a indikátor postupu vyhledávání.
- Vytvořil jsem třídu "Monitor" zodpovědnou za správu datového zdroje (souboru s dlužníky) a zpracování (tedy zastínění detailů vyhledávání před uživatelským rozhraním). Metoda "nahrajSoubor" slouží k nahrání zdrojového souboru s dlužníky (případy) a "ulozSoubor" potom k jeho uložení. Metody "bw_pracuj", "bw_zmena" a "bw_zpracovaniUkonceno" slouží k řízení více vláknového zpracování. Metoda "prohledej" se potom stará o správné použití vyhledávací třídy ("FunkceISIR"), předání vyhledávacích parametrů a následně opět předání výsledků uživatelskému rozhraní ke zobrazení. Metoda "prerusZpracovani" je potom pro

možnost validního přerušení zpracování. Třída obsahuje ještě interní třídu "Pripad", což je obalová třída pro data podle návrhového vzoru přepravka viz. [10].

- Třída "FunkceISIR" je potom funkční částí obstarávající využití webové služby vyhledávání. Tentokrát jsem nezvolil rozhraní (interface) jako u aktualizací programu z toho důvodu, že webová služba vyhledávání je zcela v mé režii (případně Společnosti) a tedy nepředpokládám nečekané změny, které by těžily z výhod rozhraní. Třída má atributy specifikující způsob připojení ke službě (http/https) a adresu služby. Metoda "hledejPripadPhp" potom slouží k zapouzdření celé logiky vyhledávání, tak jak byla popsána v předchozí kapitole. Metoda "upravaData" k převedení zadávaných dat do formátu používaného webovou službou a metoda "nactiStranku" ke komunikaci s webovou službou. Metody "hledejFyzickou" a "hledejPravnickou" potom využívají metodu "nactiStranku" jsou přetíženy s různou kombinací hledaných parametrů dlužníků. Tedy metoda "hledejPripadPhp" využívá různě parametrizované metody "hledejFyzickou"/"hledejPravnickou" ke zjištění existence dlužníka v databázi ISIR.



Obrázek 26 model tříd monitorovacího programu (Zdroj: Autor)

7. ANALÝZA/MODEL IMPLEMENTAČNÍ ISIR

Analýza implementační navazuje na předchozí části a vychází ze specifikace popsané v kapitole 4.1.3. Jejím smyslem je popsat a zvolit konkrétní prostředky pro dosažení funkcionality Systému. Výstupem bude tedy volba programovacího jazyka pro aktualizací a monitorovací program, volba databázové technologie a fyzický datový model databáze.

7.1. Úložiště dat

Úložiště dat bude realizováno pomocí technologie relační databáze. Jelikož se jedná o široce používanou technologii, můžeme pod pojmem narazit na větší množství konkrétních produktů. Pro volbu tak vybírám jen velmi známá řešení a níže uvádím kritéria, podle kterých jsem se rozhodoval.

7.1.1. Kritéria rozhodování

Jelikož prakticky všechna uvedená řešení splňují požadavky Společnosti na datové úložiště, kritéria se odvíjejí především od ceny samotného úložiště, ceny nástrojů k jeho ovládnutí, míry podpory (jak oficiální, tak komunitní) a zkušeností Společnosti s řešením.

7.1.2. Microsoft SQL Server

Jedná se o produkt společnosti Microsoft, který umožňuje správu a podporu mnoha dalším nadstavbám a aplikacím. Pomocí této datové základny jsou řešeny aplikace ERP, BI, CRM a další. Server umožňuje snadnou správu pomocí integrovaného ovládacího rozhraní SQL Server Management Studio, umožňuje přístup mnoha technologiím, splňuje nárok na použití SQL dotazovacího jazyka, má vysokou úroveň zabezpečení a zároveň přináší i nástroje pro snadnou migraci dat. Na druhou stranu komerční využití serveru je nákladné (viz Microsoft) a Společnost by tak musela investovat do jeho využití (ať už nákupem licence, nebo nákupem v MS cloudu). Míra podpory je vysoká, jak na stráně oficiální (MSDN), tak na mnoha komunitních stránkách. Dále Společnost toto řešení nikdy nevyužívala.

7.1.3. MySQL

Je databázovým řešením vyvíjeným společností Sun (později koupená Oracle). Nabízí jak komerční, tak volné využití a navíc je častým řešením používaným u hostitelů stránek. Společnost již databázi v mnoha řešeních využívala a v současné době vlastní několik připravených MySQL databází. Cena je v tomto případě prakticky nulová, protože by šlo využít již existující databázi. K ovládnutí databáze lze použít například phpAdmin, široce používané prostředí pro správu MySQL databází postavené na skriptovacím jazyku PHP, nebo MySQL administrator tools, případně novější verzi MySQL Workbench. Míra podpory je opět vysoká a to jak komunitní, tak případně na oficiálních stránkách.

7.1.4. Volba úložiště

Splnění výše definovaných kritérií shrnuje tabulka 6 a podle ní jsem se nakonec rozhodl použít k ukládání dat databázové řešení MySQL.

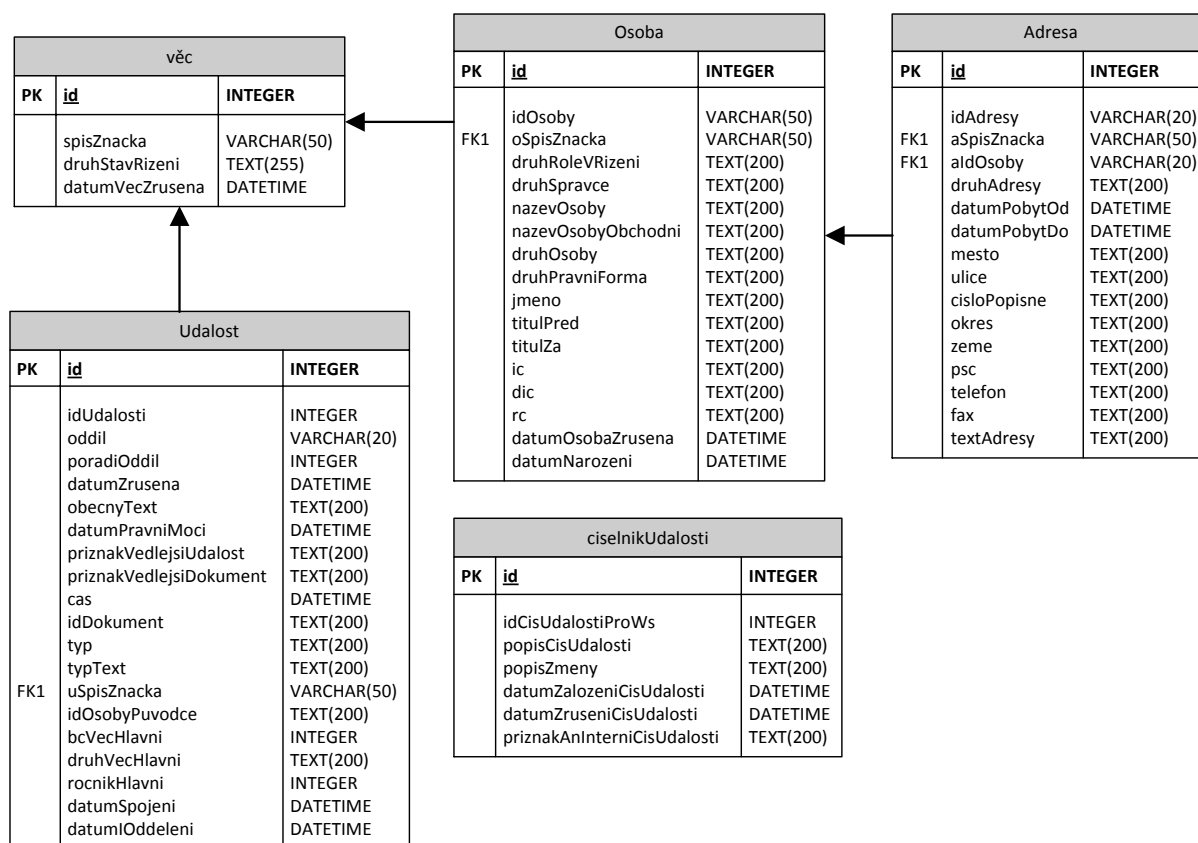
Tabulka 6 porovnání databází podle kritérií (Zdroj: Autor)

Kritérium	MS SQL Server	MySQL
Cena	Vysoká	Nulová
Podpora	Vysoká	Vysoká
Nástroje	Dobré	Dobré

Zkušenosti Společnosti	Nízké	Vysoké
------------------------	-------	--------

7.2. Fyzický datový model

Při konstrukci fyzického datového modelu jsem vycházel z modelu konceptuálního, zobrazeného na obrázku X v kapitole 5.7. Vzhledem k mým sporným zkušenostem s generováním fyzických modelů a také volbou technologie k formulaci konceptuálního modelu MS Visio 2010, která nepodporuje generování SQL (forward engineering není od verze 2003 podporován) jsem se rozhodl vytvořit fyzický model pomocí nástrojů správy databáze MySQL Workbench. Vedle toho jsem se rozhodl vytvořit i model v nástroji Visio, abych ho v případě potřeby mohl zaslat třetí straně. Model bych sice mohl kdykoliv zaslat i jako SQL skript, nebo jako obrázek vytvořený v nástroji správy, ale Visio se mi již v minulosti osvědčilo. Navíc lze zaslat i samotný model, který lze jednoduše editovat. Fyzický model je na obrázku 27.



Obrázek 27 fyzický model databáze (Zdroj: Autor)

7.2.1. Obhajoba řešení fyzického modelu

Fyzický model databáze nesplňuje některé zásady tvorby databází popsané například v [33]. Každé tabulce jsem přiřadil umělý identifikátor typu integer, který je generován databází, ačkoliv tabulky již obsahují své identifikátory z databáze ISIR a to z důvodu, že v minulosti, když jsem pracoval s databází ISIR jsem narazil na občasné nekonzistence v jednoznačných identifikátorech, respektive neošetřenou diakritiku v primárních klíích. Navíc Společnost vyžaduje používání jednoznačných

identifikátorů typu integer u všech svých databází. Co se týče normalizace databází, existují různé názory a jako velmi výstižný mi přijde pojem "přirozená míra normalizace" zmíněný například v přednášce z MS festu 2011 [MS 30], podle tohoto zdroje dále řešení s umělým identifikátorem poskytuje tyto výhody:

- neměnnost
- velmi rychlé SQL operace (JOIN atp.)
- monotónní růst = vhodný pro clustered index
- zajišťuje unikátnost klíčů pro nové záznamy

Další odchylkou modelu je neexistence referenční integrity v databázi. Cizí klíče (např. uSpisZnacka) odkazují na atribut spisZnacka, který není primárním klíčem. Ačkoliv by se toto mohlo zdát jako nevhodné řešení, tak je přijatelné, protože databáze je pouhou kopií databáze ISIR a data se z databáze nemažou. Deaktivace, respektive neplatnost záznamů je řešena pomocí atributy "datumZrušení", popřípadě "datumPobytDo" u entity Adresa. Navázání cizích klíčů na nepřímé klíče tak poskytuje určitou úsporu místa, kdy se nemusí vytvářet další atribut s identifikátorem typu integer. Druhým řešením, které by poskytovalo referenční integritu by bylo vynechání současných cizích klíčů (např. uSpisZnacka) a jejich nahrazení cizím klíčem typu integer, ale po konzultaci se zástupcem Společnosti jsme se rozhodli raději zachovat cizí klíče poskytované databází ISIR. Zároveň se tak objevuje i informace o insolvenčním řízení, ke kterému osoba/událost patří u těchto entit bez potřeby dotazovat se dále na danou věc.

7.3. Aktualizační a monitorovací program

Vzhledem k volbě platformy .NET pro vývoj jak aktualizačního, tak monitorovacího programu zbývá zvolit jeden z možných přístupů a následně konkrétní programovací jazyk pro vývoj. Zvolený jazyk by měl splňovat nároky na realizované programy, tedy možnost zobrazovat relativně přívětivé a ideálně standardní uživatelské rozhraní, komunikaci s databází a webovými službami. Pokud se zaměříme na nejnovější podporované možnosti řešení, tak se nabízejí dvě:

- WPF (Windows Presentation Foundation) technologie zaměřená na oddělení vzhledu a logiky aplikace. Uživatelské rozhraní je formulováno především pomocí jazyka XAML, který je vychází ze značkovacího XML. Řešení silně podporuje i separátní vývoj, kdy uživatelské rozhraní je vytvářeno grafikem pomocí nástrojů Expression tools a logika aplikace programátorem pomocí Visual studia.
- WinForms, starší technologie vývoje klasických aplikací operačního systému Windows. Přebírá vzhled operačního systému a je zaměřena především na snadný a rychlý vývoj typické aplikace s databází, uživatelským rozhraním a spoluprací s MS Office především pomocí událostmi řízeného modelu.

Protože Společnost nevyžaduje nestandardní uživatelské rozhraní a opět má zkušenosti především se starším modelem WinForms, rozhodl jsem se použít WinForms.

7.4. Volba konkrétního programovacího jazyka

Platforma .NET nabízí větší množství programovacích jazyků, které jsou následně kompilovány do univerzálního kódu. Volba samotného jazyka je spíše kosmetickým dilematem, případně volbou Společnosti pro sjednocení jejího programového portfolia. Jelikož většina aplikací Společnosti je realizována pomocí jazyka Visual Basic rozhodl jsem se ho použít i zde.

8. IMPLEMENTACE ISIR

V této kapitole rozebírám samotnou tvorbu Systému vycházející z analýzy a postupuji podle kapitoly 4.2. Implementace proběhla v několika fázích, nejdříve jsem vytvořil databázi a aktualizací program. Poté jsem obě řešení testoval a v přírůstcích upravoval funkcionalitu. Následovala tvorba webové služby prohledávání a na ní navazující monitorovací program. Rozhodl jsem se zachovat toto chronologické členění i v textu diplomové práce. Implementace probíhá v jednotlivých přírůstcích, iteracích, které vždy zahrnují vytvoření určité části řešení, jeho testování a následné použití (nasazení).

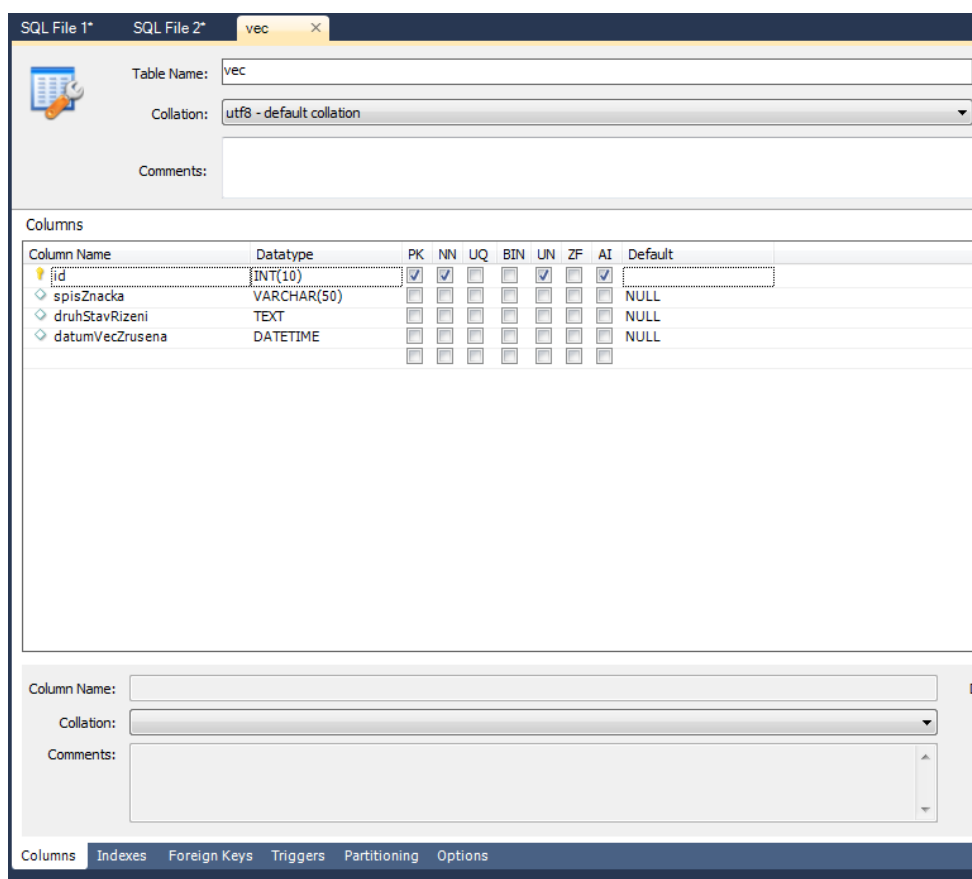
8.1. Přírůstek 1.

Cílem prvního přírůstku je založit databázi pro uchování dat a zprovoznit funkcionalitu aktualizací programu. Tento cíl byl definován Společností jako prioritní, aby ve chvíli zprovoznění monitorovacího programu již byla databáze aktualizována. Výstupem přírůstku tak je databáze a aktualizací program.

8.1.1. Tvorba databáze

Od zástupce Společnosti jsem získal přístupové údaje k serveru Společnosti a pomocí nástroje správy MySQL WorkBench jsem na serveru založil databázi pro replikaci dat ISIR. Prostředí tvorby databáze pomocí nástroje je na obrázku 28. Co se týče zabezpečení přístupu do databáze, rozhodl jsem se použít kombinaci náhodně generovaného silného hesla a filtrování IP adres, které mají přístup k zápisu.

Zároveň jsem pro účely vývoje a testování založil stejnou databázi na lokálním serveru vývojového počítače. Toto jsem udělal z důvodu snadnějšího testování (nemohou nastat problémy s připojením na databázi) a zároveň také, abych oddělil testovací a provozní prostředí a neprováděl neodzkoušené změny na provozním prostředí. Pro zajištění toho, že jsou databáze identické co se týče definice jsem založení testovací databáze provedl tak, že jsem si nechal z nástroje MySQL WorkBench vygenerovat kódy vytvoření tabulek a tyto SQL příkazy pak nechal proběhnout na lokální testovací databázi.



Obrázek 28 tvorba tabulky v prostředí MySQL WorkBench (Zdroj: Autor)

8.1.2. Tvorba aktualizací programu

Aktualizační program má odpovídat funkcionalitě popsané v kapitole 5.2.3 a 6.4. Při realizaci jsem vycházel především z publikací [10] popisující vhodné praktiky pro tvorbu řešení pomocí objektově orientovaných jazyků, [49] popisující základy práce s jazykem Visual Basic v prostředí Visual studia 2008 a [50], která se zabývá poměrně vyčerpávajícím způsobem funkcionalitou jazyka , C#. Poslední publikaci jsem zvolil především z toho důvodu, že jsem nenalezl obdobně detailní a kvalitní pro jazyk Visual Basic (pro jeho aktuální verzi) a jelikož jsou oba jazyky součástí platformy .NET, lze řešení převádět i pro jazyk Visual Basic (existují i automatizované nástroje pro převod kódu, např. [51]) . Dále jsem pro řešení specifických problémů používal především stránky podpory společnosti Microsoft [52] a dále různá komunitní fóra zabývající se tematikou vývoje aplikací, především [53].

Aktualizační program jsem vytvořil v jazyce Visual Basic ve vývojovém prostředí Visual Studio 2010.

8.1.2.1. Tvorba Uživatelského rozhraní

Uživatelské rozhraní jsem vytvořil pomocí nástrojů tvorby uživatelského rozhraní prostředí Visual Studia. Do implicitně vytvořeného okna jsem vložil textové pole pro výpis a tlačítko ovládající běh programu. Vytvořené okno odpovídá demo verzi z kapitoly 5.4.1.

8.1.2.2. **Tvorba aplikační logiky**

Při tvorbě aplikační logiky jsem postupoval především podle modelu tříd z kapitoly 6.4.4.4. Rozhraní jsem implementoval konkrétními třídami a tyto jsem odděleně od celku testoval. Testování probíhalo podle metody jednotkového testování [54] definováním vstupních a cílových výstupných dat. V případě nesplnění testu jsem funkcionality upravoval. Po splnění všech testů jsem třídy spojil do aplikačního celku a to připojil k uživatelskému rozhraní.

8.1.3. **Testování prvního přírůstku**

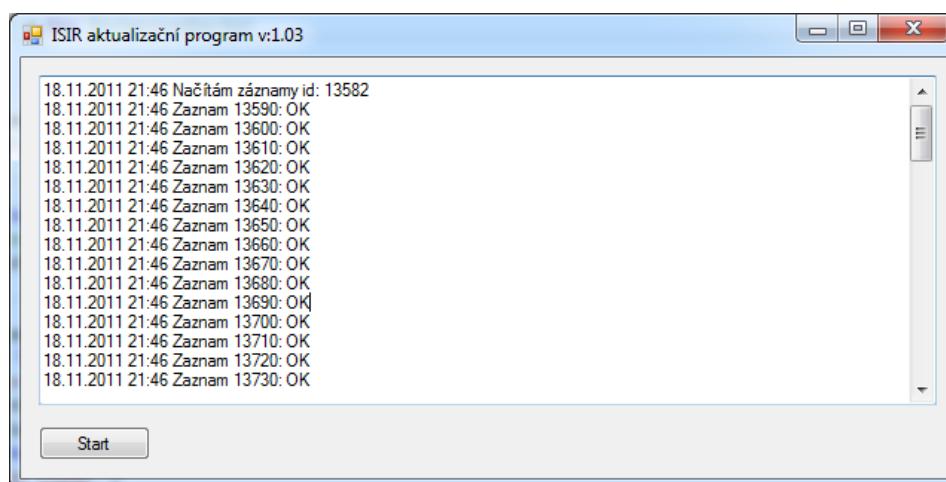
Testování prvního přírůstku mělo za cíl otestovat, zda-li aplikace správně přistupuje k databázi a zda-li aplikace správným způsobem databázi plní. Vzhledem k tomu, že databáze má být replikou insolvenčního rejstříku, tak jako úspěch řešení jsem určil shodu poskytovaných dat mezi databází a rejstříkem. Tuto shodu jsem se rozhodl porovnávat po naplnění databáze výpisem detailů o náhodně vybraných insolvenčních řízeních. Zároveň bylo předmětem testování chování programu, respektive jeho odolnost vůči neošetřeným výjimkám, které by mohli vést buď k nekonzistencím v databázi, nebo k neočekávanému ukončení činnosti programu. Toto chování jsem se rozhodl testovat dvěma metodami.

Testování nekonzistencí jsem provedl tak, že jsem v průběhu plnění náhodně přerušoval připojení k internetu a to jak v průběhu komunikace s databází, tak komunikace s webovou službou ISIR. Dále jsem náhodně přerušoval chod databázového serveru, případně odmítal připojení na něj. Zároveň jsem po naplnění databáze prováděl testy dotazující se na duplicitu dat. Ve všech případech byla pomocí transakčního systému (viz. [55]) zachována konzistence dat, nebyla zjištěna duplicita a zároveň aktualizací program neukončil svůj chod neošetřenou chybou.

Testování neočekávaného ukončení jsem provedl tak, že jsem aktualizací program pustil jen ve vývojovém prostředí s tím, že jsem jakoukoliv neočekávanou výjimku odchytil a program by detaily o chybě vypsal. V průběhu plnění databáze chyba nenastala.

V průběhu testování jsem dosáhl rychlosti přibližně 3150 událostí za minutu. Vzhledem k tomu, že webová služba ISIR poskytovala jako nejvyšší ID události 3000000, mohu v nejhorším případě očekávat, že plnění by trvalo přibližně 16 hodin. Na druhou stranu, některá ID událostí nemusejí být použita, tedy pravděpodobně by plnění trvalo kratší dobu.

Na obrázku 29 je potom aktualizací program v průběhu nahrávání dat do databáze.



Obrázek 29 aktualizací program plní data (Zdroj: Autor)

8.1.4. Nasazení prvního přírůstku

Po odladění funkcionality na lokálním prostředí jsme se rozhodli zprovoznit řešení i na serveru Společnosti. Změna vyžadovala jen úpravu připojení na databázi v aktualizacím programu. Při plnění provozní databáze jsme se však setkali s problémem rychlosti plnění. Vzhledem k tomu, že server Společnosti se nachází v Americe, docházelo k poměrně velkým zpožděním při plnění a rychlost plnění klesla na 1000 záznamů za 6 minuty. Tedy plnění databáze by trvalo přibližně 10,5 dne. Vzhledem k tomu, že Společnost chtěla zpřístupnit databázi co nejdříve třetím stranám, byla tato možnost shledána nepřijatelnou. Nabídl jsem Společnosti tři možná řešení:

- Pokusit se snížit počet dotazů na databázi. Toto řešení by zvýšilo rychlost plnění, protože testováním jsme zjistili, že k největšímu prodlení dochází mezi jednotlivými dotazy. Jelikož aktualizací program používá prakticky atomické dotazy, kdy se při zpracování jednoho záznamu nastane několik dotazů (dotaz na existenci, následně úprava, nebo vložení nového a tato sekvence probíhá pro všechny entity jednoho záznamu), šlo by prodlení snížit zasíláním komplexnějších dotazů. Řešení by vyžadovalo změny na straně aktualizacího programu, který by získával informace o více entitách/záznamech v jednom dotazu a následně po vyhodnocení by opět zaslal změny o více entitách/záznamech v jednom dotazu. Úprava by dále vyžadovala novou fázi testování.
- Naplnit databázi lokálně, což by trvalo 16 hodin a pak data přenést pomocí nástroje správy databáze. Nejprve bych vytvořil zálohu lokální databáze v jednom SQL souboru a tuto zálohu bych poté nahrál na server databáze. Výhodou řešení je to, že není potřeba úprava aktualizacího programu a zároveň díky přenosu všech dat v jedné dávce je minimalizován vliv prodlení mezi příkazy. Úprava nevyžaduje nové testování.
- Zvýšit počet připojení na databázi z aktualizacího programu. Aktualizací program by stáhl data např. o 1000 událostech, následně tato data rozdělil a potom pomocí 5 oddělených procesů a připojení nahrával, tedy rychlost plnění by teoreticky mohla být zvýšena počtem

procesů. Řešení by šlo použít za velmi významných úprav aktualizací programu, protože logika aplikace by se musela umět vyrovnat s případy toho, že jeden proces by vyžadoval data o entitě, kterou jiný proces ještě nestihl založit. Ačkoliv by vzhledem k tomu, že nad databází nikdy neprobíhá mazání dat, šlo toto řešení použít, silně bych ho nedoporučoval. Úprava by dále vyžadovala novou fázi testování.

Po konzultaci se zástupcem Společnosti jsme se rozhodli realizovat druhou variantu, tedy naplnit lokální databázi a její zálohu pak přenést na server Společnosti. Řešení nevyžaduje další testování, nebo úpravy programu, jen využijeme funkcionality programu správy databáze. Zároveň ve chvíli naplnění většiny databáze již dostačuje rychlost plnění aktualizací programu, která řádově převyšuje rychlost přírůstku nových záznamů v rejstříku ISIR.

8.1.5. Výstupy prvního přírůstku

Po realizaci prvního přírůstku jsem získal funkční a aktualizovanou databázi a zároveň funkční program zajišťující další aktualizaci.

8.2. Přírůstek 2

Cílem třetího přírůstku je zprovoznit funkcionalitu webové služby vyhledávání tak, jak byla definována v kapitole 6.3. Výstupem je potom úspěšné zprovoznění a odladění služby.

8.2.1. Tvorba webové služby vyhledávání

Službu vyhledávání jsem vytvořil pomocí textového editoru PSPad. Vytvořil jsem PHP soubor podle návrhu z kapitoly 6.3.2. K metodám (funkcím) vytváření dotazu na databázi, tedy "vytvorQueryFyzicka" a "vytvorQueryPravnicka" jsem doplnil SQL dotazy podle popisu chování webové služby. Celkem jsem takto tedy vytvořil 3 dotazy pro fyzické osoby a tři dotazy pro právnické osoby. Příklad dotazu pro fyzickou osobu se zadanými parametry jméno, příjmení a datum narození je:

```
"$query = "SELECT oSpisZnacka, max(cas) FROM `osoba` JOIN `udalost` ON (osoba.oSpisZnacka = udalost.uSpisZnacka) WHERE `jmeno` LIKE '%" . $jmeno . "%' AND `nazevOsoby` LIKE '%" . $nazev . "%' AND `datumNarozeni` LIKE '%" . $datum . "%' LIMIT 0 , 30 ";"
```

Ostatní formulované dotazy jsou obdobou s jinými atributy a parametry.

8.2.1.1. Definice rozhraní webové služby

Vzhledem k plánovanému použití webové služby i třetími stranami bylo nutné definovat i rozhraní pro validní použití webové služby. Při jeho tvorbě jsem vycházel z kapitoly 5.3.4, která definuje požadavky na toto rozhraní a pouze jsem upřesnil syntaxi a uvedl validní formáty a kombinace parametrů.

Původní požadavek Společnosti rozhraní definoval takto:

- "www.web_spolecnosti.cz/prohledej_jmeno=karel,prijmeni=novak,rc=1111111111."

Volání služby tak probíhá specifikací url a následným zavoláním stránky, která zobrazí výsledek. URL si zde můžeme rozdělit na dvě části:

- "www.web_spolecnosti.cz/prohledej" je identifikací služby.
- "_jmeno=karel,prijmeni=novak,rc=1111111111" je předání parametrů a jejich hodnot.

Toto původní zadání jsem vzhledem k syntaxi a použití php upravil takto:

- "www.web_spolecnosti.cz/prohledej.php" je identifikací služby.
- "?osoba=F" je připojením identifikace typu osoby, validní jsou hodnoty "F" jako fyzická a "P" jako právnická. Tento parametr je povinný a musí vždy následovat identifikaci služby.
- "&jmeno=Karel&nazev=Novak&RC=111111/1111" jsou nepovinné parametry služby určující již vyhledávaného dlužníka. Jejich pořadí je libovolné a syntaxe je vždy "&" + "parametr=" + hodnota. Povolené parametry jsou "jmeno, nazev, RC, ICF, datum" pro fyzické osoby a "nazevp, IC" pro právnické osoby.

Příkladem použití je potom:

- "www.web_spolecnosti.cz/prohledej.php?osoba=F&jmeno=Karel&nazev=Novak&RC=111111/1111"

Což je dotaz na vyhledání fyzické osoby Karla Nováka s rodným číslem 111111/1111.

- "www.web_spolecnosti.cz/prohledej.php?osoba=P&nazevp=Bagry&IC=12345678"

Což je dotaz na vyhledání právnické osoby s názvem Bagry a identifikačním číslem 12345678.

8.2.1.2. Definice návratových hodnot webové služby

V kapitole 6.1 jsem se rozhodl využívat webovou službu i pro dotazy monitorovacího programu, který potřebuje informace o insolvenčním řízení a datu poslední události tohoto řízení. Zároveň služba musí stále umožňovat vracet jen jednoduché ANO/NE, jak bylo požadováno Společností. Tyto dvě varianty návratových hodnot jsem se rozhodl řešit provozem dvou služeb s rozdílnou identifikací. Tedy na serveru databáze jsem nasadil dvě služby:

- "www.web_spolecnosti.cz/prohledej.php" je původní služba s návratovou hodnotou ANO, nebo NE.
- "www.web_spolecnosti.cz/prohledej2.php" je upravená služba, která vrací identifikaci insolvenčního řízení a datum poslední události tohoto řízení. Návratová data jsem se rozhodl

vracet oddělené speciálním znakem ",", který se v datech ISIR nepoužívá. V případě nenalezení dlužníka potom služba vrací NE, stejně jako původní verze.

8.2.2. Testování druhého přírůstku

Testování přírůstku proběhlo dvěma způsoby. První testování bylo manuální a náhodné, kdy na různé kombinace zadaných parametrů jsem ověřoval, že služba reaguje správným způsobem. Tedy vrací chybové hlášky při nevalidních parametrech a správné odpovědi jak v případě existence, tak neexistence dlužníka. Druhé testování bylo automatizované, kdy jsem vytvořil velmi jednoduchý program, který volal webovou službu a testoval stejně jako v případě jednotkového testování.

Obě metody testování byly úspěšné a odhalili několik nesprávných reakcí způsobených chybami v kódu webové služby. Další sady testování potom vedly ke 100% úspěšnosti.

8.2.3. Výstupy druhého přírůstku

Po dokončení druhého přírůstku Společnost získala funkční službu vyhledávání, kterou začala využívat třetí strana.

8.3. Přírůstek 3

Cílem třetího přírůstku je zprovoznit funkcionalitu monitorovacího programu a nasadit ho u třetí strany, která od Společnosti plánuje odebírat tuto funkcionalitu. Výstupem je tedy úspěšné nasazení programu u třetí strany.

8.3.1. Tvorba monitorovacího programu

Monitorovací program jsem vytvořil podle popisu v kapitole 6.5 za použití stejných postupů a nástrojů jako v kapitole 8.1.2.

8.3.1.1. Tvorba uživatelského rozhraní

Uživatelské rozhraní monitorovacího programu jsem vytvořil podle návrhu z kapitoly 5.4.2. Jako jedinou změnu jsem přidal v horní části textové pole s názvem právě otevřeného souboru. Změnu jsem provedl na základě prvního interního testování, kdy indikace jaký soubor je právě otevřený mi ulehčila práci s programem.

8.3.1.2. Tvorba aplikační logiky

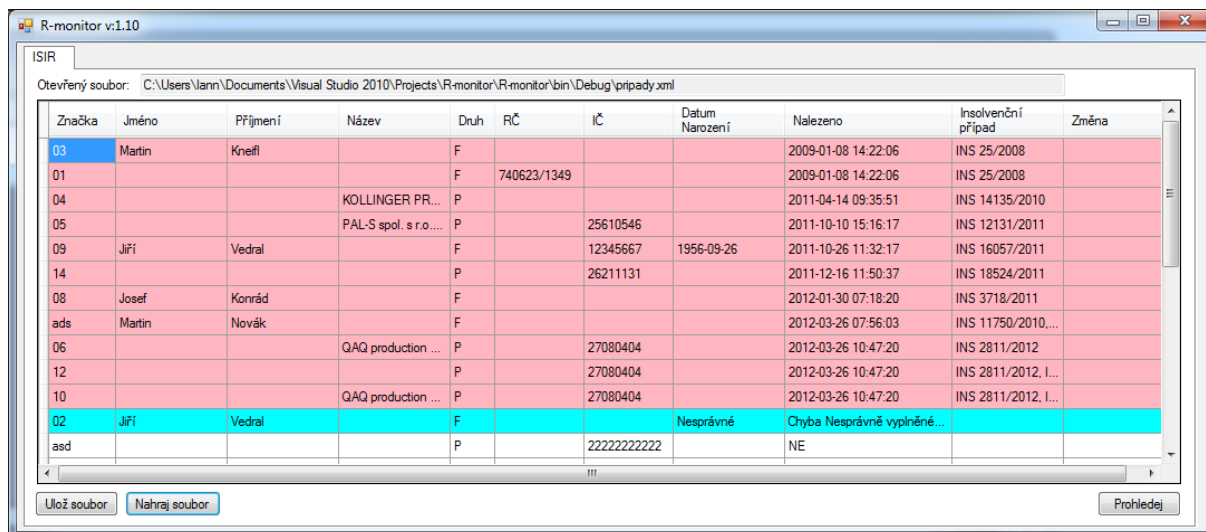
Při tvorbě aplikační logiky jsem postupoval podle modelu tříd z kapitoly 6.5.2.4. Testování probíhalo opět jednotkovými testy, tedy definováním vstupních a cílových výstupných dat. V případě nesplnění testu jsem funkcionalitu upravoval. Po splnění všech testů jsem třídy spojil do aplikačního celku a to připojil k uživatelskému rozhraní.

8.3.2. Interní testování třetího přírůstku

Před nasazení monitorovacího programu u třetí strany jsem program otestoval jako celek. Testy probíhaly v prostředí vývojového studia. Na základě testování jsem přidal barevnou indikaci u případů pro lepší orientaci, protože při zpracování většího množství případů bylo poměrně náročné se orientovat jen v bílých tabulkách. Obarvení jsem udělal podle následujícího systému:

- V případě nalezení případu je případ obarven růžově.
- V případě nesprávného formátu některého validovaného atributu (např. datum) je případ obarven modře.
- V případě nalezení případu poprvé, jeho změny, nebo události, která nastala v posledních sedmi dnech je případ obarven červeně.
- Jinak případ zůstává bílý.

Okno již funkčního programu je na obrázku 30.



Značka	Jméno	Příjmení	Název	Druh	RČ	IČ	Datum Narození	Nalezeno	Insolvenční případ	Změna
03	Martin	Kneifl		F				2009-01-08 14:22:06	INS 25/2008	
01				F	740623/1349			2009-01-08 14:22:06	INS 25/2008	
04			KOLLINGER PR...	P				2011-04-14 09:35:51	INS 14135/2010	
05			PAL-S spol. s r.o....	P		25610546		2011-10-10 15:16:17	INS 12131/2011	
09	Jiří	Vedral		F		12345667	1956-09-26	2011-10-26 11:32:17	INS 16057/2011	
14				P		26211131		2011-12-16 11:50:37	INS 18524/2011	
08	Josef	Konrád		F				2012-01-30 07:18:20	INS 3718/2011	
ads	Martin	Novák		F				2012-03-26 07:56:03	INS 11750/2010....	
06			QAQ production ...	P		27080404		2012-03-26 10:47:20	INS 2811/2012	
12				P		27080404		2012-03-26 10:47:20	INS 2811/2012, I...	
10			QAQ production ...	P		27080404		2012-03-26 10:47:20	INS 2811/2012, I...	
02	Jiří	Vedral		F			Nesprávně	Chyba Nesprávně vyplněné...		
asd				P		222222222222		NE		

Obrázek 30 okno monitorovacího programu (Zdroj: Autor)

Okno programu ve fázi vyhledávání je na obrázku 31.



Obrázek 31 okno monitorovacího programu při vyhledávání (Zdroj: Autor)

8.3.3. Akceptační testování přírůstku u třetí strany

Monitorovací program jsem nasadil k akceptačnímu testování u třetí strany. Vzhledem k tomu, že program nepotřebuje žádné soubory nastavení a XML soubory s daty si vytváří sám, tak jsem pouze zaslal výkonný program s manuálem k jeho provozu. U třetí strany byl program nejprve testován několika pracovníky a posléze byl umístěn na síťový disk a začal být testován všemi pracovníky, kteří jeho funkcionality potřebují. Testování přineslo tyto požadavky a výhrady:

- Program by v případě nalezení dlužníka měl vypsat všechny jeho insolvenční řízení, pokud se vyskytuje ve více a jen poslední (nejvyšší) datum události všech těchto řízení.
- Program by neměl nalézat osoby, které v insolvenčním řízení vystupují v jiné pozici než dlužník.
- Program by měl umožňovat vyhledávat fyzické osoby i jen podle jména a příjmení. Třetí strana byla upozorněna, že toto může vést k vyhledání jiného dlužníka, ale stejně na požadavku trvá.
- Program nevyhledává právnické osoby, u kterých může být v systému třetí strany poznámka "např. v likvidaci", která v databázi ISIR není.

8.3.3.1. Řešení akceptačních připomínek

V návaznosti na připomínky třetí strany jsem zvolil tyto úpravy řešení monitoringu:

- Pro vrácení všech insolvenčních řízení jsem se rozhodl upravit webovou službu prohledávání tak, že vrací seznam insolvenčních značek a dat posledních událostí s tím, že jednotlivé záznamy jsou odděleny speciální značkou "|", která se v navracených datech nevyskytuje. Úprava byla realizována úpravou SQL dotazů a úpravou funkce zobrazení webové služby. Monitorovací program následně sám určí, která událost je nejvyšší a zobrazí toto datum.

- Pro nenacházení osob v jiném stavu než dlužník jsem upravil opět SQL dotaz webové služby.
- Pro možnost vyhledávání fyzických osob jen podle jména a příjmení jsem přidal další možnost v logice webové služby, která teď v případě zadání jen jména a příjmení utvoří dotaz a pokusí se dlužníka nalézt.
- Vyhledávání právnických osob s poznámkou jsem se rozhodl řešit úpravou monitorovacího programu, který se v případě nenalezení právnické osoby pokusí zjistit, zda-li se v jejím názvu vyskytuje poznámka a pokud ano, tak se pokusí vyhledat osobu bez poznámky. Další případné odstraňované poznámky budou přidávány až na základě požadavků třetí strany. Případně v budoucích verzích mohu implementovat seznam filtrovaných textů.

Třetí strana i Společnost s navrženými řešeními souhlasily.

8.3.4. Nasazení programu monitoringu

Po zapracování oprav připomínek byl program monitoringu úspěšně nasazen u třetí strany. K programu byl předán manuál a proběhlo jednorázové školení pracovníků. Program nevyžadoval žádnou migraci dat, nebo specifické nastavení.

8.3.5. Výstupy třetího přírůstku

Výstupem třetího přírůstku bylo schválení akceptačního řízení a následné úspěšné nasazení monitorovacího programu u třetí strany.

9. PROVOZ A UŽITÍ ISIR

V průběhu provozu a užití programu pro monitoring ISIR zatím nenastaly žádné problémy, nebo výhrady s funkcionalitou monitoringu. Monitoring ISIR ale bude dále rozvíjen a doplněn o možnosti monitoringu ARES, kterými se pak zabývá další část práce.

10. ANALÝZA/MODEL OBSAHOVÝ ARES

Obsahová analýza ARES je obdobou analýzy ISIR, tedy kapitoly 5 a zároveň budu metodicky postupovat podle definice obsahové analýzy v kapitole 4. Smyslem kapitoly je získat požadavky Společnosti na monitoring vybraných rejstříků ARES.

10.1. Analýza základních požadavků Společnosti

Zástupce Společnosti formuloval na schůzce požadavky na řešení monitoringu ARES. Monitoring by měl být v první verzi dostupný především pro právnické osoby a pro Obchodní rejstřík (OR). Monitoring by měl být zabudován do programu monitoring ISIR a měl by fungovat podobným způsobem, tedy umožnit uživateli sledovat změny, které proběhly na obchodním rejstříku u specifikovaných osob. Vzhledem k tomu, že monitoring má být součástí již existujícího programu pro monitoring ISIR, mělo by být rozšířeno okno tak, aby umožnilo zobrazit případy z pohledu ISIR, z pohledu OR a také obecný seznam případů. Další přidávané monitory různých zdrojů by pak vždy

měly mít další vlastní pohled. Obecný pohled by měl sloužit k zobrazení univerzálních informací, například jména, příjmení, názvu apod. a také ke specifikaci, ve kterých zdrojích je případ vyhledáván.

Výhledově pro další verze by mělo být možné zobrazovat výpisy detailu subjektu z různých zdrojů ARES.

10.2. Analýza ARES

V této části uvádím, co to je ARES, jaká data obsahuje a jakým způsobem je možno k nim přistupovat. Na závěr je pak uveden model, který ukazuje daný stav řešení vzhledem k možnostem ARES.

10.2.1. Definice ARES

ARES, tedy Administrativní registr ekonomických subjektů je "informační systém, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zprostředkovává zobrazení údajů vedených v jednotlivých registrech státní správy, ze kterých čerpá data (tzv. zdrojové registry)." [56].

ARES potom obsahuje tyto majoritní zdroje:

- Obchodní rejstřík (OR), vedený rejstříkovými soudy
- Živnostenský rejstřík (RŽP), vedený Ministerstvem průmyslu a obchodu ČR
- Registr ekonomických subjektů (RES), vedený Českým statistickým úřadem
- Registr církví a náboženských společností (RCNS), vedený Ministerstvem kultury ČR
- Registr zdravotnických zařízení (RZZ), vedený Ústavem zdravotnických informací a statistiky ČR
- Seznam občanských sdružení a spolků (OSS), vedený Ministerstvem vnitra ČR
- Evidence zemědělského podnikatele (EZP), která je vedena Ministerstvem zemědělství ČR
- Seznam politických stran a hnutí (PSH), vedený Ministerstvem vnitra ČR
- Rejstřík škol a školských zařízení (RŠ), vedený Ministerstvem školství a tělovýchovy ČR

A další minoritní, případně neaktualizované zdroje, které se však připravují ke zrušení.

10.2.2. Přístup ke zdrojům ARES

ARES je přístupný dvěma způsoby. První způsobem jsou formuláře pro vyhledávání ve zdrojích, které se ale nehodí pro automatické zpracování většího množství případů. Druhou možností je pak využití webových služeb. V popisu služeb ARES (viz. [56]) se potom uvádí "Součástí informačního systému ARES je i XML rozhraní pro vyhledání subjektu a zpřístupnění jeho veřejných údajů ze zdrojových registrů. Služby XML může využívat každý, pokud bude respektovat podmínky provozu.". Podmínky provozu jsou potom především zdůrazňují omezení počtu dotazů na systém ARES, jejichž nedodržení vede k blokaci IP adresy. Omezení je 1000 dotazů od 8:00 do 18:00 a 5000 od 18:00 do 8:00. Služby

lze volat dvěma způsoby, prvním je použití webových služeb podle definice W3C [45], druhým je získávání XML souborů odpovědí pomocí volání služby identifikací URL a parametry předávanými metodou GET (tedy podobně jako u webové služby vyhledávání ISIR).

10.2.3. Definice rozhraní ARES

Rozhraní ARES je definováno v [57]. Celé rozhraní je poměrně velmi rozsáhlé a proto se při jeho popisu omezují jen na rozhraní relevantní pro řešení monitoringu OR. Důležitý je zde požadavek Společnosti sledovat změny, tedy Společnost nevyžaduje zobrazování výpisů detailů subjektu, případně tuto funkcionalitu bude požadovat až v dalších verzích. Navíc zobrazování detailů je funkcionalita, která je dostupná i na stránkách ARES, popřípadě lze do požadavku na výpis detailů specifikovat i výstupní formát PDF a následně tento soubor zobrazit uživateli. Sledovat změny v návratových hodnotách lze i bez rozboru poskytovaných hodnot a pro toto sledování jsem navrhl dva způsoby:

- První metodou je ukládání celých výpisů z OR na počítač, ze kterého je monitoring spouštěn (nebo na přístupné úložiště) a v případě nového výpisu, který se liší, zobrazit rozdílné elementy. Metoda předpokládá ukládání souborů v jednoduše formátovaném souboru, například XML. Zobrazení změn by následně nevyžadovalo ani znalost tohoto XML, šlo by jednoduše procházet jednotlivé uzly a v případě neshody tyto zobrazit. Metoda by vyžadovala vedle souborů s dlužníky, které již využívá monitoring ISIR také definovat nové úložiště pro soubory detailů. Další výhodou je možnost ukládat detaily dlužníků a potom jejich procházení bez nutnosti dalších dotazů.
- Druhou metodou je ukládání hash kódů (viz. [58]) v souboru dlužníků, jako další atribut a následné upozornění na změnu v případě jejich neshody.

Po konzultaci výhod a nevýhod obou se zástupcem Společnosti jsem se rozhodl oba způsoby zkombinovat a ukládat si tedy výpisy celé a zároveň hash kódy těchto výpisů. Při pokusném volání služby a analýze vrácených dat jsem navíc zjistil, že služba vrací i data jejího volání, jelikož tato data se samozřejmě mění, bude nutno před tvorbou hash kódu vždy tato data odstranit.

10.3. Návrh řešení monitoringu ARES

V návaznosti na požadavky Společnosti a možnosti přístupu a poskytnutých dat ARES jsem navrhl řešení monitoringu. Řešení respektuje požadavky Společnosti a je tedy rozšířením původního monitoringu ISIR. Řešení musí zachovat základní architekturu monitoringu, která byla již formulována u monitoringu ISIR.

Navržené řešení využívá volání služby ARES pro zjištění detailů subjektu a následné zobrazení zjištěných změn v monitorovacím programu.

10.3.1. Využití databáze

Rozšíření nebude využívat uložení dat o subjektech do databáze. Je to z toho důvodu, že zvolená databáze pro řešení ISIR je relační databázi MySQL a nehodí se pro ukládání velkého množství XML dat. Navíc tato data jsou neustále dostupná a nevzniká zde tak důvod pro vytvoření vlastního úložiště, jako byl v případě replikace ISIR. Rozšíření však potřebuje monitorovat počet dotazů na službu ARES z dané IP adresy. Tento počet dotazů se musí sledovat a to ideálně mimo samotný program, protože v případě využití více programů pod sdílenou IP adresou by toto sledování nebylo spolehlivé. Tedy rozšíření bude využívat databázi Společnosti pro ukládání IP adres a počtu dotazů.

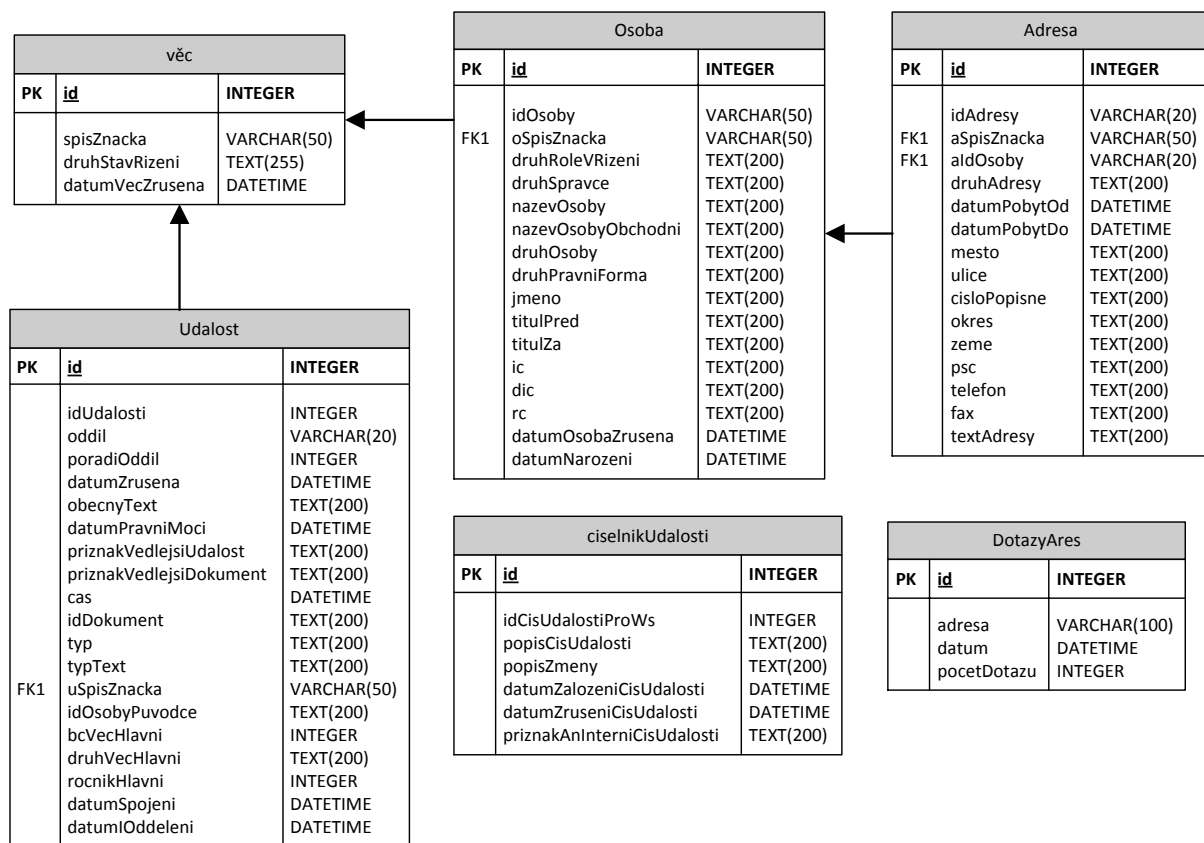
Databáze Společnosti pak musí být upravena tak, aby umožňovala uložení IP adresy, času a počtu dotazů. Vzhledem k tomu, že omezení na dotazy je platné pouze jeden den, rozhodl jsem se uložení realizovat pouze jednou tabulkou, která obsahuje atributy IP adresy, času a počtu dotazů. Proces ukládání dat pak bude realizován aplikační logikou a bude probíhat takto:

- Program se dotáže databáze, zda-li existují záznamy pro aktuální den pro danou IP adresu.
- Pokud záznamy existují, program zjistí, kolik dotazů ještě může zaslat a pokud je požadovaný počet nižší, zašle dotazy a zároveň vytvoří nový záznam v databázi. Pokud je požadovaný počet dotazů vyšší, provede jen maximální možný počet dotazů a vytvoří záznam v databázi.
- Pokud záznam neexistuje, založí ho a provede dotazy.
- Program vymaže veškeré záznamy starší dvou dnů.

Podle výše specifikované požadované funkcionality jsem pak upravil schéma databáze. Do schématu jsem přidal novou tabulku "DotazyARES" s těmito atributy:

- id - umělý primární klíč typu integer.
- adresa - atribut obsahuje IP adresu, ze které dochází k odesílání dotazů na ARES. Adresa je typu varchar s rozsahem 100.
- datum - atribut specifikuje čas, kdy dochází k dotazování ARES a je typu datetime.
- pocetDotazu - atribut určuje počet dotazů, které v daném čase proběhly na systém ARES.

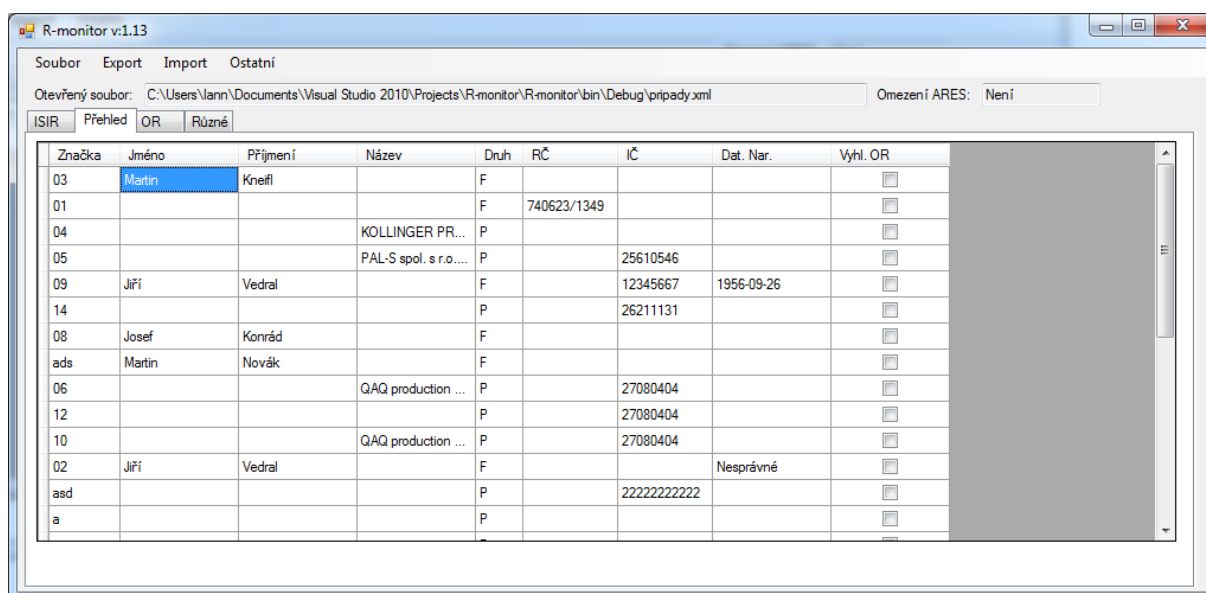
Pro úsporu místa uvádím rovnou fyzický model databáze, který je na obrázku 32.



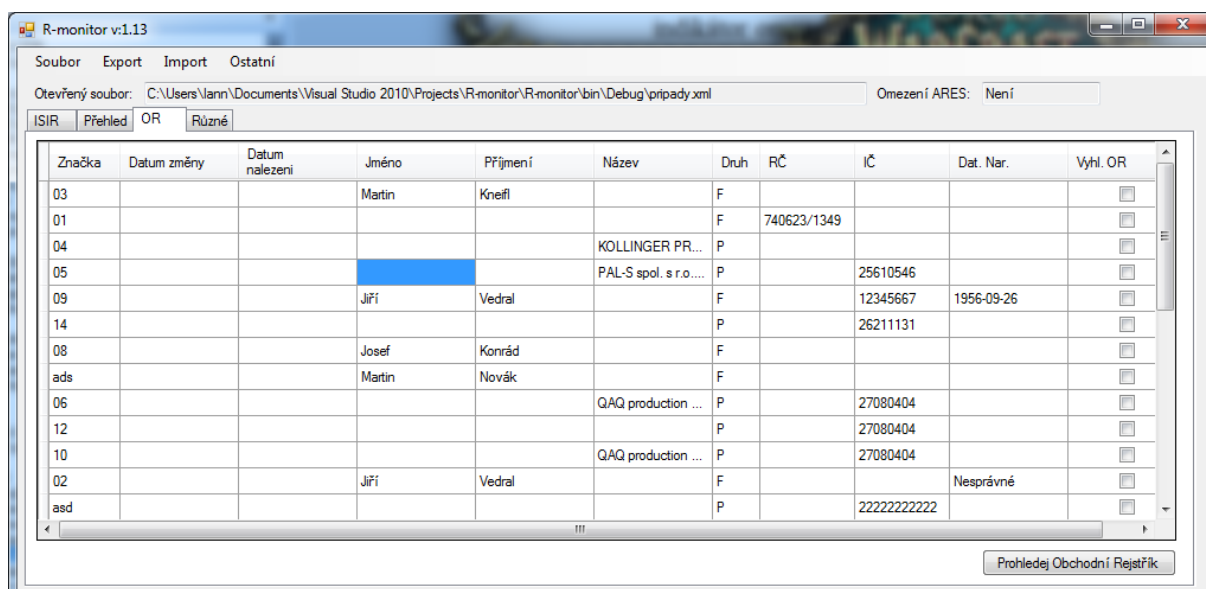
Obrázek 32 upravené schéma databáze Společnosti (Zdroj: Autor)

10.3.2. Návrh úpravy monitorovacího programu

Podle požadavků Společnosti jsem navrhl úpravu monitorovacího programu tak, že data o vyhledávaných subjektech jsou nyní zobrazeny ve třech různých pohledech, původním ISIR, obecném a zaměřeném na OR. V případě dalšího přidaného rejstříku bude stejně přidán další pohled. V návrhu je navíc složka různé, která slouží k obsazení případné další různé funkcionality. Zároveň jsem přidal indikátor omezení dotazů ARES a přesunul funkcionalitu tlačítek "otevři soubor" a "ulož soubor" do menu. Toto jsem provedl především z estetických důvodů. V návaznosti na tuto tvorbu menu jsem navíc do menu přidal záložky Export a Import a ostatní. Upravené okno je vidět na obrázku 33, který ukazuje pohled obecný (nazvaný "Přehled") a obrázku 34, který zobrazuje pohled OR.



Obrázek 33 okno monitorovacího programu s pohledem "Přehled" (Zdroj: Autor)



Obrázek 34 okno monitorovacího programu s pohledem "OR" (Zdroj: Autor)

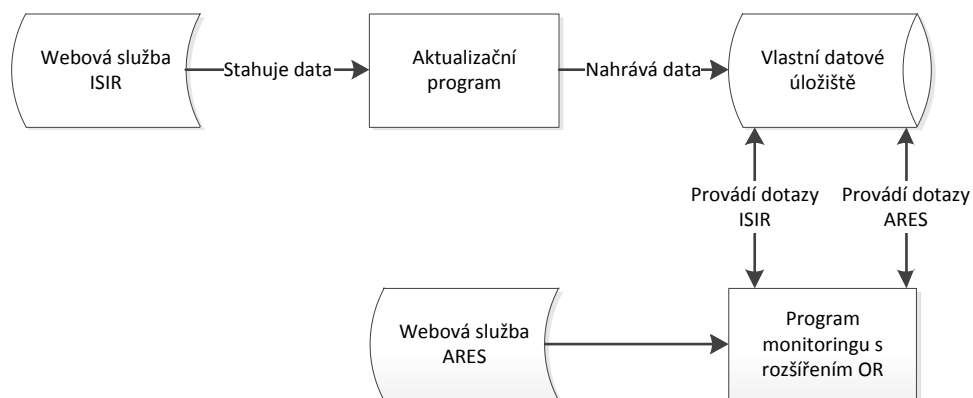
Zástupce Společnosti s navrženým řešením souhlasil.

11. ANALÝZA/MODEL TECHNOLOGICKÝ ARES

Technologická analýza ARES je obdobou analýzy ISIR, tedy kapitoly 6 a zároveň budu metodicky postupovat podle definice obsahové analýzy v kapitole 4. V kapitole se opět věnuji technologickým aspektům řešení monitoringu.

11.1. Architektura Systému

Vzhledem k rozšíření monitorovacího Systému došlo i ke změně architektury. Monitorovací program musí komunikovat se službami ARES. Pozměněná architektura je na obrázku 35.



Obrázek 35 pozměněné architektura monitorovacího Systému (Zdroj: Autor)

11.2. Monitorovací program

Monitorovací program bude rozšířen o funkcionalitu ARES OR, tedy volba platformy je .NET. Změny uživatelského rozhraní již byly specifikovány v kapitole 10.3.2. Hlavní procesy monitorovacího programu zůstávají nezměněny, jen je přidán nový proces vyhledávání ARES.

11.2.1. Komunikace se službou ARES

Komunikace musí probíhat voláním služby a zpracováním odpovědi. Toto volání může probíhat buď za využití automaticky vygenerovaného kódu podle standardu webových služeb, nebo vlastní zpracováním, kdy je načtena webová stránka s odpovědí a program si tuto dále zpracuje. Výhody a nevýhody jsou již zmíněny výše v kapitole 6.4.4.2. V případě ARES jsem se rozhodl použít vlastní metodu zpracování dat. Vedly mě k tomu dva hlavní důvody. Prvním je nefunkčnost vygenerovaného kódu, který i po úpravách (chyby způsobeny pravděpodobně generátorem platformy .NET) nebyl schopen validně pracovat se službou OR. Při pečlivém průzkumu WSDL, které jsem následně provedl jsem objevil chyby v referencování neplatných definic některých objektů, které vedly k tomu, že vygenerovaný klient se pokoušel pracovat s jinak definovanými daty, než mu byly předány. Jelikož předpokládám další změny těchto definic v budoucnosti, rozhodl jsem se, že nejlepším způsobem jak se s těmito změnami vyrovnat bude nevyužívat jejich definice. Druhým důvodem je pak to, že obě varianty zjišťování změn prakticky nepotřebují pracovat s objekty dat, ale spíše je data zajímaví jako čistý text, ze kterého se vytvoří hash kód a následně se uloží.

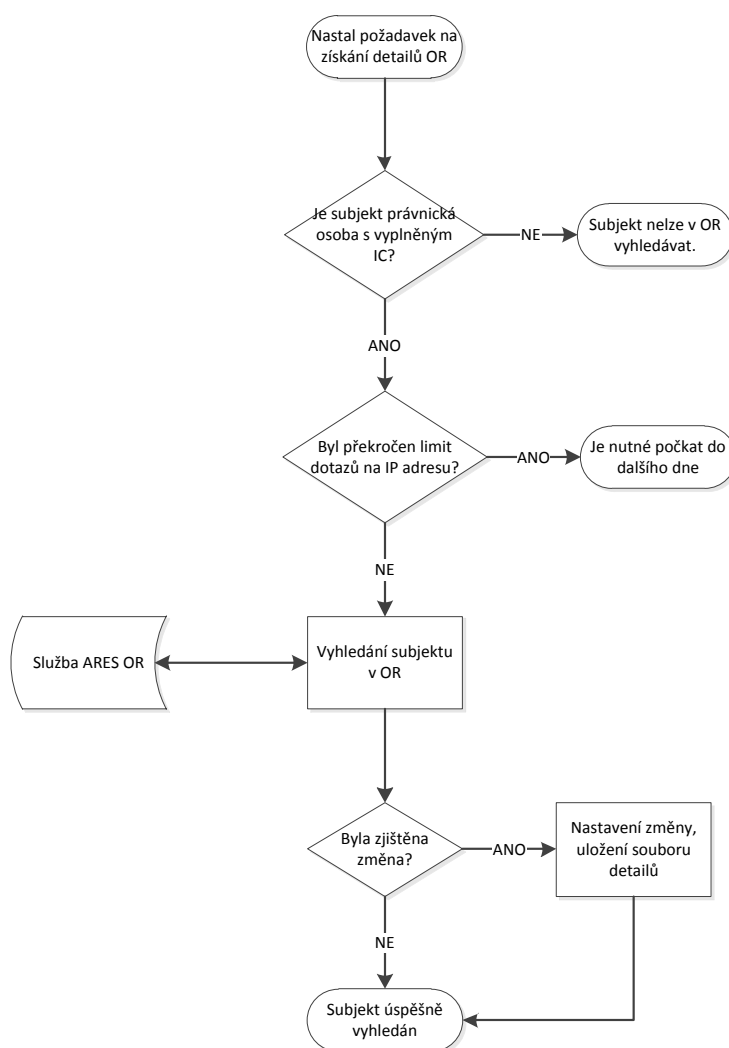
Služba ARES OR potom vyžaduje jako vstupní parametry Identifikační číslo subjektu a počítá s vyhledáváním jen právnických osob, viz. [59].

11.2.2. Procedurální popis funkcionality

Procesy monitorovacího programu zůstávají stejné jako definované v kapitole 6.5.2.2. nově je přidán proces vyhledání OR, který funguje podle následujícího postupu:

- Monitorovací program zkontroluje, zda-li je vyhledávaný subjekt právnická osoba a má vyplněné identifikační číslo.
- Monitorovací program zašle dotaz na databázi Společnosti a zjistí, zda-li nebyl překročen limit dotazů na informační systém ARES pro danou IP adresu.
- Monitorovací program zašle dotaz na službu ARES OR a stáhne výpis detailů.
- Monitorovací program porovná stažená data s daty minulými a případně indikuje změnu. V případě změny pak také uloží stažená data do souboru.

Proces vyhledání v ARES OR je na obrázku 36.



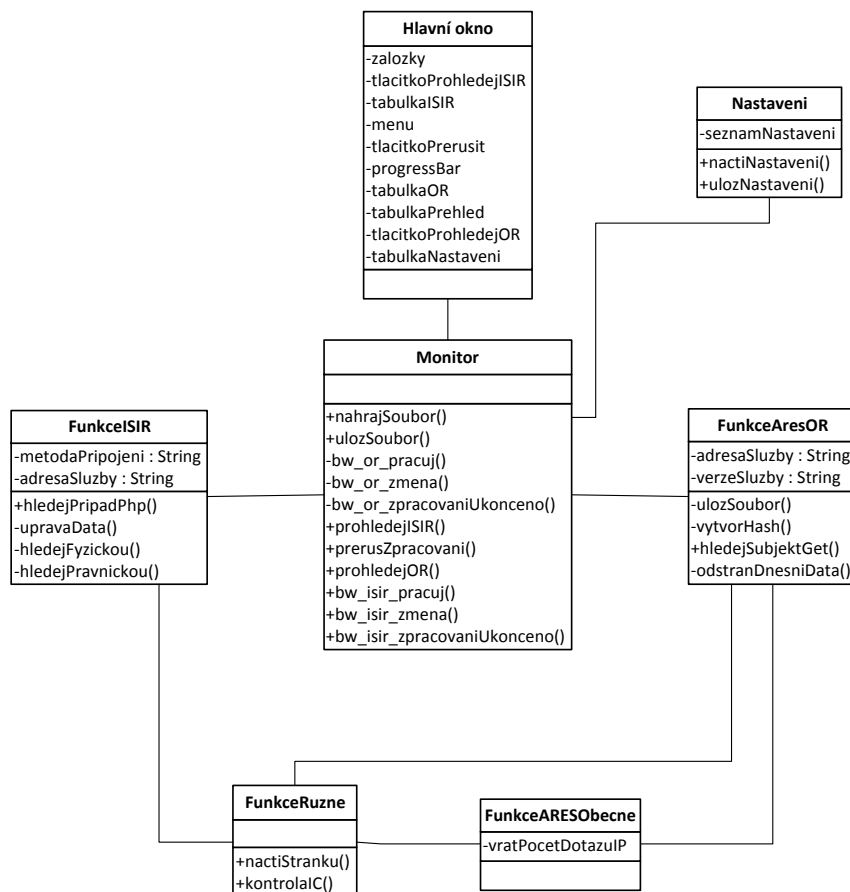
Obrázek 36 proces vyhledání v ARES OR (Zdroj: Autor)

11.2.3. Model tříd monitorovacího programu

V návaznosti na změny funkcionality jsem upravil model tříd monitorovacího programu. Vzhledem k tomu, že samotný program jsem již vytvářel s tím, aby bylo možno přidat další zdroje monitoring, nebyly tyto úpravy nijak zásadní, respektive nemění koncept programu. Úpravy probíhaly takto:

- Vzhledem k požadavkům na rozšíření uživatelského rozhraní z kapitoly 10.3.2 jsem upravil třídu "Hlavní okno", která představuje uživatelské rozhraní. Do třídy jsem doplnil tabulky pro pohled "OR" a "Přehled", dále tlačítko pro prohledání OR a také tabulku pro nastavení, pro specifikaci uložených souborů vyhledávání OR.
- Třidu monitor jsem rozšířil o metodu "prohledejOR", která slouží k prohledání OR a dále o metody "bw_or_pracuj", "bw_or_zmena" a "bw_or_zpracovaniUkonceno", které opět řídí více vláknové zpracování.
- Vytvořil jsem novou třídu "Nastaveni", která umí ukládat a načítat libovolná nastavení programu. Třída zatím bude sloužit ke zjišťování adresy souborů detailů OR, ale v případě nutnosti bude spravovat další nastavení. Třída obsahuje seznam nastavení (ve formátu "název","hodnota") a pak metody "nactiNastaveni" pro načtení nastavení a "ulozNastaveni" pro uložení těchto nastavení do souboru.
- Vytvořil jsem novou třídu "FunkceAresOr", která je obdobou třídy "FunkceISIR" a zabezpečuje komunikaci s webovou službou ARES OR pro logiku aplikace. Třída má dva atributy, "adresaSluzby", který určuje URL volání služby a "verzeSluzby", který určuje požadovanou verzi služby. Mezi metody třídy potom patří "ulozSoubor" pro ukládání souborů s detaily subjektu, "vytvorHash" pro tvorbu hash kódů, "hledejSubjektGet", která volá službu ARES OR s parametry hledaného subjektu a "odstranDnesniData", která odstraňuje časová data volání služby ze stažených souborů detailů.
- Vytvořil jsem novou třídu "FunkceARESObecne", která slouží k poskytnutí obecné funkcionality ARES. V této verzi je to metoda "vratPocetDotazuIP", což je metoda ošetřující nepřekročení limitu dotazů na informační systém ARES. V dalších verzích do této třídy budu umísťovat funkce, které se týkají systému ARES obecně a ne jen jeho částí. Důvod k vytvoření této třídy je pravidlo separace zájmů viz. [10].
- Vytvořil jsem novou třídu "FunkceRuzne", která slouží k zapouzdření obecné funkcionality, která může být využita u všech ostatních tříd, která není závislá na jednotlivých využívaných službách. Třída má metody "kontrolaIC", která validuje správnost identifikačního čísla (viz. [60]) a "nactiStranku", která načte webovou stránku podle zadaného URL a vrátí její obsah jako text.

Upravený model tříd monitorovacího programu je na obrázku 37.



Obrázek 37 upravený model tříd monitorovacího programu (Zdroj: Autor)

12. ANALÝZA/MODEL IMPLEMENTAČNÍ ARES

Vzhledem k tomu, že funkcionality ARES OR je rozšířením původního monitorovacího programu, tak i prostředky zůstávají stejné, tedy MySQL databáze a Visual Basic .NET pro tvorbu monitorovacího programu.

13. IMPLEMENTACE ARES

Kapitola se věnuje samotné úpravě Systému vycházející z analýzy a postupují podle kapitoly 4.2. Implementace opět proběhla v přírůstcích, které postupně zpřístupňovaly novou funkcionality a byly testovány.

13.1. Přírůstek 4.

Celkově čtvrtý přírůstek Systému se zabýval úpravou databáze a úpravou monitorovacího programu. Jeho výstupem je upravená funkcionality monitorovacího programu a rozšířené schéma databáze.

13.1.1. Úprava databáze

Databázi jsem upravil pomocí nástroje správy MySQL WorkBench. Změnu jsem provedl na lokální testovací databázi a vygeneroval si SQL kódy pro přenesení změny dat na provozní databázi po otestování funkcionality.

13.1.2. Úprava monitorovacího programu

Monitorovací program jsem vytvořil podle popisu v kapitole 11.2 za použití stejných postupů a nástrojů jako v kapitole 8.1.2.

13.1.2.1. Úprava uživatelského rozhraní

Uživatelské rozhraní jsem upravil podle návrhu z kapitoly 10.3.2.

13.1.2.2. Úprava aplikační logiky

Logiku monitorovacího programu jsem upravil podle návrhu z kapitoly 11.2.3. Při úpravách jsem opět definoval jednotkové testy a upravoval funkcionalitu do té doby, než všechny tyto testy byly úspěšně splněny. Po splnění jsem spojil uživatelské rozhraní s aplikační logikou.

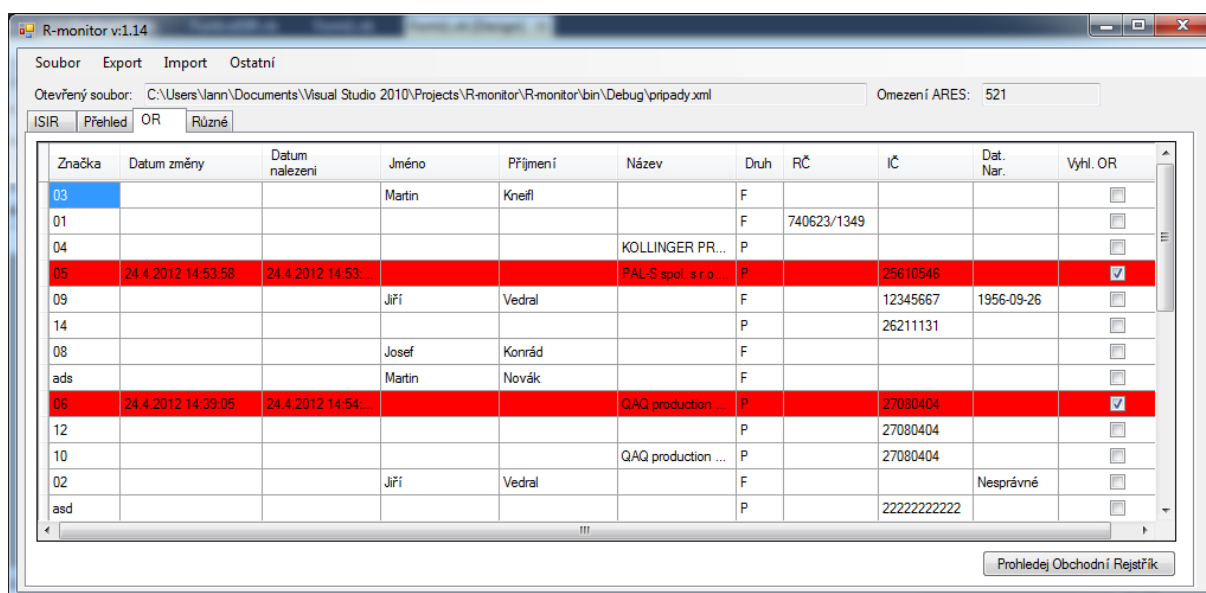
13.1.3. Interní testování čtvrtého přírůstku

Funkcionalitu monitoringu ARES OR jsem testoval s využitím lokální databáze pro monitoring IP adres. Předmětem testování bylo zajištění správné funkcionality programu, tedy především:

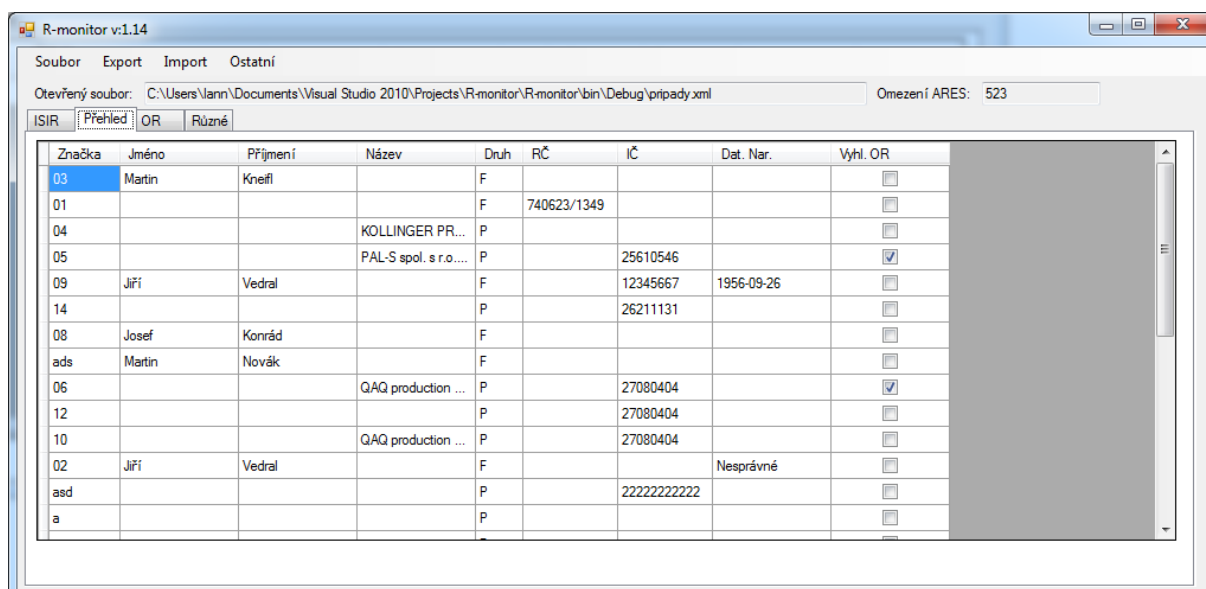
- Správná metoda vyhledávání subjektů, jak byla popsána v kapitole 11.2.2.
- Správa souborů s daty subjektů.
- Korektní sledování použitých IP adres na dotazy ARES OR (v průběhu testování jsem v jednom dnu několikrát změnil použitou IP adresu a opět se vracel).

Při testování přírůstku jsem se opět rozhodl přidat barevné zvýraznění jak nalezených subjektů (růžové) a subjektů se změnou v posledních 7 dnech (červené).

Na obrázku 38 je okno upraveného a otestovaného monitorovacího programu s pohledem "OR" a na obrázku 39 okno programu s pohledem "Přehled".



Obrázek 38 okno monitorovacího programu s pohledem "OR" (Zdroj: Autor)



Obrázek 39 okno monitorovacího programu s pohledem "Přehled" (Zdroj: Autor)

13.1.4. Akceptační testování čtvrtého přírůstku u třetí strany

Před testováním u třetí strany jsem provedl změnu provozní databáze skriptů vytvořenými v kapitole 13.1.1. Následně jsem zaslal upravený program monitoringu třetí straně spolu s manuálem. Třetí strana následně program schválila.

14. PROVOZ A UŽITÍ UPRAVENÉHO SYSTÉMU

V průběhu provozu a užití programu pro monitoring ISIR/ARES OR zatím nenastaly žádné významné problémy. Program byl následně nasazen u dvou dalších společností, které vyžadovaly přidání funkcionality importu dat. První společnost požadovala jen jednorázový import dat ze svého systému, zatímco druhá společnost požadovala opakovaný každodenní import sledovaných subjektů ze svého

systému. Jednorázový import byl přidán jako funkcionality programu, zatímco opakovaný import byl nakonec realizován pomocí systému společnosti, který data sám vkládá do zdrojového souboru monitorovacího programu.

15.ZÁVĚR

Vytvořil jsem Systém, který umožňuje replikaci databáze ISIR a její monitorování a dále umožňuje monitorování obchodního rejstříku v informačním systému ARES. Tyto dva základní zdroje byly zvoleny Společností, pro kterou jsem Systém vytvářel, jako prioritní a tedy hlavní cíl celé práce. Zároveň jsem Systém vytvořil tak, že přidání dalších zdrojů (především dalších rejstříků ARES) je poměrně jednoduché jak z hlediska aplikačního, tak z hlediska uživatelského a odpovídalo by obdobě kapitol 10-13.

Dále lze mou práci použít jako referenční zdroj pro implementaci dalších systému spolupracujících s ISIR, nebo ARES OR, protože obsahuje jak popis poskytovaných dat těchto systémů, tak popis komunikace s nimi. Práce by potom mohla být použita i jako referenční zdroj pro implementaci prakticky jakékoliv aplikace pro malou společnost s důrazem na rychlé nasazení této aplikace.

Vytvořený systém je v současnosti používán třemi jinými společnostmi, které projevíly zájem o monitoring daných rejstříků. Systém musel být v některých případech upraven o další možnosti importu dat, případně komunikace s jiným informačním systémem, ale jednalo se o jednoduché úpravy, které nevyžadovaly změnu architektury systému. Dále byl Systém u jedné společnosti integrován přímo do jejich aplikace správy pohledávek.

V budoucích verzích potom očekávám přidání dalších informačních zdrojů podle poptávky společností.

Z výše uvedených důvodů si potom myslím, že práce byla úspěšná a naplnil jsem její cíl.

16.TERMINOLOGICKÝ SLOVNÍK

.NET	Platforma řešení a programů společnosti Microsoft.	[autor]
Analýza	Postup, jehož výstupem je logický (esenciální) model vytvářeného systému	[8]
ARES	Administrativní Registr Ekonomických Subjektů	
Architektura	Zobrazení komponent systému.	[autor]
Databáze	Objekt uspořádání dat, který se nachází na počítačovém médiu a je vhodný pro práci s těmito daty, snaží se jí umožnit.	[autor]
Extrémní programování	Metodika pro malé až středně velké týmy, které vyvíjejí programové vybavení a musí se vyrovnat se zadáním, které se rychle mění, nebo není jasné.	[7]
Implementace	Proces realizace, zavedení navržené entity do reálného světa.	[autor]
ISIR	Insolvenční Rejstřík	[3]

Metodika	Návod, stanovující principy vytváření systému.	[autor]
Model	Dílčí pohled na vytvářený systém.	[8]
Server	Počítač, nabízející určitou funkcionalitu, schopný obsloužit vně přicházející požadavky; program, nabízející určitou funkcionalitu, schopný obsloužit vně přicházející požadavky.	[autor]
Společnost	Zadavatelská společnost, pro kterou je celý Systém vyvíjen.	[autor]
SQL	Structure Query Language, dotazovací jazyk	[25]
Systém	Systém pro sběr, přenos, uchování, zpracování a poskytování dat (informací, znalostí) využívaných při činnosti podniku. Jeho komponentami jsou informační a komunikační technologie, data a lidé. Jeho cílem je efektivní podpora informačních, rozhodovacích a řídicích procesů na všech úrovních organizace	[6]
Webová služba	Metoda komunikace mezi elektronickými zařízeními přes web.	[45]
XML	eXtensible Markup Language	[21]

17. SEZNAM LITERATURY

- [1] BUSINESSINFO. Pohledávky. Businessinfo.cz [online]. [cit. 2012-04-26]. Dostupné z: <http://www.businessinfo.cz/cz/clanek/orientace-v-pravnich-ukonech/pohledavky-opu/1000818/51671/#b1>
- [2] Úplné Znění - Obchodní zákoník. Ostrava: Sagit, 2009. ISBN: 978-80-7208-734-1
- [3] JUSTICE. Oficiální server českého soudnictví. justice.cz [online]. [cit. 2012-04-26]. Dostupné z: www.justice.cz
- [4] GOOGLE SCHOLAR. [online] [cit. 2012-03-16.] Dostupný z: scholar.google.cz/
- [5] GOOGLE, [online] [cit. 2012-03-16.] Dostupný z: www.google.com
- [6] VOŘÍŠEK, J. a kol. Principy a modely řízení podnikové informatiky. 1.vyd. Praha: Oeconomica, 2008. ISBN 978-80-245-1440-6.
- [7] BECK, Extrémní programování. 1.vyd. Praha: Grada, 2002. ISBN 80-247-0300-9
- [8] BUCHALCEVOVA, Alena. Metodiky budování informačních systémů. 1.vyd. Praha: Oeconomica, 2009. ISBN 978-80-245-1540-3
- [9] GÁLA, Libor. POUR, Jan, ŠEDIVÁ, Zuzana. Podniková informatika 2., přepracované a aktualizované vydání. 2. vyd. Praha: Grada Publishing. 2009. ISBN 978-80-274-2615-1

- [10] PECINOVSKÝ, Rudolf. Návrhové vzory. 1.vyd. Brno: Computer Press, 2007. ISBN 978-89-251-1582-4
- [11] REPA, Václav a kol. Metodika vývoje informačního systému s pomocí nástroje Power Designer. [online] [cit. 2012-03-16.] Dostupný z:
<http://opensoul.iquest.cz/forum/docs/publications/Metodika_vyvoje_IS_06_2006.pdf>.
- [12] JAYRATNA, Nimal. Understanding and Evaluating Methodologies. Londýn: McGraw-Hill, 2004. ISBN 0077078829
- [13] FOWLER, M; HIGHSMITH, J. The Agile Manifesto. In Dr. Dobb's Portal, 2006 [online] [cit. 2012-03-16.] Dostupný z: < <http://www.drdoobs.com/184414755?queryText=The+Agile+Manifesto> >
- [14] ITIL, [online] [cit. 2012-03-16.] Dostupný z:<<http://www.ital-officialsite.com/home/home.aspx>>
- [15] GAMMA, Erich a kol. Design Patterns: Elements of Reusable Object-Oriented Software. 1994. ISBN 978-0201633610
- [16] Rapid Application Development. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 25.3.2012 [cit. 2012-04-16.] Dostupný z: <http://en.wikipedia.org/wiki/Rapid_application_development>
- [17] PATTON, R. Testování softwaru. 1.vyd. Praha: Computer Press. 2002. ISBN:80-7226-636-5
- [18] BUSINESSINFO. Insolvenční zákon. Businessinfo.cz [online]. [cit. 2012-04-26]. Dostupné z: <<http://business.center.cz/business/pravo/zakony/insolvencni/>>
- [19] Portál veřejné správy [online]. [cit. 2012-04-26]. Dostupné z: <<http://portal.gov.cz/portal/obcan/>>
- [20] JUSTICE. Webová služba insolvenčního rejstříku. justice.cz [online]. [cit. 2012-04-26]. Dostupné z: <<http://portal.justice.cz/Justice2/ms/ms.aspx?o=1&j=1&k=170&d=209445>>
- [21] KOSEK, Jiří. XML pro každého. 1. vyd. Praha:Grada Publishing, 2000. ISBN: 80-7169-860-1
- [22] Replication. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 26.4.2012 [cit. 2012-04-27.] Dostupný z: <[http://en.wikipedia.org/wiki/Replication_\(computer_science\)](http://en.wikipedia.org/wiki/Replication_(computer_science))>
- [23] JUSTICE. Popis způsobu využívání webové služby. justice.cz [online]. [cit. 2012-04-26]. Dostupné z: <https://isir.justice.cz/isir/help/Popis_WS.pdf>
- [24] Ondra Žižka - Osobní web. Webová služba Insolvenční rejstřík. [online]. [cit. 2012-04-26]. Dostupné z: <http://ondra.zizka.cz/stranky/programovani/insolvencni_rejstrik-isir_ws/index.texy>

- [25] SQL. . In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 22.3.2012 [cit. 2012-03-25.] Dostupný z: <<http://en.wikipedia.org/wiki/SQL>>
- [26] MYSQL. [online]. [cit. 2012-04-26]. Dostupné z: <<http://www.mysql.com/>>
- [27] Data modeling. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 2.4.2012 [cit. 2012-04-6.] Dostupný z: <http://en.wikipedia.org/wiki/Data_modeling>
- [28] The Object-Oriented Database System Manifesto In: School of Computer Science [online] last modified on 26.4.1995 [cit. 2012-04-6.] Dostupný z: <<http://www.cs.cmu.edu/~clamen/OODBMS/Manifesto/htManifesto/Manifesto.html>>
- [29] Third-generation Database System Manifesto In: University of Cambridge [online] [cit. 2012-04-6.] Dostupný z: <<http://www.cl.cam.ac.uk/teaching/2003/DBaseThy/or-manifesto.pdf>>
- [30] HAKEN, Robert. Návrh schématu DB - Prezentace In: HAVIT Knowledge Base Science [online] last modified on 27.11.2011 [cit. 2012-04-6.] Dostupný z: <<http://knowledge-base.havit.cz/files/folders/ms-fest-2011/entry732.aspx>>
- [31] Klient-server. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 14.4.2012 [cit. 2012-04-20.] Dostupný z: <<http://cs.wikipedia.org/wiki/Klient-server>>
- [32] Database. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 5.4.2012 [cit. 2012-04-10.] Dostupný z: <<http://en.wikipedia.org/wiki/Database>>
- [33] Relational database. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 13.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Relational_database>
- [34] Object database. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 16.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Object_database>
- [35] Object-relational database. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 6.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Object-relational_database>

- [36] Edgar F. Codd. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 2.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Edgar_F._Codd>
- [37] Webhosting. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 2.4.2012 [cit. 2012-04-20.] Dostupný z: <<http://cs.wikipedia.org/wiki/Webhosting>>
- [38] FEUERLICHT, Jiří. Enterprise Computing: Concepts, Standards and Architectures. 1. vyd. Praha: Oeconomica. 2008 ISBN: 978-80-245-1367-6
- [39] AMAZON. Amazon Elastic Compute Cloud. www.amazon.com. [online]. [cit. 2012-04-20.] Dostupný z: <<http://aws.amazon.com/ec2/>>
- [40] MICROSOFT. Windows Azure. www.microsoft.com [online]. [cit. 2012-04-20.] Dostupný z: <<http://www.microsoft.com/cze/azure/>>
- [41] W3 TECHS. Usage of server-side programming languages for websites. w3techs.com [online]. [cit. 2012-04-20.] Dostupný z: <http://w3techs.com/technologies/overview/programming_language/all>
- [42] PHP. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 20.4.2012 [cit. 2012-04-20.] Dostupný z: <<http://cs.wikipedia.org/wiki/PHP>>
- [43] Programming Languages. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 20.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Programming_language>
- [44] MICROSOFT. .NET . www.microsoft.com [online]. [cit. 2012-04-20.] Dostupný z: <<http://www.microsoft.com/net>>
- [45] Web services. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 22.4.2012 [cit. 2012-04-24.] Dostupný z: <http://en.wikipedia.org/wiki/Web_service>
- [46] PATTON, Tony. Consume a Web service in a .NET app In: TECHREPUBLIC [online]. 1.6.2005 [cit. 2012-04-24.] Dostupný z: <<http://www.techrepublic.com/article/consume-a-web-service-in-a-net-app/5768122>>

- [47] Jak psát jednoduché webové služby v jazyce Visual Basic.NET nebo jazyka Visual Basic 2005. In: Pomoc a Podpora Microsoft. microsoft.com [online]. [cit. 2012-04-24.] Dostupný z: <<http://support.microsoft.com/kb/301273>>
- [48] How to: Implement a Form That Uses a Background Operation. In: MSDN. microsoft.com [online]. [cit. 2012-04-24.] Dostupný z: <<http://msdn.microsoft.com/en-us/library/waw3xexc.aspx>>
- [49] HALVORSON, M. Microsoft Visual Basic 2008 Krok za krokem. 1.vyd. Brno: Computer Press, 2008. ISBN 978-80-251-2221-1
- [50] NAGEL, Christian a kol. C# 2008 Programujeme profesionálně. 1. vyd. Praha: Computer Press. 2009. ISBN: 978-80-251-2401-7
- [51] CODE CONVERTER In: Developer Fusion. [online]. [cit. 2012-04-24.] Dostupný z: <<http://www.developerfusion.com/tools/convert/csharp-to-vb/>>
- [52] Visual Basic Programming Guide In: MSDN. microsoft.com [online]. [cit. 2012-04-24.] Dostupný z: <<http://msdn.microsoft.com/library/y4wf33f0>>
- [53] Codeguru. codeguru.com [online]. [cit. 2012-04-24.] Dostupný z: <<http://www.codeguru.com/>>
- [54] Unit testing. Wikipedia. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 12.4.2012 [cit. 2012-04-20.] Dostupný z: <http://en.wikipedia.org/wiki/Unit_testing>
- [55] Database transaction. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 27.4.2012 [cit. 2012-04-30.] Dostupný z: <http://en.wikipedia.org/wiki/Database_transaction>
- [56] ARES In: Administrativní registr ekonomických subjektů [online]. [cit. 2012-04-24.] Dostupný z: <<http://wwwinfo.mfcr.cz/ares/ares.html.cz>>
- [57] ARES-Jmenné prostory In: Administrativní registr ekonomických subjektů [online]. [cit. 2012-04-24.] Dostupný z: <http://wwwinfo.mfcr.cz/ares/xml_doc/schemas/index.html>
- [58] Hash function. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 12.4.2012 [cit. 2012-04-30.] Dostupný z: <http://en.wikipedia.org/wiki/Hash_function>
- [59] Doplnující informace k službě OR In: Administrativní registr ekonomických subjektů [online]. [cit. 2012-04-24.] Dostupný z: <http://wwwinfo.mfcr.cz/ares/ares_xml_or.html.cz>

[60] Jak ověřit platné IČ. Latrine [online]. [cit. 2012-04-24.] Dostupný z: <<http://latrine.dgx.cz/jak-overit-platne-ic-a-rodne-cislo>>

18. SEZNAM OBRÁZKŮ

Obrázek 1 získání popisu Systému (Zdroj: Autor).....	14
Obrázek 2 omezení Systému (Zdroj: Autor)	15
Obrázek 3 příklad použití monitoringu ISIR (Zdroj: Autor).....	17
Obrázek 4 řešení monitoringu ISIR (Zdroj: Autor).....	19
Obrázek 5 návrh uživatelského rozhraní aktualizacího programu (Zdroj: Autor).....	22
Obrázek 6 návrh uživatelského rozhraní monitorovacího programu (Zdroj: Autor).....	23
Obrázek 7 výpis detailů insolvenčního řízení (Zdroj: JUSTICE)	24
Obrázek 8 struktura poznámky (Zdroj: MANUAL ISIR)	28
Obrázek 9 diagram možných přechodů mezi stavy věci (Zdroj: MANUAL ISIR)	30
Obrázek 10 legenda přechodů mezi stavy věci (Zdroj: MANUAL ISIR).....	30
Obrázek 11 struktura elementu osoby (Zdroj: MANUAL ISIR)	31
Obrázek 12 návrh úložiště dat (Zdroj: Autor)	33
Obrázek 13 využití Systému více klienty (Zdroj: Autor)	34
Obrázek 14 varianta tlustého klienta (Zdroj: Autor)	35
Obrázek 15 varianta lehkého klienta (Zdroj: Autor)	36
Obrázek 16 popis chování webové služby vyhledání (Zdroj: Autor).....	39
Obrázek 17 vyhledání dlužníka ve webové službě prohledávání (Zdroj: Autor)	40
Obrázek 18 objektový model webové služby (Zdroj: Autor).....	41
Obrázek 19 proces nahrávání dat aktualizacím programem (Zdroj: Autor)	44
Obrázek 20 model objektů poskytovaných službou (Zdroj: Autor).....	46
Obrázek 21 proces nahrání dat do databáze (Zdroj: Autor)	48
Obrázek 22 model tříd aktualizacího programu (Zdroj: Autor)	50
Obrázek 23 procesy editace a vyhledání monitorovacího programu (Zdroj: Autor).....	52
Obrázek 24 proces tisku detailu insolvenčního řízení (Zdroj: Autor)	53
Obrázek 25 metoda vyhledávání monitorovacího programu (Zdroj: Autor).....	54
Obrázek 26 model tříd monitorovacího programu (Zdroj: Autor)	55
Obrázek 27 fyzický model databáze (Zdroj: Autor).....	57
Obrázek 28 tvorba tabulky v prostředí MySQL WorkBench (Zdroj: Autor)	60
Obrázek 29 aktualizací program plní data (Zdroj: Autor).....	62
Obrázek 30 okno monitorovacího programu (Zdroj: Autor).....	66
Obrázek 31 okno monitorovacího programu při vyhledávání (Zdroj: Autor).....	67
Obrázek 32 upravené schéma databáze Společnosti (Zdroj: Autor)	72
Obrázek 33 okno monitorovacího programu s pohledem "Přehled" (Zdroj: Autor)	73

Obrázek 34 okno monitorovacího programu s pohledem "OR" (Zdroj: Autor).....	73
Obrázek 35 pozměněné architektura monitorovacího Systému (Zdroj: Autor)	74
Obrázek 36 proces vyhledání v ARES OR (Zdroj: Autor).....	75
Obrázek 37 upravený model tříd monitorovacího programu (Zdroj: Autor)	77
Obrázek 38 okno monitorovacího programu s pohledem "OR" (Zdroj: Autor).....	79
Obrázek 39 okno monitorovacího programu s pohledem "Přehled" (Zdroj: Autor)	79

19.SEZNAM TABULEK

Tabulka 1 specifikace struktury akce (Zdroj: MANUAL ISIR).....	27
Tabulka 2 struktura událostí (Zdroj: MANUAL ISIR)	28
Tabulka 3 struktura věci/řízení (Zdroj: MANUAL ISIR)	29
Tabulka 4 specifikace osoby (Zdroj: MANUAL ISIR).....	31
Tabulka 5 specifikace adresy (Zdroj: MANUAL ISIR).....	32
Tabulka 6 porovnání databází podle kritérií (Zdroj: Autor).....	56