České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídící techniky

Diplomová práce

Řešení procesu pro práci s neplatiči v informačním systému pro výběr mýta

Bc. Lukáš Maroušek

Vedoucí práce: Ing. Tomáš Marek

Studijní program: Otevřená informatika, magisterský

Obor: Počítačové inženýrství

Leden 2012

Poděkování

Děkuji panu Ing. Tomáši Markovi, vedoucímu mé diplomové práce a jeho kolegům, zejména pak Bc. Marii Píchové, za jejich čas, ochotu a rady při zpracování této práce. Dále bych rád poděkoval rodině a přátelům za jejich podporu a trpělivost.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změnách některých zákonů (autorský zákon).

Ve Voticích dne 2.1. 2012 …….

Abstract

This thesis deals with the analysis, design and implementation of solution for working with debtors in information system for electronic toll collection.

The theoretical part of this thesis analyzes the requirements in solution for working with debtors. It compares existing solutions for processes description and their processing. Proposes software`s solution and business process of solution for working with debtors. The practical part deals implementation.

Abstrakt

Tato diplomová práce se zabývá analýzou, návrhem a implementací řešení procesu pro práci s neplatiči v informačním systému pro výběr mýta.

Teoretická část práce analyzuje požadavky pro řešení procesu s neplatiči. Porovnává existující řešení pro popis obecných procesů a jejich zpracování. Navrhuje vlastní softwarové řešení celého systému a business proces pro práci s neplatiči. Praktická část řeší implementaci.

Obsah

Seznam obrázků ..... **xv**

Seznam tabulek ..... **xvii**

1 Úvod **1**

1.1 Zadání 2

1.2 Motivace 2

1.3 Cíle 3

2 Úvodní studie **4**

2.1 Obecné rozdělení aplikací 4

2.1.1 Desktopová aplikace 4

2.1.2 Klient - server aplikace 4

2.1.2.1 Tlustý klient 5

2.1.2.2 Tenký klient 5

2.1.3 Webová aplikace 6

2.1.4 Back - end aplikace 7

2.1.5 Front - end aplikace 7

2.2 Business proces 8

2.2.1 Workflow 8

2.3 Doporučené technologie 9

2.3.1 .NET Framework 4.0 9

2.3.1.1 Technologie WCF 9

2.3.1.2 Technologie WPF 10

2.3.2 Silverlight technologie 12

2.3.3 Oracle databáze 13

3 Analýza ... 15

3.1 Požadavky na workflow proces ... 15

3.2 Uživatelské role ... 16

3.2.1 Administrátor ... 16

3.2.2 Operátor ... 16

3.2.3 Manažer workflow ... 16

3.3 Rozdělení systému ... 16

3.4 Požadavky na systém ... 17

3.4.1 Dunning application server ... 18

3.4.2 User manager ... 20

3.4.3 Task manager ... 22

3.4.4 Workflow manager ... 23

3.5 Existující řešení ... 25

3.5.1 Microsoft BizTalk Server ... 25

3.5.2 K2 blackpearl ... 27

3.5.3 Workflow Foundation ... 28

3.5.4 Porovnání řešení ... 30

4 Návrh ... 32

4.1 Workflow proces pro práci s neplatiči ... 32

4.2 Struktura systému ... 34

4.3 Dunning application server ... 35

4.3.1 Server logic ... 36

4.3.2 Logger ... 36

4.3.3 Mail sender ... 37

4.3.4 Oracle data connector ... 37

4.3.5 Dunning logic 38
4.3.6 Debtor serialization 38
4.3.7 Engine initializer 38
4.3.8 Engine starter 40
4.3.9 Task manager service 42
4.3.10 Workflow manager service 42
4.3.11 Administration service 42
4.3.12 Authentication service 43
4.4 Web server 43
4.5 Task manager 43
4.5.1 Stránka o aplikaci 44
4.5.2 Stránka správce úkolů 44
4.5.3 Dialogové okno řešení úkolu 45
4.6 Workflow manager 45
4.6.1 Stránka o aplikaci 46
4.6.2 Stránka návrhář workflow 46
4.6.2.1 Dialogové okno nový návrh 48
4.6.2.2 Dialogové okno otevřít návrh 48
4.6.3 Stránka správce workflow 48
4.7 User + debtor manager 48
4.7.1 Stránka o aplikaci 49
4.7.2 Stránka správce dlužníků 49
4.7.2.1 Dialogové okno přidat dlužníka 50
4.7.2.2 Dialogové okno detail dlužníka 50
4.7.3 Stránka správce uživatelů 50

4.7.3.1 Dialogové okno přidat uživatele 50

4.7.3.2 Dialogové okno detail uživatele 51

4.8 Oracle database 51

5 Implementace 57

5.1 Dunning application server 57

5.1.1 Struktura řešení 57

5.1.2 Reprezentace workflow diagramu 58

5.1.3 Reprezentace logiky úloh 60

5.1.4 WCF služby 61

5.2 Klientské aplikace 63

5.2.1 Struktura řešení 63

5.2.2 Webový server 65

5.3 Databázový server 66

6 Závěr 67

7 Použitá literatura 68

Příloha A – Seznam použitých zkratek 71

Příloha B – Workflow xml metadata 73

Příloha C – Ukázky klientských aplikací 75

Příloha D – Obsah přiloženého CD 77

Seznam obrázků

Obrázek 2-1: Příklad WPF aplikace 11

Obrázek 2-2: Příklad Silverlight aplikace 13

Obrázek 3-1: Diagram případu užití aplikace User Manager 22

Obrázek 3-2: Diagram případu užití aplikace Task Manager 23

Obrázek 3-3: Diagram případu užití aplikace Workflow Manager 25

Obrázek 3-4: BizTalk Orchestration Designer 26

Obrázek 3-5: Webové rozhraní K2 blackpearl pro návrh business procesů 28

Obrázek 3-6: Návrhář business procesů v Microsoft Visual Studiu 30

Obrázek 4-1: Diagram workflow procesu pro práci s neplatiči 33

Obrázek 4-2: Diagram propojení aplikací a databáze 34

Obrázek 4-3: Diagram komponent pro Dunning Application Server 35

Obrázek 4-4: Aktivity diagram pro volání uložené procedury 37

Obrázek 4-5: Diagram objektu Workflow 39

Obrázek 4-6: Diagram objektu MetodaUlohy 39

Obrázek 4-7: Aktivity diagram pro vykonání jedné úlohy workflow procesu 41

Obrázek 4-8: Layout hlavní stránky aplikace Task Manager 44

Obrázek 4-9: Layout hlavní stránky aplikace Workflow Manager 46

Obrázek 4-10: Rozložení hlavní stránky aplikace User + Debtor Manager 49

Obrázek 4-11: Relační databázový model, část 1 55

Obrázek 4-12: Relační databázový model, část 2 56

Obrázek 5-1: Struktura řešení implementace Dunning application serveru 58

Obrázek 5-2: Implementace atributu logiky úloh 60

Obrázek 5-3: Struktura WCF služby 61

Obrázek 5-4: Hostování WCF služeb 62

Obrázek 5-5: Struktura řešení implementace klientské aplikace 63

Obrázek 5-6: Uložená procedura v oracle databázi 66

Obrázek C-1: Aplikace WorkflowManager 75

Obrázek C-2: Aplikace User + Debtor Manager 75

Obrázek C-3: Aplikace TaskManager 76

Seznam tabulek

Tabulka 2-1: Srovnání edic oracle databáze 14

Tabulka 3-1: Srovnání existujících produktů pro řešení business procesů..... 31

1 Úvod

V současnosti jsou nasazovány různé informační systémy téměř do všech odvětví lidských činností.

Jedno z odvětví lidských činností je výběr mýtných poplatků. V české republice je stále částečně používáno časové zpoplatnění. Pokud tedy chceme v ČR používat zpoplatněné úseky silnic a dálnic, musíme mít na svém osobním automobilu vylepenou platnou dálniční známku. Z ekonomického pohledu je však nejefektivnější výkonové zpoplatnění. Provozovatel vozidla tak platí pouze za skutečný počet ujetých kilometrů a možné škody způsobené provozem vozidla na pozemní komunikaci. Tento druh zpoplatnění je u nás používán pro nákladní dopravu a to pouze na vybraných úsecích dálnic.

V dnešní době se ve světe, zejména v Evropě, využívá několik různých systémů elektronického výběru mýtných poplatků [1]:

- systémy založené na komunikaci na krátkou vzdálenost (mýtné brány)
- systémy založené na globálním navigačním systému
- systémy založené na automatickém rozpoznávání registračních značek

Každý takovýto systém je velice složitý. Zpravidla bývá rozdělen do několika částí, které spolu komunikují. Jednou z mnoha částí je i řešení problému, kdy uživatelé (řidiči) své poplatky neplatí. Právě tímto tématem se zabývá tato práce.

1.1 *Zadání*

Úkolem mé diplomové práce je navrhnout a implementovat řešení procesu pro práci s neplatiči mýtného poplatku. Řešení tohoto procesu zahrnuje příjem informace o neplatiči, kombinaci manuálních a automatických kroků při řešení, šíření manuálních kroků mezi operátory a notifikaci o vyřešení. Kombinaci manuálních a automatických kroků říkáme business proces [Kapitola 2.2]. Návrh tohoto procesu je součástí zadání.

Výsledný systém tedy bude obsahovat několik aplikací. Samotnou aplikaci řešící proces neplatiče, aplikaci pro návrh business procesů a aplikaci pro operátory.

Všechny aplikace budou implementovány v programovacím jazyce C# v prostředí .NET Framework 4.0. Uživatelsky interaktivní aplikace lze implementovat ve Windows Presentation Foundation [Kapitola 2.4.1.2] nebo Silverlight [Kapitola 2.4.2].

1.2 *Motivace*

V dnešní době je na trhu hned několik existujících řešení [Kapitola 3.4], které se zabývají návrhem a řešení business procesů. Všechna řešení jsou však velmi rozsáhlá a náročná na použité zdroje. S tím souvisí i jejich výkonnost. Obecné řešení je tak sice vhodné na mnoho odvětví lidských činností, ale bývá mnohokrát pomalejší. Dalším důvodem vzniku je i to, že každé řešení neobsahuje jednoduchý a uživatelsky přívětivý návrhář business procesů. Navrhovat business procesy ve vývojovém prostředí, které primárně slouží k vývoji aplikací, není pro běžného uživatele příliš šťastným řešením.

Mojí motivací je tedy vytvořit takové řešení, které bude efektivní a bude obsahovat intuitivní návrhář business procesů.

1.3 *Cíle*

Cílem této práce je tedy navrhnout a implementovat informační systém řešící proces neplatiče mýtného poplatku. S tímto hlavním cílem souvisí mnoho věcí.

První věcí je vytvoření teoretické základny pro další analýzu práce. Sem patří rozbor důležitých pojmů v oblasti vývoje aplikací, obecných business procesů a doporučených technologií pro vývoj.

Další věcí je samotná analýza. Do této skupiny patří rozdělení systému na jednotlivé aplikace, návrh uživatelských rolí, rozbor požadavků na systém a porovnání existujících systémů pro řešení business procesů.

Další v pořadí je návrh. Zde je nutné nejdříve navrhnout architekturu celého systému. Dále navrhneme jednotlivé aplikace a jejich funkční celky pomocí aktivity diagramů. Nakonec vytvoříme návrh datového úložiště.

Mezi posledními je implementace a testování. Podrobný popis implementace zde není nutný, jelikož celý systém bude k nahlédnutí na přiloženém CD. Popíšeme tedy jen důležité a zajímavé části implementace. Na otestování systému použijeme testy funkčnosti a použitelnosti. Pro testy funkčnosti je důležité navrhnout příslušné scénáře.

2 Úvodní studie

Tato kapitola rozebírá důležité pojmy v oblasti vývoje aplikací a doporučené technologie pro vývoj.

2.1 *Obecné rozdělení aplikací*

Obecně můžeme aplikace rozdělit podle různých kritérií:

- typ aplikace
    - desktopová aplikace
    - klient server aplikace
- druh aplikace
    - front - end aplikace
    - back - end aplikace

2.1.1 *Desktopová aplikace*

Pod pojmem desktopová aplikace si představíme lokálně běžící software na stolním počítači nebo notebooku. Primárně pracuje celá aplikace s lokálně uloženými daty a používá lokálně nainstalované nástroje. Aplikaci může používat jeden nebo i více uživatelů. V případě používání více uživateli je nutné každému zajistit vlastní instalaci.

Výhodou takovéto aplikace je její nezávislost na jakémkoliv připojení do sítě a tím i použití na libovolném místě. Mezi nevýhody lze zařadit téměř nulovu podporu sdílení informací mezi uživateli, jelikož každý uživatel pracuje se svými lokálními daty.

2.1.2 *Klient - server aplikace*

Klient server aplikace je opakem desktopové aplikace. Tyto aplikace jsou primárně určeny pro sdílení většího množství dat mezi uživateli. Veškerá data

jsou uložena na serveru a podle typu klienta pak dochází pouze k jejich prezentaci, nebo dalšímu zpracování.

Klienty lze rozdělit do následujících skupin [2]:

- tlustý klient
- tenký klient

2.1.2.1 *Tlustý klient*

Tento typ klienta v sobě obsahuje prezentační i aplikační vrstvu dohromady. K datovému úložišti se připojuje přímo. Typickou vlastností tohoto druhu klienta je stahování velkého objemu dat, které dále zpracovává a posílá zpět na server. Z popisu vyplývají následující výhody a nevýhody.

Výhody

- server nevyžaduje výkonný hardware
- snadný a rychlý vývoj
- rychlá odezva aplikace

Nevýhody

- klient vyžaduje výkonný hardware
- vysoké nároky na šířku pásma
- obtížné nasazení, update a správa

2.1.2.2 *Tenký klient*

Typickým příkladem tenkého klienta je webový prohlížeč, který pouze komunikuje s prezentační vrstvou přes bezestavový HTTP protokol a stará se tak pouze o prezentaci dat. Na tomto druhu klienta se nevykonává žádná aplikační logika, do které se však nepočítá např. validace dat zadaných do webového formuláře. Komunikace mezi tenkým klientem a serverem je mnohem častější, než je tomu u tlustého klienta. Objem přenášených dat je však mnohem menší.

Z důvodu rozčlenění aplikace do několika vrstev se stává vývoj aplikace pro tenkého klienta mnohem náročnějším.

Výhody

- klient nevyžaduje výkonný hardware
- nízké nároky na šířku pásma
- snadné nasazení, update a správa

Nevýhody

- server vyžaduje výkonný hardware
- náročný vývoj
- pomalá odezva aplikace

2.1.3 *Webová aplikace*

Webová aplikace je typickým příkladem klient – server aplikace s tenkým klientem. Jako tenký klient je zde použit libovolný webový prohlížeč (Internet Explorer, Mozilla Firefox, Google Chrome, aj.). Tento typ aplikace pouze zprostředkovává požadavky na webový server a zajišťuje zobrazení získaných informací, což plyne z definice tenkého klienta [Kapitola 2.1.2.2].

Podstatou vývoje webových aplikací je jejich nezávislost na operačním systému. Není tedy nutné psát aplikaci pro různé operační systémy, ale postačí nám pouze jedna verze. Z praxe ovšem vyplývá, že můžou nastat problémy při zobrazování webové aplikace v různých webových prohlížečích. Toto bývá nejčastěji způsobeno nekonzistentní implementací HTML, CSS, DOM aj., nebo uživatelským nastavením daného webového prohlížeče.

Při programování webových aplikací je nutné dbát na následující hlediska:

- bezpečnost
    - hashování přístupových údajů
    - zálohování

- rozšiřitelnost
    - počítat z modifikacemi už při návrhu
    - složitější webové aplikace vyvíjet modulárně
- rychlost
    - optimalizovat zdrojový kód na rychlost
- vysoká zátěž
    - optimalizace databáze + výpočtů
    - používat webové cache
- testování
    - testování aplikace na vývojovém serveru

2.1.4 *Back - end aplikace*

Jako back-end se označuje aplikace nebo její část, která není pro běžného uživatele přístupná.

Pokud budeme mluvit o back-end aplikaci nějakého podnikového systému, bude se s největší pravděpodobností jednat o aplikaci běžící na serveru. Tato aplikace pak obecně nabízí různá rozhraní, které používají její klientské aplikace.

V případě back-endu webové aplikace mluvíme o části, která slouží k administraci webu a ke zpracování dat. Pokud si představíme internetový obchod, tak back-end takovéto aplikace bude např. sloužit ke vkládání nového zboží, k úpravám stávajícího zboží a k nastavování vzhledu celého internetového obchodu.

2.1.5 *Front - end aplikace*

Front-end aplikace je právě naopak viditelná a použitelná pro běžného uživatele. Obecně se tedy jedná o uživatelské rozhraní, část softwaru nebo webovou stránku, kterou uživatel vidí na obrazovce a může s ní pracovat.

Pokud si opět představíme webovou aplikaci jako internetový obchod, tak front-end takovéto aplikace bude tvořit např. katalog zboží, nákupní košík a objednávkový formulář.

2.2 *Business proces*

V současnosti se k řízení podnikových činností s oblibou používají workflow systémy. Tyto systémy nacházejí využití převážně v průmyslu a službách pro řízení obchodních, řídících, komunikačních a administračních procesů. Všechny tyto procesy lze považovat za business procesy. Obecně lze tedy říci, že business proces je workflow [3]. Aplikaci, která implementuje nějaký business proces, říkáme business aplikace.

2.2.1 *Workflow*

Workflow je obecně schéma (diagram) provádění nějaké komplexní činnosti rozepsané na jednotlivé kroky a jejich vazby. Schéma workflow je deterministické, což znamená, že v každém kroku víme, které kroky budou následovat. Vykonávání workflow probíhá od začátku do konce diagramu a má ho na starosti tzv. workflow engine.

Popsat workflow můžeme dvěma jazyky k tomu určenými. Prvním je jazyk BPEL [4] (Business Process Execution Langugage). Druhým je grafický jazyk BPMN [4] (Business Process Modeling Notation). Zde jde o množinu grafických objektů a pravidel, které jsou mezi sebou propojeny relacemi. Další zkratkou vyskytující se ve světe workflow systémů je BPM [5] (Business Process Modeling), kterou je označováno celkové řešení. Celkovým řešením je myšleno modelování business procesů za použití BPMN nebo BPEL, workflow engine, reportování výsledků, práce s rolemi a možnost integrace s jinými systémy.

V případě této práce si pod pojmem workflow představíme schéma provádění procesu pro neplatiče mýtného poplatku. Pokud tedy v dalším textu

narazíme na slovo workflow, workflow proces nebo workflow diagram, stále je tím myšleno schéma provádění procesu pro neplatiče mýtného poplatku.

2.3 *Doporučené technologie*

Následující kapitoly popisují doporučené technologie k vývoji celého systému.

2.3.1 *.NET Framework 4.0*

Pojem .NET Framework představuje rozsáhlou softwarovou platformu určenou pro vývoj aplikací do různých prostředí [6]. Za pomoci této platformy můžeme vyvíjet klasické aplikace pro Windows, webové aplikace, ale i aplikace pro mobilní zařízení.

Celý framework je složen z velké sady knihoven, funkcí a samotného běhového prostředí (CLR), které zajišťuje běh a kompilaci aplikací. Mnoho obsažených funkcí ulehčuje vývojářům práci a tím tak dochází i k menšímu výskytu chyb. Vývoj v .NET Frameworku je tak rychlý a méně náchylný na chyby programátora.

S příchodem nové verze .NET Framework 4.0 došlo oproti předešlým verzím k následujícím inovacím:

- vylepšené běhové prostředí (CLR), sada knihoven a funkcí
- vyšší výkon včetně podpory vícejádrových procesorů a uvolnění paměti na pozadí
- vylepšené technologie WCF, WPF a WF
- inovativní funkce paralelního programování

2.3.1.1 *Technologie WCF*

Window Comunication Foundation je jednotný framework pro vytváření servisně orientovaných aplikací [7]. S výhodou ho lze použít

pro meziprocesorovou, ale především pro síťovou komunikaci. Celé WCF je založené na komunikaci pomocí zpráv. Příkladem zprávy může být např. HTTP požadavek. Model této technologie rozlišuje mezi klientem a službou. Klient je aplikace, která započíná komunikaci. Služba je aplikace, která čeká na požadavky od klientů.

Základním prvkem služby je jeden nebo více endpointů. Endpoint, nebo-li koncový bod, je místo na příjem nebo odesílání zpráv. Každý endpoint se skládá ze třech částí:

- adresa – udává, kam jsou zprávy odesílány
- binding – určuje druh komunikace (HTTP, TCP, aj.), zahrnuje i zabezpečení
- contract – rozhraní služby obsluhující náš požadavek

Nastavovat endpointy můžeme za pomoci kódu, konfigurace, nebo kombinací obou dvou možností. Výhodou nastavení endpointů za pomoci konfiguračního souboru je, že konfiguraci nemusí nutně provádět vývojář, jelikož není nutné rekompilovat kód.

Pro posílání zpráv existují tři různé modely:

- jednosměrné posílání zpráv (simplex)
- asynchronně obousměrné posílání zpráv (duplex)
- synchronně obousměrné posílání zpráv (request reply)

WCF služby můžeme hostovat za pomoci IIS, nebo lze použít vlastní hostovací aplikace, která pak kontroluje celý životní cyklus služby. Další možností je pak využití služeb windows.

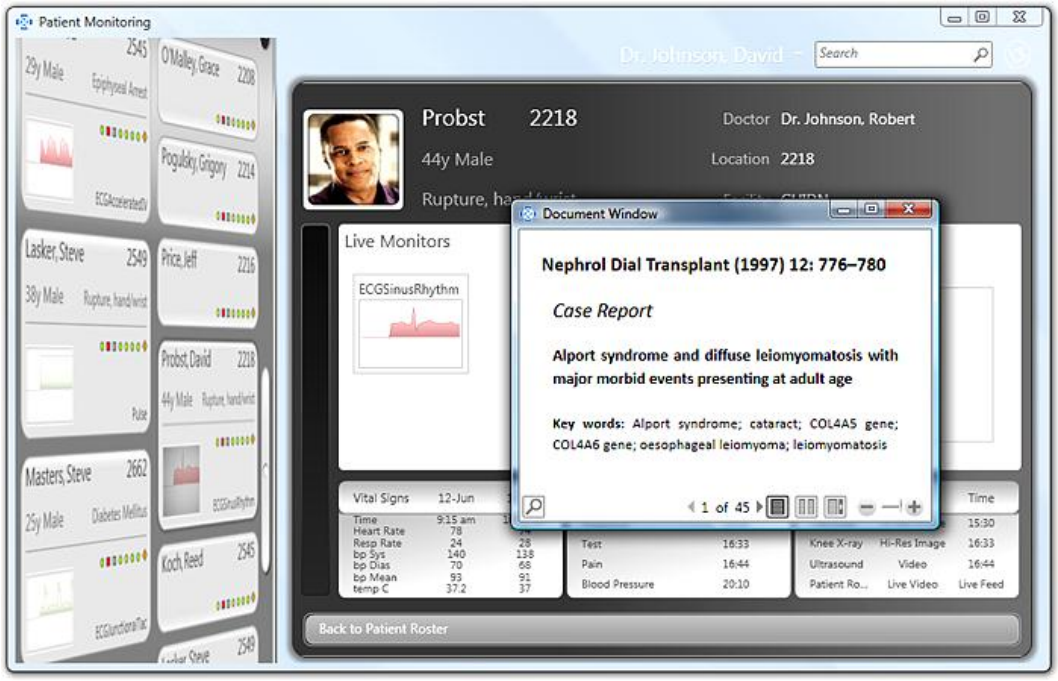
2.3.1.2 *Technologie WPF*

Window Presentation Foundation je grafický framework navržený pro vytváření windows aplikací [8]. WPF je nástupcem Windows Forms. Nejde

ovšem hovořit o náhradě Windows Forms. WPF je pouze další možností jak vyvíjet windows aplikace. V této technologii lze tedy vytvářet více uživatelsky bohaté aplikace, než tomu bylo u Windows Forms.

Jedná se zde hlavně o možnost využití následujících vylepšeních:

- vektorová grafika
    - bezeztrátová změna velikosti prvků
    - používání geometrických tvarů
- animace
    - časová, nebo za použití rámců
    - rotace, přechod barev, 3D animace
- multimédia - práce s audiem a videem (avi, mpeg, wmv)
- efekty - stíny, záře, rozostření, průhlednost a zrcadlení
- interaktivní 3D aplikace - poskytují podmnožinu funkcí z Direct3D
- data binding – možnost provázání dat v aplikaci



Obrázek 2-1: Příklad WPF aplikace

Vývoj vzhledu aplikace je u WPF oproti Windows Forms rozšířen. Nabízejí se zde dvě možnosti. První možností je umístění grafických prvků do okna aplikace pomocí drag and drop. U těchto prvků se pak dají nastavit jejich další vlastnosti. Tento model vývoje vzhledu aplikace byl použit i u Windows Forms. Technologie WPF však přišla s další možností a tím je značkovací jazyk XAML (čtěte zammel). Výhody tohoto jazyka jsou zejména v čistotě, jednoduchosti a rychlosti psaní kódu. Zároveň dochází k oddělení aplikační logiky a vzhledu, neboť jeden XAML soubor definuje vzhled a druhý soubor v některém z .NET jazyků definuje logiku aplikace.

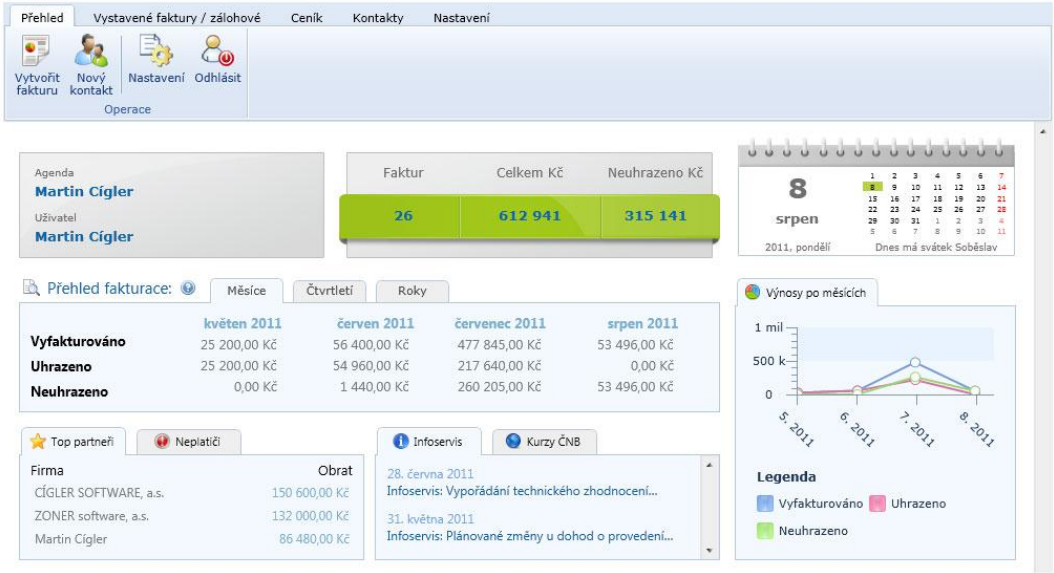
2.3.2 *Silverlight technologie*

Silverlight je moderní technologie pro internetové prohlížeče [9]. V současné době (prosinec 2011) je ke stažení již pátá verze tohoto oblíbeného produktu. Jedná se o platformu určenou pro tvorbu dynamického online obsahu a interaktivní práci s ním. Tato technologie kombinuje text, vektorovou i bitmapovou grafiku, animace a video.

Po technické stránce jde o tzv. softwarový doplněk (plug-in), který se instaluje jako rozšíření do internetového prohlížeče. Silverlight funguje na platformách Windows a Mac OS X. Na Linuxu je dostupný pod názvem Moonlight. Podporuje nejznámější internetové prohlížeče jako jsou Internet Explorer, Mozilla FireFox, Safari, Opera a Google Chrome. Po jednoduché instalaci se Silverlight plně integruje do internetového prohlížeče. Jednou nainstalovaný plug-in Silverlightu nevyžaduje již žádnou další obsluhu, neboť zde dochází k automatickým aktualizacím. Z pohledu uživatele je tak úplně transparentní a ještě k tomu úplně bezplatný.

Tato technologie v sobě obsahuje zredukovanou verzi běhového prostředí (CLR) jako tomu je u .NET Frameworku [Kapitola 2.4.1]. V mnoha případech tak lze použít kód původně psaný pro obyčejný .Net i v Silverlight aplikaci. Aplikace

napsané v Silverlightu mohou volat webové služby typu ASP.NET, nebo WCF. Jelikož technologie Silverlight vychází z WPF [Kapitola 2.4.1.2], je zde použit i stejný jazyk XAML pro definici uživatelských rozhraní. Opět je tedy možné využít zdrojový kód původně pro WPF i u webové Silverlight aplikace. Z toho plyne, že i v Silverlightu lze vytvářet uživatelsky bohaté aplikace a využívat tak možností vektorové grafiky, animací, multimédií, efektů, provázání dat a mnoho dalšího.



Obrázek 2-2: Příklad Silverlight aplikace

2.3.3 *Oracle databáze*

Oracle databáze je nejrozšířenějším typem relační databáze. V současnosti (prosinec 2011) její aktuální verze Oracle Database 11g [10] nabízí nepřetržitou dostupnost a možnost rozšiřitelnosti díky technologii Oracle Real Application Clusters. Díky této technologii tak dochází k minimalizaci nákladů způsobenou výpadky systému. Pomocí nástrojů pro automatizovanou správu se snižují náklady na administraci a dochází tak k poskytování nejvyšší možné úrovně služeb. Celá databáze poskytuje širokou škálu bezpečnostních mechanismů. Tyto mechanismy tak databázi předurčují pro použití v prostředích s velkými nároky na bezpečnost dat. Široké možnosti komprese dat jednak zvyšují

výkon databáze, ale i šetří náklady na diskový prostor ukládáním dat podle jejich četnosti využívání.

Oracle databáze existuje ve čtyřech základních edicích. Každá edice je vhodná pro prostředí s jinými nároky. Následující tabulka [Tabulka 2-1] popisuje a porovnává jednotlivé edice. Z tabulky vyplývá, že Express Edition je spíše vhodná pro testovací a vzdělávací účely.

	Express Edition	Standart Edition One	Standart Edition	Enterprise Edition
Maximum Procesorů	1	2	4	bez limitu
Velikost RAM	1GB	OS max	OS max	OS max
Velikost databáze	4GB	bez limitu	bez limitu	bez limitu
Platforma	Windows, Linux	Windows, Linux, Unix	Windows, Linux, Unix	Windows, Linux, Unix
64bit podpora	ne	ano	ano	ano

Tabulka 2-1: Srovnání edic oracle databáze

Pro usnadnění vývoje databázových aplikací podporuje Oracle následující jazyky:

- SQL/XML – rozšíření standartu SQL 2003 o podporu XML
- PL/SQL – k jazyku SQL přidává konstrukci procedurálního programování
- .NET – výrazná podpora pro vývoj databázových aplikací pod .NET technologií
- XML DB – podpora vysoce výkonných úložných a vyhledávacích technologií zaměřených na XML
- PHP – podpora skriptovacího jazyka pro vývoj webových aplikací

3 Analýza

V této kapitole analyzujeme požadavky na workflow proces, uživatelské role, rozdělíme systém na jednotlivé aplikace, provedeme rozbor požadavků systému a porovnáme existující systémy pro návrh a řešení business procesů.

3.1 *Požadavky na workflow proces*

Workflow proces pro práci s neplatiči je identifikován názvem, jménem autora a svoji verzí. Celé řešení workflow procesu je časově omezeno. Konfigurace workflow je dána kombinací automatických, manuálních úloh a vazeb mezi nimi. Každá úloha má svůj jedinečný název a výčet možných řešení. Vazby mezi jednotlivými úlohami můžou mít definované určité časové zpoždění.

Při řešení workflow procesů pro neplatiče mýta, lze s výhodou použít upomínací techniky pro vymáhání dluhu [11]. Nečastější upomínání nebo informování dlužníků probíhá za pomoci emailů. Další možností je pak telefonní hovor, nebo obyčejný dopis. U obou těchto možností je však potřeba nějaká lidská podpora (operátor). V některých případech po uplynutí určité doby dochází k placení jistého administračního poplatku a tím tak k navýšení celého dluhu. V krajních případech může proces vymáhání dlužné částky přejít k procesu exekuce. Vzhledem k těmto poznatkům lze definovat základní automatické a manuální úlohy, ze kterých se workflow proces může skládat.

Mezi automatické úlohy a jejich možné stavy řešení může patřit:

- pošli upomínací email (Posláno)
- navyš dlužnou částku (Navýšeno)

Mezi manuální úlohy a jejich možné stavy řešení může patřit:

- telefonuj dlužníkovi a upomínej ho (Provedeno / Dlužník neodpovídá)
- potvrď exekuční řízení (Ano / Ne)

3.2 *Uživatelské role*

V informačním systému pro řešení procesu pro práci s neplatiči můžeme nalézt následující uživatelské role.

3.2.1 *Administrátor*

Administrátor je osoba s nejvyššími právy a přistupuje tak do všech částí systému. Má tedy stejná oprávnění jako operátor a manažer workflow. Spravuje všechny uživatele a neplatiče mýtných poplatků. Správa neplatičů je zde pouze z testovacích důvodů. V reálném systému se o to administrátor starat nebude.

3.2.2 *Operátor*

Tato osoba řeší manuální úlohy workflow procesů. Při řešení manuálních úloh komunikuje s dlužníkem a nahlíží ke všem jeho důležitým údajům, jako jsou výše dluhu, čas vzniku atd. Vidí všechny doposud provedené automatické a manuální úlohy dlužníkova workflow procesu a může o tom dlužníka informovat.

3.2.3 *Manažer workflow*

Osoba navrhující a spravující workflow diagramy. Navrhuje workflow diagramy z definovaných automatických a manuálních úloh v systému. Kontroluje správnost navržených diagramů. Nasazuje a odstraňuje nepoužívané diagramy ze systému.

3.3 *Rozdělení systému*

Celý systém pro řešení procesu pro práci s neplatiči se skládá z několika aplikací. Všechny tyto aplikace jsou typu klient-server [Kapitola 2.1.2]. Klientem zde bude vždy webová aplikace [Kapitola 2.1.3]. Jako server bude použita back-end aplikace [Kapitola 2.1.4] řešící proces neplatiče mýtného a nabízející rozhraní všem klientským webovým aplikacím.

V systému nalezneme následující aplikace (stejné pojmenování bylo použito i pro jednotlivé projekty v implementaci):

- **Dunning application server**

  Upomínací aplikační server je back-end aplikace řešící proces neplatiče mýtného poplatku. Tato aplikace zároveň poskytuje rozhraní pro klientské webové aplikace.

- **User manager**

  Správce uživatelů spravuje všechny uživatele v systému. Součástí této aplikace je i správa dlužníků, která je zde pouze pro testovací účely. Vytváření dlužníků bude v reálném systému zcela automatické a tato část aplikace tedy nebude potřeba. Řešení automatického vytváření dlužníků není však už tématem této práce.

- **Task manager**

  Správce manuálních úloh distribuuje manuální úlohy workflow procesů mezi operátory.

- **Workflow manager**

  Správce workflow slouží pro návrh a správu workflow diagramů.

User manager, Task manager i Workflow manager jsou webové aplikace komunikující s Dunning application serverem.

3.4 *Požadavky na systém*

Následující podkapitoly popisují požadavky na jednotlivé aplikace v systému. Pro popis funkčních požadavků u klientských webových aplikací jsou použity UML diagramy případů užití.

3.4.1 *Dunning application server*

Upomínací aplikační server je back-end aplikace řešící proces neplatiče mýtného poplatku. Pro tuto aplikaci lze specifikovat následující funkční a nefunkční požadavky.

Mezi funkční požadavky patří:

- **Správa upomínacího aplikačního serveru**

  Správa serveru zajišťuje samotné spouštění a vypínání aplikačního serveru.

- **Zajištění logiky upomínání**

  Logika obsahuje kompletní seznam automatických a manuálních úloh. Jednotlivé úlohy definuje a v případě automatických úloh zajišťuje jejich kompletní vykonání v rámci workflow procesu. V případě manuálních úloh logika zajišťuje šíření důležitých informací o úloze a dlužníkovi, mezi operátory.

- **Řešení workflow procesu neplatiče mýtného**

  Řešení workflow procesu je založeno na vykonávání automatických a manuálních úloh. Spolu s vykonáváním manuálních úloh souvisí pozastavování a znovuspuštění workflow procesu. Workflow proces je pozastavován z důvodu čekání na vyřešení manuální úlohy operátorem. Dále může být celé řešení ukončeno z důvodu zaplacení dluhu, nebo může být úplně zrušeno z důvodu vypršení časového omezení na vyřešení. Jelikož je schéma workflow procesu deterministické, známe tedy vždy bezprostředního následníka aktuální úlohy. Tato vlastnost nám tedy zajistí postupné řešení celého workflow procesu.

- **Odesílání emailů**

  Upomínání probíhá pomocí odesílání emailových zpráv na dlužníkův účet. V těle zprávy se uvádí výše dluhu, datum vzniku a případně další související informace. Jako předmět zprávy je uveden důvod pro zaslání upomínky.

- **Logování chyb**

  Pro snadnější ladění a testování zavádíme logování vyskytujících se chyb při běhu aplikace. Chyby zapisujeme do textových souborů, kam uvádíme místo výskytu a druh chyby.

- **Vytváření statistik**

  Z důvodu případného zkvalitňování služeb nabízí aplikace vytváření statistik provedených úkonů. Aplikace vytváří statistiky o počtu celkově vyřešených manuálních úloh operátory, o počtu dlužníků, o počtu odeslaných upomínek.

- **Zajištění rozhraní k datovému úložišti**

  Rozhraní k datovému úložišti umožňuje vybírat a vkládat příslušná data z/do oracle databáze. K rozhraní existuje v rámci této aplikace globální přístup.

- **Zajištění rozhraní pro klientské aplikace**

  Toto rozhraní zajišťuje klientským webovým aplikacím User Manager, Task Manager a Workflow Manager komunikaci se samotným upomínacím aplikačním serverem.

Mezi nefunkční požadavky patří:

- **Konfigurovatelnost**

  Aplikaci lze konfigurovat za pomocí konfiguračního souboru. Konfigurujeme zde zejména rozhraní k datovému úložišti, rozhraní pro klientské aplikace, odchozí emailové účty, vytváření statistik a logování chyb.

- **Bezpečnost**

  Tím, že budou klientské webové aplikace přístupné z globální sítě internet a vzhledem k charakteru informací vyskytujících se v aplikacích, bude celý systém komunikovat s okolím pomocí zabezpečeného protokolu. Každý uživatel bude před vstupem do klientských aplikací ověřen uživatelským jménem a heslem.

- **Modularita**

  Modularita zajišťuje snadnou rozšiřitelnost systému. Z tohoto důvodu se snažíme aplikaci navrhnout tak, aby byla tvořena několika nezávislými moduly. Tyto moduly pak budeme snadno v případě změn nahrazovat.

- **Snadné nasazení**

  Instalace aplikace bude jednoduchá bez dodatečného programování. Vše potřebné bude konfigurovatelné viz *konfigurovatelnost*. Operační systém nebude vyžadovat netradiční nastavení.

3.4.2 *User manager*

Správce uživatelů je webová aplikace jejíž součástí je i správa dlužníků, viz rozdělení systému [Kapitola 3.2]. Do aplikace bude umožněn přístup pouze registrovaným uživatelům s administrátorským oprávněním. Funkční požadavky této aplikace popisují následující odstavce a [Obrázek 3-1].

- **Logování**

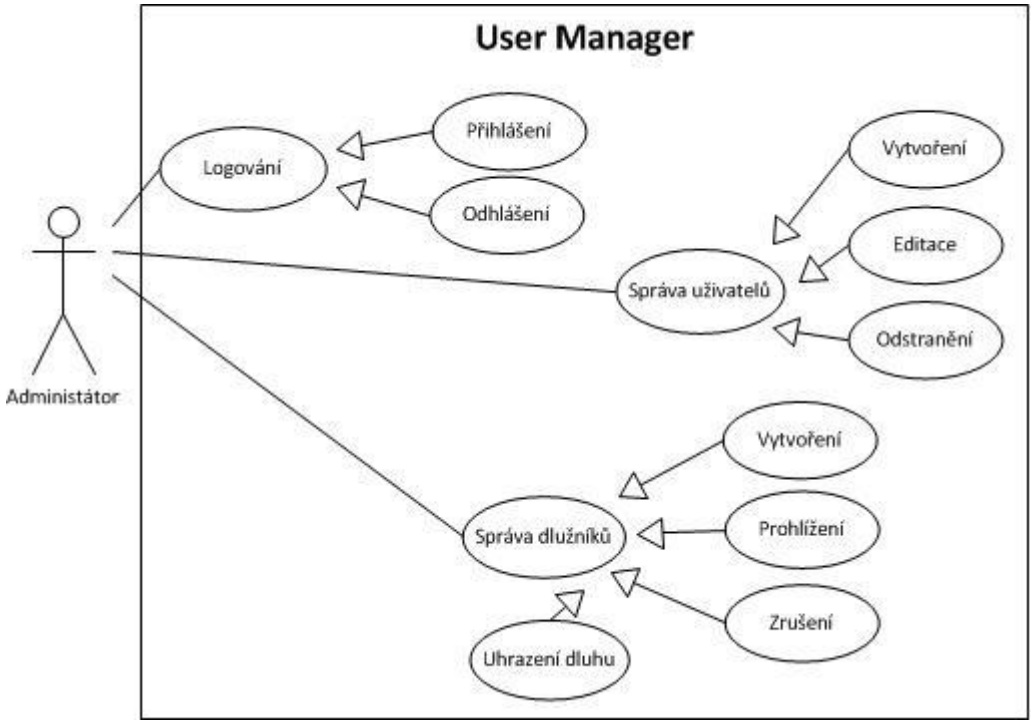
  Požadavek logování v aplikaci zahrnuje přihlašování a odhlašování uživatelů. Jméno přihlášeného uživatele je v aplikaci viditelné.

- **Správa uživatelů**

  Správa uživatelů zahrnuje vytváření, editaci a mazaní uživatelů v systému. Každý uživatel má své přihlašovací jméno, přihlašovací heslo, jméno, příjmení, email a přidělenou uživatelskou roli.

- **Správa dlužníků**

  Správa dlužníků obsahuje funkce pro vytváření, prohlížení a rušení dlužníků. Při vytváření dlužníka jsou potřeba údaje jako jméno, příjmení, email, telefon, adresa, výše dluhu a vybraný workflow proces pro vymáhání dluhu. Jeden workflow proces lze přiřadit více dlužníkům. Spolu s vytvořením dlužníka dojde ke spuštění vymáhacího procesu podle zvoleného workflow. Vytvořený dluh lze kdykoliv v průběhu vymáhacího procesu uhradit. U každého dlužníka můžeme prohlížet jeho identifikační údaje (jméno, příjmení, adresa, atd.), stav běžícího workflow procesu a samotný diagram workflow procesu použitého pro vymáhání.



Obrázek 3-1: Diagram případu užití aplikace User Manager

3.4.3 *Task manager*

Správce manuálních úloh je webová aplikace do které mají přístup zejména operátoři, ale přístup je povolen i uživatelům v roli administrátora. Operátoři řeší manuální úlohy právě probíhajících workflow procesů. Následující odstavce a [Obrázek 3-2] popisují funkční požadavky této aplikace.

- **<u>Logování</u>**

  Zde platí stejné požadavky jako pro logování do aplikace User Manager [Kapitola 3.3.2].

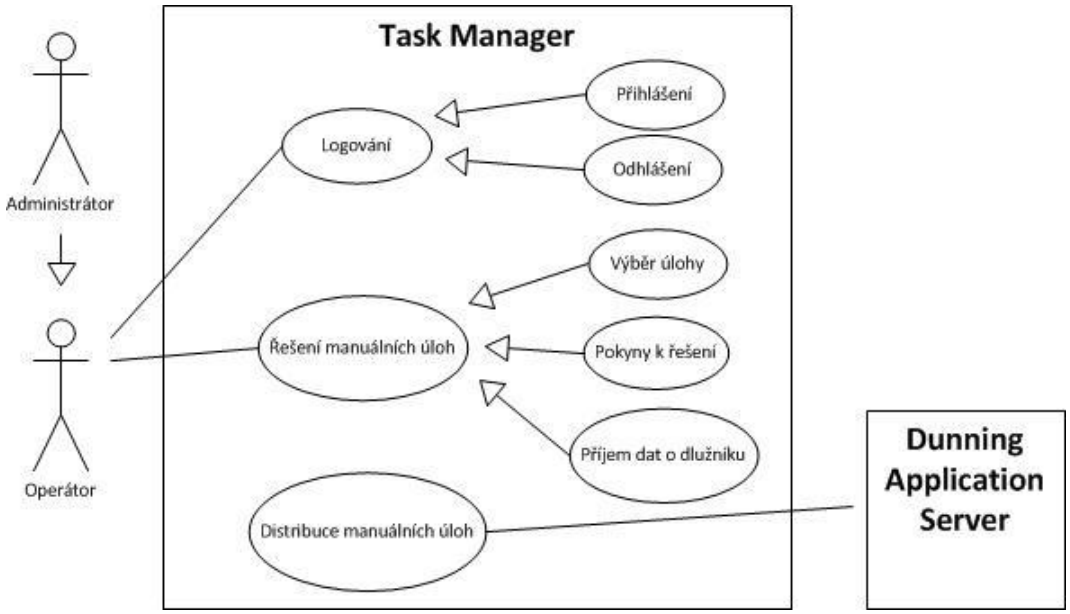
- **<u>Řešení manuálních úloh</u>**

  Řešení manuálních úloh zahrnuje výběr úlohy, pokyny k řešení a příjem informací o dlužníku. Operátor při řešení manuální úlohy vidí pokyny k řešení úlohy a příslušná potřebná data o samotném dlužníkovi. Ke každé řešené úloze lze zapsat doplňující poznámku. Každou manuální úlohu

může řešit pouze jeden operátor. Pokud operátor vybranou úlohu nevyřeší, musí být zajištěna její opětovná distribuce.

- **Distribuce manuálních úloh**

  Šíření manuálních úloh v této aplikaci zajišťuje upomínací aplikační server. Šíření úloh je tedy zcela automatické.



Obrázek 3-2: Diagram případu užití aplikace Task Manager

3.4.4 *Workflow manager*

Tato webová aplikace slouží pro tvorbu a správu workflow diagramů. Přístup do aplikace je povolen pouze uživatelům v rolích manažer workflow a administrátor. Funkční požadavky popisují následující odstavce. Tyto požadavky jsou zároveň popsány formou diagramu na [Obrázek 3-3].

- **Logování**

  Zde platí stejné požadavky jako pro logování do aplikace User Manager [Kapitola 3.3.2].

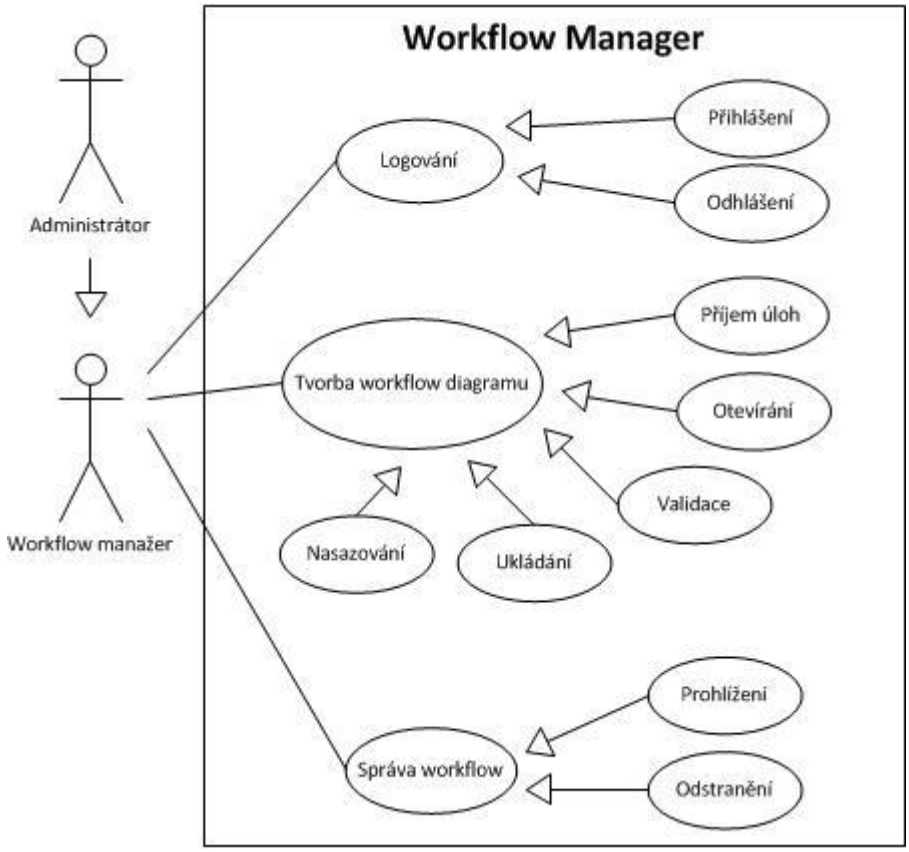
- **Tvorba workflow diagramu**

  Tvorba workflow diagramu zahrnuje příjem automatických a manuálních úloh, otevírání existujících, validaci, ukládání a nasazování workflow diagramů. Pro tvorbu workflow diagramu potřebujeme jeho základní stavební bloky, což jsou automatické a manuální úlohy. Bez těchto úloh nelze diagram sestavit. Mezi úlohami můžeme vytvářet vazby, které tak jednotlivé úlohy spojují. Vazby mohou obsahovat určité časové zpoždění. Dále můžeme při tvorbě diagramu definovat jeho vlastnosti jako název, autora, verzi a časovou platnost. Před samotným nasazením nebo uložením diagramu je jej potřeba validovat. Správně navržené diagramy mají vazbami provázány všechny úlohy a definované všechny vlastnosti. Nasazení znamená, že je možné diagram využívat při vymáhání dluhu

  u neplatičů mýta. Ukládáním umožníme diagramy využívat jako předlohy pro další verze. S ukládáním potřebujeme existující diagramy otevírat.

- **Správa workflow**

  Správa workflow obsahuje funkce pro prohlížení nasazených a odstraňování již nepoužívaných workflow diagramů. Workflow diagram, který je systémem právě používán k vymáhání dluhu, nelze odstranit.



Obrázek 3-3: Diagram případu užití aplikace Workflow Manager

3.5 *Existující řešení*

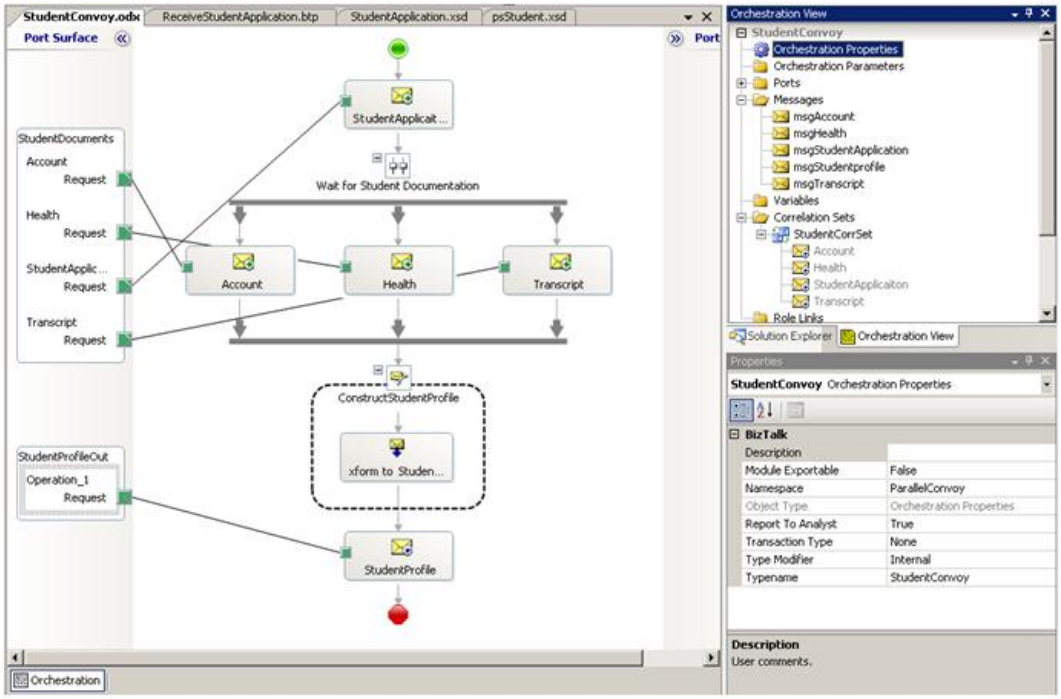
Pro popis workflow procesů a jejich zpracování existuje celá řada již existujících řešení. V této podkapitole rozebereme několik z nich. Nakonec tyto řešení porovnáme a vyhodnotíme jejich možnosti použití vzhledem k požadavkům našeho systému.

3.5.1 *Microsoft BizTalk Server*

Jedná se o komerční produkt od společnosti Microsoft [12], jehož cílem je snadná integrace interních podnikových aplikací, bezpečné spojení s obchodními partnery přes internet a automatizace komplexních business procesů realizovaných organizacemi. V případě business procesů nabízí účastníkům jejich zpřístupnění pomocí více kanálů (např. pomocí webové aplikace, nebo call centra).

Pro úspěšnou realizaci instalace potřebujeme několik důležitých komponent. Mezi tyto komponenty patří MS Windows Server, MS SQL Server a MS Visio. Pro spojení s ostatními aplikacemi lze využít webových služeb nebo WCF [Kapitola 2.3.1.1].

Přínos tohoto produktu je zejména v podpoře návrhu, vytváření a provádění dynamických business procesů. Pro návrh a implementaci business procesů používá vlastní nástroj *BizTalk Orchestration Designer* [Obrázek 3-4], který je založen na aplikaci Microsoft Visio. Systém BizTalk využívá celkem čtyři druhy databáze. Jedna z databází je *Orchestration Persistence Database*. Do této databáze jsou ukládány stavy o běžících business procesech. Pro sledování stavu těchto procesů slouží nástroj BizTalk Tracing. Tento nástroj je důležitý zejména při řešení nenadálých problémů.



Obrázek 3-4: BizTalk Orchestration Designer

Dalšími přínosy tohoto produktu jsou bezpečnost, spolehlivost poskytovaných služeb a škálovatelnost. Škálovat systém (zvyšovat výkon) lze

dvěma způsoby. První způsob spočívá v posilování hardwaru nebo hardwarové konfigurace. Druhý způsob je založen na rozkládání zátěže na více serverů. Tento druhý způsob je pro optimalizaci BizTalk Serveru nejvíce používán.

Tento produkt dále nabízí nástroje pro vytváření různých statistik, reportů a samotnou správu MS BizTalk Serveru.

Výhody

- hlavní release každých 18 měsíců
- žádné proprietární metody vzdáleného přístupu

Nevýhody

- podporuje pouze MS SQL
- nekompabilita s UML

3.5.2 *K2 blackpearl*

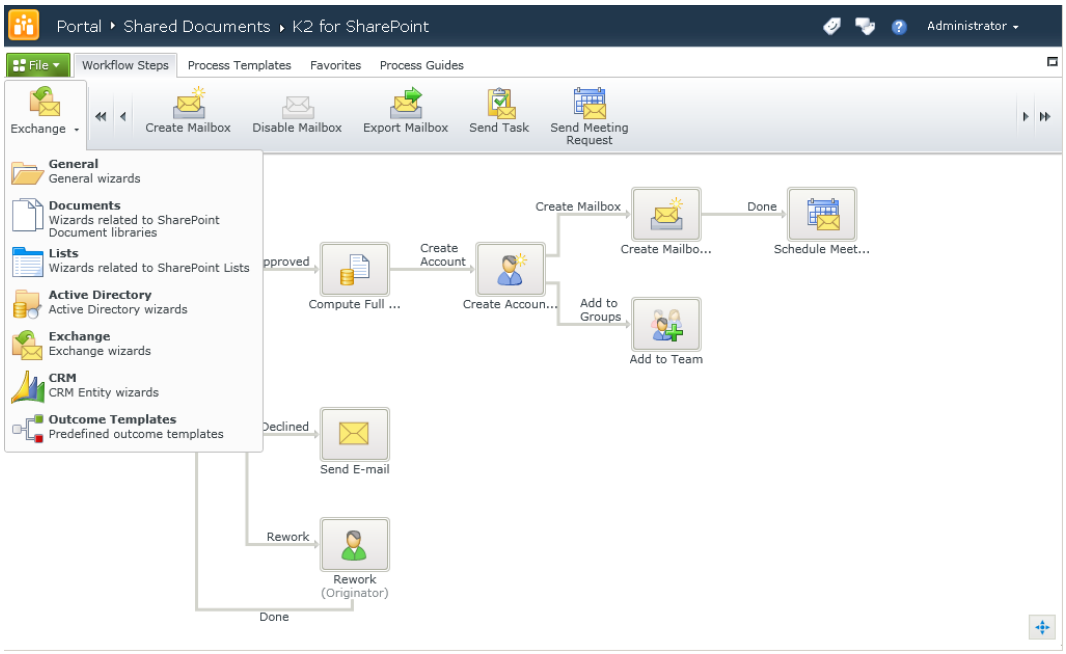
Produkty K2 pocházejí z dílny jihoafrické společnosti SourceCode, kam patří i K2 blackpearl [13]. Tento komerční produkt je dalším zástupcem kompletního BPM workflow systému. Jedná se tedy o komplexní řešení pro návrh, běh a správu business procesů.

Na pozadí celého systému běží Window Workflow Foundation [Kapitola 3.5.3]. K2 blackpearl používá MSSQL databázi, kam ukládá stavy běžících business procesů. Pro integraci s externími systémy lze opět využít možností webových služeb nebo WCF [Kapitola 2.3.1.1].

Business procesy lze modelovat přes webové rozhraní [Obrázek 3-5], v prostředí MS Visual Studia nebo MS Visia. Další možností je aplikace K2 Studio. Pokud nebudeme potřebovat z procesů volat externí kód nebo webové služby, lze návrh provést i s Web Designerem přímo v SharePointu. Pro správu business

procesů si můžeme na příslušném místě ze šablon vytvořit samotný správcovský web. V tomto webu pak upravujeme parametry samotného systému K2 blackpearl, spravujeme definice procesů a jejich běžící instance. Dále tento produkt podporuje vytváření široké škály reportů. Některé základní reporty jsou již v systému obsaženy od základu a vlastní reporty lze vytvářet za pomoci SQL Reporting Services.

K2 blackpearl je v dnešní době asi nejlepším řešením v prostředí MS Windows. K systému je dodávána rozsáhlá dokumentace a mnoho tutoriálů. Zároveň nabízí pokročilé možnosti reportování, ale především spolehlivý, výkonný a škálovatelný workflow engine.



Obrázek 3-5: Webové rozhraní K2 blackpearl pro návrh business procesů

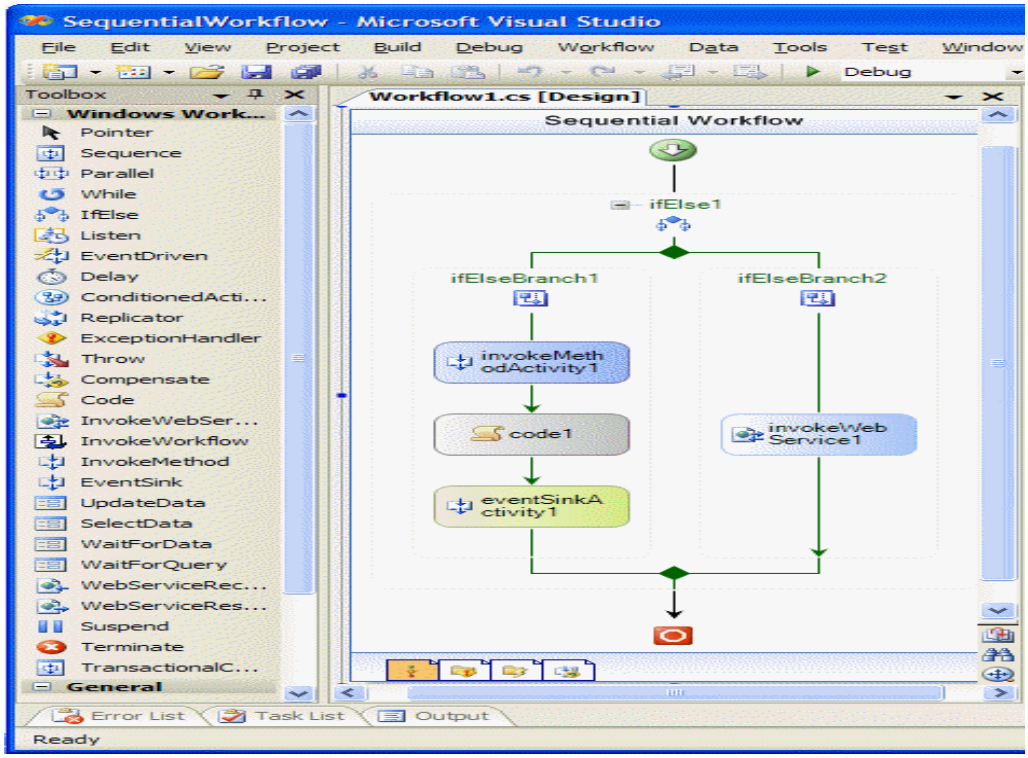
3.5.3 *Workflow Foundation*

Workflow Foundation je programový model, stroj a z velké části nástroj pro tvorbu workflow aplikací [14]. Je jednou ze základních součástí Microsoft .NET Frameworku již od verze 3.0.

Od předešlých produktů se liší hlavně tím, že nejde o komplexní BPM workflow systém. Nenabízí tedy spoustu nástrojů pro reporting a správu business procesů a nelze z něj tedy okamžitě postavit nějaký informační systém na řízení obchodních činností. Jedná se tedy spíše o čistý programový model s několika nástroji.

Mezi komponenty Workflow Foundation frameworku patří základní knihovna úloh (stavebních bloků workflow), běhové jádro, běhové služby, hostující proces a workflow návrhář. Běhové služby zahrnují služby jako sledování, komunikaci přes webové služby, WCF, řízenou Eventy apod. Hostujícím procesem je myšlena aplikace, která hostuje běhové jádro pro vykonávání business procesů. Workflow návrhář [Obrázek 3-6] je součástí MS Visual Studia a business procesy lze tedy navrhovat v něm. Procesy můžeme navrhovat jako sekvenční workflow nebo také jako stavové stroje. Každý druh lze obecně využít při různé činnosti. Workflow Foundation nepotřebuje pro svůj běh žádnou databázi, neboť všechny workflow jsou spouštěny v runtimu. Naproti tomu, oba dva předešlé produkty spouštějí workflow v procesu.

V případě tohoto produktu tedy nelze okamžitě spravovat běžící business procesy, využívat reporty a škálovat celý systém. Tyto funkce, lze ale za pomoci do tohoto frameworku implementovat.



Obrázek 3-6: Návrhář business procesů v Microsoft Visual Studiu

3.5.4 *Porovnání řešení*

V případě BizTalk Serveru a K2 blackpearlu se jedná o komerční produkty, které lze pořídit za nemalý obnos peněz. Jejich rozsáhlost, kvalita zpracování, spolehlivost a výkonnost však těmto cenám odpovídá. Jedná se tedy spíše o produkty, jejichž řešení využijí střední a velké organizace pro řízení svých business procesů. V našem případě takto rozsáhlý systém nepotřebujeme. Zbytečně by se zvyšovali nároky na jeho provoz. Dalším důvodem je i vysoká cena.

Samotné Workflow Foundation je řešení, kterého by se dalo využít. Jeho cena je nulová, neboť je součástí .NET Frameworku, je dostupné, dobře dokumentované a neustále se rozvijí. Srovnání všech třech existujících řešení najdeme v [Tabulka 3-1]. Nevýhodou je však absence návrháře business procesů, který je pouze součástí MS Visual Studia, což je velmi nešikovné. Dalším problémem může být samotný výkon tohoto frameworku. Jedná se především

o framework pro řešení obecných business procesů, který může být v některých případech pomalejší než vlastní proprietární řešení systému. Dalším důvodem je i to, že workflow spuštěné pomocí Workflow Foundation jsou spuštěné v runtimu. To znamená, že pokud uzavřeme hostující proces přijdeme o všechny právě běžící workflow. Z těchto důvodů jsem se rozhodl pro návrh a implementaci vlastního řešení.

	BizTalk Server	K2 blackpearl	MS Workflow Foundation
Licence	komerční	komerční	součást .NET Frameworku
Nutnost databáze	ano	ano	ne
Škálovatelnost	ano	ano	ne
Návrhář workflow	BizTalk Orchestration Designer, MS Visio	MS Visual Studio, MS Visio, K2 studio, webové rozhraní	MS Visual Studio
Správa a reporty	ano	ano	ne

Tabulka 3-1: Srovnání existujících produktů pro řešení business procesů

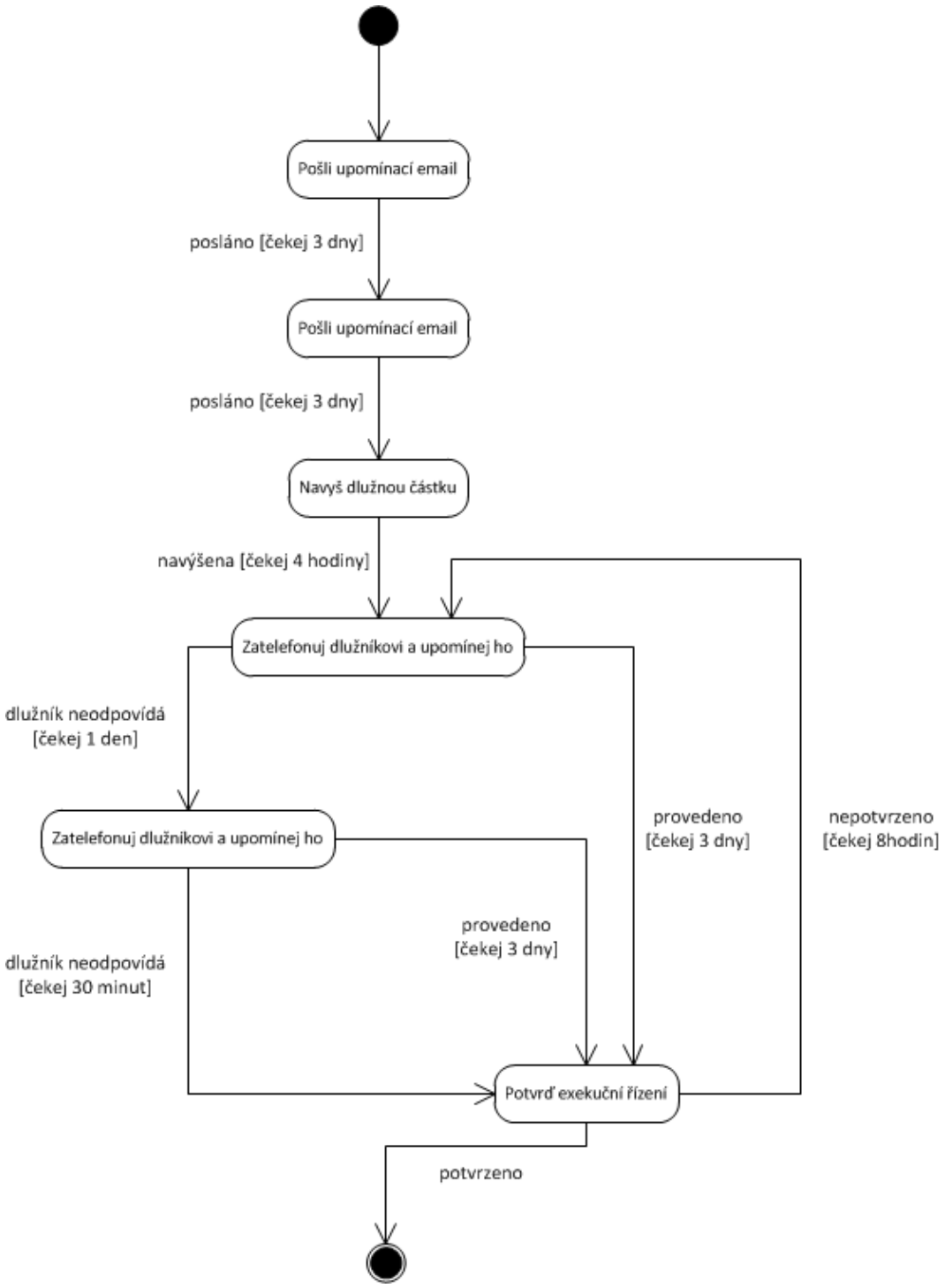
4 Návrh

Z analýzy jsme získali uživatelské role a rozdělení systému na jednotlivé aplikace. K těmto aplikacím máme zároveň specifikované jejich požadavky. Z existujících řešení jsme získali určitý přehled o možnostech vykonávání obecných workflow procesů. V této kapitole navrhneme business proces pro práci s neplatiči, strukturu celého systému, samotné aplikace a jejich důležité funkce prostřednictvím aktivity diagramů. Nakonec navrhneme databázový model.

4.1 *Workflow proces pro práci s neplatiči*

Z analýzy workflow procesu pro práci s neplatiči [Kapitola 3.1] jsme získali možné automatické a manuální úlohy. Pokud tyto úlohy vhodně provážeme přechody s určitým časovým zpožděním, získáme návrh workflow procesu [Obrázek 4-1].

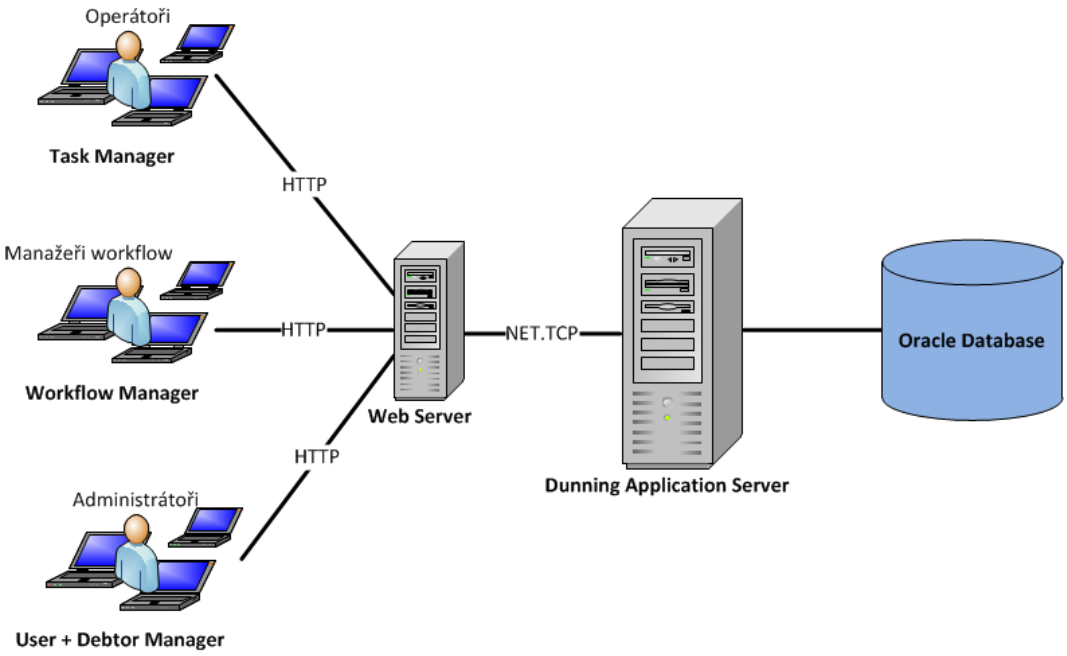
Při spuštění procesu jsou poslány dva upomínací emaily v rozmezí tří dnů. Pokud do této doby dlužník svůj dluh neuhradí, dochází k jeho navýšení. Po uplynutí čtyř hodin od navýšení dluhu je vytvořen úkol pro operátora, jehož zadáním je zatelefonovat dlužníkovi a upozornit ho, že má neuhrazené mýtné poplatky. Pokud se operátor vůbec nedovolá, je tato úloha opakována s jednodenním zpožděním. Jestliže ani po jednom z telefonátů, dlužník svůj dluh do určité doby neuhradí, je vytvořen další úkol pro operátora. Tento úkol spočívá v potvrzení nebo nepotvrzení exekučního řízení. Operátor tedy opět telefonuje dlužníkovi a v případě jeho nedostupnosti exekuční řízení potvrzuje. V případě kladné odezvy, operátor exekuční řízení nepotvrzuje a upomínání dlužníka se s osmi hodinovým zpožděním opakuje od úkolu prvního telefonátu. Po několika opakováních těchto smyček, operátor exekuční řízení potvrzuje i v případě dlužníkova slibu o splacení dluhu.



Obrázek 4-1: Diagram workflow procesu pro práci s neplatiči

4.2 *Struktura systému*

Pro návrh celého systému je důležité popsat propojení jednotlivých klientských aplikací, upomínacího aplikačního serveru a databáze. Propojení aplikací lze vidět na [Obrázek 4-2].



Obrázek 4-2: Diagram propojení aplikací a databáze

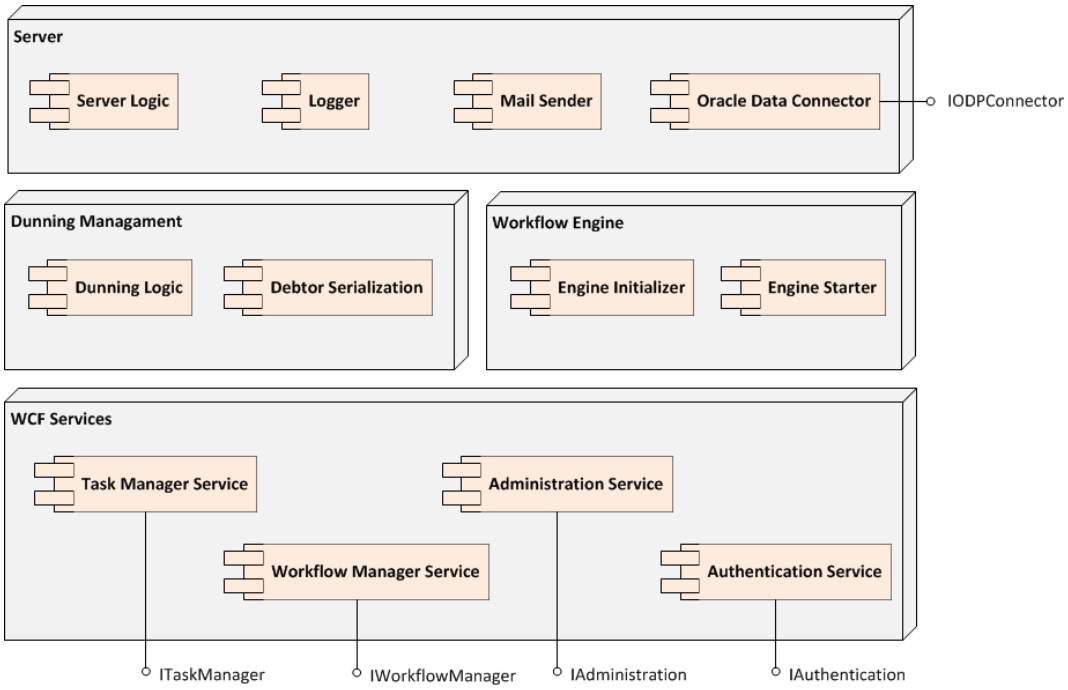
Klientské aplikace TaskManager, WorkflowManager UserManager jsou webové aplikace. Tyto aplikace běží v oknech internetových prohlížečů. Jejich spustitelný kód je nahraný na příslušném webovém serveru. Komunikace mezi internetovým prohlížečem a webovým serverem probíhá za pomoci HTTP požadavků. Na webovém serveru jsou umístěny příslušné webové služby pomocí, kterých je umožněna klientským aplikacím komunikace s upomínacím aplikačním serverem. Komunikace je založena na technologii WCF, konkrétně na protokolu NET.TCP. Upomínací aplikační server tedy musí nabízet tři různá rozhraní pro klientské aplikace a jedno společné rozhraní pro autorizaci uživatelů. Tento server dále zajišťuje komunikaci s oracle databází, která je nainstalovaná na datovém serveru.

Návrh architektury systému může být i jiný. Webový server se může např. do databáze připojovat přímo a nemusí k tomu využívat Dunning Application Server. Pokud bychom vzali v úvahu tuto architekturu, měli bychom do databáze dvě různá rozhraní, protože Dunning Application Server svůj přístup mít musí. Tímto by došlo k navýšení časové náročnosti implementace a k dalšímu možnému výskytu chych. Z tohoto důvodu jsme radši zůstali u původního návrhu.

V následujících kapitolách navrhneme a popíšeme strukturu jednotlivých entit z [Obrázek 4-2].

4.3 *Dunning application server*

Začněme upomínacím aplikačním serverem, neboť jde o nejdůležitější a nejsložitější součást celého systému. Celý server obsahuje několik komponent, které popisuje následující obrázek.



Obrázek 4-3: Diagram komponent pro Dunning Application Server

4.3.1 *Server logic*

Tako komponenta je vstupním bodem celé aplikace. Obsahuje konfigurační xml soubor, kterým lze nastavit důležité atributy celého serveru. V tomto souboru máme možnost zapínat nebo vypínat logování chyb a vytváření statistik. Definujeme zde přístupové údaje do oracle databáze, intervaly pravidelnosti spouštění workflow logiky, vlastnosti WCF služeb a emailové adresy, ze kterých jsou odesílány upomínky dlužníkům.

Dále je zde umístěn zavaděč celého serveru, který tvoří konzolová aplikace. Tato aplikace umí server spustit v konzoly nebo pomocí příslušného přepínače jako službu windows. Instalátor služby je součástí této komponenty. Spuštění celého serveru zajišťuje startovací metoda komponenty Engine starter [Kapitola 4.2.7].

4.3.2 *Logger*

Logger nabízí podporu pro logování chyb a vytváření statistik. Logovat chybové stavy lze kdekoliv v rámci celého upomínacího aplikačního serveru. Zápis chyb je směřován do čtyř textových souborů, podle toho v jaké části aplikace k chybě došlo. Mezi tyto čtyři soubory patří WorlfowEngineLog, ODPConnectorLog, MailSenderLog a DunningManagamentLog. Do souborů nejsou zapisovány pouze chybové stavy, ale i informace o tom v jakých stavech se daná část systému právě nachází. Logovací soubory jsou k nahlédnutí po spuštění serveru a při jeho vypnutí je jejich obsah smazán.

Vytvářet statistiky můžeme z počtu odeslaných upomínek prostřednictvím emailu, počtu právě běžících workflow procesů a z počtu doposud nevyřešených manuálních úloh operátory. Pro zobrazování výsledků využijeme aplikaci Performance Monitor, která je součástí operačního systému Windows 7. Z důvodu snadného přístupu k výsledkům je důležité v této aplikaci vytvořit

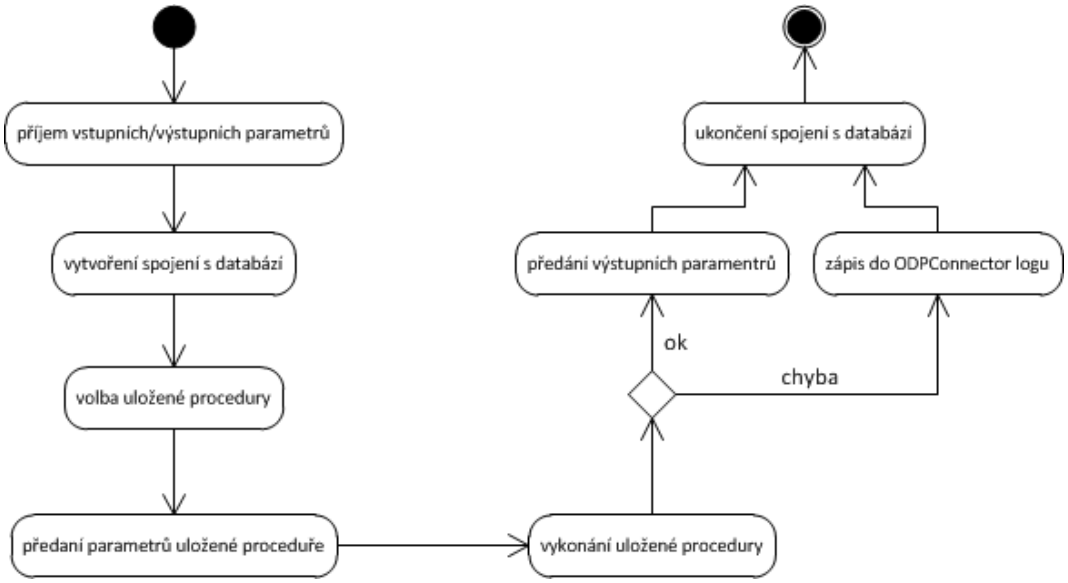
pro všechny druhy statistik jednu společnou kategorii s názvem např. DunningAppServer Category.

4.3.3 *Mail sender*

Komponenta Mail sender zajišťuje asynchronní odesílání emailových zpráv. Asynchronní odesílání je zde zvoleno záměrně, neboť při synchronním odesílání by docházelo, ke zbytečným prostojům při zpracování workflow procesu. Mail sender potřebuje jako vstupní parametry adresu odesílatele, příjemce, předmět a samotné tělo zprávy.

4.3.4 *Oracle data connector*

Oracle data connector nabízí rozhraní *IODPConnector* mezi oracle databází a upomínacím aplikačním serverem. Umožňuje tedy vytvořit spojení, vybrat nebo uložit data a spojení opět uzavřít. Součástí oracle databáze jsou uložené procedury viz [Kapitola 4.5.3], a proto metody této komponenty používají přímo je a nevytvářejí tak SQL dotazy. Následují aktivity diagram [Obrázek 4-4] ukazuje strukturu jedné metody této komponenty.



Obrázek 4-4: Aktivity diagram pro volání uložené procedury

Z diagramu je vidět, že pokud dojde při vykonávání uložené procedury k chybě dozvíme se o tom v příslušném logovacím souboru.

4.3.5 *Dunning logic*

Dunning logic definuje automatické a manuální úlohy, ze kterých se může skládat workflow proces neplatiče mýtného. Pro tyto úlohy zároveň zajišťuje jejich logiku vykonávání. Manuální úlohy rozděluje na počáteční a koncové. Automatické a počáteční manuální úlohy jsou volány automaticky při běhu workflow procesu. Počáteční manuální úlohy zajišťují typ formuláře a dlužníkova data. Koncové manuální úlohy jsou volány při vyřešení úkolu operátorem. Dále jsou zde definovány množiny řešení pro každou úlohu. Výčet úloh byl specifikován v analýze [Kapitola 3.1].

V souvislosti s vykonáváním úloh zde můžeme přistupovat k datům o dlužníku a některé z nich modifikovat. Například máme možnost dluh smazat nebo k němu přidat další komentář.

4.3.6 *Debtor serialization*

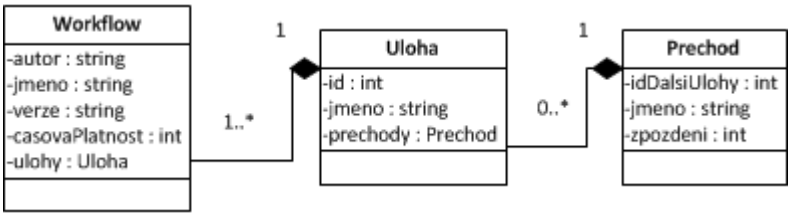
V průběhu řešení procesu dlužníka dochází ke změnám některých jeho dat viz [Kapitola 4.2.5]. Každý běžící proces si tyto data musí pamatovat a tudíž budou ukládány do databáze. Ukládat data jednotlivě by však nebylo dobré řešení, a proto je potřeba data serializovat. Spolu se serializací je nutné umět data i deserializovat. Pro serializaci, případně deserializaci dlužníkových dat, slouží právě tato komponenta.

4.3.7 *Engine initializer*

Tato komponenta slouží k inicializaci celého workflow enginu. Existují zde metody pro test spojení s databází, načtení a rozparsování všech nasazených workflow a načtení automatických a manuálních úloh z Dunning logic. Pokud některá s inicializačních metod selže nelze upomínací aplikační server spustit.

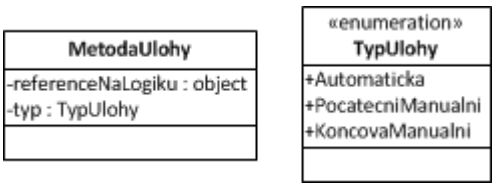
Připojení do databáze je testováno z důvodu dostupnosti databázového serveru, neboť se k němu připojujeme často. Tímto vyloučíme případné pozdější chyby ve spojení s databází.

Nasazená workflow načteme z databáze a pomocí této komponenty rozparsujeme do jednotlivých *Workflow* objektů [Obrázek 4-5]. Tento objekt obsahuje jméno autora, jméno workflow, verzi, časovou platnost a slovník všech objektů automatických a manuálních úloh. V tomto slovníku lze podle identifikace úlohy vyhledávat. U každého objektu *Uloha* nalezneme jméno úlohy a slovník objektů možných přechodů mezi úlohami. Objekt *Prechod* nabízí jméno a zpoždění vazby na další úlohu.



Obrázek 4-5: Diagram objektu Workflow

Dále zde načítáme logiku automatických a manuální úlohy z komponenty Dunning logic. Důležitým a jedinečným parametrem u těchto úloh je jejich název. Z načtených úloh vytváříme slovník objektů *MetodaUlohy* ve kterém pak jednotlivé úlohy podle názvu vyhledáváme. Mezi vlastnosti objektu MetodaUlohy [Obrázek 4-6] patří odkaz na logiku metody a typ úlohy, který může nabývat hodnot Automaticka, PocatecniManualni a KoncovaManualni.



Obrázek 4-6: Diagram objektu MetodaUlohy

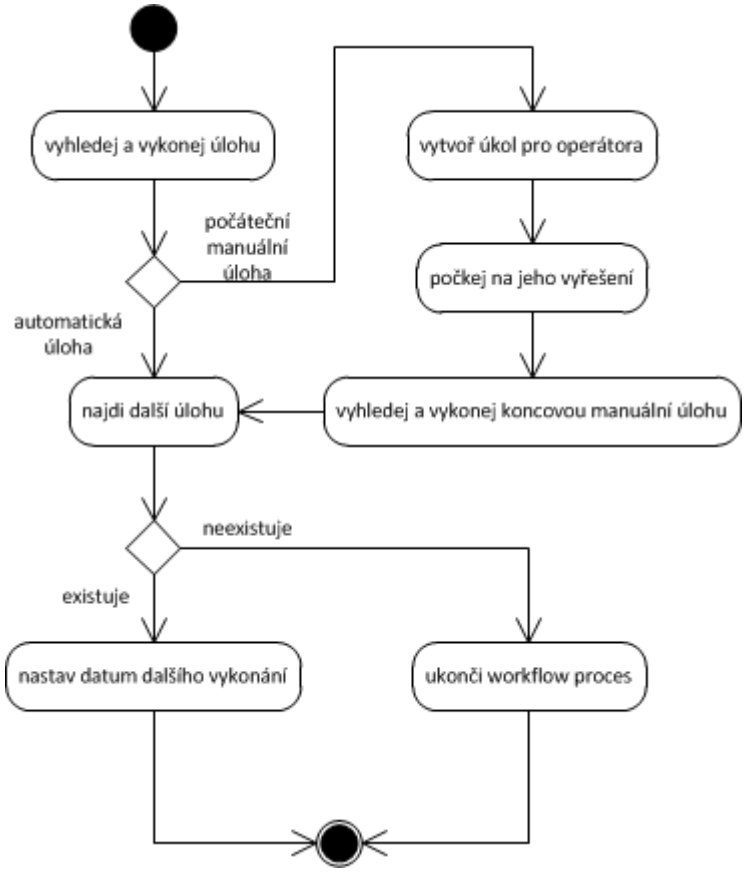
4.3.8 *Engine starter*

Engine starter zajišťuje spuštění celého workflow enginu. Dále je zde umístěna logika pro vykonávání a monitorování běžících workflow procesů.

Samotné spuštění workflow enginu probíhá v několika krocích. Nejdříve se pomocí komponenty Engine initializer spustí všechny potřebné inicializační kroky potřebné pro běh workflow enginu viz [Kapitola 4.2.7]. Následně se spustí logika pro vykonávání a monitorování workflow procesů. Nakonec spustíme hostující WCF služby (Task Manager Service, Workflow Manager Service, Administration Service a Authentication Service) pro klientské aplikace. Tímto máme úspěšně spuštěn celý server. Pokud dojde v některém kroku k chybě, nelze server spustit.

Logika pro vykonávání běžících workflow procesů pracuje na základě pravidelného dotazování záznamů v tabulce *Workflow_instance* [Kapitola 4.8]. Kde hledáme workflow procesy, které splňují následující podmínky: jsou aktivní, nevypršela jim časová platnost a uplynul čas zpoždění přechodu z předcházející úlohy do nové. Jak logika pracuje s takovýmto workflow procesem je vidět na [Obrázek 4-7].

Z obrázku vyplývá, že nejdříve vykonáme příslušnou úlohu, kterou najdeme ve slovníku objektů *MetodaUlohy* [Obrázek 4-5]. Pokud jde o počáteční manuální úlohu pozastavíme proces vykonávání workflow a vytvoříme příslušný úkol pro operátora. Jakmile operátor vyřeší úkol, vykonáme koncovou manuální úlohu. Dále nalezneme podle předcházejícího řešení úlohy další úlohu v pořadí. Tuto úlohu najdeme v objektu *Workflow* [Obrázek 4-5]. Pokud žádná další úloha neexistuje, workflow proces ukončíme. V případě její existence nastavíme datum dalšího vykonání workflow procesu. Datum určíme jako součet aktuálního data a doby trvání zpoždění přechodu na tuto nově nalezenou úlohu.



Obrázek 4-7: Aktivity diagram pro vykonání jedné úlohy workflow procesu

Logika pro monitorování běžících workflow procesů zajišťuje kontrolu časové platnosti a přiřazenosti úkolů operátorům. Opět pracuje na principu pravidelného dotazování záznamů v tabulce *Workflow_instance* [Kapitola 4.8]. Pokud dojde u běžícího workflow procesu k vypršení jeho platnosti je takovýto proces terminován, což zamezuje jeho další vykonávání. Dalším úkolem této logiky je kontrola práce operátorů. Operátoři si ve své webové aplikaci přiřazují a řeší úkoly. Pokud operátor nevyřeší přiřazený úkol v určitém časovém rozmezí, tato logika zajistí odejmutí úkolu a tím tak umožní dalším operátorům jeho přiřazení. O této nedůslednosti operátora se dozví příslušný vedoucí pracovník prostřednictvím emailu.

4.3.9 *Task manager service*

Komponenta Task manager service nabízí rozhraní *ITaskManager*. Služba vytváří rozhraní, s kterým komunikují klientské aplikace typu TaskManager. Prostřednictvím tohoto rozhraní lze zasílat seznam nevyřešených úkolů a podrobnosti k nim. Operátoři pak mají možnost si jednotlivé úkoly přiřazovat, řešit a posílat notifikace o vyřešení.

Informace o nevyřešených úkolech jsou načítány z tabulky *Ukol* [Kapitola 4.8], kam se i průběžně zapisuje jejich stav. Jakmile operátor svůj úkol vyřeší, pošle prostřednictvím tohoto rozhraní řešení a workflow engine podle něj může dále řešit upomínací proces.

4.3.10 *Workflow manager service*

Workflow manager service je služba, která nabízí rozhraní *IWorkflowManager*. S tímto rozhraním komunikují klientské aplikace typu WorkflowManager pomocí něhož mohou spravovat uložené a nasazené diagramy workflow procesů. Zároveň lze pomocí tohoto rozhraní získávat seznamy workflow diagramů (uložených, nasazených) a seznamy úloh (automatických i manuálních).

Uložené diagramy jsou v tabulce *Workflow_navrhar_data* [Kapitola 4.8]. Nasazené diagramy nalezneme v tabulce *Workflow_data*. Seznam definic automatických a manuálních úloh získáme z komponenty Dunning logic [Kapitola 4.3.5].

4.3.11 *Administration service*

Komponenta Administration service zprostředkovává rozhraní *IAdministration*. Prostřednictvím tohoto rozhraní komunikují klientské aplikace typu UserManager. Jsou zde nabízeny funkce pro správu dlužníků a uživatelů

v systému. V souvislosti se správou dlužníků, zajišťuje rozhraní další funkce pro spouštění, terminaci workflow procesů a úhradu dlužné částky.

Dlužníci jsou uloženy v tabulce *Dluznik* [Kapitola 4.8]. Uživatele nalezneme v tabulce *Uzivatel*. Pro spuštění workflow procesu stačí vytvořit příslušný záznam v tabulce Workflow_instance s aktivním stavem. Pro terminaci procesu změníme tento stav na terminovaný.

4.3.12 *Authentication service*

Authentication service nabízí rozhraní *IAuthentication*. Služba vytváří rozhraní se kterým komunikují všechny klientské aplikace (TaskManager, WorkflowManager a UserManager). Rozhraní slouží pouze k ověření uživatelských přístupových údajů. Mezi tyto údaje patří uživatelské jméno, heslo a role v systému, které jsou uloženy v tabulce *Uzivatel* [Kapitola 4.8].

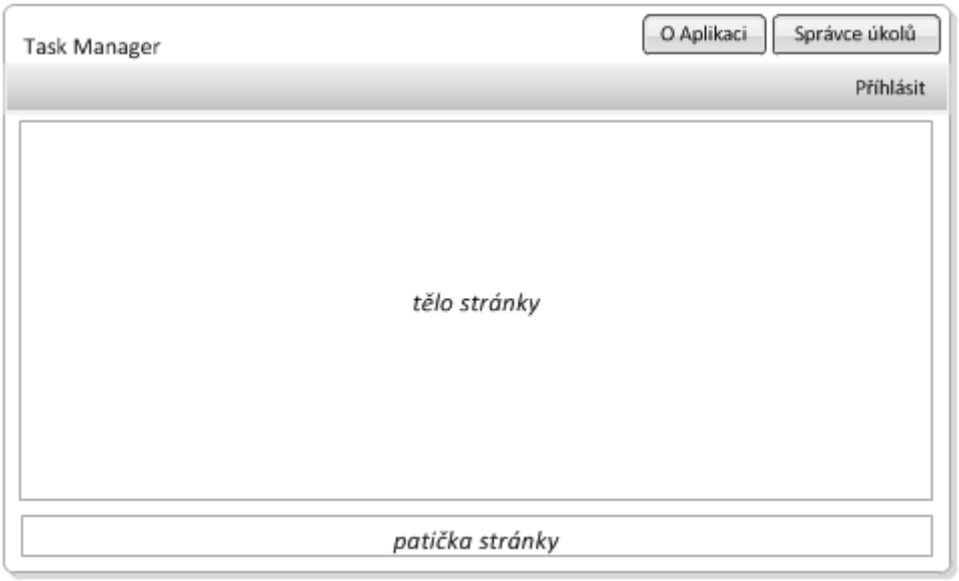
4.4 *Web server*

Webový server je prostředníkem v komunikaci mezi webovým prohlížečem s klientskou aplikací a upomínacím aplikačním serverem. Existují zde tedy obdobné funkce jako u rozhraní *ITaskManager*, *IWorkflowManager*, *IAdministration* a *IAuthentication*, neboť webový server tyto rozhraní využívá. Dále jsou pomocí tohoto serveru hostovány všechny klientské aplikace. Typickým příkladem tohoto serveru v prostředí Windows je Microsoft IIS (Internet Information Server).

4.5 *Task manager*

Task manager je webová aplikace běžící v okně internetového prohlížeče pomocí níž operátoři řeší úkoly. Tyto úkoly vytváří upomínací aplikační server při vykonávání workflow procesů. Stránku aplikace lze rozdělit na hlavičku, logovací lištu, tělo a patičku [Obrázek 4-8]. V hlavičce je vlevo umístěn název aplikace a na pravé straně jsou odkazy na všechny stránky aplikace. Mezi stránky

aplikace patří samotný správce úkolů a stránka s informacemi o aplikaci. Logovací lišta slouží pro zobrazení informací o přihlášeném uživateli, kde je i tlačítko pro přihlášení případně odhlášení uživatele z aplikace.



Obrázek 4-8: Layout hlavní stránky aplikace Task Manager

V aplikaci existují dvě hlavní stránky a jedno dialogové okno. Jedna stránka pro samotný správce úloh a druhá pro jednoduchý popis aplikace. V dialogovém oknu, které se zobrazuje v popředí hlavní stránky, řeší operátor svůj přiřazený úkol.

4.5.1 *Stránka o aplikaci*

Patička je v případě této stránky prázdná. V těle stránky je stručný popis o tom k čemu aplikace slouží a jak se používá.

4.5.2 *Stránka správce úkolů*

V těle stránky je zobrazen seznam nevyřešených úkolů. Tento seznam je stránkovatelný, a proto jsou v patičce zobrazeny čísla jednotlivých stránek. U každého nevyřešeného úkolu jsou zobrazeny informace jako název úkolu, datum vytvoření, jméno dlužníka a jeho dluh. Dále je zde tlačítko pro přiřazení

úkolu operátorovi. Seznam nevyřešených úkolů je v pravidelných intervalech obnovován, aby byla zajištěna jeho aktuálnost.

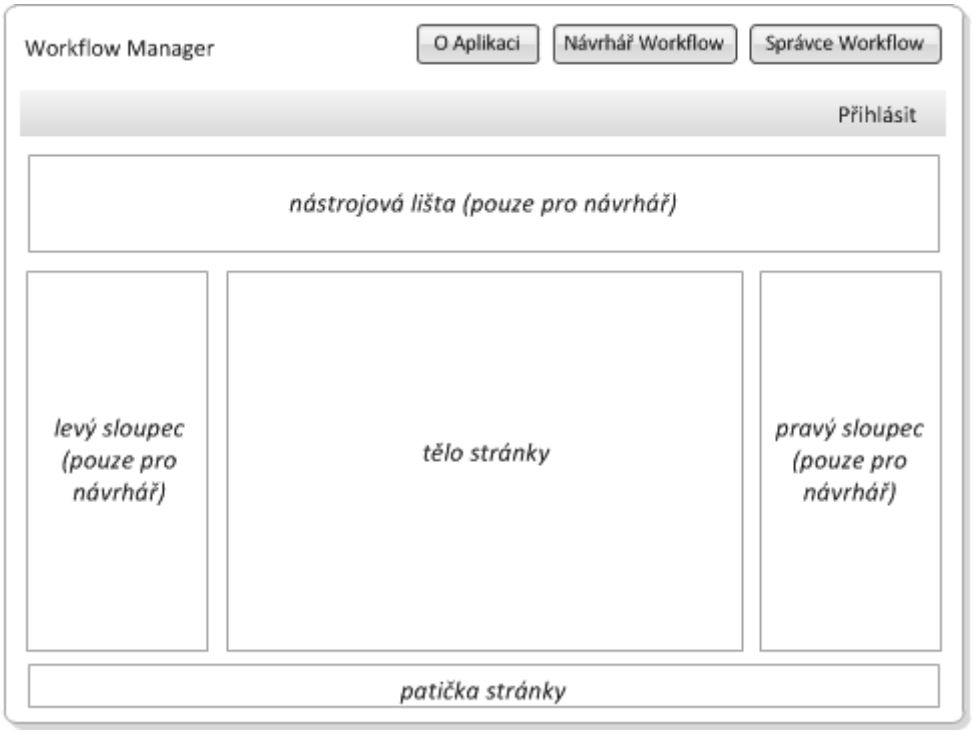
4.5.3 *Dialogové okno řešení úkolu*

Typ dialogového okna k řešení úkolu určuje počáteční manuální úloha komponenty Dunning logic [Kapitola 4.3.5]. V okně je zobrazeno zadání úkolu, stav aktuálního workflow procesu a informace o dlužníku. Dále zde má operátor k dispozici tlačítka, které charakterizují vyřešení úkolu. V případě nejednoznačného řešení úkolu může operátor přidat k řešení svůj vlastní komentář.

4.6 *Workflow manager*

Workflow manager je webová aplikace běžící v okně internetového prohlížeče. Manažeři workflow pomocí ní spravují diagramy workflow procesů. Rozložení stránky aplikace je na [Obrázek 4-9].

Mezi hlavní stránky patří návrhář a správce workflow. Dále je zde stránka s informacemi o aplikaci.



Obrázek 4-9: Layout hlavní stránky aplikace Workflow Manager

4.6.1 *Stránka o aplikaci*

Zde platí stejná funkčnost jako pro stránku o aplikaci Task Manager [Kapitola 4.5.1].

4.6.2 *Stránka návrhář workflow*

V nástrojové liště jsou umístěné tlačítka pro nový návrh, otevírání, validaci, ukládání a nasazování workflow diagramů. Pro nový a otevírání stávajících návrhů používáme dialogová okna. Levý sloupec obsahuje stavební prvky workflow, což jsou automatické a manuální úlohy. Tělo stránky slouží jako plocha pro návrh workflow diagramu. Pravý sloupec obsahuje seznam vlastností workflow, které lze v průběhu návrhu měnit. Mezi tyto vlastnosti patří název, autor, verze a časová platnost workflow. Časovou platnost lze definovat pomocí dnů, hodin a minut. Do patičky stránky se v případě validace diagramu vypisují chyby.

Sestavování workflow diagramu je prováděno stylem drag and drop. Jednotlivé automatické a manuální úlohy jsou přetahovány ve formě krabiček na plochu návrháře. Krabička úlohy má definován název a možné stavy řešení. Tyto stavy řešení jsou pomocí přechodů (šipek) propojovány s dalšími úlohami. Ze stavu řešení úlohy může existovat pouze jeden přechod na další úlohu. Krabičky s úlohami lze libovolně po ploše návrháře přemísťovat.

Pomocí kontextové nabídky vyvolané pravým klinutím myši na krabičku úlohy lze:

- nastavit startovací úlohu (první úloha která se vykonává)
- odstranit úlohu z plochy návrháře

Pomocí kontextové nabídky vyvolané pravým kliknutím myši na přechod mezi krabičkami úloh lze:

- odstranit přechod z plochy návrháře
- nastavit vlastnosti přechodu (zpoždění)

Pro nastavení zpoždění přechodu je vyvoláno dialogové okno. V tomto okně definujeme zpoždění pomocí dnů, hodin a minut.

Při návrhu nového nebo otevírání stávajícího workflow diagramu kontrolujeme obsah plochy návrháře. V případě, že se na ploše nachází workflow diagram informujeme o tom uživatele za pomoci informačního dialogového okna. Uživatel následně zváží, zda-li chce opravdu přijít o obsah plochy návrháře.

Validaci navrženého workflow spouštíme manuálně nebo automaticky při ukládání nebo nasazování. Správný návrh má definován všechny vlastnosti workflow a navržený diagram má všechny úlohy provázán přechody. Zároveň v diagramu existuje jedna startovací úloha.

4.6.2.1 *Dialogové okno nový návrh*

V dialogovém okně pro nový návrh je možné nastavit vlastnosti navrhovaného workflow diagramu. Lze tedy definovat název, autora, verzi a časovou platnost workflow. Spodní část okna obsahuje dvě tlačítka. Jedno tlačítko slouží pro vytvoření nového prázdného návrhu a druhé pro zrušení akce.

4.6.2.2 *Dialogové okno otevřít návrh*

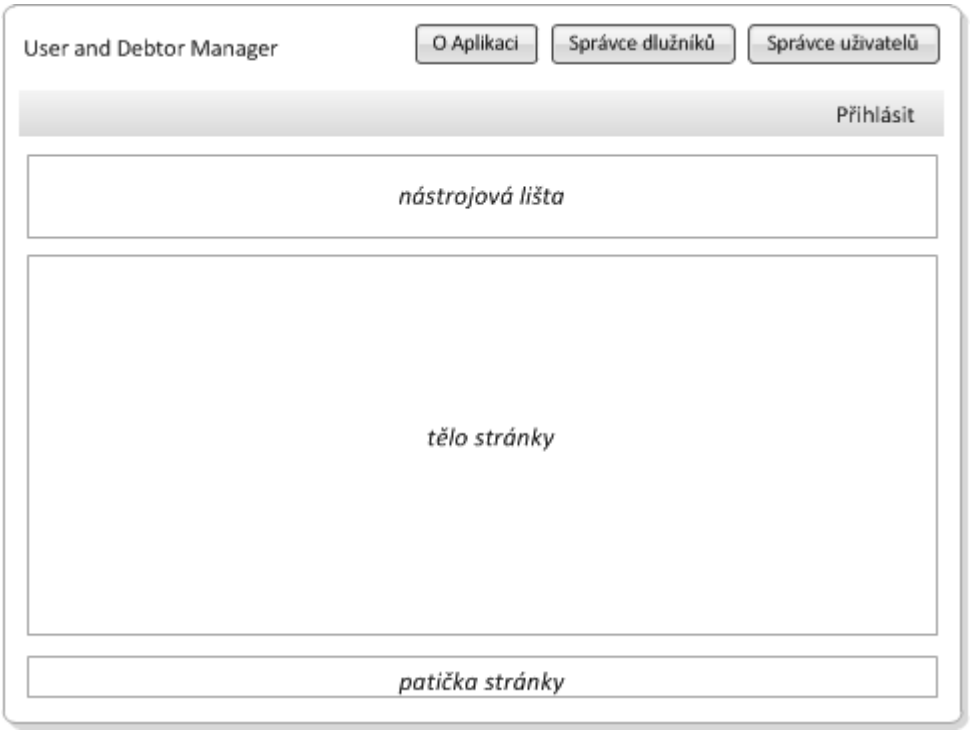
Toto okno slouží pro otevření v minulosti vytvořených návrhů workflow diagramů. Umožňuje otevírat uložené a nasazené workflow diagramy. Seznamy těchto diagramů jsou stránkovatelné. Nasazené workflow diagramy jsou v návrháři otevřeny pouze pro čtení a nelze je nijak upravovat. Ve spodní části okna jsou tlačítka pro otevření návrhu a zrušení akce.

4.6.3 *Stránka správce workflow*

V případě této stránky je z layoutu [Obrázek 4-9] použita pouze patička a tělo stránky, kde je zobrazen seznam všech uložených nebo nasazených workflow. Mezi těmito typy lze vybírat nebo je můžeme zobrazit hromadně. Seznamy jsou stránkované a v patičce stránky jsou zobrazeny jejich čísla. U každého workflow jsou zobrazeny informace o jeho názvu, verzi a datu vytvoření. Dále je zde umístěno tlačítko, kterým lze vybraný workflow diagram odstranit. Odstraňovat lze uložené, nebo nasazené diagramy, které však nesmějí být právě používané v běžících workflow procesech. Pokud bychom takovýto diagram odstranili, upomínací aplikační server by po restartu nemohl, s takovýmto běžícím workflow procesem pokračovat ve vykonávání.

4.7 *User + debtor manager*

User + debtor manager je webová aplikace běžící v okně internetového prohlížeče. Administrátoři za pomoci této aplikace spravují dlužníky a uživatele v systému. Rozložení hlavní stránky této aplikace je na [Obrázek 4-10].



Obrázek 4-10: Rozložení hlavní stránky aplikace User + Debtor Manager

Mezi hlavní stránky patří správce dlužníků a uživatelů. Dále je zde stránka s informacemi o aplikaci.

4.7.1 *Stránka o aplikaci*

Zde platí stejná funkčnost jako pro stránku o aplikaci Task Manager [Kapitola 4.5.1].

4.7.2 *Stránka správce dlužníků*

V nástrojové liště je umístěné tlačítko pro vytvoření nového dlužníka. Tělo stránky obsahuje stránkovatelný seznam všech dlužníků v systému. V patičce stránky jsou zobrazeny stránky tohoto seznamu. Každý dlužník v seznamu je identifikován jménem, výší dluhu, přiřazeným workflow procesem a stavem tohoto procesu. Dále jsou zde tlačítka pro zaplacení dluhu, odstranění dlužníka a zobrazení jeho detailu. Seznam dlužníků je v pravidelných intervalech obnovován.

4.7.2.1 *Dialogové okno přidat dlužníka*

V okně pro přidání dlužníka definujeme informace o samotném dlužníkovi, jeho dluhu a vybíráme pro něj vhodný workflow proces pro vymáhání dluhu. Pro vytvoření dlužníka vyplňujeme celé jméno, email, telefon, adresu, výši dluhu, volitelný komentář a ze seznamu vybíráme vhodný workflow proces. Spodní část okna obsahuje dvě tlačítka, jedno pro vytvoření dlužníka a druhé pro zrušení akce.

4.7.2.2 *Dialogové okno detail dlužníka*

V detailu vidíme informace o dlužníkovi a jeho workflow procesu. Máme možnost tedy vidět celé jméno, email, telefon, adresu, výši dluhu a sled již provedených automatický a manuálních úloh z workflow procesu. Dále vidíme podrobný stav workflow procesu, zejména datum vytvoření, datum vypršení platnosti, aktuálně vykonanou úlohu a datum vykonání další úlohy v pořadí.

4.7.3 *Stránka správce uživatelů*

Stránka správce uživatelů je podobná stránce správce dlužníků. V nástrojové liště se nachází tlačítko pro vytvoření nového uživatele. V těle stránky je umístěn stránkovatelný seznam všech uživatelů. V patičce jsou opět umístěny jednotlivé stránky seznamu. Položka seznamu jedno uživatele je identifikována loginem, celým jménem, datem vytvoření, emailem a rolí uživatele. Seznam uživatelů lze třídit podle rolí. K položkám seznamu dále patří tlačítka pro odstranění uživatele a zobrazení jeho detailu.

4.7.3.1 *Dialogové okno přidat uživatele*

Dialogové okno pro přidání uživatele umožňuje zapsat údaje jako jsou login, heslo, celé jméno, roli a email uživatele. Zapisování hesla je skryto pomocí jednotných znaků. Z důvodu ověřování správnosti hesla jej musí uživatel zadávat dvakrát. Ve spodní části okna jsou dvě tlačítka. Jednot tlačítko slouží pro uložení uživatele a druhé pro zrušení akce.

4.7.3.2 *Dialogové okno detail uživatele*

Toto okno je téměř stejné jako dialogové okno pro přidání uživatele [Kapitola 4.7.3.1]. Není zde pouze možnost změny loginu.

4.8 *Oracle database*

Všechna data, které jsou potřeba v systému uchovávat jsou uložena na databázovém serveru. Na tomto serveru je nainstalovaná oracle databáze, konkrétně typ SE (Standart Edition). Tato edice byl zvolena záměrně, neboť edice EE (Express Edition, vhodná pro studentské a testovací účely) nemá podporu pro 64bitový operační systém Windows, pod kterým je celý systém vyvíjen.

V databázi najdeme příslušné tabulky z uloženými daty, uložené procedury, používané pro operace nad daty a spouště pro automatickou inkrementaci čísel primárních klíčů při vkládání záznamů.

Pro vytvoření databáze je důležitý návrh jejího relačního modelu. Návrh tohoto modelu najdeme na [Obrázek 4-11] a [Obrázek 4-12]. Tučně zvýrazněné položky tabulek nesmějí obsahovat hodnotu NULL. Relační model obsahuje následující tabulky:

- **Tabulka Dluh**

  Do této tabulky se ukládají informace o zákazníkově dluhu mezi které patří jeho výše, komentář a datum vytvoření.

- **Tabulka Dluh status**

  Tato tabulka obsahuje záznamy o všech možných stavech dluhů. Rozlišujeme mezi stavy aktivní a terminovaný.

- **Tabulka Dluznik**

  Zde jsou uloženy záznamy o dlužnících. Každý záznam je identifikován jménem, příjmením, emailem, telefonem, adresou, poštovním směrovacím číslem, městem a zemí.

- **Tabulka Uzivatel status**

  Tato tabulka obsahuje záznamy o všech možných stavech uživatelů. Rozlišujeme mezi stavy aktivní a smazaný.

- **Tabulka Uzivatel**

  Do této tabulky jsou ukládány záznamy o uživatelích v systému. Každý uživatel je identifikován loginem, heslem, jménem, příjmením, emailem a datem vytvoření.

- **Tabulka Uzivatel role**

  V tabulce *Uzivatel_role* jsou uloženy všechny uživatelské role vyskytující se v systému. Mezi tyto role patří administrátor, operátor a manažer workflow.

- **Tabulka Workflow instance**

  Do této tabulky jsou ukládané všechny spuštěné workflow procesy. Spuštěnému workflow procesu lze říkat i workflow instance. Tato tabulka je velmi důležitá pro komponentu Engine starter [Kapitola 4.3.8]. Jsou zde např. uloženy informace o úloze, která čeká na vykonání. Mezi tyto informace patří datum vykonání úlohy, její číslo a název. Mezi další důležitá data patří serializovaná dlužníkova data a časovou platnost celé workflow instance.

- **Tabulka Workflow instance status**

  Tato tabulka obsahuje záznamy o všech možných stavech workflow instance. Rozlišujeme mezi stavy aktivní, pozastavená, ukončená a terminovaná. Instance je pozastavená v případě řešení úkolu operátorem. Ukončená instance je taková, která už nemá žádné úlohy k vykonání.

- **Tabulka Ukol**

  Tabulka *Ukol* obsahuje úkoly, které řeší operátoři. S vytvořením úkolu, se zde ukládá číslo formuláře, ve kterém pak operátor tento úkol řeší. S vybráním úkolu operátorem dochází k vyplnění jména vlastníka úkolu, data přiřazení a vstupních serializovaných dat dlužníka. Jakmile operátor úkol vyřeší, dojde k zápisu řešení a případně i změněných výstupních serializovaných dat dlužníka.

- **Tabulka Ukol status**

  Tato tabulka obsahuje záznamy o všech možných stavech úkolů v průběhu jejich existence. Rozlišujeme mezi stavy nevyřešený, přiřazený a vyřešený.

- **Tabulka Workflow data**

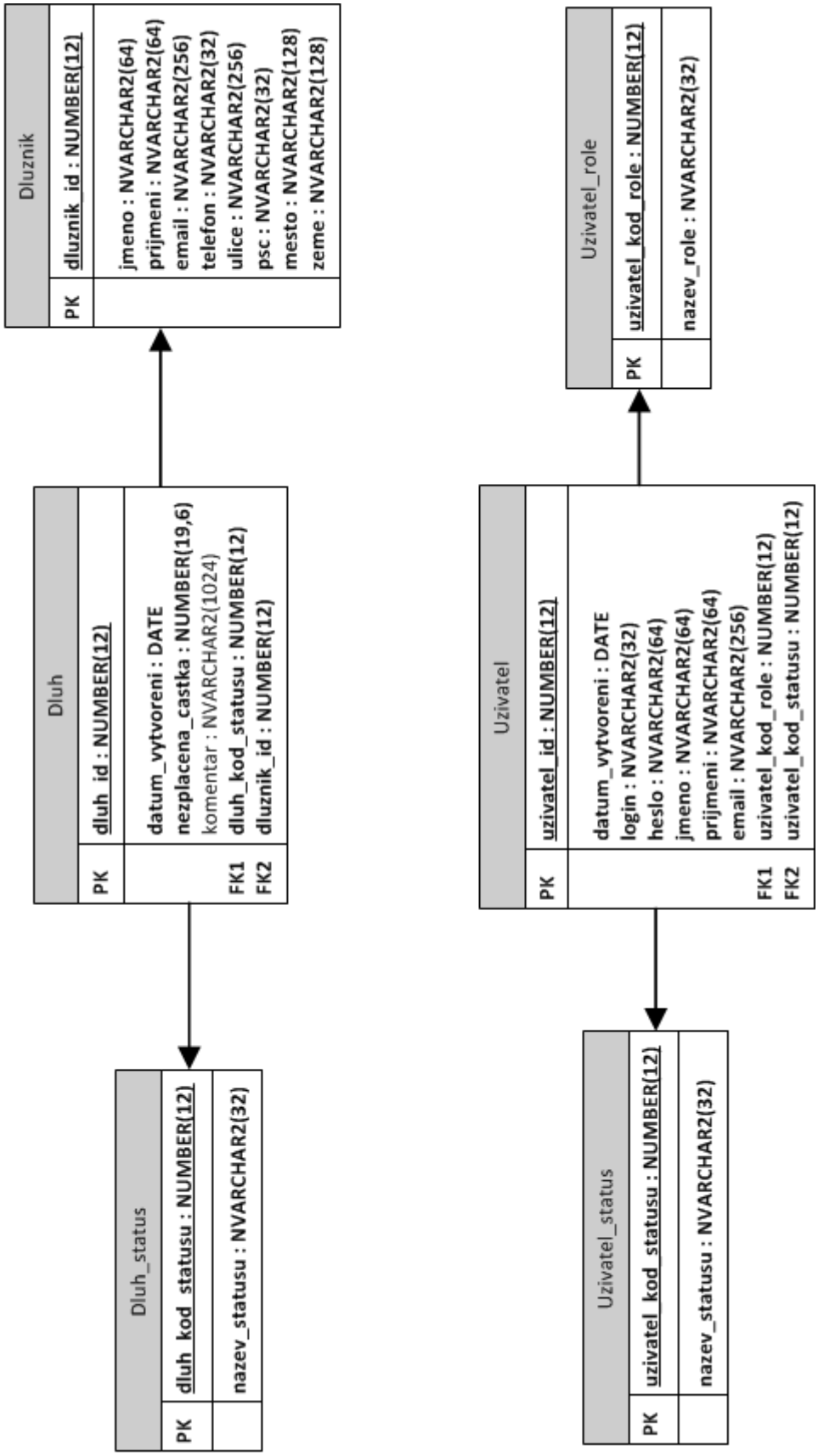
  Tabulka *Workflow_data* obsahuje nasazené worfklow diagramy. Tyto záznamy jsou důležité pro komponentu Engine initializer [Kapitola 4.3.7]. Každý nasazený workflow diagram je identifikován názvem, verzí, datem vytvoření a samotnými daty diagramu.

- **Tabulka Workflow navrhar data**

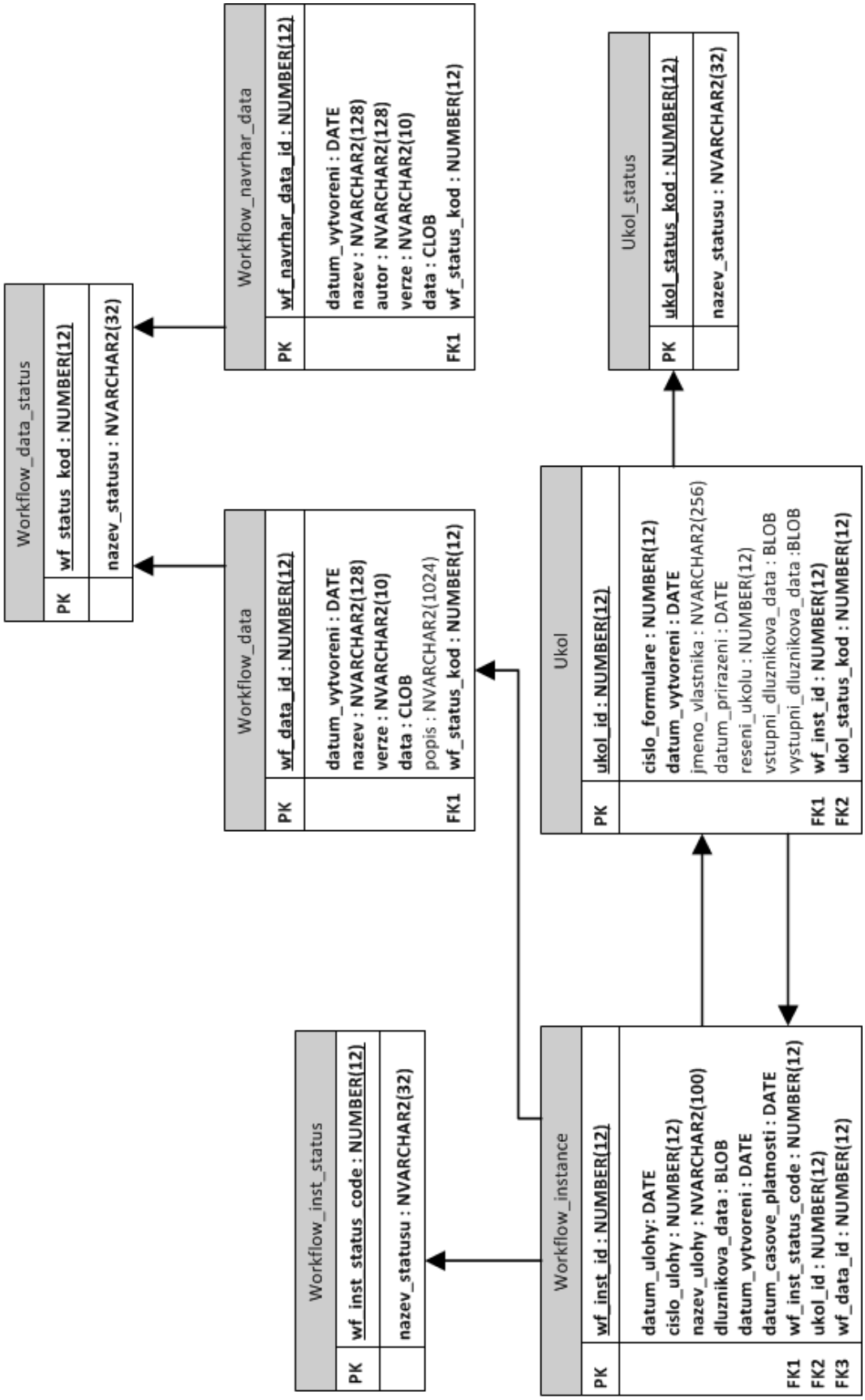
  Tabulka *Workflow_navrhar_data* obsahuje uložené workflow diagramy, které slouží pro případné budoucí nasazení. Každý uložený diagram je identifikován názvem, jménem autora, verzí, datem vytvoření a daty diagramu.

- **<u>Tabulka Workflow data status</u>**

  Tato tabulka obsahuje záznamy o všech možných stavech workflow diagramů. Rozlišujeme mezi stavy aktivní a smazaný.



Obrázek 4-11: Relační databázový model, část 1



Obrázek 4-12: Relační databázový model, část 2

5 Implementace

Serverová i klientská část systému je impelmentována pomocí jazyka C#. Jako vývojové prostředí bylo použito Microsoft Visual Studio 2010. V serverové části systému je využita technologie .NET Framework 4.0. Klientské webové aplikace jsou založeny na technologii Silverlight 4.0, kde byl pro implemetaci vzhledu aplikace, mimo jazyku C#, použit jazyk XAML.

Vzhledem k tomu, že se celá implementace nachází na přiloženém CD, není potřeba v této kapitole uvádět kompletní popis implementace systému. Kapitola se tedy spíše zaměřuje na popis důležitých případně něčím zajímavých řešení.

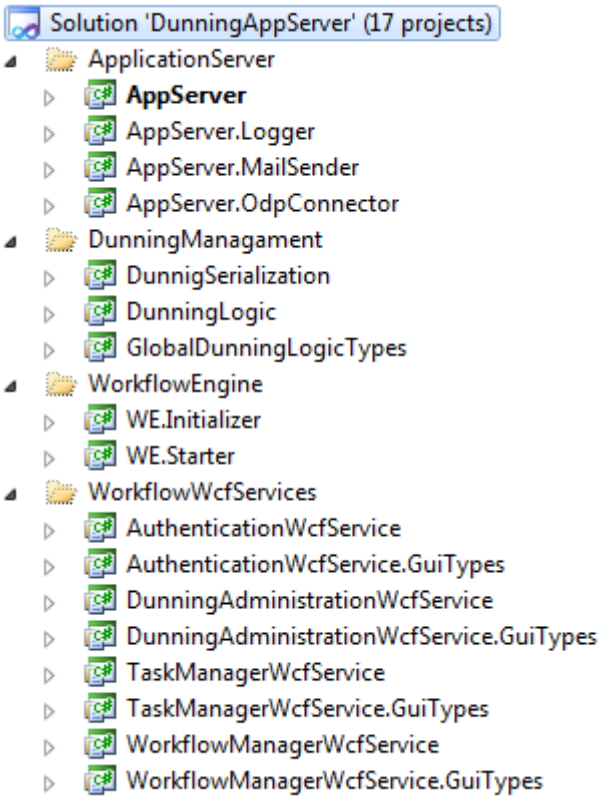
5.1 *Dunning application server*

V této kapitole popíšeme strukturu řešení aplikace, reprezentaci workflow diagramu, reprezentaci logiky úloh a implementaci s hostováním WCF služeb.

5.1.1 *Struktura řešení*

Struktura řešení byla vytvořena podle navržených komponent v podkapitole analýzy [Kapitola 4.3]. Celé řešení obsahuje 17 projektů [Obrázek 5-1], které je logicky rozděleno do čtyř skupin. Mezi tyto skupiny patří ApplicationServer, DunningManagament, WorkflowEngine a WcfServices.

ApplicationServer je skupina projektů, které zajišťují logiku běhu serveru, logování částí systému, odesílání emailů a rozhraní s oracle databází. Skupina DunningManagament zahrnuje kompletní správu nad automatickými a manuálními úlohami workflow procesů. Skupina WorkflowEngine zajišťuje počáteční inicializaci a běh workflow enginu. Poslední skupina WcfServices definuje příslušné služby, které implementují rozhraní ke klientským aplikacím. Dále jsou zde definice datových objektů, které služby nabízejí.



Obrázek 5-1: Struktura řešení implementace Dunning application serveru

Projekt AppServer je startovací a obsahuje metodu main. Je tedy koncipován jako spustitelná assembly *.exe. Všechny ostatní projekty jsou nespustitelné knihovny *.dll, které mohou ostatní projekty referencovat.

5.1.2 *Reprezentace workflow diagramu*

Nejjednoduší způsob jak reprezentovat navržený workflow diagram [Kapitola 4.1] je použít nějaký strukturální jazyk, který je obecně známý a je podporován .NET Frameworkem 4.0. Volba padla na značkovací jazyk XML. Celý xml soubor reprezentující navržený workflow diagram pro práci s neplatiči [Obrázek 4-1] lze nalézt v [Příloha B]. Tento xml soubor lze rozdělit na části *common* a *workflow*. V oblasti ohraničené tagy *common* nalezneme vlastnosti workflow diagramu. Mezi vlastnosti patří název, jméno autora, verze a časová platnost workflow diagramu. V oblasti ohraničené tagy *workflow* je pak

definován samotný diagram, který je složen z automatický *automaticStep* nebo manuálních *manualStep* úloh. Každá úloha *automaticStep* nebo *manualStep* pak může mít definovány vazby *transition* na další úlohy.

S touto reprezentací workflow diagramu pracujeme v projektu WE.Initializer. Konkrétně jde o metodu zajišťující parsování xml souboru. Popišme si tedy tuto samotnou metodu knihovny WE.Initializer.

Knihovna WE.Initializer pro práci s xml využívá jmené prostory System.Xml a System.Xml.Schema. Metoda zajišťující parsování xml reprezentujících nasazené workflow diagramy se jmenuje *InitializeWorkflowsMetadata*. Je to metoda, která nemá žádné vstupní parametry. Na konci pouze vrací výčet typu *EngineResult*, který reprezentuje úspěšnost inicializace (parsovaní) nasazených workflow diagramů. Prvním krokem je načtení nasazených workflow diagramů z databáze. O to se stará instance *DunningPackage* typu singleton, která obsahuje všechny metody, potřebné systémem, pro práci s daty umístěnými v databázi. Pomocí této instance a její příslušné metody, obdržíme všechny aktivní nasazené workflow diagramy. Tyto diagramy obdržíme jako tabulku, která má v každém řádku dvojici klíč a hodnota. Klíčem je zde jedinečné identifikační číslo workflow diagramu a hodnotou je objekt reprezentují samotný diagram. Tento objekt mimo xml metadat obsahuje ještě název, jméno autora a verzi diagramu. Nyní potřebujeme všechny xml metadata z tabulky reprezentovat jako objekt *Workflow* [Kapitola 4.3.7]. K tomuto účelu nám slouží metoda *ParseAndLoadWorkflowMetadata*, která má jako vstupní parametr výše uvedenou tabulku.

Metoda *ParseAndLoadWorkflowMetadata* nejdříve xml metadata validuje oproti schématu. V případě že metadata neodpovídají schématu, nelze pokračovat v parsování. Pokud je vše v pořádku dochází k postupnému hledání hodnot xml elemetů reprezentujících název, autora, verzi a časovou platnost

workflow diagramu. Následně dochází k procházení všech elementů reprezentujích automatické a manuální úlohy. S každou takovouto úlohou je kontrolováno zda ji upomínací aplikační server umí vykonávat. Toho je docíleno porovnáváním jedinečných názvů těchto úloh s jejich logikou vykonávání, které je definována v projektu DunningLogic. Více o reprezentaci této logiky lze nalézt v [Kapitola 5.1.3]. Pokud by logika k příslušné úloze neexistovala, nebylo by možné tento diagram používat pro vykonávání workflow procesů. Pokud je tedy vše v pořádku načteme u každé úlohy její identifikační číslo a název. Dále parsujeme všechny možné přechody těchto úloh spolu s jejich parametry. Parametry přechodů jsou název, cílová úloha a zpoždění. Všechny tyto rozparsované workflow diagramy pak reeprezentují jednotlivé objekty *Workflow* [Obrázek 4-5] umístěné v tabulce. Klíčem tabulky je identifikace workflow diagramu a hodnotou pak samotný objekt *Workflow*.

5.1.3 *Reprezentace logiky úloh*

Definice logiky automatických a manuálních úloh je v projektu DunningLogic. Pro každou automatickou úlohu je zde implementována jedna metoda. Manuální úlohy zde mají implementovány metody dvě. Jedna je definována jako počáteční a druhá jako koncová viz [Kapitola 4.3.5]. Jednotlivé rozpoznávání těchto metod je řešeno pomocí specializovaných atributů metod [Obrázek 5-2]. Je zde tedy implementován *StepMethodAttribute*, který dědí od System.Attribute. Konstruktor tohoto atributu obsahuje název úlohy (*methodName*) a typ metody (*methodType*). Typ metody je výčtem automatické, počáteční manuální nebo koncové manuální metody.

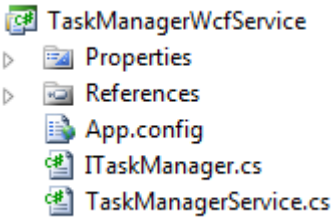
```
public class StepMethodAttribute : Attribute
{
    public StepMethodAttribute(string methodName, StepMethodTypes methodType);

    public string MethodName { get; }
    public StepMethodTypes MethodType { get; }
}
```

Obrázek 5-2: Implementace atributu logiky úloh

DunningLogic je reprezentována jako knihovna *.dll, což už bylo řečeno v [Kapitola 5.1.1]. Tato knihovna je pak ve WE.Initializer dynamicky, pomocí reflexe prohledávána. Hledání jednotlivých metod zajišťujících vykonání automatických a manuálních metod pak probíhá právě díky těmto specifickým atributům. Samotné spouštění metod je realizováno reflexí.

5.1.4 *WCF služby*

Struktura a implementace všech WCF služeb je podobná, proto zde popíši např. službu TaskManagerWcfService [Obrázek 5-3].



Obrázek 5-3: Struktura WCF služby

Každá WCF služba je opět koncipována jako knihovna *.dll. Knihovna obsahuje definici vlastností *Properties*, reference *References*, konfiguraci *App.config*, definici rozhraní *ITaskManager* a implementaci samotné služby *TaskManagerService*.

Ve vlastnostech služby je uveden název, popis, použitá kultura a verze sestavení knihovny. Reference obsahují odkazy na další knihovny, které jsou ve službě využívany při její implementaci nebo k definici rozhraní.

Konfigurační soubor obsahuje nastavení služby. Definuje se zde zejména trojice adresa, binding a contract viz [Kapitola 2.3.1.1]. Konfiguraci služby lze provéct i v aplikaci Microsoft Service Configuration Editor, která je součástí vývojového prostředí.

Rozhraní služby které je identifikováno pomocí atributu *ServiceContract* obsahuje hlavičky všech metod, které služba implementuje. Každá hlavička metody je identifikována atributem *OperationContract*.

Samotná služba pak pouze implementuje všechny vlastnosti a metody daného rozhraní. Každá z metod má navratový typ *ServiceResult*, který určuje zda při vykonávání metody nedošlo k chybě.

Aby byly služby přístupné pro webový server a tím i pro klientské aplikace, musíme každou službu na našem serveru hostovat. Hostování služeb je prováděno ve WE.Starter. Tento modul má mimo jiné ještě na starosti spuštění logiky vykonávání a monitorování workflow procesů. Pro spuštění hostujících služeb, je důležité překopírovat konfigurační soubor každé služby do hlavního konfiguračního souboru serveru AppServer. Pokud bychom toto neudělali nebylo by možné službu hostovat. Spuštění a ukončení hostování služeb ukazuje [Obrázek 5-4].

```
ServiceHost taskManagerServiceHost = new ServiceHost(typeof(TaskManagerService));
taskManagerServiceHost.Open();

if (taskManagerServiceHost.State == CommunicationState.Opened)
{
    taskManagerServiceHost.Close();
}
```

Obrázek 5-4: Hostování WCF služeb

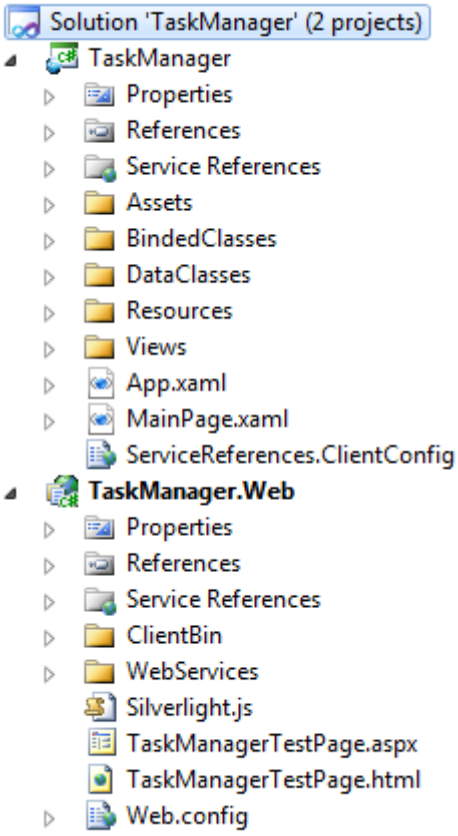
Pro hostování služeb musíme vytvořit instanci třídy *ServiceHost*, která v parametru konstruktoru přijímá typ naší služby. Poté stačí zavolat metoda *Open* a naše služba je hostována. Při ukončování běhu serverové aplikace, ukončujeme i hostování našich služeb. Před samotným zavoláním metody *Close* je vhodné zkontrolovat, zda je služba opravdu ve stavu *Opened*.

5.2 *Klientské aplikace*

Tato kapitola popisuje společnou strukturu řešení silverlight klientských aplikací (TaskManager, WorkflowManager a UserManager), realizaci webového serveru a jeho služeb.

5.2.1 *Struktura řešení*

Struktura řešení implementace klientských aplikací obsahuje 2 projekty [Obrázek 5-5]. Pro názornost byla vybrána aplikace TaskManager. Prvním projektem TaskManager je samotná silverlight aplikace. Druhý projekt TaskManager.Web je vývojový webový server, který hostuje silverlight aplikaci. Jako startovací projekt slouží právě TaskManager.Web, který zajistí spuštění okna internetové prohlížeče a zobrazení silverlight aplikace.



Obrázek 5-5: Struktura řešení implementace klientské aplikace

Každá silverlight klientská aplikace obsahuje *ServiceReferences*, *Assets*, *BindedClasses*, *Resources*, *Views*, *App.xaml* a *MainPage.xaml*.

- **ServiceReferences**

  SericeReferences obsahuje reference na webové služby, která silverlight aplikace používá. Každá z aplikací využívá referenci na webovou službu *AuthenticationWebServiceRef*. Další reference se již liší podle druhu aplikace. Aplikace typu TaskManager referencuje webovou službu TaskManagerWebServiceRef, apod. Všechny metody těchto webových služeb jsou volány pouze v asynchroním řežimu. Silverlight aplikace jiný druh volání neumožňuje.

- **Assets**

  Zde jsou umístěny jednotlivé styly uživatelských prvků v sliverlight aplikaci. Styly jsou vytvořeny v jazyce XAML.

- **BindedClasses**

  V BindedClasses jsou umístěné třídy, které nám umožňují provázání dat s uživatelským rozhraním. Pokud je potřeba zobrazovaná data validovat, třídy implementují rozhraní *INotifyPropertyChanged* a *IDataErrorInfo*.

- **Resources**

  Ve zdrojích jsou uloženy soubory, které aplikace používá. Povětšinou jsou zde uloženy pouze ikony uživatelských rozhraní.

- **Views**

  Zde jsou uloženy všechny uživatelské obrazovky spolu s dialogovými okny. Vzhled uživatelských rozhraní je implementován v jazyce XAML, samotná logika pak v jazyce C#. Logika a vzhled je tedy oddělená.

- **MainPage.xaml**

  MainPage je hlavní uživatelská obrazovka zobrazená při startu aplikace. Vzhled a logika je opět oddělená.

5.2.2 *Webový server*

Realizace vývojového webového serveru je jednoduchá. Vývojové prostředí MS Visual Studia 2010 nám tuto možnost přímo při vytváření projektu pro silverlight aplikaci nabízí. Pro reálné nasazení klientských aplikací je však potřeba využít komerčních řešení webových serverů (IIS 7 nebo Apache 2.0). Pro naše účely jsou však možnosti vývojového webového serveru plně dostačující.

Mezi důležité prvky webového serveru patří [Obrázek 5-5]:

- **ServiceReferences**

  Zde jsou reference na WCF služby upomínacího aplikačního serveru. Mezi tyto reference služeb patří:

  - AuthenticationWcfServiceRef
  - UserManagerWcfServiceRef
  - TaskManagerWcfServiceRef
  - WorkflowManagerWcfServiceRef

- **WebServices**

  Tady implementuje jednotlivé webové služby, které referencují klientské aplikace. Při implementaci využíváme reference na WCF služby z předchozího odstavce.

- **Page.html**

  Hlavní html stránka do níž silverligth umísťuje jednotlivé uživatelské obrazovky.

5.3 *Databázový server*

Na databázovém serveru jsou umístěné uložené procedury, které obsahují SQL dotazy. Jedna procedura může obsahovat i několik dotazů. Použitím těchto metod jsme přispěli k přehlednosti kódu v AppServer.OdpConnector [5.1.1].

Příklad takovéto metody je na [Obrázek 5-6]. Na obrázku je metoda, která přiřazuje úkol operátorovi. Výstupní parametry *o_result* a *o_error_message* slouží k odhalení chyby při vykonávání procedury. Procedura jednoduše vyvolá vyjímku a tím dojde k zápisu kódu chyby a chybové hlášky. Pokud žádná vyjímka nenastane, vykoná se příslušný SQL dotaz. V tomto případě jde o *UPDATE*.

```
PROCEDURE assign_task
(
  o_result                    OUT NUMBER,
  o_error_message             OUT VARCHAR2,
  i_task_id                   IN task.task_id%TYPE,
  i_owner_user_name           IN task.owner_user_name%TYPE
)
IS
BEGIN
  o_result := c_ok;

  UPDATE task
    SET task_status_code = c_accepted_task,
        owner_user_name = i_owner_user_name,
        assigned_date =  sys_extract_utc(systimestamp)
    WHERE task_id = i_task_id;

EXCEPTION

  WHEN OTHERS THEN
    o_result := c_other_error;
    o_error_message := SQLERRM;

END assign_task;
```

Obrázek 5-6: Uložená procedura v oracle databázi

6 Závěr

V dnešní době se workflow systémy používájí stále ve větší míře. Jejich použitím lze řešit různá odvětví lidských činností. My jsme využili workflow k upomínání neplatičů mýtných poplatků.

V rámci analýzy jsme s pomocí zadání rozdělili celý systém na několik aplikací. Díky tomuto rozdělení jsme přišli na možné uživatelské role. Další kapitola se zabývala specifikací funkčních a nefunkčních požadavků na jednotlivé aplikace. Poslední částí analýzy byl rozbor existujících řešení, který nám posloužil jako dobrý základ pro další návrh. Zejména pak uživatelská rozhraní pro vytváření workflow diagramů.

Na základě získaných poznatků byl proveden návrh. Došlo k návrhu jednoho business procesu, který zajišťuje upomínání neplatiče. Navrhnutím architektury systému jsme mohli pokračovat s návrhem jednotlivých aplikací a datového uložiště.

Poslední částí byla implementace, kde jsme realizovali všechny navržené aplikace a datové uložiště. V průběhu implementace probíhalo jednoduché testování.

Celá práce byla pro mě velkým přínosem. Seznámil jsem se s novou technologií Silverlight, která je používána v oblasti pro vývoj uživatelsky bohatých aplikací. Při implementaci jsem si zároveň vyzkoušel možnosti komunikace za pomoci technologie Windows Comunication Foundation.

7 Použitá literatura

[1] **Elektronický výběr poplatků**

http://www.odbornecasopisy.cz/index.php?id_document=29025

[2] **Vícevrstvá architektura**

http://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/

[3] **Workflow**

http://en.wikipedia.org/wiki/Workflow

[4] **Jazyk BPEL a BPMN**

http://en.wikipedia.org/wiki/Business_Process_Execution_Language

[5] **BPM systém**

http://en.wikipedia.org/wiki/Business_process_management

[6] **.NET Framework 4**

http://msdn.microsoft.com/en-us/library/w0x726c2.aspx

[7] **Windows Comunication Foundation**

http://msdn.microsoft.com/en-us/library/dd456779.aspx

[8] **Windows Presentation Foundation**

http://msdn.microsoft.com/en-us/library/aa970268.aspx

[9] **Silverlight**

http://msdn.microsoft.com/en-us/library/gg130945%28v=vs.95%29.aspx

[10] **Oracle Database**

http://www.oracle.com/us/products/database/index.html

[11] **Jak na dlužníky**

http://proklepni.kletch.cz/jak-na-dluzniky.html

[12] **BizTalk Server**

http://www.microsoft.com/biztalk/en/us/default.aspx

[13] **K2 blackpearl**

http://www.k2.com/en/blackpearl.aspx

[14] **Workflow Foundation**

http://msdn.microsoft.com/en-us/library/ee342461.aspx

[15] **Oracle documentation**

http://www.oracle.com/pls/db111/homepage

[16] **HostingWCF service**

http://debugmode.net/2010/05/25/wcfconsolevideo/

[17] **VIBlend Silverlight Menu Control (free)**

http://www.viblend.com/products/net/silverlight/controls/silverlight_menu_control.aspx

[18] **Web services in Silverlight**

http://www.silverlight.net/learn/data-networking/network-services-%28soap,-rest-and-more%29/web-services-and-silverlight

Příloha A – Seznam použitých zkratek

ASP - Active Server Pages

CLR - Common Language Runtime

CSS - Cascading Style Sheets

DOM - Document Object Model

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

SQL - Structured Query Language

TCP - Transmission Control Protocol

WCF - Windows Comunication Foundation

WF - Workflow Foundation

WPF - Windows Presentation Foundation

XAML - Extensible Application Markup Language

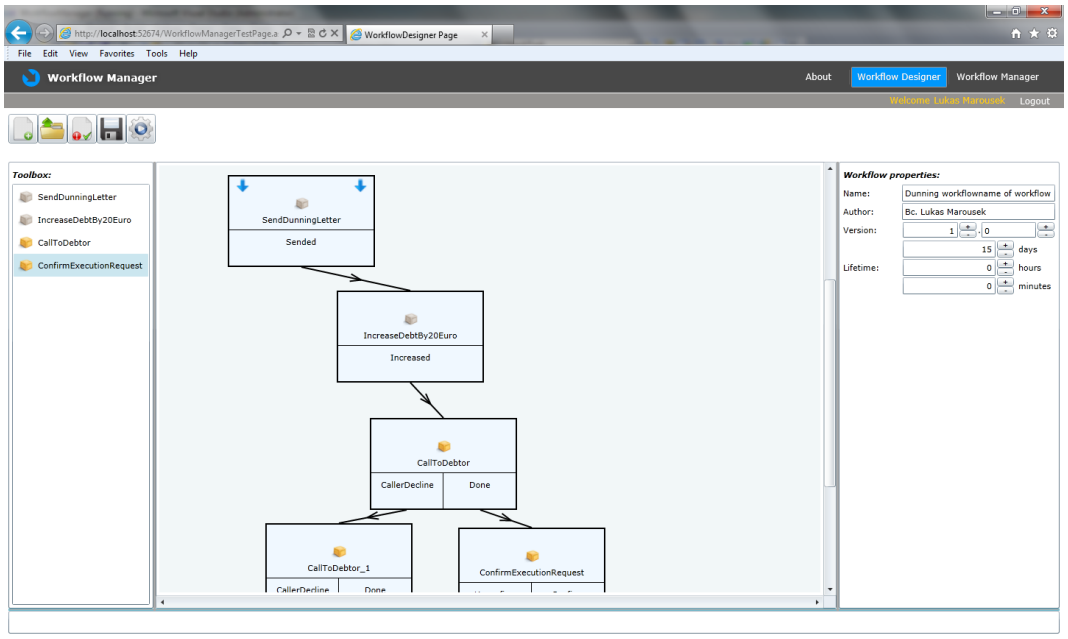
XML - eXtensible Markup Language

XSD - XML Schema Definition

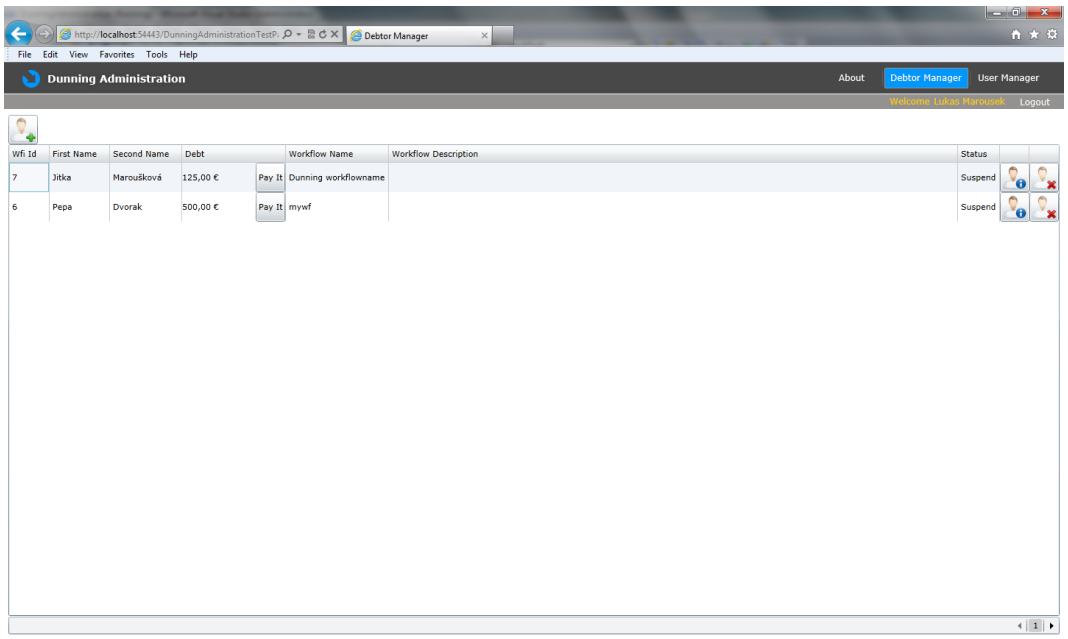
Příloha B – Workflow xml metadata


```xml
<?xml version="1.0" encoding="utf-8"?>
<main>
  <common>
    <author>Lukas Marousek</author>
    <lifetime>P15D</lifetime>
    <workflowName>Debtor's workflow</workflowName>
    <workflowVersion>1.0</workflowVersion>
  </common>
  <workflow>
    <automaticStep id="0" name="SendDunningLetter">
      <transition condition="Sended" target="1" delay="P3D" />
    </automaticStep>
    <automaticStep id="1" name="SendDunningLetter">
      <transition condition="Sended" target="2" delay="P3D" />
    </automaticStep>
    <automaticStep id="2" name="IncreaseDebtBy20Euro">
      <transition condition="Increased" target="3" delay="PT4H" />
    </automaticStep>
    <manualStep id="3" name="CallToDebtor">
      <transition condition="CallerDecline" target="4" delay="P1D" />
      <transition condition="Done" target="5" delay="P3D" />
    </manualStep>
    <manualStep id="4" name="CallToDebtor">
      <transition condition="CallerDecline" target="5" delay="PT30M" />
      <transition condition="Done" target="5" delay="P3D" />
    </manualStep>
    <manualStep id="5" name="ConfirmExecutionRequest">
      <transition condition="Unconfirm" target="3" delay="PT8H" />
    </manualStep>
  </workflow>
</main>
```

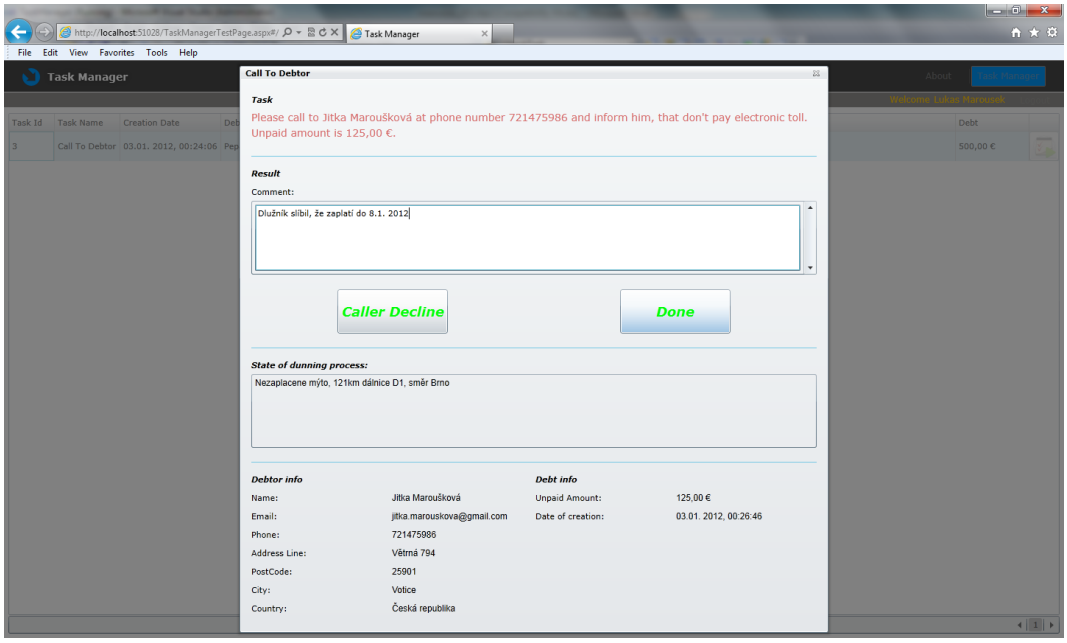
Příloha C – Ukázky klientských aplikací



Obrázek C-1: Aplikace WorkflowManager



Obrázek C-2: Aplikace User + Debtor Manager



Obrázek C-3: Aplikace TaskManager

Příloha D – Obsah přiloženého CD

Na CD jsou následující adresáře:

- /zdrojove_kody/
- /dokumentace/
- /instalacni_prirucka/
- /text_dp/