

# **Jádro informačního systému Skeleton**

The Core of an Information System Skeleton

Bc. Michal Němec

---

Diplomová práce  
2012



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2011/2012

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Michal NĚMEC**  
Osobní číslo: **A10719**  
Studijní program: **N 3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**

Téma práce: **Jádro informačního systému Skeleton**

Zásady pro vypracování:

1. Zpracujte teoretický přehled základů problematiky informačních systémů.
2. Provedte analýzu současného stavu informačního systému Skeleton.
3. Na základě předcházející analýzy navrhnete strukturu nového jádra systému.
4. Naprogramujte nové jádro systému Skeleton a popište použité technologie.
5. Vytvořte klientskou testovací aplikaci a otestujte vlastnosti navrženého jádra.
6. Vypracujte závěrečné zhodnocení a navrhnete směry budoucího vývoje systému.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **NASH, Trey. C 2010: rychlý průvodce novinkami a nejlepšími postupy.** Vyd. 1. Brno: Computer Press, 2010, 624 s. ISBN 978-802-5130-346.
2. **LACKO, L'uboslav. Silverlight: výukový průvodce tvorbou interaktivních aplikací.** Vyd. 1. Brno: Computer Press, 2010, 464 s. ISBN 978-802-5127-162.
3. **AGARWAL, Vidya Vrat a James HUDDLESTON. Databáze v C 2008: průvodce programátora.** Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2009, 424s. ISBN 978-802-5123-096.
4. **PIALORSI, Paolo a Marco RUSSO. Microsoft LINQ: kompletní průvodce programátora.** Vyd. 1. Brno: Computer Press, 2009, 615 s. ISBN 978-802-5127-353.
5. **PECINOVSKÝ, Rudolf. Návrhové vzory.** Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-802-5115-824.
6. **WRÓBLEWSKI, Piotr. Algoritmy: datové struktury a programovací techniky.** Vyd. 1. Překlad Marek Michalek, Bogdan Kiszka. Brno: Computer Press, 2004, 351 s. ISBN 80-251-0343-9.

Vedoucí diplomové práce:

**Ing. Bc. Pavel Vařacha, Ph.D.**

Ústav informatiky a umělé inteligence


Datum zadání diplomové práce:

**24. února 2012**

Termín odevzdání diplomové práce:

**21. května 2012**

Ve Zlíně dne 24. února 2012

  
prof. Ing. Vladimír Vašek, CSc.  
*děkan*



  
doc. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*

## ABSTRAKT

Cílem práce je vytvořit jádro informačního systému, do kterého bude možno zasouvat jednotlivé aplikační moduly a na straně serveru volat jejich funkce za pomoci rozhraní služby WCF. Každý serverový modul bude navenek prezentovat své uzly, které mohou klientské aplikace volat. Nad systémem uzlů bude vybudován systém přístupových práv. Jádro systému bude také obsahovat vlastní servisní uzly, které budou umožňovat správu a dohled nad vnitřním chodem jádra. Jako protistrana k serveru bude vytvořena modulární klientská aplikace. Jejím úkolem bude vytvářet hostitelské prostředí pro jednotlivé zásuvné moduly, které budou obsahovat hlavní logiku interakce s uživatelem.

*Klíčová slova:* Informační systém, Skeleton, WCF, WPF, SOAP

## ABSTRACT

The goal of the thesis is to create an information system kernel. It will be possible to plug in individual application modules into the kernel and call their functions via the WCF-service application interface. On the outside, each server module will represent its nodes, which will then be able to call the client applications. On top of the system of nodes will be an access control system. The system kernel will also contain its own service nodes, which will allow for administration and monitoring of the inner kernel processes. A modular client application will be created to complement the server. Its purpose will be to host an environment for each individual plug-in module. These modules will contain the main logic behind the application user interface.

*Keywords:* Information System, Skeleton, WCF, WPF, SOAP

Informace je informace, není to ani hmota, ani energie. Materialismus, který toto nepřipouští, nemůže přetrvat dnešek (Norbert Winer, 1948).

Děkuji vedoucímu mé diplomové práce panu Ing. Bc. Pavlu Vařachovi, Ph.D. za příkladné vedení a věcné konzultace.

### Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## OBSAH

ÚVOD .....	10
<b>I TEORETICKÁ ČÁST</b> .....	<b>11</b>
<b>1 OBECNÁ PROBLEMATIKA INFORMAČNÍCH SYSTÉMŮ</b> .....	<b>12</b>
1.1 POJEM INFORMAČNÍHO SYSTÉMU .....	12
1.2 SOUČASNÝ SMĚR VÝVOJE .....	12
1.3 PODNIKOVÉ INFORMAČNÍ SYSTÉMY .....	12
1.3.1 Informace .....	12
1.3.2 Základní komponenty .....	13
1.3.3 Proces .....	14
1.3.4 Výpočetní modely .....	15
1.3.5 Distribuované zpracování .....	16
1.3.6 Model klient/server .....	17
1.3.7 Závěry .....	19
<b>2 SOUČASNÝ STAV INFORMAČNÍHO SYSTÉMU SKELETON</b> .....	<b>20</b>
2.1 O INFORMAČNÍM SYSTÉMU SKELETON .....	20
2.2 POPIS STÁVAJÍCÍHO SYSTÉMU .....	21
2.3 NEDOSTATKY STÁVAJÍCÍHO SYSTÉMU .....	21
<b>3 POŽADAVKY NA NOVÝ SYSTÉM</b> .....	<b>22</b>
3.1 NÁVRH STRUKTURY NOVÉHO JÁDRA INFORMAČNÍHO SYSTÉMU SKELETON .....	22
3.2 NÁVRH SERVEROVÉ ČÁSTI .....	23
3.2.1 Rozhraní .....	23
3.2.2 Zásuvné moduly .....	24
3.2.3 Systém uzlů .....	24
3.2.4 Uživatelské účty .....	25
3.2.5 Protokol událostí .....	25
3.3 NÁVRH KLIENTSKÉ ČÁSTI .....	25
3.3.1 Pracovní rámec .....	26
3.3.2 Pracovní prostor .....	26
3.3.3 Dialogy .....	26
3.3.4 Zásuvné moduly .....	27
3.3.5 Návrhový vzor MVVM .....	27
3.4 ROZDĚLENÍ DO SHROMÁŽDĚNÍ (ASSEMBLY) .....	28
3.5 ROZDĚLENÍ DO JMENNÝCH PROSTORŮ (NAME SPACES) .....	29
<b>II PRAKTICKÁ ČÁST</b> .....	<b>31</b>

<b>4</b>	<b>SERVER</b> .....	<b>32</b>
4.1	DIAGRAMY A POPIS ZODPOVĚDNOSTI JEDNOTLIVÝCH TŘÍD.....	32
4.1.1	Třídy služby serveru .....	33
4.2	POPIS FUNKCE ROZHRAŇÍ SLUŽBY.....	35
4.2.1	Komunikační rámec .....	36
4.2.2	Zpracování požadavku.....	37
4.3	VLASTNÍ UZLY SYSTÉMU .....	39
4.4	KONFIGURACE SERVERU .....	39
4.5	IMPLEMENTACE ZÁSUVNÉHO MODULU .....	40
4.5.1	Implementace rozhraní.....	40
4.5.2	Implementační doporučení a pravidla.....	41
<b>5</b>	<b>KLIENT</b> .....	<b>43</b>
5.1	ÚVOD DO TECHNOLOGIE WPF.....	43
5.2	DIAGRAMY A POPIS ZODPOVĚDNOSTI JEDNOTLIVÝCH TŘÍD.....	44
5.2.1	Třídy pracovního rámce .....	44
5.2.2	Třídy pracovního prostoru.....	45
5.3	UŽIVATELSKÉ ROZHRAŇÍ.....	46
5.3.1	Pracovní rámec .....	46
5.3.2	Pracovní prostor.....	47
5.3.3	Rámy pracovního prostoru.....	47
5.3.4	Přiřazení view k modelu.....	48
5.3.5	Dialogy pracovního prostoru .....	49
5.4	KONFIGURACE KLIENTA .....	50
5.5	IMPLEMENTACE ZÁSUVNÉHO MODULU .....	51
5.5.1	Implementace rozhraní.....	51
5.5.2	Implementační doporučení a pravidla.....	52
5.6	OBSLUHA KLIENTSKÉ APLIKACE.....	52
<b>6</b>	<b>TESTOVACÍ APLIKACE ROZHRAŇÍ SLUŽBY SERVERU</b> .....	<b>56</b>
<b>7</b>	<b>MOŽNOSTI VYUŽITÍ ROZHRAŇÍ NA JINÝCH PLATFORMÁCH</b> ..	<b>58</b>
<b>8</b>	<b>SHRNUTÍ VÝSLEDNÉ IMPLEMENTACE</b> .....	<b>60</b>
8.1	ZHODNOCENÍ VÝSLEDNÉ IMPLEMENTACE .....	61
8.2	BUDOUCÍ SMĚR VÝVOJE SYSTÉMU.....	61
<b>9</b>	<b>INSTALACE A SPUŠTĚNÍ APLIKACE</b> .....	<b>62</b>
9.1	POŽADAVKY NA SYSTÉM .....	63
	<b>ZÁVĚR</b> .....	<b>64</b>
	<b>SEZNAM POUŽITÉ LITERATURY</b> .....	<b>67</b>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....	68
SEZNAM OBRÁZKŮ .....	70
SEZNAM TABULEK .....	71
SEZNAM PŘÍLOH .....	72

## ÚVOD

Tato diplomová práce se zabývá návrhem nového jádra informačního systému Skeleton. Skeleton je registrovaná značka firmy FS Software s. r. o.<sup>1)</sup>, která se zabývá od roku 1990 vývojem a správou software zaměřeným na dopravní společnosti a přepravu osob. Firma nejprve fungovala jako sdružení fyzických osob a v roce 1997 přešla do právní formy s. r. o. S touto firmou spolupracuji již od roku 2005 a nyní jsem byl pověřen zpracováním nového konceptu na informační systém Skeleton z důvodů dnes již nevyhovujících parametrů tohoto systému. Taktéž mi bylo firmou umožněno zpracovat tuto problematiku formou diplomové práce. V současné době je informační systém Skeleton tvořen mnoha samostatnými moduly (aplikacemi), řešícími jednotlivé problémy z oblasti přepravy osob, a to jak z pohledu přepravce tak i zákazníka. Z důvodu, že celý systém je v dnešní době postaven již na nevyhovujících platformách, vznikl požadavek na co nejefektivnější přechod na nové platformy, které umožní další rozvoj software zejména v oblasti internetu a mobilních aplikací.

Jednotlivé moduly stávajícího informačního systému Skeleton, jsou vytvořeny jako samostatné aplikace s dvouvrstvou architekturou typu klient-server. Server zde zastává relační databáze, kde je v podobě uložených procedur částečně implementována takzvaná obchodní logika a zbylá, větší část této obchodní logiky a celé uživatelské rozhraní je zapouzdřeno v samostatném spustitelném souboru, který tak tvoří tlustého klienta. Většina modulů používá relační databázi Firebird a v menší míře MSSQL. U systému Firebird také hrozí zneužití kódu v uložených procedurách neoprávněnými osobami a z tohoto důvodu je potřeba tuto obchodní logiku přesunout do jiných vrstev systému. Tyto uvedené skutečnosti, jsou tak jen dalšími oprávněnými důvody k přechodu na novější platformy a vícevrstvé architektury.

Tato diplomová práce se zabývá, jak bylo uvedeno, návrhem nového jádra informačního systému Skeleton. Jádrem v tomto novém navrhovaném systému se rozumí jednotné prostředí s hostitelskými funkcemi pro jednotlivé moduly toho systému. V první, teoretické části práce, bude obecně charakterizována problematika informačních systémů, poté budou uvedeny hlavní nedostatky stávajícího provedení informačního systému Skeleton a v neposlední řadě bude provedena analýza a návrh nového jádra informačního systému Skeleton. V druhé, praktické části této práce, bude provedena implementace návrhu serverové i klientské části a budou popsány všechny části a vrstvy navrhovaného systému. V závěru této diplomové práce bude její zhodnocení a vyjádření budoucího směru vývoje informačního systému Skeleton.

---

<sup>1)</sup>Další informace o firmě je možné nalézt na stránkách <http://www.fssoftware.cz>

# I. TEORETICKÁ ČÁST

## 1 OBECNÁ PROBLEMATIKA INFORMAČNÍCH SYSTÉMŮ

### 1.1 Pojem informačního systému

Přesná definice informačního systému neexistuje a nelze ji ani jednoduše vytvořit. Informační systém (zkráceně IS) lze však charakterizovat jako soubor lidí, technologických prostředků a metod, které zabezpečují sběr, zpracování a uchování dat za účelem tvorby prezentace informací pro potřeby uživatelů.

### 1.2 Současný směr vývoje

Zatím, co v minulosti byly informační systémy zaměřeny spíše na sběr a uchování dat, v dnešní době se hlavně věnují zpracování a prezentaci dat. Sběr a uchování dat, stejně tak i jejich přenos se dnes téměř považuje za samozřejmost. Dále je nutno podotknout, že i data nasbírané v minulosti (mnohdy i desítky let nazpět) mají dnes velký význam. Umíme je dnes mnohem kvalitněji zpracovat a prezentovat, a tím poskytnou uživateli nový pohled a pochopení. Dnešní prezentace dat se čím dál více přesouvá ze samostatných pracovních stanic na internet, a tím se stává dostupnější pro širší spektrum uživatelů v privátních i veřejných sítích. Dostupnost pro více uživatelů a pro jejich různorodé a někdy i protichůdné požadavky přinesla další škálování informačních systémů z hlediska zpracování a prezentace dat. Slovem *Dostupnost* a čím dál častěji používaným slovním spojením *Vysoká dostupnost* se dnes jako zaklínadlem označují právě systémy sloužící velkému okruhu uživatelů. Škálováním z hlediska prezentace dat se pak rozumí vytváření různých pohledů na data podle účelu jejich použití různými skupinami uživatelů. Z toho dále vyplývá potřeba dělit či sdružovat uživatele do skupin. Současný směr vývoje informačních systémů tak jde jednoznačně směrem k uspokojení požadavků co nejširšího okruhu uživatelů. Od nasazení informačního systému se také zároveň očekává ekonomický přínos.

### 1.3 Podnikové informační systémy

Na informatiku se můžeme dívat na zcela obecné úrovni a na úrovni jejích aplikací v určitém oboru lidské činnosti, tj. na aplikovanou informatiku, a konkrétněji na informatiku v podnikovém prostředí, tedy na podnikovou informatiku. Informatika je založena na širokém spektru technických, programových a komunikačních prostředků, resp. technologické infrastruktury. Text v této kapitole a podkapitolách této kapitoly včetně obrázků byl čerpán ze zdroje [7].

#### 1.3.1 Informace

Tak, jak se vyvíjela informatika, vyvíjel se i obsah pojmu informace. Pojem „**informace**“ používáme intuitivně v průběhu celého našeho života. Není bez zajímavosti,

že historicky se s tímto pojmem setkáváme již ve středověku – vždy v nejdůležitějších sférách: v obchodě, v soudnictví a v církevním životě (tedy v ideologii). Samostatný výraz informace (z lat. *informatio*, resp. *informace* = dát tvar, formovat, tvořit) je zaznamenán poprvé roku 1274 ve významu souboru aktů, které vedou k prokázání důkazů trestného činu a k odhalení jeho pachatelů. Informační kancelář bývala zřizována při každém větším peněžním ústavu a podávala důvěrná sdělení o finančním stavu jednotlivých obchodníků.

Dnes se setkáváme s různým chápáním pojmu informace. V pojetí podnikové informatiky je podstatné, že informace je článkem zpracovatelského řetězce. V tomto kontextu se data označují jako „surovina“ pro přípravu informací.

### 1.3.2 Základní komponenty

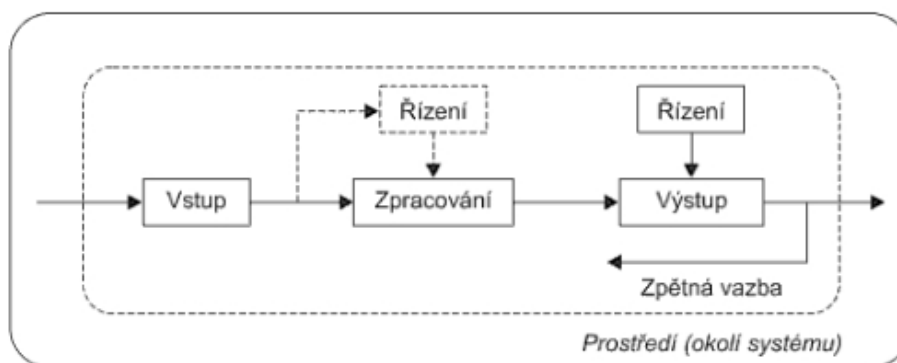
Prvky informačního systému můžeme seskupit do tří základních komponent:

- **vstup** – zahrnuje prvky, umožňující zachytit informační a další vstupy, které mají být předmětem zpracování, případně vstupy vzájemně propojit;
- **zpracování** – zahrnuje prvky, které zajišťují transformaci vstupů do požadovaného výstupu;
- **výstup** – představuje prvky, které jsou schopny přenést informační a další výstupy k jeho příjemci (uživateli).

Takový systém je pak rozšířen o komponenty, které zajišťují jeho řízení a zpětnou vazbu.

Komponenta řízení zahrnuje následující aktivity – nastavení standardů zpracování, měření vyhovění standardům a vyvolání akcí vedoucích k minimalizaci odchylek od standardů. Řízení lze provádět na výstupu a v závislosti na výsledku lze korigovat vstupy a zpracování, aby bylo vyhověno definovaným standardům.

Zpětná vazba zastupuje mechanismus, kterým na základě vyhodnocení ovlivňuje budoucí vstup do zpracování. Může také ovlivňovat zpracování samotné anebo jeho řízení. Mechanismus zpětné vazby je základem systémů pro podporu rozhodování, kdy výstup (tj. rozhodnutí) koriguje v budoucnu vstup do procesu, který vede k dalšímu rozhodnutí. Obr. 1 ukazuje schématicky vazby mezi komponentami informačního systému s informační zpětnou vazbou. Informační systém existuje v určitém prostředí a je subsystém systému jiného.



Obr. 1. Komponenty informačního systému

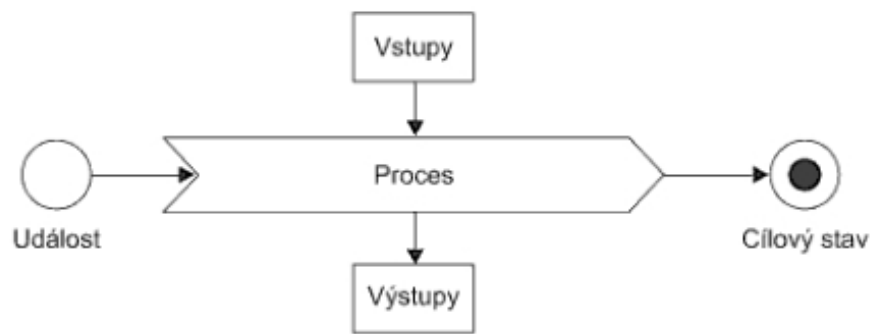
Obvykle se setkáváme s těmito informačními systémy:

- s neformálním informačním systémem, který je reprezentován výměnou i zpracováním informací lidmi a kdy vedle mluvy využíváme i další komunikační techniky (mimika, gesta apod.);
- s formálním informačním systémem, který je založen na formalizovaných pracovních a informačních tocích, realizovaných na základě popsaných politik, cílů, strategií, pravidel a předpisů;
- s informačním systémem založeným na počítačích.

Předmětem našeho dalšího zájmu již bude pouze informační systém, který je založen na počítačích. Jak jsme uvedli výše, prvky tohoto systému tvoří obecně lidé, vhodné nástroje a metody. Nástroji budeme rozumět technické prostředky (hardware), metody pak tvoří programové vybavení (software). Navíc doplníme systém o data, jakožto v minulosti zaznamenaná fakta. Hardware a software souhrnně označujeme pojmem *technologie*, a protože jsou orientovány na zpracování informací, používá se pro ně termín *informační technologie*. Vzdávající akcent na zajištění komunikace v systému vede k rozšíření pojmu *informační technologie* na označení *informační a komunikační technologie* (ICT – Information and Communication Technology). ICT a data dále chápeme jako *informatické zdroje*.

### 1.3.3 Proces

Proces je definován jako soubor vzájemně souvisejících nebo vzájemně působících činností, který přeměňuje vstupy na výstupy. Činnosti využívají zdrojů (lidí, nástrojů, materiálů apod.). Proces může mít více vstupů a také více výstupů. Obr. 2 schématicky ukazuje model procesu.



Obr. 2. Model procesu

Proces je spouštěn definovanou událostí. Ty mohou být v praxi velmi různorodé. Jejich základní typy jsou:

- **vstup** (informací, lidí, materiálu) do podniku, například příchod objednávky od zákazníka spustí proces jejího vyřízení, příchod dodávky materiálu od dodavatele s dodacím listem spustí proces přijetí dodávky apod.;
- **čas**, časová událost – spouští události na základě stanoveného data nebo časového okamžiku, například každé pondělí se spouští proces zpracování přehledu ukončených a rozpracovaných zakázek;
- **interní potřeba změny**, například vznik potřeby inovace produktu nebo služby (nápad některého z pracovníků), požadavek na změnu projektu, dokumentu;
- **výjimečný stav** – může mít povahu výpadku nebo poruchy, příkladem může být porucha výrobní linky spouštějící proces řízení oprav nebo náhradního provozu.

Informační a hmotně-energetické vstupy procesu představují všechny vstupy do procesu na jeho začátku, případně v jeho průběhu. Některý z těchto vstupů může být spojen s jeho spouštěcí událostí. Příkladem může být materiál u výrobního procesu, objednávka apod. Výstupy procesu mají obdobnou povahu jako vstupy, tj. jsou informační a hmotně-energetické. Příkladem může být hotový výrobek, odeslané zboží zákazníkovi apod. Proces pak končí definovaným cílovým stavem, například výrobek byl dokončen, objednávka byla realizována apod.

S procesem je spojena celá řada charakteristik. Jedná se především o formulaci jeho cíle a účelu, tj. odpovědi na otázku, proč vlastně probíhá, jaké jsou pro něj důvody.

#### 1.3.4 Výpočetní modely

Výpočetní model je model, kterým charakterizujeme zpracování aplikací informačního systému a který zároveň formuluje principiální požadavky na technologie. Historicky

bylo použití konkrétního výpočetního modelu v zásadě předurčeno aktuální úrovní a možnostmi technologií.

Výpočetní modely se postupně vyvíjely od plně centralizovaného systému přes decentralizovaný systém až po distribuovaný systém. Za centralizovaný systém považujeme takový systém, jehož komponenty jsou omezeny na jedno místo. U těchto systémů se veškeré zpracování provádí na jednom místě či zařízení. Za decentralizovaný systém považujeme takový systém, jehož komponenty jsou na různých místech a mezi jeho komponentami neexistuje žádná vzájemná koordinace anebo je pouze částečná. Distribuovaný systém je takový systém, jehož komponenty jsou autonomní mechanismy, které koordinují svoje zpracování prostřednictvím nějakého globálního mechanismu a znalostí, přičemž ale disponují také lokálními mechanismy a znalostmi. Komponenta umožňuje samostatnou práci, ale to, kdy má práci vykonat a jak, je dáno požadavky a stavem jiných komponent.

V současné době v IS/ICT využíváme pozitivních vlastností jednotlivých modelů, které jsou postupně zdokonalovány, a nelze říci, že by se některý z modelů dnes vůbec neuzíval.

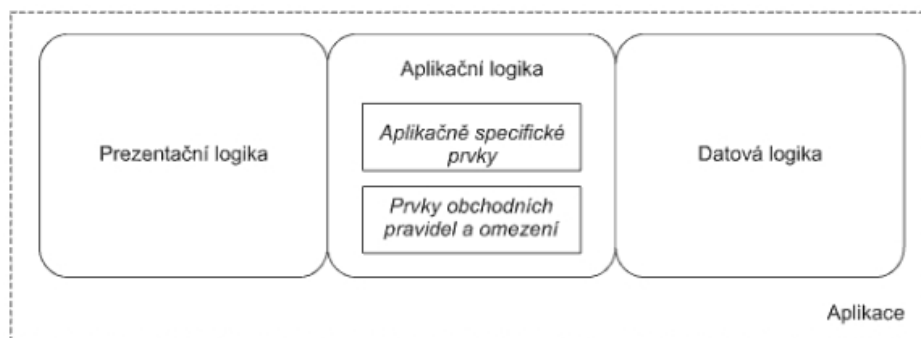
### 1.3.5 Distribuované zpracování

Distribuované zpracování aplikací vzniklo na základě využití všech výhod předešlých modelů (centralizovaného a decentralizovaného). Klíčový princip nového modelu lze charakterizovat následovně: „data by se měla zpracovávat především tam, kde jsou k dispozici“. To znamená, že pokud jsou data uložena na nějakém vzdáleném počítači, tak jejich zpracování by mělo proběhnout na tomto počítači a výsledky (např. pouze vybraná data, výsledky výpočtů apod.) by měly být zaslány tomu, kdo o zpracování žádal. Koncept vychází z dalšího rozvoje počítačových sítí a databázových systémů a reaguje na požadavky vytvořit kooperativní distribuované zpracování. U tohoto konceptu je komponenta, která o něco žádá, označena jako **klient**, a komponenta, která poskytuje službu pojmem **server**.

Aby bylo možné stanovit, co realizuje klient a co bude realizovat server, je aplikace vnitřně členěna na logiky (oblasti), viz obr. 3. Pro libovolnou aplikaci informačního systému je pak definována:

- **Prezentační logika** – obsahuje vstup dat modelů a zajišťuje přípravu vhodného formátu výsledku z závislosti na tom, kdo bude příjemcem výsledků (např. formátuje formulář vstupních dat dle příslušného prostředí uživatele, odpověď na požadavek, zprávu jinému programu apod.).

- **Aplikační logika** – realizuje provedení zpracování věcné (obchodní) logiky, neboli tzv. „byznys funkcí“, pro kterou byla aplikace sestavena.
- **Datová logika** – realizuje operace s daty, které vyžaduje konkrétní aplikační logika. Zpravidla to je kontrola, zda jsou data k dispozici a v potřebném tvaru (např. pokud jsou data v nevhodném pořadí či formátu, zajistí jejich vhodnou transformaci), a dále to je realizace čtení a zápisu dat.



Obr. 3. Logiky aplikace

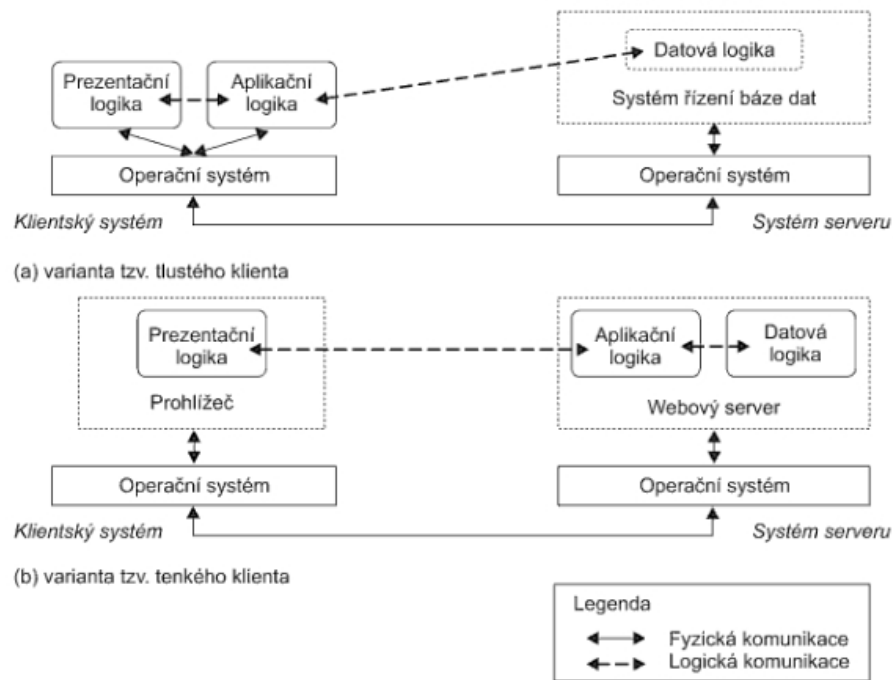
### 1.3.6 Model klient/server

Klient/server je model, kterým charakterizujeme distribuované zpracování aplikací informačního systému. Popisuje komunikaci mezi programovými komponentami aplikace (prezentační, aplikační a datovou logikou) a způsob, jak je zpracování rozděleno (distribuováno) v počítačovém prostředí.

Způsoby rozmístění zpracování byly popsány řadou modelů jejichž praktická implementace vede k tzv. dvouúrovňovému, tříúrovňovému a n-úrovňovému modelu klient/server.

U dvouúrovňového modelu je zpracování aplikace mezi klientským systémem a systémem serveru. Klientským systémem rozumíme zpravidla prostředí osobního počítače uživatele, které umožní spustit a provozovat příslušnou část aplikace. Systémem serveru rozumíme typicky systém řízení báze dat, ale také různé servery spojené s prostředím internetu (webový server, server elektronické pošty apod.). Podle toho, jak je rozděleno zpracování aplikace mezi klientský systém a systém serveru, rozlišujeme variantu tzv. tlustého klienta a variantu tenkého klienta, viz obr. 4.

Varianta tlustého klienta byla prvním implementovaným modelem distribuovaného zpracování aplikací. Umožnila efektivní sdílení dat, které v modelu zajišťuje systém řízení báze dat. Zároveň mezi serverem a klientem jsou přenášena pouze data nezbytná pro zpracování aplikační logiky. Za nevýhody této varianty lze považovat to, že klade

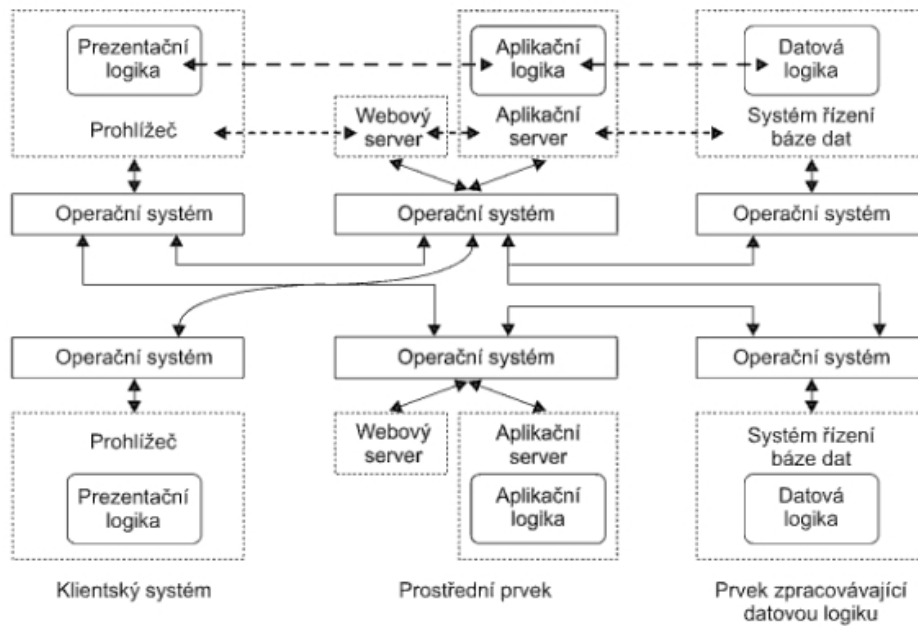


Obr. 4. Dvouúrovňový model klient/server

vyšší nároky na technickou konfiguraci klientského systému, a také to, že správce musí část aplikace instalovat do každého klientského systému.

Varianta tenkého klienta souvisí jednak s dalším rozvojem databázových systémů, které nyní umožňují zpracování aplikační logiky (tzv. uložené procedury databázového systému), a jednak s nástupem technologií internetu. Serverový systém zajišťuje kompletní zpracování aplikační a datové logiky a připravuje nezbytná data pro prezentační logiku. Klientský systém zajišťuje zpracování prezentační logiky, tedy obsluhu vstupu od uživatele a zobrazení uživatelského rozhraní aplikace s daty, které dodá aplikační logika. Výhodou této varianty je snížení nároků na správu aplikace. Nevýhodou pak je to, že server se vedle zajištění datové logiky musí starat i o provádění aplikační logiky.

Reakcí na nedostatky variant dvouúrovňového modelu je vznik tříúrovňového modelu klient/server, viz obr. 5. V tomto tříúrovňovém modelu je aplikační logika zpracována novým prvkem systému. Tento prvek vystupuje vůči klientskému systému, který zpracovává prezentační logiku, jako server. Zároveň vůči systému, který zpracovává datovou logiku, vystupuje jako klient.



Obr. 5. Tříúrovňový model klient/server

### 1.3.7 Závěry

Technologie představují širokou škálu rychle se rozvíjejících technických prostředků, programového vybavení a prostředků telekomunikačních a počítačových sítí. Konkrétní výběr, uspořádání a nasazení technologií závisí na požadavcích organizace, které jsou implementovány jako aplikace podnikové informatiky.

## 2 SOUČASNÝ STAV INFORMAČNÍHO SYSTÉMU SKELETON

V této kapitole bude představen současný informační systém Skeleton, budou popsány jeho části a uvedeny jeho stávající nedostatky.

### 2.1 O informačním systému Skeleton

Informační systém Skeleton je produktem firmy FS Software s. r. o., která je na trhu od roku 1997. Firma se od počátku zabývá tvorbou informačních systémů pro dopravní podniky a subjekty s podobným zaměřením. V prvotní fázi byl software produkován na platformě operačního systému MS-DOS a v průběhu času přešel na platformu MS Windows. Firma tak má letité zkušenosti s tímto oborem dopravy a plně je využívá k dalšímu rozvoji.

Informační systém Skeleton se zabývá dopravou v oblasti přepravy osob. Poskytuje řadu aplikačních modulů spojených s touto problematikou a je využíván dopravními podniky různých měst v České Republice a na Slovensku. Jednotlivé aplikační moduly tohoto informačního systému, které jsou podrobněji popsány v příloze P I, se zabývají vždy jen nějakou logickou částí z této problematiky a tím jako celek tvoří systém. Jednotlivé aplikační moduly také musí respektovat požadavky a zvyklosti konkrétních dopravních podniků a také požadavky našich i zahraničních zákonů. Firma dále poskytuje k systému technickou podporu, servis a pravidelné školení.

Tento systém je plně modulární a je sestavován pro každého zákazníka individuálně dle jeho možností a požadavků. Po nasazení bývá systém po nějakou dobu v takzvaném záběhovém režimu, kdy je aktivně přizpůsobován těmto požadavkům a poznatkům daného zákazníka. Zákazníci se tak velkou měrou podílí na směru vývoje tohoto informačního systému.

Problematika zaměřená na dopravní podniky je specifická zejména vtom, že skýtá mnoho typů dopravních prostředků, mnoho zákazníků využívající možnosti úsekově kombinované dopravy a spoustu zákonných požadavků na řízení dopravy v oblasti přepravy osob. Systém musí umět vytvářet dlouhodobé plány a stejně tak umět pružně reagovat na neplánované problémy, jako jsou například výluky a havárie jakéhokoliv druhu.

## 2.2 Popis stávajícího systému

Jak již bylo popsáno, současný systém Skeleton se skládá z několika samostatných aplikací více či méně spolupracujících. V jednotlivých aplikacích je implementována převážná část logiky systému a veškeré grafické uživatelské rozhraní. Tyto aplikace tak tvoří tlustého klienta. Jako server zde převážně slouží přímo databáze<sup>2)</sup>, ve které je ve formě uložených procedur také částečně implementována logika systému. V současné době také zároveň probíhá migrace aplikací z platformy Delphi 6<sup>3)</sup> na Delphi 2007 a Delphi XE2. Tato migrace je prováděna z důvodu částečného oživení a umožnění pružnější údržby stávajícího systému. Migrace má také hlavně umožnit snadnější postupné přepojení stávajících aplikací na nové serverové rozhraní služby WCF<sup>4)</sup> nového systému Skeleton.

## 2.3 Nedostatky stávajícího systému

Nedostatky stávajícího systému spočívají v používání již zastaralých vývojových prostředků a s tím související nemožnost dalšího dynamického rozvoje systému. Dalším nedostatkem je současná architektura systému, která již neodpovídá současným požadavkům trhu. Zejména se jedná o nemožnost publikování jednotlivých modulů či částí modulů na internet nebo do mobilních aplikací. Současně začíná být problémem nutné sdílení kódů mezi jednotlivými moduly, které se většinou provádí pouhým kopírováním daných jednotek a jejich případnou úpravou. Tento neduh je hlavně způsoben právě použitím již zmíněných zastaralých vývojových prostředků. Posledním významným nedostatkem je nepřítomnost vícevrstvé architektury systému, která způsobuje obtížnou údržbu tak rozsáhlého systému.

---

<sup>2)</sup>Firebird, MS SQL

<sup>3)</sup>Delphi je integrované grafické vývojové prostředí firmy Borland

<sup>4)</sup>Windows Communication Foundation

### 3 POŽADAVKY NA NOVÝ SYSTÉM

Cílem je vyvinout nový vícevrstvý modulární systém, tak aby nynější jednotlivé aplikace, které tvoří současný systém, byly v novém systému zastoupeny pouze jako moduly. Nový systém Skeleton, respektive jádro nového systému Skeleton tak bude sloužit jako hostitelské prostředí pro jednotlivé moduly a bude jim poskytovat potřebné funkce pro jejich hostování. Systém bude rozdělen na dvě základní části, serverovou a klientskou. Serverová část bude tvořena rozhraním implementovaným pomocí technologie WCF a bude umožňovat spouštět funkce definované v jednotlivých zásuvných modulech. Moduly hostované v serverové části, budou zejména obsahovat takzvanou business logiku a budou tak tvořit výkonnou část systému. Serverové rozhraní bude také předpokládat i možné kontrolované připojení jiných systémů třetích stran. Klientská část bude vytvořena jako tenká aplikace, která umožní hostování jednotlivých modulů s grafickým uživatelským rozhraním. Tato tenká klientská aplikace bude poskytovat uživatelům nejnútnejší interakci se systémem a zprostředkovávat interakci definovanou v konkrétních zásuvných modulech. Celý systém bude plně konfigurovatelný, tak aby umožnil sestavení systému z daných modulů pro konkrétního zákazníka.

Implementace bude provedena na technologiích firmy Microsoft, konkrétně ve Visual Studiu 2010 a jazyku C#. Klientská aplikace bude vytvořena jako spustitelný soubor nebo jako XBAP<sup>5)</sup> aplikace. Serverová část bude vytvořena v podobě knihovny, kterou bude možno cílově hostovat jako službu systému. Pro testovací účely však bude použito hostování v konzolové aplikaci. Použití jednotlivých modulů bude možno konfigurovat v konfiguračním souboru klientské či serverové aplikace. Nově vytvářený systém předpokládá spouštění výhradně na novějších verzích operačních systémů Microsoft.

Toto je pouze část požadavků kladených na nový informační systém Skeleton, která je náplní této diplomové práce. Úplná *Úvodní studie* je v příloze P II. Následný vývoj, který bude probíhat po ukončení části popisované v této práci, by měl postupně naplňovat další požadavky kladené v této *Úvodní studii*.

#### 3.1 Návrh struktury nového jádra informačního systému Skeleton

V návrhu struktury se budu postupně zabývat jednotlivými částmi a vrstvami systému z různých pohledů implementačního prostředí. Jen pro úplnost připomínám použité implementační prostředí MS Visual Studio 2010, prostředí .NET Framework 4 a jazyk C#. Zároveň je také předpokládána čtenářova základní znalost tohoto prostředí.

---

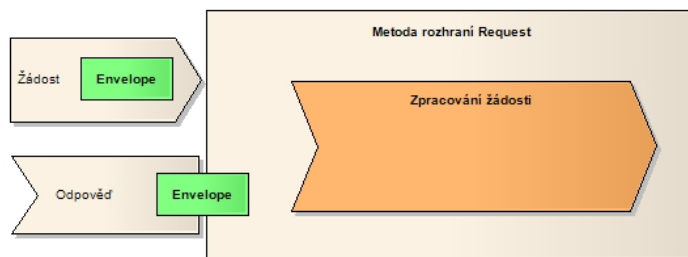
<sup>5)</sup>XAML Browser Application

### 3.2 Návrh serverové části

V návrhu serverové části jsou popsány jednotlivé klíčové prvky systému. Návrh popisuje jakým způsobem by měla být provedena implementace. Tento návrh vychází plně ze zkušeností řešitele s podobnými systémy a komunikačními rozhraními.

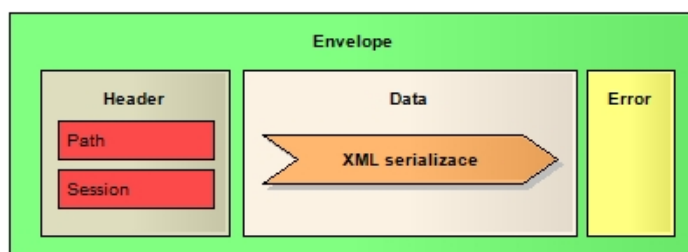
#### 3.2.1 Rozhraní

Pro komunikaci klientských aplikací se serverem bude použito rozhraní WCF. V rozhraní bude implementována pouze jedna metoda *Request*. Dále bude definován komunikační rámec *Envelope*, který tato metoda rozhraní bude jako jediný parametr přebírat a zároveň bude i její návratovou hodnotou. Schéma volání metody *Request* je znázorněno na obr. 6. Komunikační rámec *Envelope* bude obsahovat hlavičku *Header*, datovou složku *Data* a případnou informaci o výskytu chyby *Error* během zpracování požadavku. V hlavičce *Header* bude uvedena cesta *Path* k požadovanému uzlu systému a identifikátor relace *Session*.



Obr. 6. Metoda Request

Datová složka *Data* komunikačního rámce *Envelope* bude typu **string**, do kterého je cílově možné převést jakýkoli jiný typ. Jako primární konvertor dat bude použit serializátor do formátu XML<sup>6)</sup>. Objekty určené pro přenos přes rozhraní tak musí být definovány jako serializovatelné. Schéma komunikačního rámce je znázorněno na obr. 7.



Obr. 7. Komunikační rámec

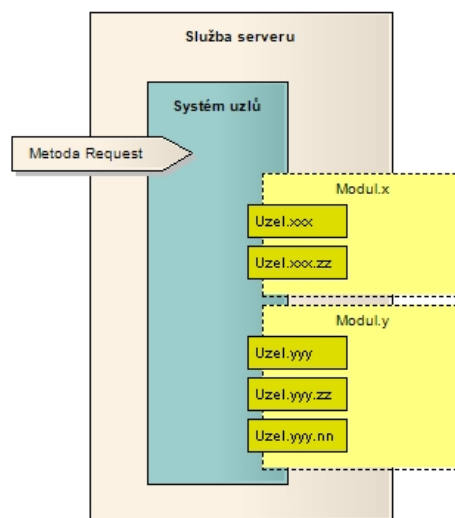
Jakékoli rozhraní dle obecných definic by mělo být co nejužší, což tento návrh maximálně splňuje. Zároveň rozhraní umožní přenos jakéhokoliv objektu, čímž ho tato

<sup>6)</sup>Extensible Markup Language

vlastnost činí univerzálním. Takto navržené rozhraní splňuje i požadavek na robustnost, protože v jeho úzkém hrdle je možno plně kontrolovat komunikaci.

### 3.2.2 Zásuvné moduly

Běhové prostředí služby serveru bude vytvářet hostitelské prostředí pro jednotlivé zásuvné moduly uvedené v konfiguračním souboru. Po inicializaci služby serveru budou nataženy všechny moduly a načteno jejich rozhraní, které bude definováno pomocí atributů u jednotlivých metod tříd modulu. Služba serveru pak umožní volání takto označených metod skrze rozhraní serveru. Veškeré neošetřené výjimečné události vzniklé ve volané metodě, budou zachyceny obslužným systémem služby serveru a vhodným způsobem indikovány klientovi. V jednotlivých zásuvných modulech pak bude implementována veškerá výkonná logika informačního systému. Pro účely této diplomové práce bude vytvořen jeden testovací modul pro ověření správné funkce systému. Schématické znázornění zásuvného modulu v rámci služby serveru je na obr. 8.



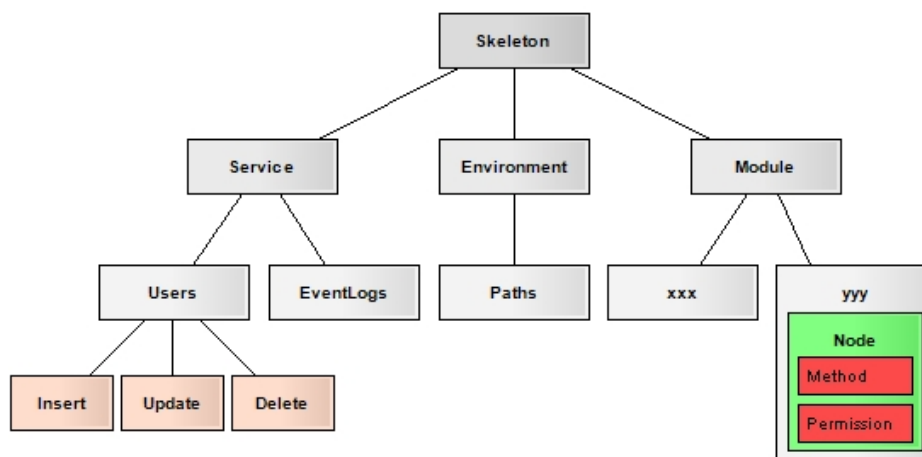
Obr. 8. Zásuvné moduly serveru

### 3.2.3 Systém uzlů

Systém uzlů bude mít podobu prefixového stromu, kde jeho kořen bude představovat vstupní bod do systému uzlů. Strom bude sestavován během načítání zásuvných modulů do systému dle jednotlivých parametrů atributů uvedených u uzlových metod definovaných v těchto zásuvných modulech. Vznikne tak hierarchie, ke které bude možné přiřadit přístupová práva. Pokud přístupové právo uzlu nebude definováno, bude zděděno z nadřazeného uzlu. Příklad stromu je na obr. 9.

Tímto způsobem tak dle obr. 9 vzniknou následující cesty v systému:

Skeleton  
 Skeleton.Service  
 Skeleton.Service.Users  
 Skeleton.Service.EventLog  
 Skeleton.Service.Environment  
 :



Obr. 9. Strom uzlů

Tyto cesty bude následně možné dotazovat pomocí metody *Request* v rozhraní služby serveru. Při přijetí požadavku na daný uzel, se tento vyhledá v prefixovém stromu a po ověření přístupového práva se spustí obslužná metoda uzlu.

### 3.2.4 Uživatelské účty

Pro řízení přístup k jednotlivým uzlům systému bude navržen systém uživatelských účtů a práv. Budou vytvořeny potřebné uzly dostupné přes rozhraní serverové služby, pomocí kterých bude možné uživatelské účty a přístupové práva spravovat.

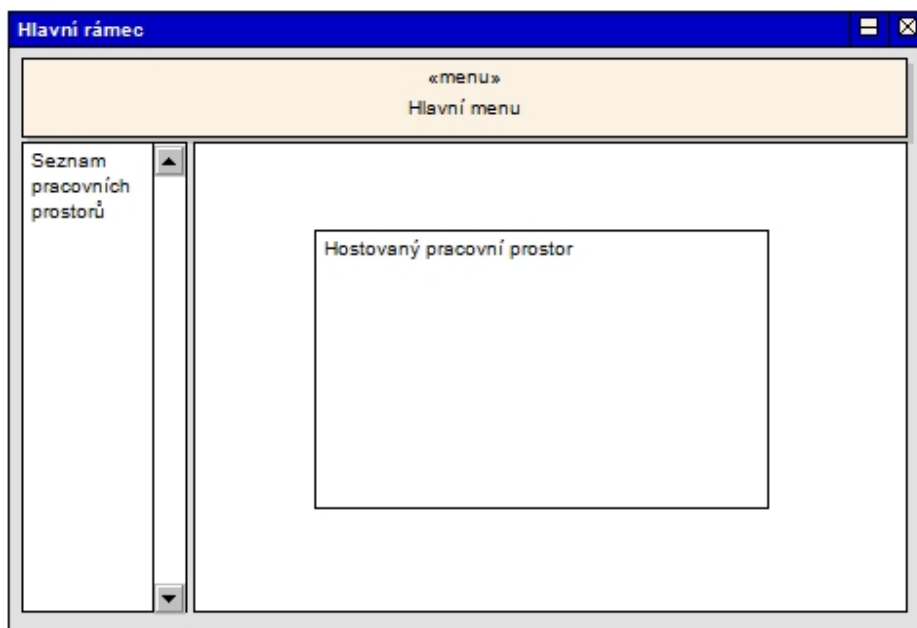
### 3.2.5 Protokol událostí

Ve službě serveru bude vytvořen protokol událostí, do kterého budou zaznamenávány všechny důležité události. Budou vytvořeny potřebné uzly dostupné přes rozhraní serverové služby, pomocí kterých bude možné protokol událostí vyčítat.

## 3.3 Návrh klientské části

Jako protistrana k serveru bude navržena klientská aplikace, která bude obsahovat jen nejnútnejší logiku potřebnou pro interakci s uživatelem. Klientská aplikace bude taktéž navržena modulárním systémem a bude poskytovat hostitelské prostředí pro jednotlivé moduly. Pro implementaci uživatelského rozhraní bude použita technologie

WPF<sup>7)</sup>. Na obr. 10 je návrh grafického uživatelského rozhraní s následným popisem jednotlivých segmentů.



Obr. 10. Hlavní pracovní rámeček

### 3.3.1 Pracovní rámeček

Pracovní rámeček bude tvořen hlavní nabídkou, seznamem aktivních pracovních prostorů a aktuálně vybraným pracovním prostorem. Seznam aktivních pracovních prostorů a hlavní nabídka bude zobrazena až po přihlášení uživatele do systému.

### 3.3.2 Pracovní prostor

Pracovní prostor bude zapouzdřovat uživatelské rozhraní implementované v zásuvném modulu a bude typicky vytvářen na příkaz z hlavní nabídky pracovního rámečku. Pracovní rámeček se bude chovat jako hostitelské prostředí pro tyto pracovní prostory a bude nad nimi držet kompletní správu. Pro implementaci pracovních prostorů bude používán návrhový vzor MVVM<sup>8)</sup>.

### 3.3.3 Dialogy

Dialog má za úkol přerušit interakci uživatele s pracovním prostorem a vynutit si uživatelskou akci definovanou v dialogu. Dialog, do doby než bude zrušen, vhodným způsobem překryje pracovní prostor, který jej vyvolal a ostatní pracovní prostory blokovat nebude. Tento způsob chování tak vytvoří multimodální pracovní rámeček.

<sup>7)</sup>Windows Presentation Foundation

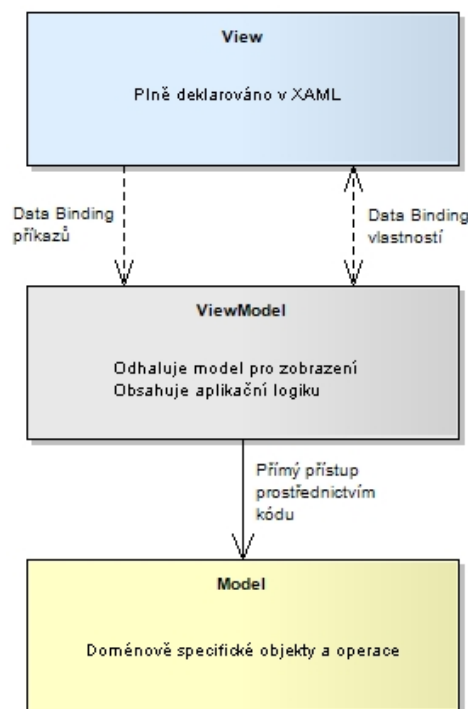
<sup>8)</sup>Model-View-ViewModel

### 3.3.4 Zásuvné moduly

Zásuvný modul klientské strany, obdobně jako modul severu, bude načten do systému v inicializační části pracovního rámce. Seznam zásuvných modulů bude uveden v konfiguračním souboru klientské aplikace a v jednotlivých zásuvných modulech budou pomocí atributů označeny vstupní metody, na základě kterých se v hlavní nabídce pracovního rámce vygenerují příslušné ovládací prvky. Programátor daného zásuvného modulu bude dále zodpovědný za to, co tato vstupní metoda vykoná. Typicky by však měla vytvářet nový pracovní prostor.

### 3.3.5 Návrhový vzor MVVM

Návrhový vzor MVVM (Model-View-ViewModel) je architektonický vzor v softwarovém inženýrství, který vznikl ve firmě Microsoft jako specializace pro používání platformy uživatelského rozhraní WPF (Windows Presentation Foundation) a z velké části je založen na MVC (Model-View-Controller) vzoru. MVVM umožňuje jasné oddělení grafického uživatelského rozhraní zvaného *View* od obchodní logiky zvané jako *Model*. *ViewModel* zde slouží jako prostředník zodpovědný za odhalení datových objektů z *Modelu*. Nespornou výhodou tohoto řešení je oddělení pohledu od modelu a možnost tak vytvořit více jednoduchých pohledů na jeden datový model. Schématické zobrazení spolupráce a funkce jednotlivých vrstev návrhového vzoru MVVM je na obr. 11.



Obr. 11. Návrhový vzor MVVM

### 3.4 Rozdělení do shromáždění (assembly)

Z důvodu potřebného sdílení kódu mezi serverovou a klientskou částí je navrženo s tímto ohledem i rozdělení do shromáždění. Pojem shromáždění znamená fyzické uspořádání do knihoven DLL<sup>9)</sup>. Rozdělení bude provedeno ve třech hlavních částech. Serverová část, část uživatelského rozhraní a modulová část, viz tab. 1. Všechny názvy shromáždění budou začínat prefixem *Skeleton* a jednotlivé části budou odděleny tečkou. Tento systém pojmenování je zvolen s ohledem na doporučení MSDN [8].

Tab. 1. Rozdělení do shromáždění

Shromáždění	Popis
<i>Serverová část</i>	
<b>Skeleton.Service.dll</b>	Výkonný kód služby serveru.
<b>Skeleton.Core.dll</b>	Kód společný pro vrstvu serveru.
<b>Skeleton.Common.dll</b>	Kód společný napříč vrstvami systému.
<i>Část uživatelského rozhraní</i>	
<b>Skeleton.UI.dll</b>	Výkonný kód uživatelského rozhraní.
<b>Skeleton.UI.Core.dll</b>	Kód společný pro vrstvu UI.
<i>Modulová část</i>	
<b>Skeleton.Module.xxx.dll</b>	Výkonný kód modulu xxx.
<b>Skeleton.Module.xxx.UI.dll</b>	Výkonný kód UI modulu xxx.
<b>Skeleton.Module.xxx.Common.dll</b>	Kód společný napříč vrstvami modulů.

Dále následuje podrobnější popis jednotlivých navržených shromáždění.

- **Skeleton.Service.dll** – Shromáždění bude obsahovat výkonný kód služby serveru. Toto shromáždění nesmí být odkazováno žádným jiným shromážděním, kromě vlastní hostitelské aplikace.
- **Skeleton.Core.dll** – Shromáždění bude obsahovat kód společný pro vrstvu serveru. Může být odkazován vlastní službou serveru nebo serverovými moduly.
- **Skeleton.Common.dll** – Shromáždění bude obsahovat kód společný napříč vrstvami systému. Může být odkazován v kterékoliv části systému.
- **Skeleton.UI.dll** – Shromáždění bude obsahovat výkonný kód pracovního rámce uživatelského rozhraní. Toto shromáždění nesmí být odkazováno žádným jiným shromážděním, kromě vlastní hostitelské aplikace.
- **Skeleton.UI.Core.dll** – Shromáždění bude obsahovat kód společný pro vrstvu uživatelského rozhraní. Může být odkazováno vlastní vrstvou pracovního rámce nebo moduly s uživatelským rozhraním.

<sup>9)</sup>Dynamic-link library

- **Skeleton.Module.xxx.dll** – Shromáždění vlastního zásuvného modulu na straně serveru. Toto shromáždění bude zapsáno v konfiguračním souboru serveru a pomocí atributů v něm budou označeny vstupní metody uzlů systému. Znaky *xxx* zde zastupují vlastní název modulu.
- **Skeleton.Module.xxx.UI.dll** – Shromáždění vlastního zásuvného modulu uživatelského rozhraní. Toto shromáždění bude zapsáno v konfiguračním souboru klientské aplikace a pomocí atributů v něm budou označeny vstupní metody příkazů hlavní nabídky systému. Znaky *xxx* zde zastupují vlastní název modulu.
- **Skeleton.Module.xxx.Common.dll** – Shromáždění bude obsahovat kód společný napříč vrstvami systému. Může být odkazováno v kterékoliv části systému. Zejména se předpokládá sdílení datových entit přenášených přes rozhraní služby serveru. Znaky *xxx* zde zastupují vlastní název modulu.

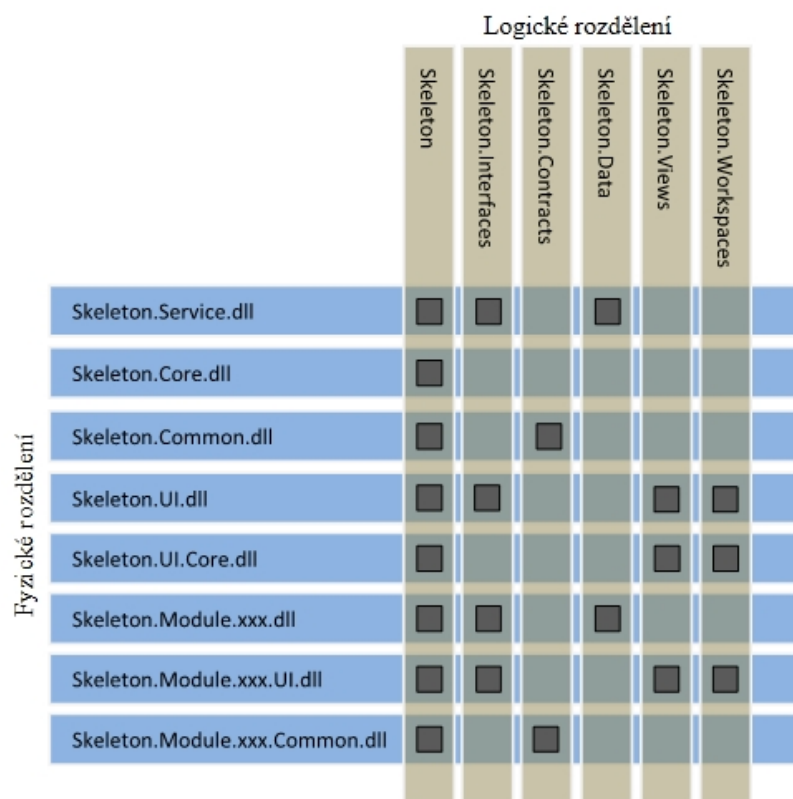
### 3.5 Rozdělení do jmenných prostorů (name spaces)

Jmenné prostory jsou logické uspořádání a měly by být v rámci systému dle doporučení MSDN [8] definovány nezávisle na fyzickém shromáždění. Jednotlivé jmenné prostory se tak rozkládají napříč jednotlivými shromážděními. S ohledem na architekturu systému je navrženo několik základních jmenných prostorů, které by programátoři modulů měly respektovat, viz tab. 2.

Tab. 2. Rozdělení do jmenných prostorů

Jmenný prostor	Popis
<b>Skeleton</b>	Základní třídy systému.
<b>Skeleton.Interfaces</b>	Rozhraní systému.
<b>Skeleton.Contracts</b>	Datové objekty určené pro přenos.
<b>Skeleton.Data</b>	Datové objekty datové vrstvy.
<b>Skeleton.Views</b>	Pohledy na pracovní prostory uživatelského rozhraní.
<b>Skeleton.Workspaces</b>	Pracovní prostory uživatelského rozhraní.

Princip rozdělení do shromáždění a jmenných prostorů je znázorněno na obr. 12. V obrázku jsou použity konkrétní názvy shromáždění a jmenných prostorů tohoto projektu. Vyznačené průsečíky znázorňují použití jednotlivých jmenných prostorů v daném shromáždění.



Obr. 12. Rozdělení do shromáždění a jmenných prostorů

## **II. PRAKTICKÁ ČÁST**

## 4 SERVER

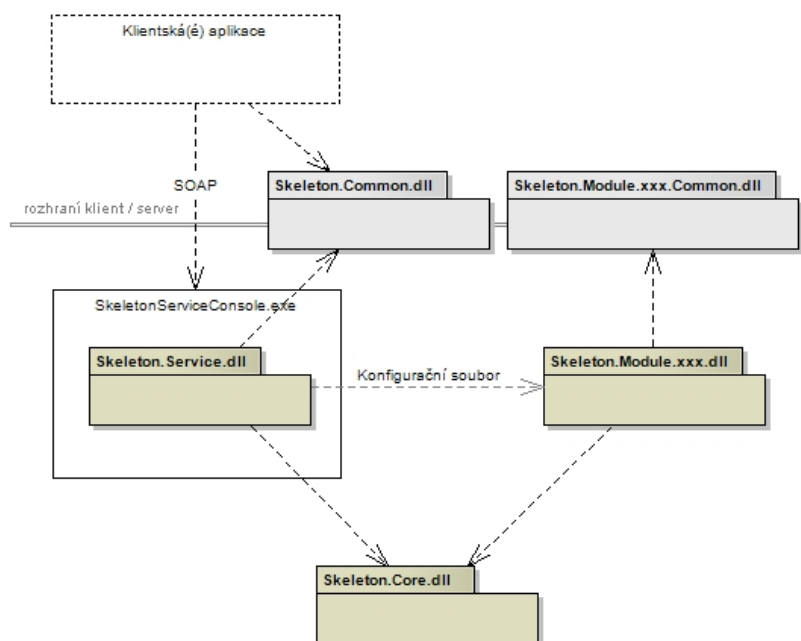
S ohledem na návrh serverové části budou vytvořeny následující knihovny a aplikace. Schématické znázornění závislostí jednotlivých modulů je na obr. 13.

**Skeleton.Service.dll** – Bude obsahovat výkonný kód služby serveru, který nebude nikde jinde sdílen. Pro testovací účely v této diplomové práci, bude tato knihovna zapouzdřena do konzolové aplikace viz obr. 42. V produkčním prostředí pak bude tato knihovna cílově zapouzdřena jako služba systému.

**Skeleton.Core.dll** – Bude obsahovat kód společný pro vrstvu serveru. Serverovou vrstvou se rozumí samotná služba serveru a zásuvné moduly na straně serveru.

**Skeleton.Common.dll** – Bude obsahovat kód společný napříč vrstvami systému. Může být sdílena v jakékoliv části systému.

**SkeletonServiceConsole.exe** – Konzolová aplikace zapouzdřující vlastní službu serveru. V okně konzole se budou online vypisovat potřebné ladící informace.



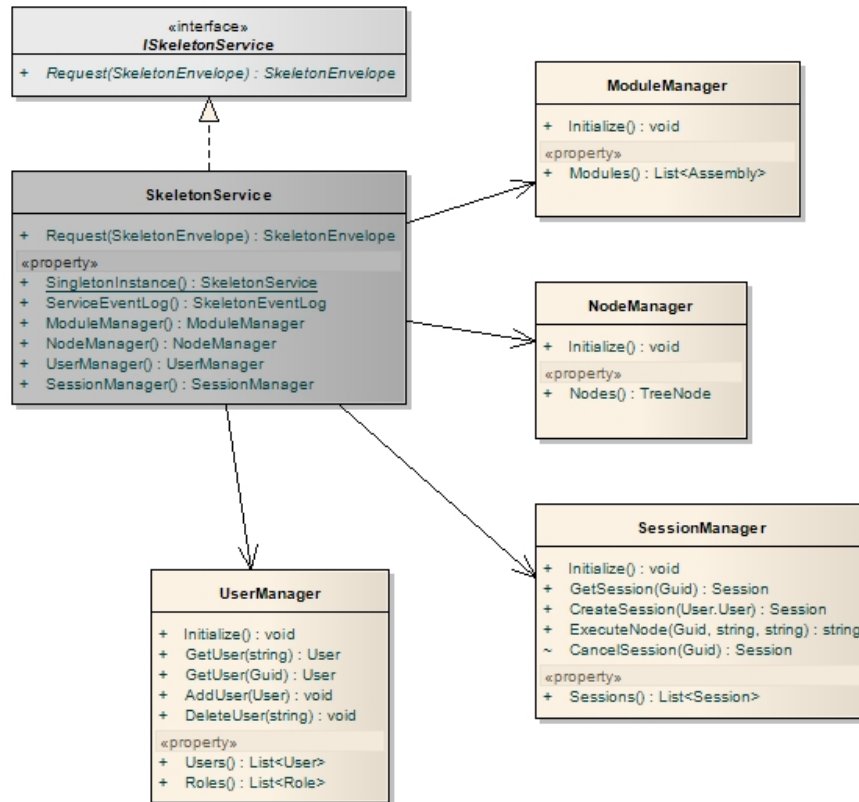
Obr. 13. Závislosti modulů serverové strany

### 4.1 Diagramy a popis zodpovědnosti jednotlivých tříd

V následujících podkapitolách budou popsány jednotlivé zodpovědnosti a funkčnosti tříd služby serveru. V diagramech tříd budou znázorněny jen nejdůležitější součásti systému. Jednotlivé třídy tak ve skutečnosti obsahují mnohem více atributů a operací, které jsou zejména zapotřebí pro správné zapouzdření tříd.

#### 4.1.1 Třídy služby serveru

Následující obr. 14. znázorňuje základní třídy služby serveru, které mají za úkol spravovat čtyři hlavní funkční části.



Obr. 14. Třídy služby serveru

**ISkeletonService** – Rozhraní definující metodu *Request* WCF služby.

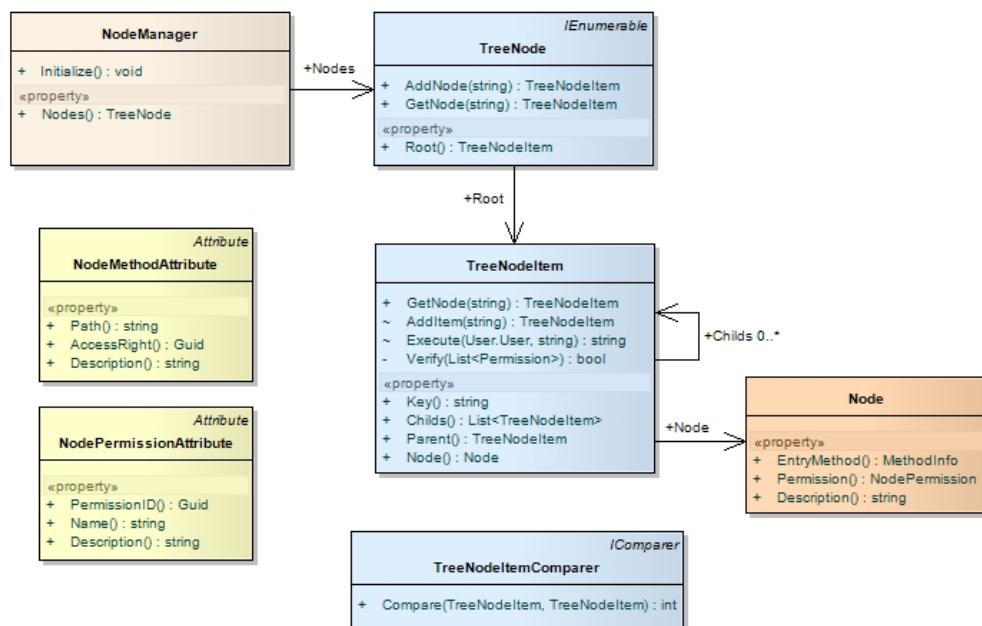
**SkeletonService** – Hlavní třída služby serveru. Třída implementuje rozhraní *ISkeletonService* a společně s tímto rozhraním tvoří vstupní bod služby serveru. Tato třída je zodpovědná za obsluhu rozhraní WCF a za řízení všech tříd správců (třídy označené xxxManager). Při vytváření instance této třídy, zajistí obslužná metoda konstruktoru vytvoření a inicializaci všech tříd správců. Po úspěšné inicializaci je instance třídy připravena přijímat požadavky z rozhraní WCF. V inicializaci je zajištěno načtení celého běhového prostředí, což znamená načtení všech modulů, vytvoření stromu uzlů a načtení uživatelských účtů.

**ModuleManager** – Třída správce modulů. Třída je zodpovědná za načtení všech modulů uvedených v konfiguračním souboru serveru. Načítání modulů probíhá pouze v inicializační metodě třídy. Pokud nastane výjimečná událost v některém kroku načítání modulu, je zaznamenána do listu událostí systému a je pokračováno v načítání dalšího modulu.

**NodeManager** – Třída správce uzlů. Třída je zodpovědná za vytvoření a správu stromu uzlů. Po provedení inicializace správce modulů následně proběhne inicializace správce uzlů. Možné vzniklé výjimečné události jsou zaznamenány do listu událostí systému. Třídy správce uzlů jsou na obr. 15.

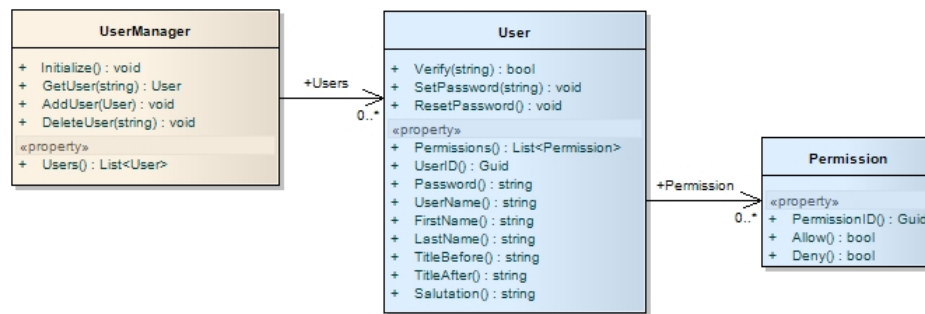
Správce uzlů v inicializační sekci provádí postupné sestavování stromu uzlů z informací, které získá pomocí reflexe postupně z každého načteného zásuvného modulu ve správci modulů. Pomocí reflexe se vyhledají všechny metody tříd, které jsou označeny atributem `NodeMethodAttribute` a na základě vlastností atributu se provede vytvoření a vložení uzlu do stromu. Pokud u metody existuje i atribut `NodePermissionAttribute`, provede se i přiřazení přístupového práva. Třída `TreeNodeItemComparer` slouží k seřazení uzlů na stejné úrovni stromu.

Významnou metodou třídy `TreeNodeItem` je metoda `Execute`, která má za úkol na žádost rozhraní systému spustit obslužnou metodu uzlu. Spouštěcí mechanismus bude podrobněji popsán v kap. 4.2.



Obr. 15. Třídy správce uzlů

**UserManager** – Třída správce uživatelských účtů. Třída je zodpovědná za správu uživatelských účtů systému a v inicializační sekci provádí načtení těchto uživatelských účtů z databáze. Třída dále reaguje na požadavky přidání, odebrání a změnu uživatelského účtu a jeho aktualizaci v databázi. Třídy správce uživatelských účtů jsou na obr. 16. Systém tuto třídu využívá pro ověřování uživatele a uživatelských práv k uzlům systému.



Obr. 16. Třídy správce uživatelů

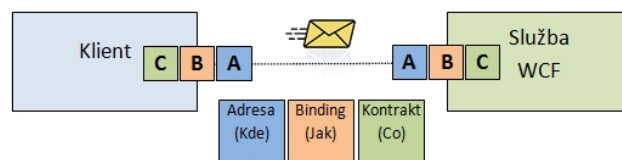
**SessionManager** – Třída správce relací. Třída je zodpovědná za správu jednotlivých přihlášení k systému (relací). Při požadavku přihlášení k systému třída ověří uživatele a vytvoří jedinečný identifikátor relace, pomocí kterého probíhá další komunikace. Pokud k přihlášení do systému nedojde, tak komunikace probíhá jako anonymní. Třídy správce relací jsou na obr. 17. Dále na základě přidělení identity v systému probíhá řízení přístupu k jednotlivým uzlům systému.



Obr. 17. Třídy správce sezení

## 4.2 Popis funkce rozhraní služby

Komunikační rozhraní služby serveru je postaveno na architektuře WCF, která využívá principy SOA<sup>10)</sup> a komunikace probíhá pomocí protokolu SOAP<sup>11)</sup>. Zjednodušené schéma komunikace je znázorněno na obr. 18.



Obr. 18. Schéma komunikace WCF

Aby se klient mohl připojit ke službě, musí znát její koncový bod. Koncový bod (EndPoint) se skládá z *Adresy*, která je ve formátu URL<sup>12)</sup>, dále z *Vazby* (*Binding*), která

<sup>10)</sup>Service Oriented Architecture

<sup>11)</sup>Simple Object Access Protocol

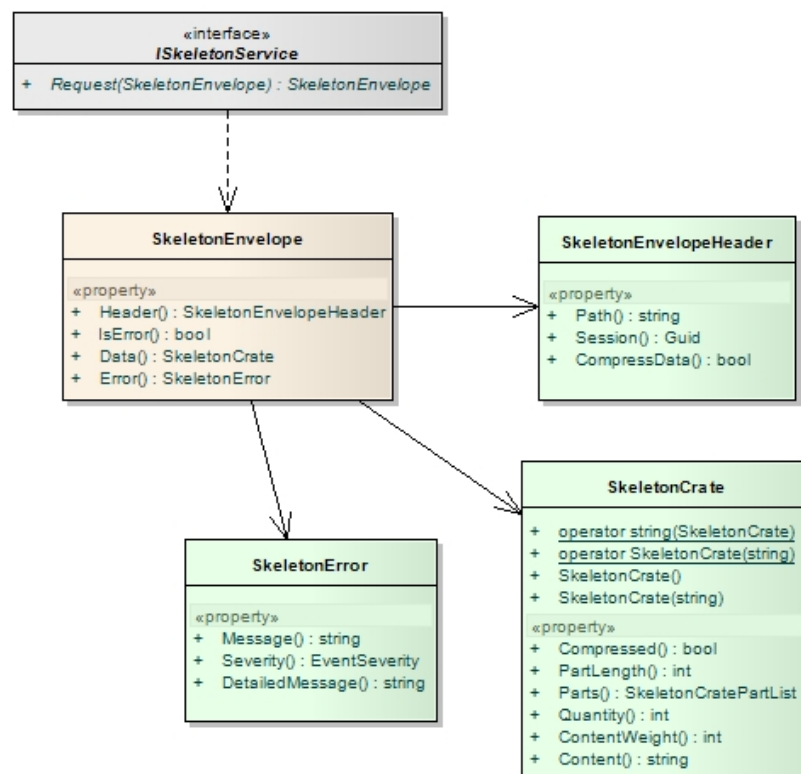
<sup>12)</sup>Uniform Resource Locator

specifikuje komunikační mechanismus a jako poslední *Kontrakt*, který specifikuje přenášenou zprávu. Vlastní komunikace pak probíhá v podobě přenosu zprávy ve formátu XML, jejíž struktura je definována SOAP. Možnosti použití a konfigurace WCF služby jsou velké a přesahují rámec této práce. Podrobnější popis této technologie je možné čerpat z [4], kap. 18.

#### 4.2.1 Komunikační rámec

Dle návrhu komunikačního rozhraní služby serveru byla vytvořena v rozhraní služby serveru jedna vstupní metoda *Request*, která má za úkol zpracovávat všechny požadavky klientů. Vzniká tak velmi úzké vstupní hrdlo do systému, ve kterém lze dobře řídit, zabezpečovat a monitorovat veškerou komunikaci.

Dále je vytvořena třída *SkeletonEnvelope*, která tvoří komunikační rámec systému a zároveň ji lze nazvat komunikační jednotkou. Třídy tohoto komunikačního rámce jsou na obr. 19. Pomocí komunikačního rámce a pomocí datové přepravky *SkeletonCrate*, která je popsána dále, lze přenášet jakékoliv serializovatelné objekty.



Obr. 19. Třídy komunikačního rámce

Aby bylo možné pomocí jediné metody rozhraní přenést jakýkoliv objekt, musí být definována datová složka *Data* jako typ **string**. Z důvodu, že není možné ve formátu XML přenášet řetězce libovolné délky, byla definována datová přepravka *SkeletonCrate*, která nahrazuje typ **string** tím způsobem, že rozděluje původní řetězec velké délky na

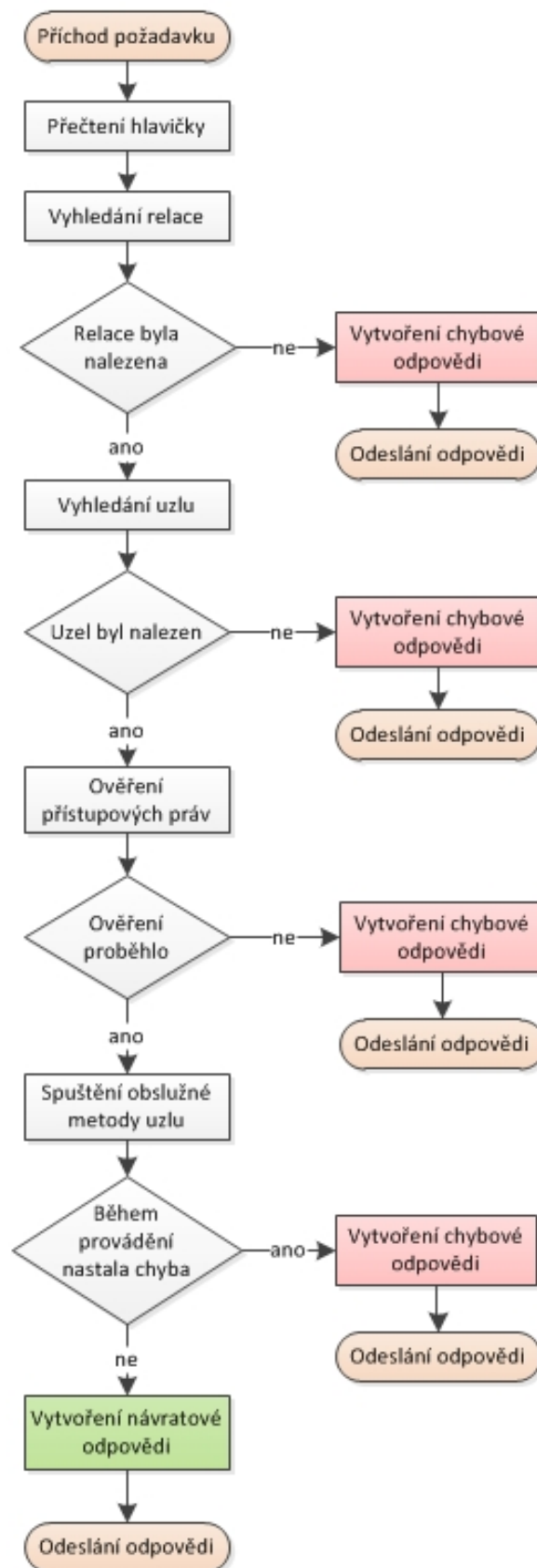
části definované délky. Tímto způsobem tak umožňuje přenést jakkoliv velký objekt v serializované podobě. Další vlastností datové přepravky je možnost komprese dat, což v případě formátu XML značně snižuje nároky na objem přenášených dat.

Komunikační rámec `SkeletonEnvelope` dále obsahuje hlavičku `Header`, která slouží pro identifikaci požadavku. Ve vlastnosti `Path` je uložena cesta k požadovanému uzlu, který má být proveden. Vlastnost `Session` identifikuje komunikační relaci a ve vlastnosti `CompressData` je možné povolit kompresi datové složky `Data` komunikačního rámce `SkeletonEnvelope`.

Poslední, neméně významnou složkou komunikačního rámce `SkeletonEnvelope` je příznak chyby `IsError`. Pokud kdekoliv při zpracování požadavku vyvolaného metodou `Request` v rozhraní služby `ISkeletonService` dojde k vyjímce, je tato vyjímka odchycena a pomocí příznaku `IsError` indikována klientovi a podrobnosti o chybě jsou nastaveny do vlastnosti `Error` komunikačního rámce `SkeletonEnvelope`.

#### 4.2.2 Zpracování požadavku

Diagram na obr. 20 znázorňuje základní operace při zpracování požadavku `Request`. Při příchodu požadavku do systému se nejprve přečte hlavička komunikačního rámce. Dle identifikátoru relace uvedeného v hlavičce se tato relace vyhledá v běhovém prostředí služby serveru. Po nalezení této příslušné relace, tzn. ztotožnění (identifikace) uživatele, a po ověření přístupových práv k uzlu je spuštěna příslušná obslužná metoda uzlu. Po ukončení provádění této obslužné metody uzlu je její návratová hodnota zabalená do datové položky komunikačního rámce a odeslána jako odpověď klientovi. Pokud v jakékoliv části tohoto procesu dojde k chybě, zpracování požadavku se ukončí a klientovi je odeslána odpověď s příslušnou chybou.



Obr. 20. Diagram zpracování požadavku

### 4.3 Vlastní uzly systému

Pro správu systému byly vytvořeny vlastní uzly systému, které jsou dotazovatelné přes jeho rozhraní. Seznam cest uzlů s popisem je uveden v tab. 3. Z důvodu zkrácení cest uzlů v této tabulce je vynechána jejich kořenová část *Skeleton.*, která logicky předchází všem cestám uzlů systému. Systém navíc umí tuto část při její neexistenci automaticky doplnit.

Tab. 3. Seznam uzlů systému

Uzel	Popis
<b>Skeleton</b>	Vrátí text identifikující systém.
<b>Echo</b>	Vrátí předaný řetězec jako ozvěnu.
<b>Service.Environment.Paths</b>	Vrátí seznam cest systému.
<b>Service.Environment.Nodes</b>	Vrátí seznam informací o uzlech systému.
<b>Service.EventLogs</b>	Vrátí seznam událostí systému.
<b>Service.Login</b>	Provede přihlášení uživatele.
<b>Service.Logout</b>	Provede odhlášení uživatele.
<b>Service.Users</b>	Vrátí seznam uživatelských účtů.
<b>Service.Users.Update</b>	Aktualizuje uživatelský účet.
<b>Service.Users.Insert</b>	Vloží nový uživatelský účet.
<b>Service.Users.Delete</b>	Odstraní uživatelský účet.
<b>Service.Users.Password.Reset</b>	Provede reset uživatelského hesla.
<b>Service.Users.Password.Update</b>	Provede změnu uživatelského hesla.
<b>Service.Users.Permissions</b>	Vrátí seznam oprávnění uživatele.
<b>Service.Users.Permissions.Update</b>	Aktualizuje seznam oprávnění uživatele.

### 4.4 Konfigurace serveru

V konfiguračním souboru serveru je potřeba nastavit dvě základní funkcionality systému a připojení k databázi, která slouží jako persistentní úložiště běhového prostředí služby serveru. Pro názornost uvádím jeho zkrácený výpis.

```

:
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="HttpStreaming" maxReceivedMessageSize="67108864"
        transferMode="Streamed" />
    </basicHttpBinding>
  </bindings>
  <services>
    <service name="Skeleton.Service.SkeletonService">
      <endpoint binding="basicHttpBinding"
        bindingConfiguration="HttpStreaming" name="EndPoint1"
        contract="Skeleton.Service.ISkeletonService">
      </endpoint>
    </service>
  </services>
</system.serviceModel>

```

```
<host>
  <baseAddresses>
    <add baseAddress="http://127.0.0.1:7000" />
  </baseAddresses>
</host>
</service>
</services>
</system.serviceModel>
:
<skeleton>
  <modules>
    <add name="TestModule" module="Skeleton.Module.Test.dll" />
  </modules>
</skeleton>
:
<connectionStrings>
  <add name="SkeletonEntities" ... />
</connectionStrings>
:
```

Pro publikování služby je důležité nastavit její koncový bod uvedený v sekci *services/service/endpoint*, který se skládá z adresy (sekce *services/service/host/baseAddresses*) na níž je služba k dispozici a z vazby (sekce *bindings*), která má podobu transportního protokolu. Z důvodu plánovaného využití služby i pro internetové aplikace bude výhradně používán transportní protokol HTTP<sup>13</sup>).

Dále je nutné pro načtení modulu do systému tento modul uvést v sekci *skeleton/modules*. Protože systém využívá jako persistentní úložiště databázi, je potřeba patřičný přípojovací řetězec v sekci *connectionStrings* se jménem *SkeletonEntities* správně nastavit.

## 4.5 Implementace zásuvného modulu

Pro vytvoření zásuvného modulu je potřeba ve VisualStudiu založit projekt typu *Class Library*, implementovat rozhraní zásuvného modulu a výkonný kód obslužných metod uzlů. Pro následné připojení zásuvného modulu do systému, je nutné tento modul zapsat do konfiguračního souboru služby serveru, tak jak je uvedeno v kap. 4.4.

### 4.5.1 Implementace rozhraní

Pro vytvoření rozhraní zásuvného modulu, které má být načteno službou serveru, je potřebné v těle třídy (třídy s rozhraním uzlů) tohoto modulu definovat příslušné metody uzlů tak, jak je uvedeno v následujícím příkladě.

---

<sup>13</sup>Hypertext Transfer Protocol

```
namespace Skeleton.Interfaces
{
    public class NodeInterface
    {
        [NodePermission("79E0309A-9537-4A9B-8523-3BFC6919F5A5",
            Name = "Přístup do celého systému Skeleton",
            Description = "Uděluje oprávnění pro přístup do celého stromu funkcí systému Ske...
            ..leton.")]
        [NodeMethod("Skeleton", Description = "Vrátí text identifikující systém.")]
        public static string Skeleton()
        {
            return "Služba informačního systému SkeletonŽ";
        }

        [NodePermission("75E9D6F6-94E4-4B77-A4C4-E4C657FACA80")]
        [NodeMethod("Echo", Description = "Vrátí předaný textový parametr jako ozvěnu.")]
        public static string Skeleton_Echo(string echo)
        {
            return echo;
        }
    }
}
```

Atribut `NodeMethod` označuje konkrétní metodu třídy jako metodu uzlu systému. První povinný parametr uvádí cestu k uzlu a druhý parametr *Description* uvádí doplňující popis uzlu. Takto označená metoda bude tedy dostupná přes rozhraní služby serveru.

Atribut `NodePermission` označuje přístupové právo k danému uzlu. První povinný parametr uvádí jedinečný identifikátor práva, druhý parametr *Name* název práva a třetí parametr *Description* doplňující popis práva. Toto přístupové právo lze poté pomocí systému pro konkrétního uživatele povolit nebo zakázat. Jednotlivá přístupová práva mají hierarchickou strukturu podle toho, jak jsou definované jednotlivé cesty k daným uzlům, které mají taktéž hierarchickou strukturu. Pokud nebude přístupové právo u metody uzlu definováno, zdědí se z nadřazeného uzlu. V těle takto označené metody je potřebné deklarovat výkonný kód, který má být při volání uzlu proveden. Jako typ vstupních a výstupního parametru může být použita jakákoliv serializovatelná třída. Třídy, jejichž instance prochází skrze rozhraní služby serveru budou dále nazývány jako kontrakty.

#### 4.5.2 Implementační doporučení a pravidla

Při vytváření modulů systému je potřebné dodržovat následující pravidla.

- Dodržet následující název modulu `Skeleton.Module.nazev_modulu.dll`.
- Třídy rozhraní deklarovat ve jmenném prostoru `Skeleton.Interfaces`.
- Třídy jednotlivých kontraktů rozhraní deklarovat ve jmenném prostoru `Skeleton.Contracts` a ve fyzickém modulu s příponou `.Common`, tedy `Skeleton.Module.nazev_modulu.Common.dll`.

- Pokud bude implementována datová vrstva, tuto deklarovat ve jmenném prostoru *Skeleton.Data*, případně oddělit do samostatného modulu s příponou *.Data*, tedy `Skeleton.Module.nazev_modulu.Data.dll`.
- Při pojmenovávání uzlů dodržovat jejich hierarchickou strukturu a vždy začínat prefixem *Skeleton.Module.nazev\_modulu..*

## 5 KLIENT

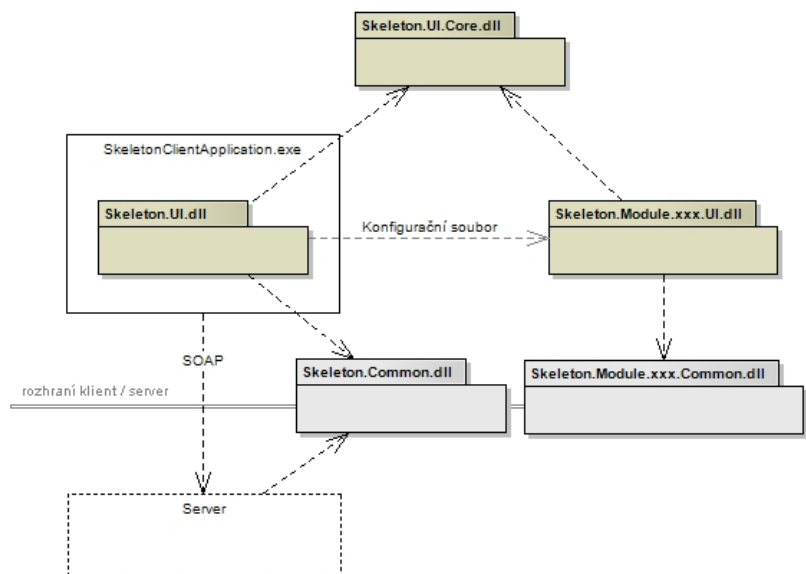
Jako protistrana k serveru budou s ohledem na návrh klientské části vytvořeny následující knihovny a aplikace. Schématické znázornění závislostí jednotlivých modulů je na obr. 21. Implementace uživatelského prostředí je provedena technologií WPF a s použitím návrhového vzoru MVVM.

**Skeleton.UI.dll** – Bude obsahovat základní logiku hostitelského prostředí potřebnou pro interakci s uživatelem. Tato knihovna nebude nikde sdílena (referencovaná) a bude zapouzdřena do spustitelného souboru.

**Skeleton.UI.Core.dll** – Bude obsahovat logiku hostitelského prostředí společnou pro klientskou vrstvu.

**Skeleton.Common.dll** – Tato knihovna je již definována v serverové části a obsahuje kód společný napříč vrstvami systému.

**SkeletonClientApplication.exe** – Spustitelný soubor klientské aplikace.



Obr. 21. Závislosti modulů klientské strany

### 5.1 Úvod do technologie WPF

Windows Presentation Foundation (WPF) je platforma využívající jazyk XAML<sup>14)</sup> pro vytváření uživatelsky bohatého rozhraní. Technologie WPF je integrovaná do Windows Vista, Windows 7 a Windows Server 2008 a je ji možno doinstalovat i do Windows XP a Windows Server 2003. Díky jazyku XAML je od sebe oddělená funkčnost a vzhled

<sup>14)</sup>Extensible Application Markup Language

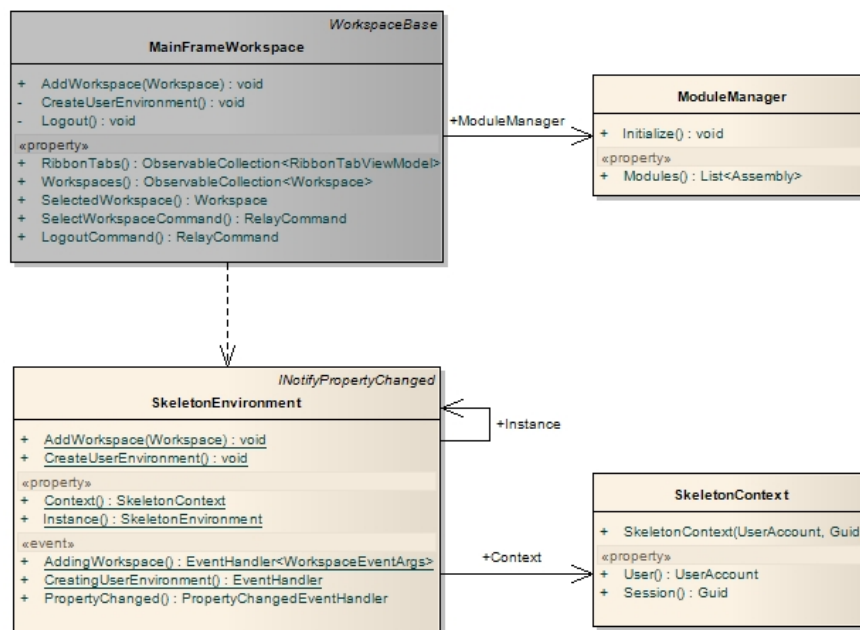
aplikace. XAML je značkovací jazyk (podobný HTML), který je určen k popisu grafického rozhraní v aplikacích nové generace. Způsob práce s komponentami WPF a jazykem XAML je možné čerpat z [2].

## 5.2 Diagramy a popis zodpovědnosti jednotlivých tříd

V následujících podkapitolách budou popsány jednotlivé zodpovědnosti a funkčnosti tříd hostitelského prostředí klientské aplikace. V diagramech tříd budou znázorněny jen nejdůležitější součásti systému. Jednotlivé třídy tak ve skutečnosti obsahují mnohem více atributů a operací, které jsou zejména zapotřebí pro správné zapouzdření tříd.

### 5.2.1 Třídy pracovního rámce

Následující obr. 22. znázorňuje základní třídy hostitelského prostředí, které tvoří logickou část uživatelského prostředí. K třídám označeným jako *Model* existuje podle návrhového vzoru MVVM i jejich pohled (*View*), který je přiřazen pomocí šablony. Tato technika použití šablon je popsána v kap. 5.3.4.



Obr. 22. Třídy pracovního rámce

**MainFrameWorkspace** – Model hlavního rámce. Třída je zodpovědná za vytvoření uživatelského prostoru a správu pracovních prostorů. Obsahuje funkce pro přidání, odebrání a aktivaci pracovního prostoru. Dále poskytuje funkce pro přihlášení a odhlášení uživatele.

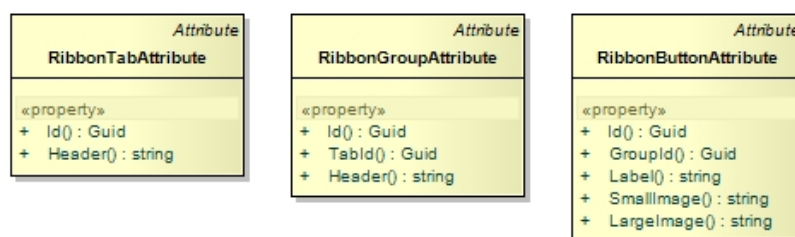
Hlavní rámec v inicializační sekci provádí postupné sestavování hlavního menu

(*RibbonTabs*) z informací, které získá pomocí reflexe postupně z každého načteného modulu ve správci modulů. Pomocí reflexe se vyhledají všechny atributy typu *RibbonTabAttribute*, *RibbonGroupAttribute*, *RibbonButtonAttribute* (obr. 23) a dle hodnot jejich vlastností se provede sestavení tohoto menu. Pomocí každého atributu se vytvoří jiná část menu, která vyplývá z jejich názvů.

**ModuleManager** – Třída správce modulů. Třída je zodpovědná za načtení všech modulů uvedených v konfiguračním souboru aplikace.

**SkeletonEnvironment** – Třída prostředí aplikace typu jedináček (Singleton). Tato třída běhového prostředí aplikace vytváří prostředníka mezi jednotlivými moduly a hlavním pracovním rámcem. Třída poskytuje funkce pro přidání pracovního prostoru do hlavního rámce a získání informací o kontextu aplikace.

**SkeletonContext** – Třída kontextu aplikace. Poskytuje informace o aktuálním kontextu aplikace.



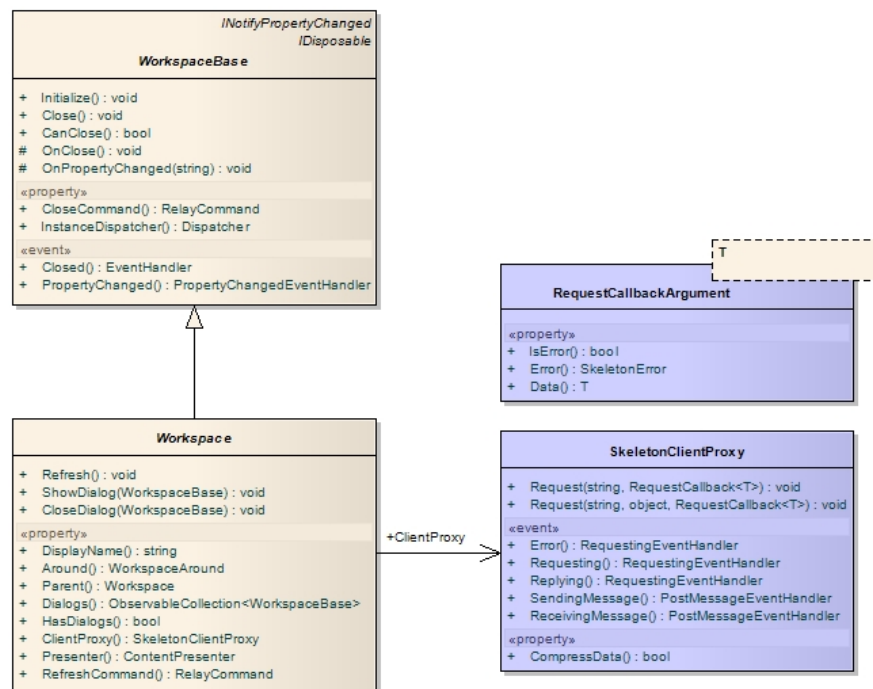
Obr. 23. Třídy atributů hlavního menu

### 5.2.2 Třídy pracovního prostoru

Pracovní prostor je základní stavební jednotkou v zásuvných modulech uživatelského rozhraní. Pracovní prostor je prvek, který vystupuje v roli hosta a je hostován hlavním pracovním rámcem aplikace. Tento pracovní rámec mu tak poskytuje veškeré zázemí pro jeho funkcionalitu. Třídy modelu pracovního prostoru jsou znázorněny na obr. 24.

**WorkspaceBase** – Předek všech modelů pracovních prostorů. Poskytuje základní funkcionalitu pro pracovní prostor. Z této třídy je možné odvodit třídy dialogů systému.

**Workspace** – Předek všech modelů pracovních prostorů, které jsou hostované pracovním rámcem. Poskytuje rozšířenou funkcionalitu pro pracovní prostor. Z této třídy musí být odvozeny všechny pracovní prostory hostované pracovním rámcem.



Obr. 24. Třídy modelu pracovního prostoru

**SkeletonClientProxy** – Třída zástupce klientského rozhraní WCF pro službu severu.

Třída je zodpovědná za komunikaci s rozhraním WCF služby serveru. Poskytuje asynchronní funkce pro volání požadavků na server. Tyto funkce jsou generické a návratová hodnota v podobě zpětného volání vrací data přímo v požadovaném typu.

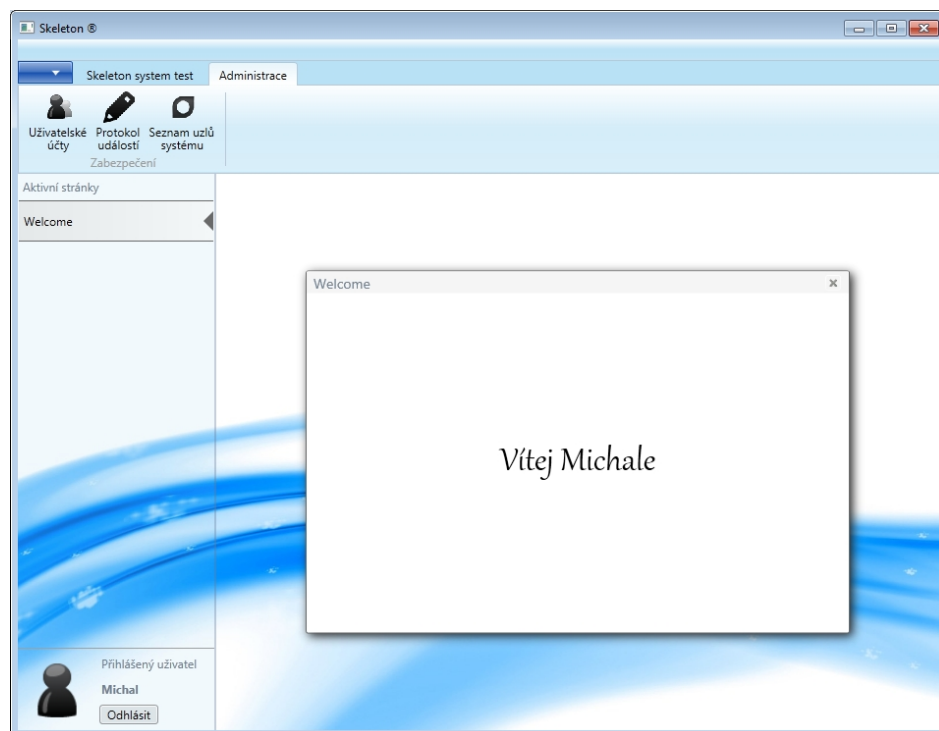
**RequestCallbackArgument** – Generická třída argumentu zpětného volání metody *Request* třídy *SkeletonClientProxy*.

### 5.3 Uživatelské rozhraní

Uživatelské rozhraní slouží k přímé interakci s uživatelem. Z hlediska návrhového vzoru MVVM reprezentuje část zvanou *View*. Na platformě WPF je pro definici uživatelského rozhraní používán jazyk XAML. Pro přiřazení *View* k *Modelu* se používá šablona (template) s jejíž pomocí vznikne spojení s vrstvou zvanou *ViewModel*.

#### 5.3.1 Pracovní rámec

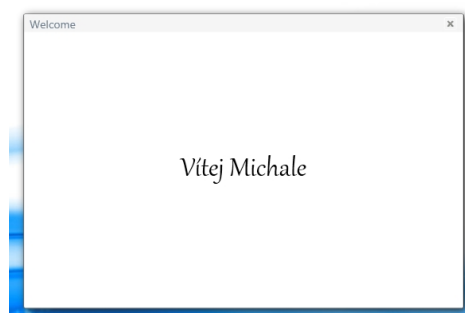
Dle návrhu bylo vytvořeno uživatelské rozhraní hlavního pracovního rámce. V horní části je lišta s příkazovým menu vygenerovaným z jednotlivých zásuvných modulů uvedených v konfiguračním souboru aplikace. Názorný výsledek implementace uživatelského rozhraní pracovního rámce je na obr. 25.



Obr. 25. Hlavní pracovní rámeč

### 5.3.2 Pracovní prostor

Pracovní prostor, jak již bylo uvedeno, je základním stavebním kamenem uživatelského rozhraní implementovaným v jednotlivých zásuvných modulech klientské aplikace. Zásuvný modul, na příkaz z menu vytvoří instanci daného modelu pracovního prostoru a za pomoci funkce prostředí (třídy `SkeletonEnvironment`) vloží tuto instanci pracovního prostoru do hlavního pracovního rámeč. Na obr. 26 je příklad implementace pracovního prostoru v podobě uvítací stránky.



Obr. 26. Pracovní prostor

### 5.3.3 Rám pracovního prostoru

Rám pracovního prostoru má za úkol vytvořit základní grafickou logiku pracovního prostoru. Prozatím byly definovány tři typy rámeč. Všechny tyto tři typy rámeč umí

zobrazit a překrýt plochu daného pracovního prostoru dialogem, vyvolaným z tohoto pracovního prostoru a dále se pak od sebe liší účelem použití. V následujícím odstavci jsou uvedeny názvy těchto rámců i s popisem jejich použití.

**FramedWorkspaceTemplate** – Vytváří základní rám pracovního prostoru s titulkovým pruhem a tlačítkem pro uzavření.

**SimpleWorkspaceTemplate** – Vytváří zjednodušený rám pracovního prostoru bez titulkového pruhu.

**DialogWorkspaceTemplate** – Vytváří základní rám pro dialogy pracovních prostorů.

Tyto jednotlivé typy rámců je možné použít v definici přiřazení pohledu (view) k modelu popisované v následující kapitole 5.3.4.

#### 5.3.4 Přiřazení view k modelu

Pro správné grafické zobrazení modelu je potřeba přiřadit k tomuto modelu jeho view viz obr. 27. Přiřazení se definuje pomocí šablony (template) umístěné ve výchozím zdrojovém slovníku zásuvného modulu. Výchozím zdrojovým slovníkem se rozumí projektový soubor *Generic.xaml* umístěný v adresáři *Themes* jehož obsah vypadá následovně.

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:v="clr-namespace:Skeleton.Views"
                    xmlns:w="clr-namespace:Skeleton.Workspaces">

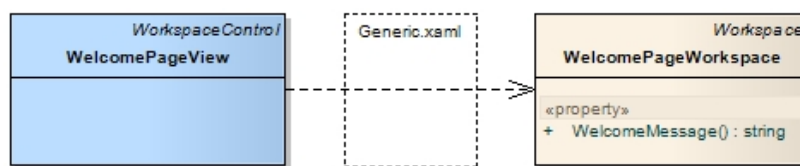
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="pack://application:,,,/Skeleton.UI.Core;component/
            Templates/FramedWorkspaceTemplate.xaml"/>
        <ResourceDictionary Source="pack://application:,,,/Skeleton.UI.Core;component/
            Templates/SimpleWorkspaceTemplate.xaml"/>
        <ResourceDictionary Source="pack://application:,,,/Skeleton.UI.Core;component/
            Templates/DialogWorkspaceTemplate.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <DataTemplate DataType="{x:Type w:UserLoginWorkspace}">
        <v:UserLoginView Template="{StaticResource SimpleWorkspaceTemplate}"/>
    </DataTemplate>

    <DataTemplate DataType="{x:Type w:UserAccountsWorkspace}">
        <v:UserAccountsView/>
    </DataTemplate>

    <DataTemplate DataType="{x:Type w>WelcomePageWorkspace}">
        <v>WelcomePageView/>
    </DataTemplate>

    :
```



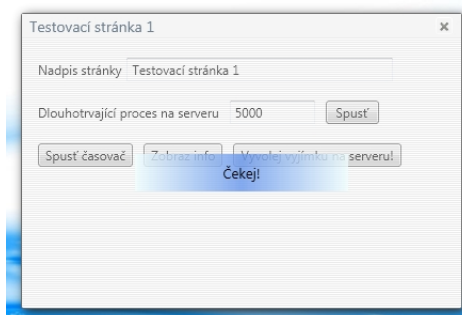
Obr. 27. Přiřazení view k modelu

V sekci *ResourceDictionary/MergedDictionaries* jsou připojeny šablony s definicí rámců pro pracovní prostory. Pokud je pohled (view) odvozen z třídy *WorkspaceControl* je implicitně použita šablona *FramedWorkspaceTemplate*, v opačném případě je potřeba tuto šablonu uvést explicitně. V sekcích *DataTemplate* jsou uvedeny vazby jednotlivých pohledů (view) na jejich model s případným explicitním použitím rámcové šablony.

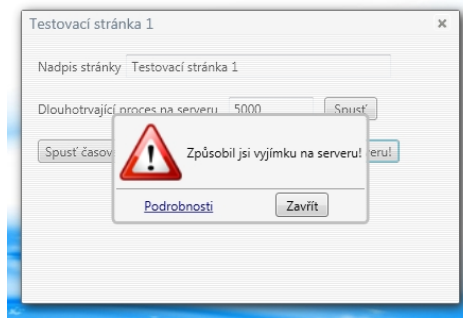
### 5.3.5 Dialogy pracovního prostoru

Za účelem vynucení vstupu od uživatele nebo zobrazení informační zprávy (např. o průběhu operace, atd. . .) je implementován systém dialogů. Dialogem může být jakýkoliv pracovní prostor (instance třídy odvozené od třídy *WorkspaceBase*) aktivovaný pomocí funkce *ShowDialog* třídy *Workspace*.

Příklad uživatelského rozhraní dialogů implementovaných v této klientské části systému je na obr. 28 a obr. 29. Při vyvolání dialogu dojde k překrytí pracovní plochy daného pracovního prostoru dialogem a tím k znemožnění interakce uživatele s tímto pracovním prostorem. Po vykonání vynucené interakce a zavření dialogu je opět daný pracovní prostor k dispozici. Ostatní pracovní prostory, které nemají aktivovaný žádný dialog jsou nadále plně funkční a uživateli dostupné k interakci. Takto je docílena v návrhu zmiňovaná multimodálnost aplikace.



Obr. 28. Informační dialog pracovního prostoru



Obr. 29. Chybový dialog pracovního prostoru

## 5.4 Konfigurace klienta

V konfiguračním souboru klienta je potřeba nastavit dvě základní funkcionality systému. Pro názornost uvádím jeho zkrácený výpis.

```

<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="EndPoint1" closeTimeout="00:01:00" openTimeout="00:01:00"
        receiveTimeout="00:10:00" sendTimeout="00:01:00" allowCookies="false"
        bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
        maxBufferSize="67108864" maxBufferPoolSize="524288" maxReceivedMessageSi...
        ..ze="67108864"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Streamed"
        useDefaultWebProxy="true">
      <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
        maxBytesPerRead="4096" maxNameTableCharCount="16384" />
      <security mode="None">
        <transport clientCredentialType="None" proxyCredentialType="None"
          realm="" />
        <message clientCredentialType="UserName" algorithmSuite="Default" />
      </security>
    </binding>
  </basicHttpBinding>
</bindings>
<client>
  <endpoint address="http://127.0.0.1:7000/" binding="basicHttpBinding"
    bindingConfiguration="EndPoint1" contract="SkeletonService"
    name="EndPoint1" />
</client>
</system.serviceModel>
:
<skeleton>
  <modules>
    <add name="TestModule" module="C:\Projects\Skeleton.NET\Skeleton.Module.Test.UI\...
      ...bin\Debug\Skeleton.Module.Test.UI.dll"/>
  </modules>
</skeleton>
:

```

Pro přístup ke službě serveru je důležité nastavit její koncový bod uvedený v sekci *client/endpoint*, který se skládá z adresy *address*, na níž je služba k dispozici a z vazby *binding*, která je definovaná v sekci *bindings*. Toto nastavení se musí přesně shodovat s nastavením služby na serverové straně.

Pro načtení modulu s uživatelským rozhraním do systému, je nutné tento modul uvést v sekci *skeleton/modules*.

## 5.5 Implementace zásuvného modulu

Pro vytvoření zásuvného modulu je potřeba ve VisualStudiu založit projekt typu *WPF User Control Library*, implementovat rozhraní a kód obslužných příkazů menu. Pro připojení modulu do systému, je nutné tento modul zapsat do konfiguračního souboru aplikace, tak jak je uvedeno v kap. 5.4.

### 5.5.1 Implementace rozhraní

Pro vytvoření rozhraní zásuvného modulu, které má být načteno klientskou aplikací, je potřebné v těle třídy (třídy s rozhraním menu) tohoto modulu definovat příslušné atributy tak, jak je uvedeno v následujícím příkladě.

```
using Skeleton.Interfaces;

[assembly: RibbonTab("3E77ED30-6D5D-4509-9F72-1AE4B5FA9216", Header = "Administrace")]
[assembly: RibbonGroup("7039A1AE-2D68-44B8-9D53-65B151372141", "3E77ED30-6D5D-4509-9...
...F72-1AE4B5FA9216", Header = "Zabezpečení")]

namespace Skeleton.Interfaces
{
    public static class RibbonInterface
    {
        [RibbonButton("3E77ED30-6D5D-4509-9F72-1AE4B5FA9216", "7039A1AE-2D68-44B8-9D5...
...3-65B151372141", Label = "Uživatelské účty", LargeImage = "pack://applica...
...tion:,,,/Skeleton.UI.Core;component/Resources/Users64.png")]
        public static void ShowUserList()
        {
            SkeletonEnvironment.AddWorkspace(Workspaces.UserAccountsWorkspace.OneInstance);
        }

        [RibbonButton("331D0F0A-35C7-4CD8-B47B-B2D7F0639497", "7039A1AE-2D68-44B8-9D5...
...3-65B151372141", Label = "Protokol událostí", LargeImage = "pack://applica...
...tion:,,,/Skeleton.UI;component/Images/Pen48.png")]
        public static void ShowEventLog()
        {
            SkeletonEnvironment.AddWorkspace(Workspaces.EventLogsWorkspace.OneInstance);
        }

        [RibbonButton("F45F36CC-B60C-40E5-8A09-32F7FCF2727E", "7039A1AE-2D68-44B8-9D5...
...3-65B151372141", Label = "Seznam uzlů systému", LargeImage = "pack://applica...
...tion:,,,/...Skeleton.UI;component/Images/Node48.png")]
        public static void ShowNodes()
        {
            SkeletonEnvironment.AddWorkspace(Workspaces.NodesWorkspace.OneInstance);
        }
    }
}
```

Atribut *RibbonTab* slouží k definici záložky v liště menu hlavního pracovního rámce. První povinný parametr udává jedinečný identifikátor záložky v menu a druhý parametr *Header* uvádí text v hlavičce záložky.

Atribut `RibbonGroup` slouží k definici skupiny na dané záložce v liště menu hlavního pracovního rámce. První povinný parametr udává jedinečný identifikátor skupiny v menu. Druhý povinný parametr udává jedinečný identifikátor nadřízené záložky v menu a třetí parametr `Header` uvádí text v hlavičce skupiny.

Atribut `RibbonButton` slouží k definici tlačítka v dané skupině a záložce lišty hlavního menu pracovního rámce. První povinný parametr udává jedinečný identifikátor tlačítka v menu. Druhý povinný parametr udává jedinečný identifikátor nadřízené skupiny v menu. Třetí uvedený parametr `Label` uvádí text tlačítka a čtvrtý parametr `LargeImage` uvádí zdroj obrázku, který má být zobrazen v daném tlačítku. Při stisknutí takto vytvořeného tlačítka v liště menu hlavního pracovního rámce, dojde ke spuštění metody u které je tento atribut definován. Z ukázky je patrné, že se v tomto případě provede přidání nového pracovního prostoru do běhového prostředí aplikace.

### 5.5.2 Implementační doporučení a pravidla

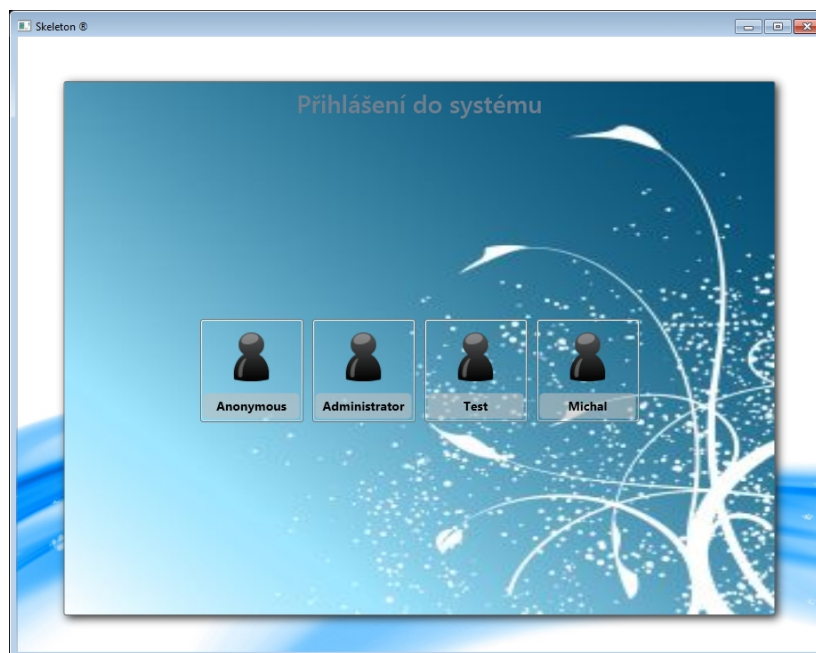
Při vytváření modulů s uživatelským rozhraním systému je potřebné dodržovat následující pravidla.

- Dodržet následující název modulu `Skeleton.Module.nazev_modulu.UI.dll`.
- Třídy rozhraní deklarovat ve jmenném prostoru `Skeleton.Interfaces`.
- Třídy pracovních prostorů deklarovat ve jmenném prostoru `Skeleton.Workspaces`.
- Třídy pohledů pracovních prostorů deklarovat ve jmenném prostoru `Skeleton.Views`.

## 5.6 Obsluha klientské aplikace

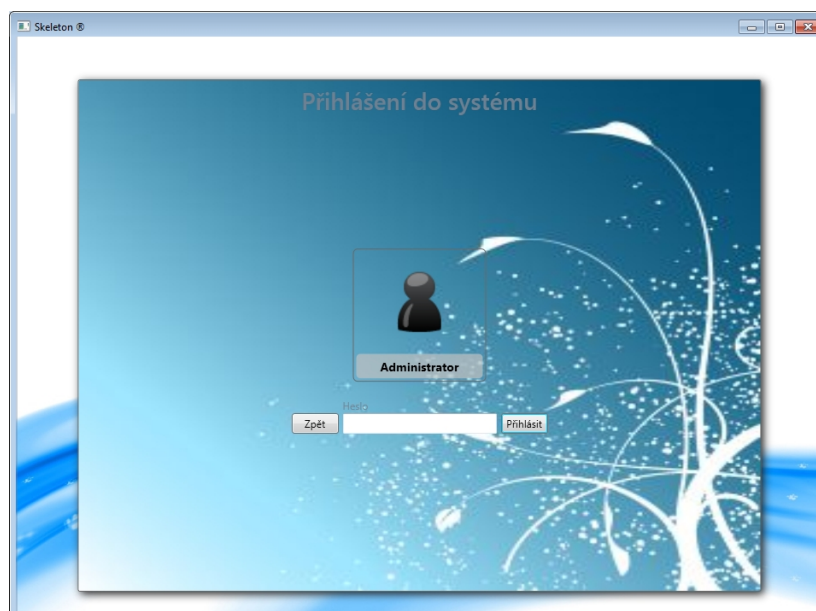
Z důvodu, že v této prvotní fázi implementace klientské aplikace není předpokládána tvorba uživatelské příručky, uvádím alespoň základní popis obsluhy této aplikace.

Před spuštěním aplikace je nutné, aby byla správně nastartovaná serverová služba systému. Po spuštění klientské aplikace a získání seznamu uživatelů systému ze služby serveru je zobrazena úvodní obrazovka s tímto seznamem uživatelů, viz obr. 30. Na této obrazovce je možné si vybrat jeden z uživatelských účtů, na základě jehož chceme provést přihlášení do systému.



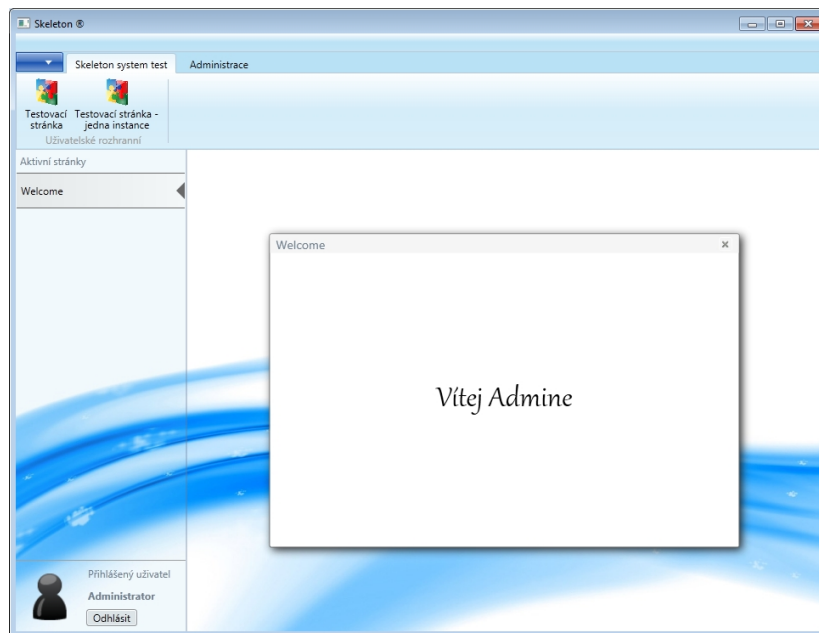
Obr. 30. Klientská aplikace – úvodní obrazovka

Po provedení výběru uživatelského účtu se zobrazí obrazovka (obr. 31) s možností zadání hesla k tomuto uživatelskému účtu a možností provedení přihlášení k systému. V této instalaci jsou hesla nastavena na prázdný řetězec (není potřeba nic zadávat).

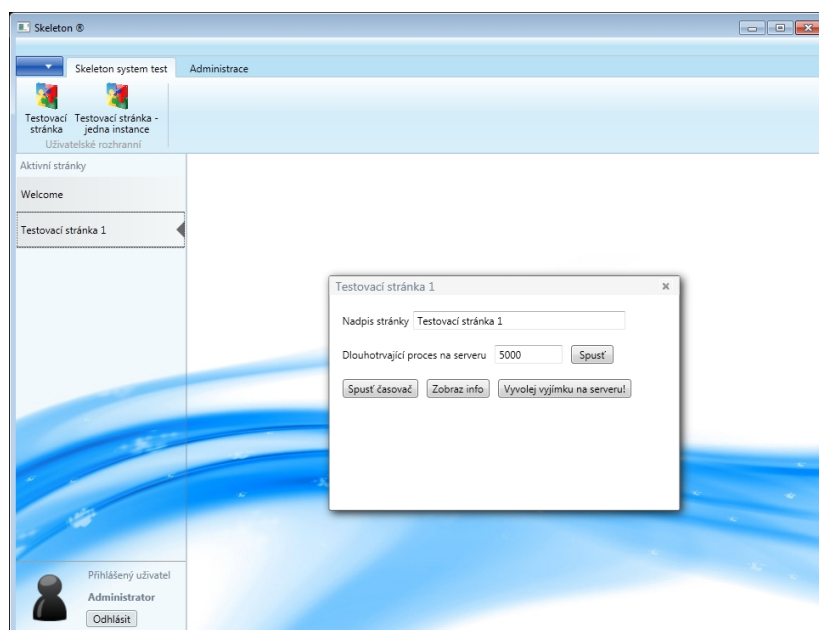


Obr. 31. Klientská aplikace – přihlášení

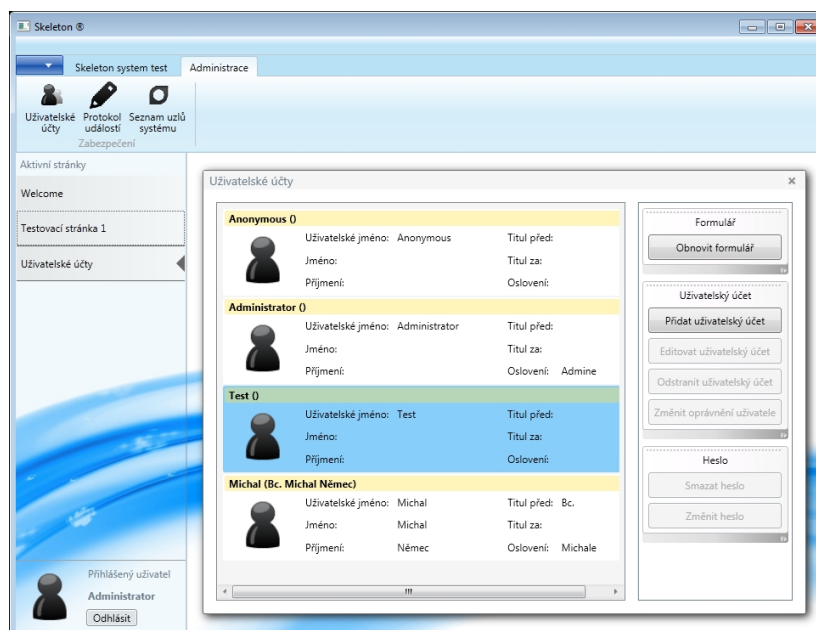
Po úspěšném přihlášení do systému je zobrazena uvítací obrazovka (obr. 32). Dále je možné provádět volby v hlavní nabídce aplikace a vyvolat například *Testovací stránku* (obr. 33), stránku *Uživatelské účty* (obr. 34) nebo stránku *Seznam událostí systému* (obr. 35).



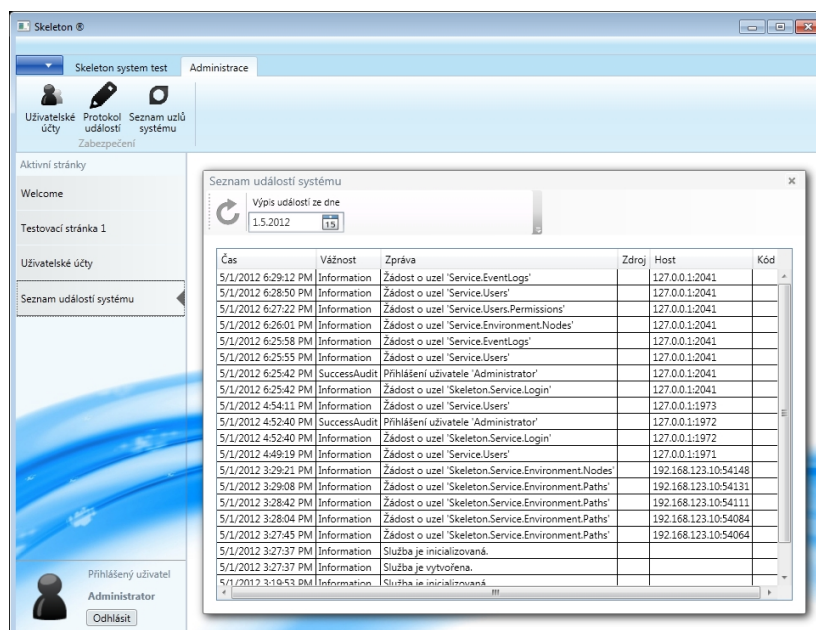
Obr. 32. Klientská aplikace – uvítací obrazovka



Obr. 33. Klientská aplikace – Testovací stránka



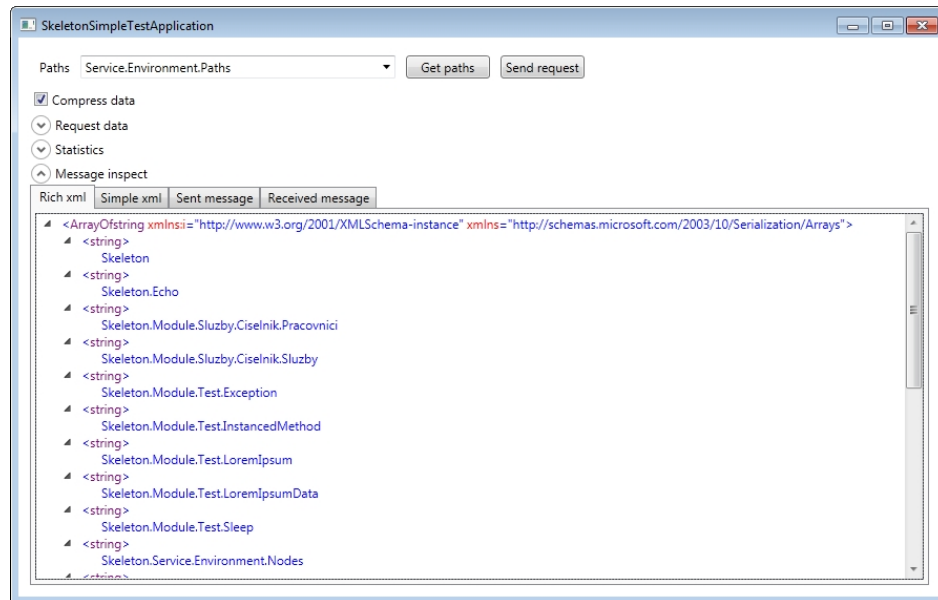
Obr. 34. Klientská aplikace – Uživatelé účty



Obr. 35. Klientská aplikace – Seznam událostí systému

## 6 TESTOVACÍ APLIKACE ROZHRAŇÍ SLUŽBY SERVERU

Tato klientská aplikace byla vytvořena pro účely základního testování komunikačního rozhraní služby serveru. Náhled obrazovky této aplikace je na obr. 36.



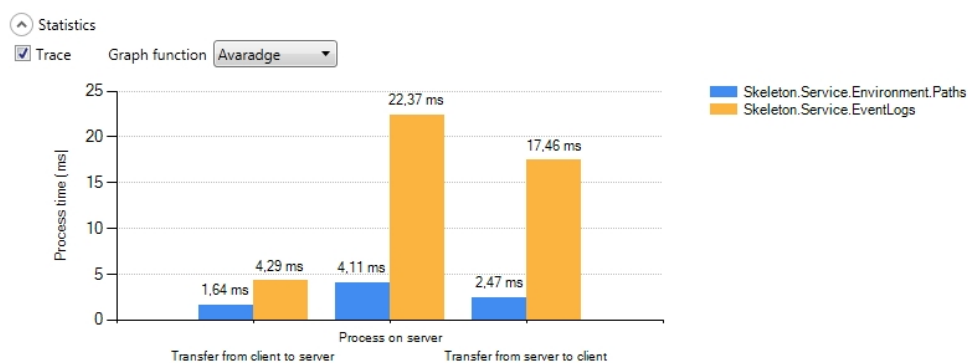
Obr. 36. Testovací aplikace rozhraní služby serveru

V horní části formuláře je možno tlačítkem *Get paths* načíst do výběrového boxu seznam cest uzlů systému. Poté je možné vybrat jednu z cest k uzlu systému a provést vyžádání tlačítkem *Send request*. Aplikace, pokud je správně nakonfigurována dle kap. 5.4, vyšle žádost na server a po přijetí odpovědi zobrazí datovou část této odpovědi v záložce *Rich xml* nebo *Simple xml* v části *Message inspect*. V této části je také možno nahlédnout na tvar odesílané a přijímané zprávy protokolu SOAP, viz obr. 37.



Obr. 37. Testovací aplikace rozhraní služby serveru – kontrola zpráv

Dále aplikace disponuje jednoduchým statistickým zobrazením komunikace se serverem, viz obr. 38 a vyhodnocování je orientováno na jednotlivé uzly systému.



Obr. 38. Testovací aplikace rozhraní služby serveru – statistika

Část aplikace *Request Data* slouží pro zkonstruování dat ve formátu XML, které mají být odeslány v žádosti, viz obr. 39. K dispozici je výběrový box s předdefinovanými vzory.

The screenshot shows the 'Request data' interface. It features an 'Insert pattern' field with a dropdown menu containing the XML snippet: `<?xml version="1.0" encoding="utf-16"?> <string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">t`. Below the field is a large text area containing the full XML snippet: `<?xml version="1.0" encoding="utf-16"?> <string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">text</string>`. There are 'Insert' and 'Clear' buttons next to the field.

Obr. 39. Testovací aplikace rozhraní služby serveru – data žádosti

## 7 MOŽNOSTI VYUŽITÍ ROZHŘANÍ NA JINÝCH PLATFORMÁCH

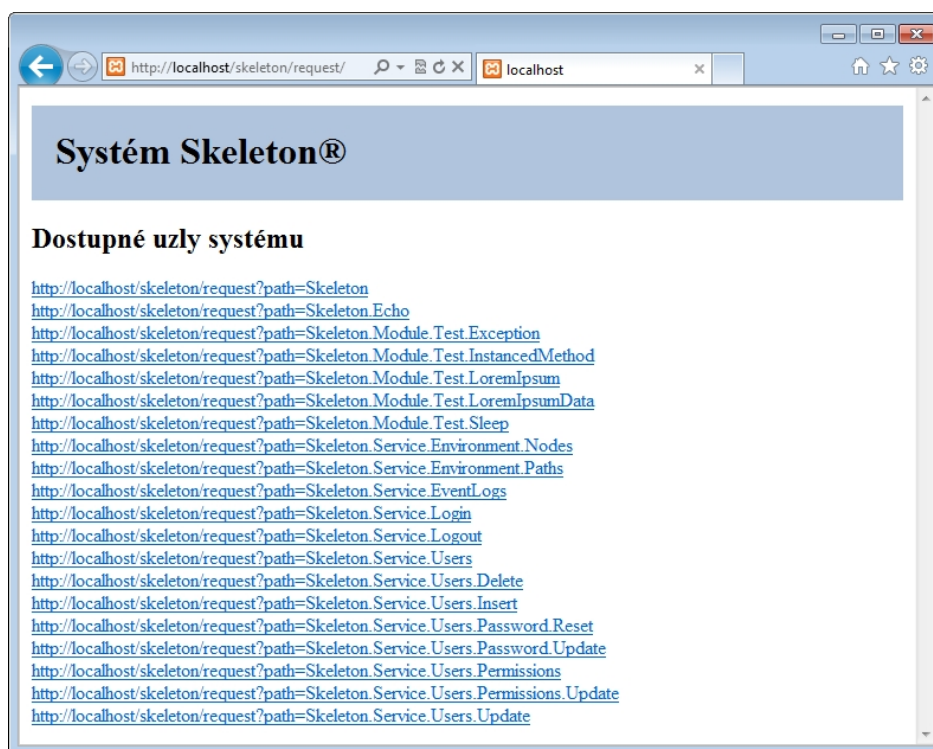
Rozhraní WCF služby umožňuje připojení na různých platformách, které je dané použitým protokolem SOAP této služby. V rámci informačního systému Skeleton je uvažováno připojení z Delphi XE2 a PHP. Obě tyto platformy obsahují potřebné rozšíření pro snadné vytvoření tohoto typu připojení a současně poskytují odstínění od přenosového formátu.

Nad rámec zadání této diplomové práce byl vytvořen testovací projekt (klientská aplikace) v kódu PHP. Tento projekt obsahuje dva soubory *index.php* a *SkeletonClientProxy.php*, jejichž popis je uveden v tab. 4.

Tab. 4. Soubory PHP klienta

Soubor	Popis
<b>index.php</b>	Hlavní stránka PHP klienta.
<b>SkeletonClientProxy.php</b>	Zástupný modul pro WCF službu serveru.

Tento projekt je možné hostovat ve webovém prostředí s podporou PHP 5. Po zadání odkazu na výchozí stránku do internetového prohlížeče se zobrazí stránka se seznamem dostupných uzlů systému, viz obr. 40. Po aktivaci odkazu příslušného uzlu a zpracování požadavku serverem budou zobrazena výsledná data, viz obr. 41.



Obr. 40. Ukázka PHP klienta – hlavní stránka

```

<?xml version="1.0" encoding="UTF-16"?>
- <ArrayOfNode xmlns="http://schemas.datacontract.org/2004/07/Skeleton.Contracts"
  xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  - <Node>
    <Path>Skeleton</Path>
    <Description>Vrátí text identifikující systém.</Description>
    <EntryMethod>String Skeleton()</EntryMethod>
    <Class>Skeleton.Interfaces.NodeInterface</Class>
    <Module>Skeleton.Service.dll</Module>
  </Node>
  - <Node>
    <Path>Skeleton.Echo</Path>
    <Description>Vrátí předaný textový parametr jako ozvěnu.</Description>
    <EntryMethod>String Skeleton_Echo(String)</EntryMethod>
    <Class>Skeleton.Interfaces.NodeInterface</Class>
    <Module>Skeleton.Service.dll</Module>
  </Node>
  - <Node>
    <Path>Skeleton.Module.Test.Exception</Path>
    <Description>Vyvolá výjimku na serveru.</Description>
    <EntryMethod>String Module_Test_Exception()</EntryMethod>
    <Class>Skeleton.Interfaces.NodeInterface</Class>
    <Module>Skeleton.Module.Test.dll</Module>
  </Node>
  - <Node>
    <Path>Skeleton.Module.Test.InstanceMethod</Path>
    <Description>Při volání této metody se vytvoří instance její třídy. V návratovém textu
      je vypsán stav počítadel instance.</Description>
    <EntryMethod>String Module_Test_InstanceMethod()</EntryMethod>
    <Class>Skeleton.Interfaces.NodeInterface</Class>
    <Module>Skeleton.Module.Test.dll</Module>
  </Node>
- <Node>

```

Obr. 41. Ukázka PHP klienta – data

Tato implementace je jen krátkou demonstrací možnosti, jak lze využít komunikační rozhraní služby serveru systému Skeleton. Takto získané data lze dále zpracovat téměř jakýmkoliv způsobem, například použít XSL<sup>15)</sup> transformace a převést tak tento dokument do formátu HTML nebo do obyčejného textového souboru.

<sup>15)</sup>Extensible Stylesheet Language

## 8 SHRNU TÍ VÝSLEDNÉ IMPLEMENTACE

V průběhu této práce byly vytvořeny čtyři výsledné aplikace. Jako první vznikla konzolová aplikace zapouzdřující službu serveru. Rozhodnutí, zapouzdřit službu serveru do konzolové aplikace, bylo učiněno čistě z praktických důvodů za účelem testování služby. V produkční verzi je plánováno zapouzdření do služby systému (Windows Service). Aplikace, respektive knihovna služby serveru ke svému chodu používá MS SQL databázi, která je implementována do systému pomocí technologie Entity Framework knihovny ADO.NET. Způsob práce s touto technologií a s databází MS SQL byl čerpán z [4] a [3]. Pro návrh různých datových struktur, které byly v této části implementace zapotřebí, jsem využíval informace ze zdroje [6].

Jako druhá vznikla aplikace pro testování rozhraní služby serveru. Tato aplikace pomohla odladit veškerou komunikaci a strukturu zpráv mezi klientem a serverem.

Jako třetí vznikla hlavní klientská aplikace uživatelského rozhraní tohoto systému. Uživatelské rozhraní aplikace je implementováno pomocí technologie WPF, jazyka XAML a návrhového vzoru MVVM. Pro pomoc při implementaci této části jsem využíval zdroje [2] a [5]. Spustitelný soubor této aplikace tvoří pouhou obálku nad implementací uživatelského rozhraní, které je zapouzdřeno do knihovny tříd (DLL). Tento způsob produkce tak umožňuje cílové zapouzdření i do jiného typu aplikace. Pro otestování této možnosti byl v tomto implementačním řešení vytvořen projekt s názvem *SkeletonBrowserApplication*, který tuto hlavní knihovnu uživatelského prostředí zapouzdřuje do aplikace typu XBAP, která může být hostována ve webovém prohlížeči. Informace k platformě XBAP jsem čerpal ze zdroje [9].

Jako čtvrtá vznikla webová klientská aplikace napsaná v jazyku PHP, která měla posloužit k otestování možnosti připojení ke službě serveru i z jiné platformy než Microsoft .NET. Zároveň je počítáno s možným použitím tohoto rozhraní ve webovém redakčním systému založeném na jazyku PHP. Jako referenční příručku jazyka PHP jsem využíval zdroj [10].

Průběžně, po dobu celé implementace, jsem používal jako referenční příručku k jazyku C# a k běhovému prostředí CLR<sup>16)</sup> aplikačního rámce .NET zdroj [1]. Dále jsem po dobu celé implementace věnoval maximální úsilí při dodržování obecně platných zásad implementace, jako jsou pojmenovávací konvence, zapouzdření jednotlivých tříd, rozdělení systému na logické a fyzické části a v neposlední řadě také použití návrhových vzorů. V této posledně jmenované kategorii mi byl velkým průvodcem zdroj [5].

---

<sup>16)</sup>Common Language Runtime

## 8.1 Zhodnocení výsledné implementace

V závěru této práce se podařilo naplnit všechny body zadání. Byla vytvořena serverová i klientská část jádra systému s moduly. V klientské části byly dokonce vytvořeny tři klientské aplikace oproti uvažované jedné. Všechny tyto části systému jsou plně funkční a strukturované tak, aby byl možný jejich další rozvoj. Za velký úspěch považuji výslednou implementaci komunikačního rozhraní služby serveru v podobě jediné metody a komunikačního rámce. Toto řešení přináší nezávislost na komunikační platformě WCF a umožňuje případný přechod nebo rozšíření na jakoukoliv jinou komunikační platformu se zachováním vlastností systému. Zároveň se podařilo celý systém dobře strukturovat v rámci rozdělení do jednotlivých shromáždění a jmenných prostorů. Dalším podstatným přínosem této implementace, je modulárnost celého navrženého systému v podobě zásuvných modulů. Poslední věcí, kterou bych rád uvedl, je úspěšná implementace uživatelského rozhraní pomocí návrhového vzoru MVVM. Toto řešení umožní snadnou změnu vzhledu aplikace bez zásahu do logiky uživatelského rozhraní.

## 8.2 Budoucí směr vývoje systému

Po ukončení této diplomové práce budou následně implementovány jednotlivé zásuvné moduly systému. Pro účel nasazení systému u zákazníka bude služba serveru zapouzdřena do služby operačního systému. Pro účely plynulého přechodu stávajících modulů systému implementovaných v prostředí Delphi, budou vytvořeny komponenty v tomto prostředí Delphi potřebné pro připojení k novému komunikačnímu rozhraní WCF služby serveru. Další směr vývoje systému Skeleton půjde vstříc k internetovým aplikacím a bude věnován rozvoji na platformě jazyka PHP a ASP.NET<sup>17)</sup>.

---

<sup>17)</sup>Technologie pro vývoj webových aplikací založená na Microsoft .NET Frameworku

## 9 INSTALACE A SPUŠTĚNÍ APLIKACE

Před začátkem instalace je nutné ověřit, zda je v systému nainstalován Microsoft .Net Framework 4. V případě nepřítomnosti Microsoft .Net Framework 4 instalátor upozorní na tuto skutečnost a instalace bude přerušena. Dále je nutné mít nainstalovaný Microsoft SQL Server 2008 verze Express, který je potřebný pro úspěšné spuštění služby serveru.

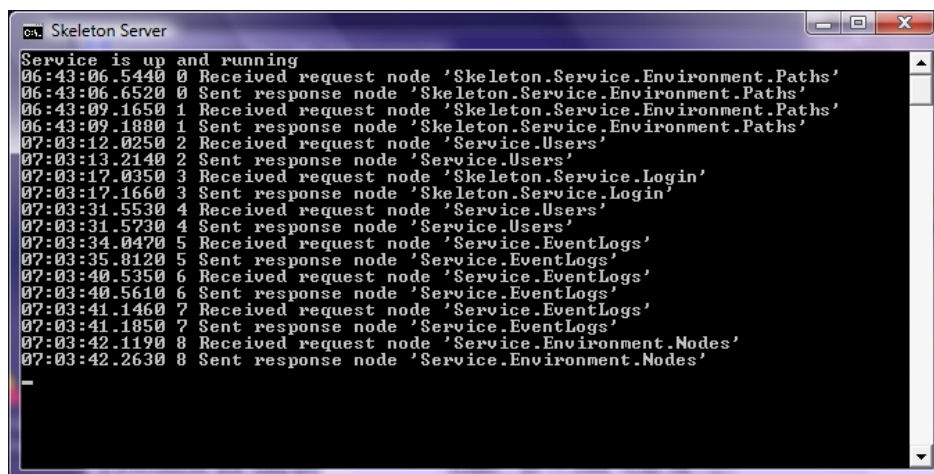
Instalaci proveďte spuštěním instalačního souboru `SkeletonSetup.exe` z instalačního CD a pokračujte dle pokynů na obrazovce.

Po úspěšném dokončení instalace přibudou na Ploše a v nabídce Start (pokud nebylo změněno výchozí nastavení instalace) tři zástupci těchto aplikací:

**Skeleton Server** – provede spuštění serveru systému Skeleton. Pokud spuštění serveru proběhne v pořádku, tak systém v okně konzole vypíše *Service is up and running* a dále vypisuje požadavky přicházející z komunikačního rozhraní služby, viz obr. 42. V opačném případě došlo k chybě ve spouštění serveru a je potřeba provést nápravu. Případná chyba může být v nastavení konfiguračního souboru.

**Skeleton Client** – provede spuštění aplikace s hlavním klientským uživatelským rozhraním. Pokud spuštění aplikace proběhne v pořádku, zobrazí se seznam uživatelů systému. V opačném případě se zobrazí chybové hlášení. Případná chyba může být v nastavení konfiguračního souboru. Obsluha aplikace je popsána v kap. 5.6.

**Skeleton Test** – provede spuštění testovací aplikace služby serveru. Obsluha aplikace je popsána v kap. 6.



```
ca. Skeleton Server
Service is up and running
06:43:06.5440 0 Received request node 'Skeleton.Service.Environment.Paths'
06:43:06.6520 0 Sent response node 'Skeleton.Service.Environment.Paths'
06:43:09.1650 1 Received request node 'Skeleton.Service.Environment.Paths'
06:43:09.1880 1 Sent response node 'Skeleton.Service.Environment.Paths'
07:03:12.0250 2 Received request node 'Service.Users'
07:03:13.2140 2 Sent response node 'Service.Users'
07:03:17.0350 3 Received request node 'Skeleton.Service.Login'
07:03:17.1660 3 Sent response node 'Skeleton.Service.Login'
07:03:31.5530 4 Received request node 'Service.Users'
07:03:31.5730 4 Sent response node 'Service.Users'
07:03:34.0470 5 Received request node 'Service.EventLogs'
07:03:35.8120 5 Sent response node 'Service.EventLogs'
07:03:40.5350 6 Received request node 'Service.EventLogs'
07:03:40.5610 6 Sent response node 'Service.EventLogs'
07:03:41.1460 7 Received request node 'Service.EventLogs'
07:03:41.1850 7 Sent response node 'Service.EventLogs'
07:03:42.1190 8 Received request node 'Service.Environment.Nodes'
07:03:42.2630 8 Sent response node 'Service.Environment.Nodes'
```

Obr. 42. Konzolová aplikace

## 9.1 Požadavky na systém

- Podporované operační systémy: Windows 7; Windows Server 2003 Service Pack 2; Windows Server 2008; Windows Server 2008 R2; Windows Vista Service Pack 1; Windows XP Service Pack 3
- Běhové prostředí: Microsoft .NET Framework 4, Microsoft SQL Server 2008 Express
- Procesor: Pentium 1 GHz nebo vyšší
- RAM: 512MB nebo více
- Pevný disk: 10MB volného místa nebo více, při současné instalaci balíčku Microsoft .NET Framework 4 až 850MB volného místa

## ZÁVĚR

Výsledný software založený na návrhu popisovaném v této práci není finální produkt, nýbrž jen začátek (jádro) rozsáhlého informačního systému s příznačným názvem Skeleton. Vůbec než padlo rozhodnutí začít s vývojem nové platformy pro tento informační systém, bylo stráveno mnoho hodin s programátory současných modulů a vedením firmy. Z analýzy těchto rozhovorů jsem načerpal potřebné rámcové informace o zbývajících částech současného informačního systému, které sám nemám na starosti. Bez těchto informací a bez mé dlouholeté spolupráce na části tohoto systému, by nebylo možné provést analýzu a návrh nového jádra systému správným směrem. Na základě zmiňované dlouholeté zkušenosti, jsem byl firmou pověřen vedením této práce a zároveň mi bylo umožněno zpracovat tento projekt v podobě diplomové práce.

Výsledkem této diplomové práce je vícevrstvý modulární systém prezentovaný v podobě dvou aplikací se zásuvnými moduly. Celé řešení projektu bylo vedeno se zájmem použít co nejmodernější prostředky a technologie firmy Microsoft. Přestože tento projekt je pouhým začátkem rozsáhlého systému, je v této podobě plně funkční a připravený k prvním implementacím zásuvných modulů. Nutno podotknout, že proběhla i implementace zásuvného modulu, který nese část skutečné funkcionality informačního systému Skeleton. Tento modul, současně s nově vyvinutou serverovou částí se plánuje k nasazení do zkušebního provozu u zákazníka. Funkce poskytované tímto modulem budou použity jako zdroj dat pro prezentaci ve webovém redakčním systému. Uvedení serverové části do provozu, tak představuje první fázi nasazování tohoto produktu. Ve druhé fázi, která je uvažována do konce roku 2012, se plánuje i nasazení klientské části s příslušnými zásuvnými moduly, které budou teprve vyvíjeny.

Úvodní teoretická část, která se věnuje návrhu řešení nového jádra informačního systému Skeleton, byla dokonce prezentována v podobě příspěvku na elektronické konferenci zaštitěnou pod Univerzitou Tomáše Bati ve Zlíně [11]. Tento příspěvek si kladl za důraz uvést význam dnešních informačních systémů z pohledu jejich modularity a znovu využitelnosti.

Tuto práci považuji za přínosnou, už jen z hlediska jejího praktického nasazení. Analýza, návrh a implementace vyžadovaly mnoho časových prostředků a programátorských zkušeností. Další řada zkušeností, zejména s prostředím .NET a jazykem C# pak byla načerpána během práce na této diplomové práci, což považuji za velký osobní i profesní přínos.

## ZÁVĚR V ANGLIČTINĚ

The final software based on the proposal described herein is not the final product yet. Rather, it is but a start (i.e., the kernel) of an extensive information system fittingly called Skeleton. Before the decision to start the development of a new platform for this information system was even made, many hours were spent discussing the matter with the programmers of the modules currently in use and with the company's management. By analyzing these discussions, I have obtained the necessary framework information on other parts of the current information system-parts that I am not in charge of. Without this intelligence and without my long-term cooperation on a part of this system, the proposal and well-directed analysis of a new kernel would not have been possible. It was based on this long-term experience that the company put me in charge of this assignment. At the same time, I was allowed to process the project into the form of a diploma thesis.

The resulting product of this diploma thesis is a modular, multi-layer system consisting of two plugin-enabled applications. The whole management of the project has been led with the intention to make use of the most up-to-date Microsoft technology and tools. Even though the project is but a beginning of an extensive system, it is already fully functional as it is, and plug-in modules can now be readily put to use for the first time. It must be noted that a plug-in module has been implemented that delivers part of the real-world functionality of the Skeleton system. Along with the newly developed server side, this module is scheduled for inclusion in customer testing. Functions provided by this module will be used as a data source for presentation in the web-based CMS. Introducing the server part into operation is thus the first-phase to the introduction of the product. The second phase, planned to take place before the end of 2012, should introduce the client application part along with its plug-in modules, which are yet to be developed.

The introductory, theoretical part, which deals with the proposal of a new kernel solution for the Skeleton information system, has even been presented at an electronic conference held under the auspices of Tomas Bata University in Zlín [11]. The goal of this contribution was to present the importance of today's information systems in regard to their modularity and reusability.

I consider this thesis to be valuable, if not only for the real-world application of its subject. The analysis, proposal, and implementation have required much computer-programming experience and many hours of work. A new whole amount of experience, especially in relation to the .NET environment and the C# programming language,

has been earned during the work on the thesis. This I consider to be a great personal victory in my career.

## SEZNAM POUŽITÉ LITERATURY

- [1] NASH, Trey. *C# 2010: rychlý průvodce novinkami a nejlepšími postupy*. Vyd. 1. Brno: Computer Press, 2010, 624 s. ISBN 978-802-5130-346.
- [2] LACKO, Luboslav. *Silverlight: výukový průvodce tvorbou interaktivních aplikací*. Vyd. 1. Brno: Computer Press, 2010, 464 s. ISBN 978-802-5127-162.
- [3] AGARWAL, Vidya Vrat a James HUDDLESTON. *Databáze v C# 2008: průvodce programátora*. Vyd. 1. Preklad Lukáš Krejčí. Brno: Computer Press, 2009, 424s. ISBN 978-802-5123-096.
- [4] PIALORSI, Paolo a Marco RUSSO. *Microsoft LINQ: kompletní průvodce programátora*. Vyd. 1. Brno: Computer Press, 2009, 615 s. ISBN 978-802-5127-353.
- [5] PECINOVSKÝ, Rudolf. *Návrhové vzory*. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN978-802-5115-824.
- [6] WRÓBLEWSKI, Piotr. *Algoritmy: datové struktury a programovací techniky*. Vyd.1. Překlad Marek Michálek, Bogdan Kiszka. Brno: Computer Press, 2004, 351 s. ISBN 80-251-0343-9.
- [7] GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika. 2., preprac. a aktualiz. vyd.* Praha: Grada, 2009, 496 s. Expert (Grada). ISBN 978-80-247-2615-1.
- [8] Guidelines for Names. In: *MSDN* [online]. 2012 [cit. 2012-03-10]. Dostupné z: <http://msdn.microsoft.com/cs-cz/library/ms229002.aspx>
- [9] WPF XAML Browser Applications Overview. In: *MSDN* [online]. 2012 [cit. 2012-03-20]. Dostupné z: <http://msdn.microsoft.com/cs-cz/library/aa970060.aspx>
- [10] *PHP Manual* [online]. 2012 [cit. 2012-05-01]. Dostupné z: <http://www.php.net/manual/en/index.php>
- [11] VAŘACHA, Pavel a Michal NĚMEC. NÁVRH INOVACE INFORMAČNÍHO SYSTÉMU SKELETON. In: *INTERNET, COMPETITIVENESS AND ORGANIZATIONAL SECURITY: Process Management and the Use of Modern*. Tomas Bata University: Zlín, 2012, s. 196-203. ISBN 978-80-7454-142-1.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

CLR	Common Language Runtime
CMS	Content Management System
DLL	Dynamic-link library
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IS	Informační systém
IE	Internet Explorer
MSDN	Microsoft Developer Network
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
PHP	Hypertext Preprocessor
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XBAP	XAML Browser Applications

**SEZNAM OBRÁZKŮ**

Obr. 1. Komponenty informačního systému .....	14
Obr. 2. Model procesu .....	15
Obr. 3. Logiky aplikace .....	17
Obr. 4. Dvouúrovňový model klient/server .....	18
Obr. 5. Tříúrovňový model klient/server .....	19
Obr. 6. Metoda Request .....	23
Obr. 7. Komunikační rámec .....	23
Obr. 8. Zásuvné moduly serveru .....	24
Obr. 9. Strom uzlů .....	25
Obr. 10. Hlavní pracovní rámec .....	26
Obr. 11. Návrhový vzor MVVM .....	27
Obr. 12. Rozdělení do shromáždění a jmenných prostorů .....	30
Obr. 13. Závislosti modulů serverové strany .....	32
Obr. 14. Třídy služby serveru .....	33
Obr. 15. Třídy správce uzlů .....	34
Obr. 16. Třídy správce uživatelů .....	35
Obr. 17. Třídy správce sezení .....	35
Obr. 18. Schéma komunikace WCF .....	35
Obr. 19. Třídy komunikačního rámce .....	36
Obr. 20. Diagram zpracování požadavku .....	38
Obr. 21. Závislosti modulů klientské strany .....	43
Obr. 22. Třídy pracovního rámce .....	44
Obr. 23. Třídy atributů hlavního menu .....	45
Obr. 24. Třídy modelu pracovního prostoru .....	46
Obr. 25. Hlavní pracovní rámec .....	47
Obr. 26. Pracovní prostor .....	47
Obr. 27. Přiřazení view k modelu .....	49
Obr. 28. Informační dialog pracovního prostoru .....	49
Obr. 29. Chybový dialog pracovního prostoru .....	50
Obr. 30. Klientská aplikace – úvodní obrazovka .....	53
Obr. 31. Klientská aplikace – přihlášení .....	53
Obr. 32. Klientská aplikace – uvítací obrazovka .....	54
Obr. 33. Klientská aplikace – Testovací stránka .....	54
Obr. 34. Klientská aplikace – Uživatelské účty .....	55
Obr. 35. Klientská aplikace – Seznam událostí systému .....	55
Obr. 36. Testovací aplikace rozhraní služby serveru .....	56
Obr. 37. Testovací aplikace rozhraní služby serveru – kontrola zpráv .....	56

---

Obr. 38. Testovací aplikace rozhraní služby serveru – statistika .....	57
Obr. 39. Testovací aplikace rozhraní služby serveru – data žádosti .....	57
Obr. 40. Ukázka PHP klienta – hlavní stránka.....	58
Obr. 41. Ukázka PHP klienta – data .....	59
Obr. 42. Konzolová aplikace .....	62

**SEZNAM TABULEK**

Tab. 1. Rozdělení do shromáždění .....	28
Tab. 2. Rozdělení do jmenných prostorů.....	29
Tab. 3. Seznam uzlů systému .....	39
Tab. 4. Soubory PHP klienta .....	58

## SEZNAM PŘÍLOH

- P I. Popis stávajících aplikačních modulů informačního systému Skeleton
- P II. Úvodní studie
- P III. Kompletní schéma modulů systému
- P IV. Struktura přiloženého CD

## **PŘÍLOHA P I. POPIS STÁVAJÍCÍCH APLIKAČNÍCH MODULŮ INFORMAČNÍHO SYSTÉMU SKELETON**

Tato příloha obsahuje popis stávajících aplikačních modulů informačního systému Skeleton. Popis byl čerpán z <http://www.fssoftware.cz> (záložka *Základní informace*).

### **Jízdní řády**

V jízdnicích řádech je aplikována konstrukční a grafická technologie zaměřená na uživatelský komfort při návrhu všech typu jízdnicích řádu. Technologie umožňuje snadný návrh jízdnicích řádu pro Internet, dispečery, vozidla, stanice a kapesní jízdnicí rády. Jízdní rády odpovídají náročným kritériím designérů i cestujících.

### **Plán a skutečnost služeb řidičů**

V současné době je veškerý provoz orientován na spolehlivou orientaci a organizaci práce s jízdnicími pracovníky. Plán zajišťuje dokonalý přehled o všech výkonech prováděných v provozu jízdnicími pracovníky i samotnými vozidly. Modul zajišťuje ideální plánování provozu a jeho realizaci. Plán se postupně stává realitou a vytváří všechny nezbytné doklady.

### **Plán a skutečnost služeb dílen**

Provoz dílen v nepřetržitých provozech je zajišťován za pomoci pravidelného oběhu pracovníku po službách. Cyklus umožňuje kvalitní pokrytí potřeb jednotlivých provozů. Pro dodržení všech požadavků zákona je zajištěna dokonalá souhra všech údajů a zajištěn plný a ucelený přehled všech norem práce.

### **Dispečerský deník**

Prostředek určený k práci dispečerů bez ohledu na jejich organizační zaměření. Tuto variabilitu umožňuje vlastní programovatelné prostředí. Z této informace plyne možnost nastavení vlastního uživatelského rozhraní každému uživateli v závislosti na jeho pracovní specifikaci. Zobrazuje informace o výpravě a denní události v provozu.

### **Detailní statistiky**

Generátor provozních podkladů pro přesné zpracování dopravních charakteristik vytvořených v konstrukci a plánování dopravy. Detailní statistiky věnují pozornost i podrobnému vyhodnocení a porovnání provozních parametrů z již uskutečněného provozu.

### **Mapy**

Jednotlivé druhy map zajišťují podle druhu určení geografickou či schématickou orientaci v zajišťovaném provozu. Mapy dělíme podle jejich zaměření na mapy určené konstruktérům a provozním pracovníkům. Ty první zajišťují dokonalou

simulaci plánovaného provozu a druhé přesně popisují stav prováděné dopravy, případně dopravy již vykonané. Vše je zajištěno z mnoha úhlu pohledu.

#### **Vymáhání pohledávek od černých pasažéru**

Jedním z největších problémů veřejné dopravy je přeprava neplatících pasažéru. Modul zabývající se evidencí a technologickým postupem při vymáhání dlužných částek pomáhá zabezpečit správný postup při vymáhání, získat dokonalý přehled o jednotlivých pohledávkách a zajistit v souladu s používaným předprodejem maximální ochranu vůči těmto neplatícím. Postup je veden od základních upomínek až po exekuce na majetek.

#### **Předprodej**

Prostředek pro zajištění detailní evidence prodaného časového jízdného a evidencí výhod pro cestující veřejnost. Předprodej zajišťuje i prodej různého zboží včetně jízdenek smluvním prodejcům a evidenci jízdenek určených pro doplňkový prodej u řidičů. Jsou vytvořeny všechny podklady a výměny dat v rámci IDS (integrovaných dopravních systémů) mezi jednotlivými dopravci a koordinátorem dopravy.

#### **Technické prohlídky**

Technické prohlídky jsou zprostředkovatelem dokonalého přehledu o pohonných hmotách podle mnoha kritérií. Zajišťují informace o pneumatikách a všech kapalinách ve vozidlech. Na základě zakázkového systému vzniká přehled o technickém stavu provozovaných vozidel, nebo jednotlivých agregátů. V celém systému technických prohlídek je zabezpečen perfektní přehled o finanční náročnosti jednotlivých vozidel.

## **PŘÍLOHA P II. ÚVODNÍ STUDIE**

### **Úvodní studie jádra informačního systému Skeleton**

Cílem je vytvořit jádro modulárního distribuovaného informačního systému. Systém se bude dělit na dvě základní části, server a klient. Do obou těchto částí bude možno „zasouvat“ jednotlivé moduly, které budou implementovat skutečnou funkcionalitu daného informačního systému. Jádro systému bude pouze spouštěcí prostředí těchto modulů a bude definovat potřebné rozhraní a funkce pro tyto moduly. V části serveru budou spouštěny moduly s takzvanou business logikou a v klientské části moduly s uživatelským rozhraním a minimem logických funkcí. Obě části jádra systému budou spojena rozhraním služby WCF. Serverové a klientské moduly budou spolu komunikovat jen za pomoci funkcí, které jim poskytne samotné jádro systému. Jádro systému musí zachytit všechny neošetřené výjimky z modulu a přívětivým způsobem je oznámit uživateli.

### **Požadavky na server**

Server bude poskytovat komunikační rozhraní na platformě služby WCF. V prvotní fázi bude rozhraní služby WCF tvořit jedna synchronní metoda, která bude přijímat jako vstupní parametr název uzlu a množinu parametru, které budou předány do metody uzlu definované v daném modulu. Server bude mít také vlastní modul, který bude prezentovat potřebné uzly pro správu serveru.

### **Správou serveru se rozumí:**

1. Správa uživatelských účtů – založení, úprava, smazání, přihlášení, odhlášení.
2. Informace o běhovém prostředí – seznam prezentovaných uzlů, seznam přihlášených uživatelů.
3. Logování událostí – přístupy uživatelů, chybové události.
4. Správa plánovače – vytvoření, spuštění, zastavení, smazání a stav úkolu.
5. Generování statistik pro jednotlivé volání uzlu – rozlišení na session i uživatele.

Server bude dále držet správu nad uživatelskými účty, rolemi a jednotlivými session uživatelů. Ke každému uživateli i k jeho session bude generovat statistiku volání uzlu a udržovat detailní statistiku volání posledního uzlu (čas, paměťové nároky . . .). Systém bude poskytovat dva defaultní uživatele – Anonymous, pro identifikaci všech nepřihlášených uživatelů a Root, pro výhradní správu systému.

Systém umožní vytváření, mazání a úpravu uživatelských účtů a rolí. Přístupová práva budou mít stromovou strukturu dědění a budou se řídit stavy *Povolit*, *Odepřít* a *Ne-definováno*. Uživatelská práva bude možno definovat k jednotlivým rolím i uživatelům. Jednotlivým uživatelům bude možno přiřadit i více rolí, případně žádnou.

### **Požadavky na klienta**

Klient bude zastřešovat v podobě kontejneru jednotlivé uživatelské rozhraní modulu. Na žádost klientského modulu vytvoří pracovní prostor pro uživatelské rozhraní předané modulem a bude držet správu nad těmito pracovními prostory. Bude poskytovat funkce pro snadné integrování modulu do systému.

### **Funkcemi se rozumí:**

1. Přístup ke komunikačnímu rozhraní služby WCF.
2. Vytvoření pracovního prostoru rámcového uživatelského rozhraní.
3. Uložení a načtení uživatelských nastavení a konfigurací.
4. Zápis do logu.

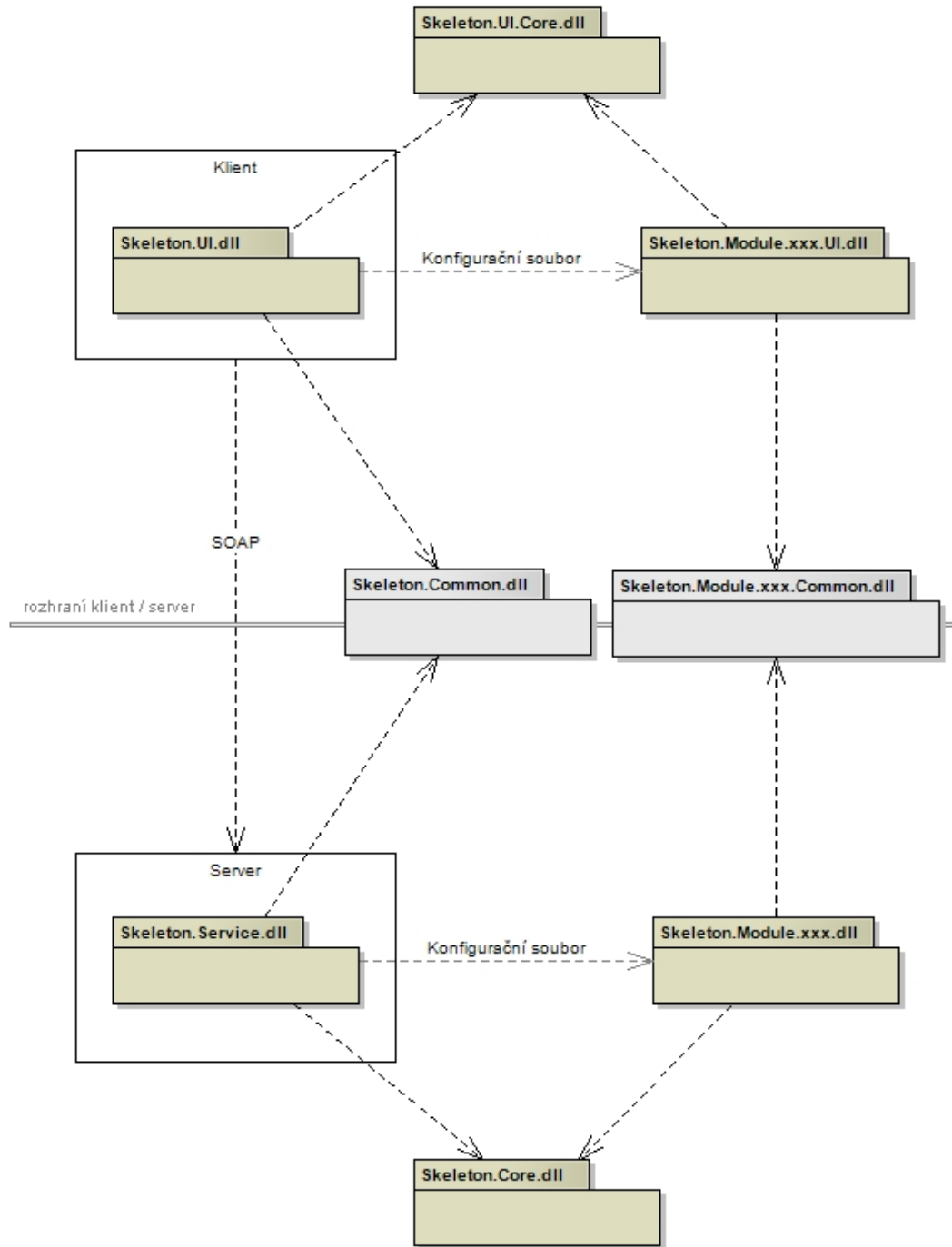
### **Požadavky na serverový modul**

V modulu za pomoci atributu budou označeny jednotlivé uzlové metody. Modul bude pro přístup k externím datům využívat funkce serveru (brány serveru). Moduly budou z pohledu systému vytvářeny jako pasivní.

### **Požadavky na klientský modul**

V modulu za pomoci atributů budou označeny jednotlivé příkazové metody. Příkazové metody budou nejčastěji vytvářet jednotlivé pracovní prostory uživatelských rozhraní. Modul bude striktně dodržovat standarty návrhového vzoru MVVM. Modul bude vytvořen s ohledem na možnou lokalizaci.

# PŘÍLOHA P III. KOMPLETNÍ SCHÉMA MODULŮ SYSTÉMU



## PŘÍLOHA P IV. STRUKTURA PŘILOŽENÉHO CD

Přiložené CD obsahuje tyto adresáře (složky):

`bin` projekt připravený ve spustitelné podobě

`src` zdrojové kódy projektu

`doc` všechny dokumenty, které byly použity při tvorbě tohoto projektu

`install` instalační balíček programu (podrobnosti k instalaci jsou uvedeny v kap. 9.)