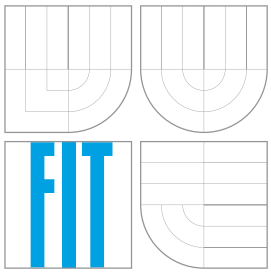
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KONFIGUROVATELNÝ TELEMETRICKÝ A DOHLEDOVÝ SYSTÉM

CONFIGURABLE TELEMETRIC AND MONITORING SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — PETR VOBORNÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ RYŠAVÝ, Ph.D.

SUPERVISOR

BRNO 2008

Abstrakt

Cílem bakalářské práce je návrh a implementace základu modulárního systému pro telemetrické a zabezpečovací účely. Systém se skládá z modulů pro zpracování příchozích dat, webového administračního a prezentačního rozhraní a souboru webových služeb. Systém je implementován v jazyce C# a pomocí technologií .NET Framerwork a ASP.NET

Klíčová slova

telemetrický systém, zabezpečovací systém, modul, modulární systém, telemetrie, bezpečnost, informační systém, C#, ASP.NET

Abstract

A goal of this thesis is design and implementation of modular system for telemetry and security purposes. System consists of modules for processing incoming data, web administration and presenation interface and set of web services. For implementation is used C# language along with .NET Framework and ASP.NET.

Keywords

telemetric system, security system, module, modular system, telemetry, security, information system, C#, ASP.NET

Citace

Petr Voborník: Konfigurovatelný telemetrický
a dohledový systém, bakalářská práce, Brno, FIT VUT v Brně, 2008

Konfigurovatelný telemetrický a dohledový systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ondřeje Ryšavého, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Voborník
13. května 2008

Obsah

Kapitola 1

Úvod

Žijeme v době, kdy technika je nedílnou součástí života. Připojení k Internetu je každým rokem dostupnější. V roce 2007 počet domácností připojených k Internetu vzrostl o 5 % na celkových 32 %. [1] Tato tendence nabízí tvorbu celé řady nových služeb dostupných pomocí Internetu. Mezi tyto služby lze zařadit oblast telemetrie,vzdáleného řízení a zabezpečení. Největší zastoupení v této oblasti tvoří rozsáhlé systémy s velkým množstvím měřících míst stavěné na zakázku pro teplárenské společnosti, společnosti mající na starost správu vodních toků aj.

V oblasti telemetrie popř. zabezpečení lze nalézt zákazníky, kteří potřebují zpracovávat data z malého množství měřících míst. Může se jednat např. o sledování malých vodních elektráren nebo zabezpečení chat či domácností. Využití rozsáhlých systémů pro tyto potřeby je často nevhodné a zbytečně nákladné.

Tímto se otevírá možnost vytvořit systém určený pro tuto cílovou skupinu. Jako většina produktů by měl být levný, jednoduchý na obsluhu a dostupný kdykoliv pomocí Internetu.

Často bývá nejdůležitějším požadavkem nízká cena, bohužel tento požadavek nelze uskutečnit, pokud by měl být systém budován zákazníkovi na míru.

Aby byl systém kdykoliv dostupný prostřednictvím Internetu, musí být provozován na serveru běžícím neustále a připojeným dostatečně rychlou linkou. Domácí připojení se k tomuto účelu z rychlostních či dostupnostních problémů většinou nehodí. Další možností je využít služeb serverhousingu a server provozovat tam. Avšak i tato možnost přináší nevýhody v podobě velkých nákladů na pronájem místa a linky.

Pro splňení všech tří požadavků je nutné vytvořit dostatečně univerzální systém, aby byl použitelný k celé řadě různých účelů. Zároveň by měl běžet na serveru využitelný více uživateli. Tím se mezi ně rozloží náklady a navíc se často využije výpočetní výkon, který by pravděpodobně nebyl využit. Takový systém je předmětem této práce.

Kapitola 2

Architektura systému

2.1 Požadavky na systém

Cílem práce je navrhnout a implementovat základ systému pro sběr a zpracování dat od telemetrických stanic (dále jen jednotek). Data by měla být přenášena po Internetu. Systém by měl bý schopný komunikovat s jednotkami různých typů od různých výrobců. Při přijmu dat vyhovujících definovaným mezím by měl systém vykonat definované akce.

Součástí systému by mělo být webové uživatelské rozhraní umožňující administraci systému a zobrazení naměřených dat v rozvržení definovaném uživatelem. Rozhraní by mělo být víceuživatelské s možností filtrace obsahu dle uživatelských oprávnění.

Pro spolupráci s jinými projekty by měl systém obsahovat sadu webových služeb poskytující naměřená data. Jméno systému nebylo stanoveno, proto byl pracovně nazván *CTMS*– dle iniciálů názvu práce v angličtině.

2.2 Složení systému a komunikace s okolím

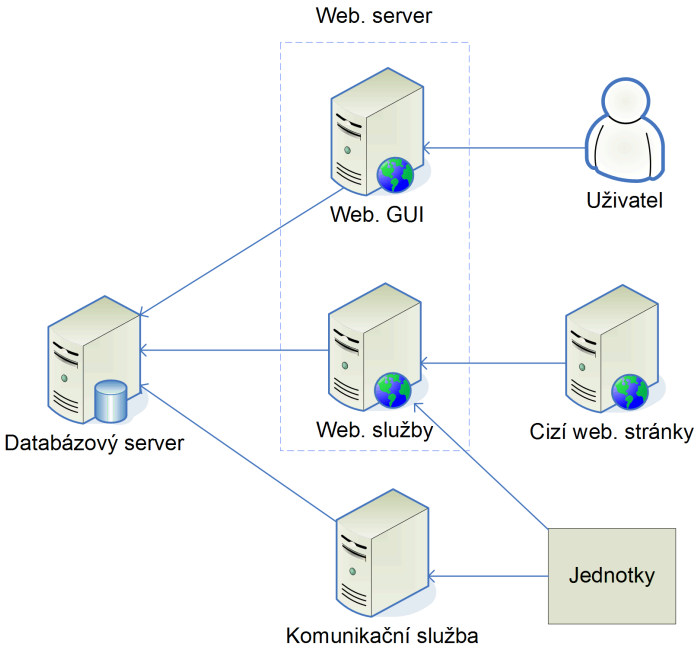
Dle požadavků byl navržen systém skládající se z:

- Webového uživatelského rozhraní (GUI)
- Webových služeb
- Komunikační služby
- Databázového serveru

Systém komunikuje s:

- Uživateli
- Cizími webovými stránkami
- Jednotkami

Složení systému a komunikaci popisuje obrázek 2.1.



Obrázek 2.1: Architektura systému

2.2.1 Komunikace s jednotkami

Cílem systému je, aby byl schopný komunikovat se všemi jednotkami podporovaných typů připojených k Internet. Bohužel v současné době mnoho poskytovalů internetového připojení (ISP) poskytuje připojení, kdy je zákazník připojen do privátní sítě poskytovatele a komunikace do internetu probíhá prostřednictvím brány, kde je využíváno překladu síťových adres (`NAT`). Pokud ISP neumožňuje zakoupení veřejné IP adresy, nebo přesměrování určitých portů, nemůže se nikdo z internetu připojit na zařízení u zákazníka. Proto při komunikaci s jednotkami systém plní funkci serveru, který čeká na příchozí komunikaci inicializovanané jednotkou.

2.3 Třívrstvá architektura

Dle obr. 2.1 komunikační podsystémy využívají pro ukládaní dat společnou databázi. Data uložená v databázi představují stav systému. Protože zmiňované podsystémy jsou součástí stejného systému, měly by jeho stav měnit dle společných pravidel. Je tedy vhodné využít stejného prostředníka, pro všechny podsystému. Pro zajištění tohoto principu byl vybrán model třívrstvé architektury [6]

Třívrstvá architektura je tvořena prezenční vrstvou, aplikační vrstvou a datovou vrstvou. Části prezenční vrstvy pracují s objekty aplikační vrstvy. Aplikační vrstva provádí logiku systému s objekty tvořící abtraktní modely reálných částí systému. Pro manipulaci

s daty a převod dat na objekty aplikační vrtvy slouží datová vrstva.

Při aplikaci na vyvíjený systém lze říci, že komunikační podsystémy tvoří prezenční vrstvu, společný prostředník aplikační vrstvu a databáze datovou vrstvu. Aby se aplikační vrstva domluvila s databází, musí datová vrstva obsahovat ještě prostředníka, který implementuje komunikační rozhraní s aplikační vrstvou.

Rozdělením systému na tři vrstvy dosáhneme těchto výhod:

- Jednodušší implementace komunikátorů (Využívají sadu abstraktních modelů systemů).
- Změny v aplikační logice pro všechny komunikátory se prováději na jednom místě.
- Nezávislost komunikátorů na konkretním typu databáze a tím možnost snažší výměny databáze.

2.4 Implementační technologie

Již v zadání je uvedeno, že jako implementační technologie se využije *.NET Framework* a jazyk *C#*. Volba této technologie je výhodná, protože umožňuje pomocí jednoho jazyka vyvíjet služby běžící na pozadí, webové stránky pomocí technologie *ASP.NET*, webové služby pomocí technologie *ASP.NET Web Services* a dynamické knihovny využívané všemi předchozími technologiemi.

Jednotlivé vrstvy, popř. subsytémy z pohledu typu aplikací:

- Aplikační vrstva tvoří projekt kompilovaný do *.NET knihovny.*
- Jako databáze je využit *Microsoft Sql Server 2005.*
- Prostředník mezi databází a aplikační vrstvou je pro *Sql Server 2005* součástí knihovny aplikační vrstvy. Při použití jiného *SŘBD* by byla potřeba vytvořit knihovna s vlastním prostředníkem.
- Webové uživatelské rozhraní tvoří *ASP.NET Web Application.*
- Webové služby využívají technologie *ASP.NET Web Services*, mohou být součástní aplikace webového uživatelského rozhraní.
- Komunikační služba je tvořena jako služba systému Windows (*Windows Service*).

Při vývoji bylo využito *Microsoft Visual Studio 2008* a *Microsoft Sql Server Management Studio.*

2.5 Požadavky na hostitelský systém

Dle využitých implementačních technologií by hostitelský systém měl obsahovat:

- *Microsoft .NET Framework 3.5*
- Webový server s možností provozu *ASP.NET 3.5* aplikací např. *Internet Information Services 6.0* a vyšší
- *Microsoft Sql Server 2005* nebo jiný kompatibilní databázový server

Z hlediska rozložení výkonu lze celý systém provozovat na jednom počítači, nebo na více počítačích. Samostatně mohou běžet všechni části systému uvedeny v 2.1.

Kapitola 3

Aplikační vrstva

Aplikační vrstva zabezpečuje logiku systému a poskytuje objekty, které tvoří modely reálných částí systému.

Při popisu objektů systémů, se často odkazuje na rozhraní, které daný objekt definuje. Všechny podsystémy aplikační vrstvy se nacházejí ve jmenném prostoru `Ctms.Engine`.

3.1 Konceptuální model

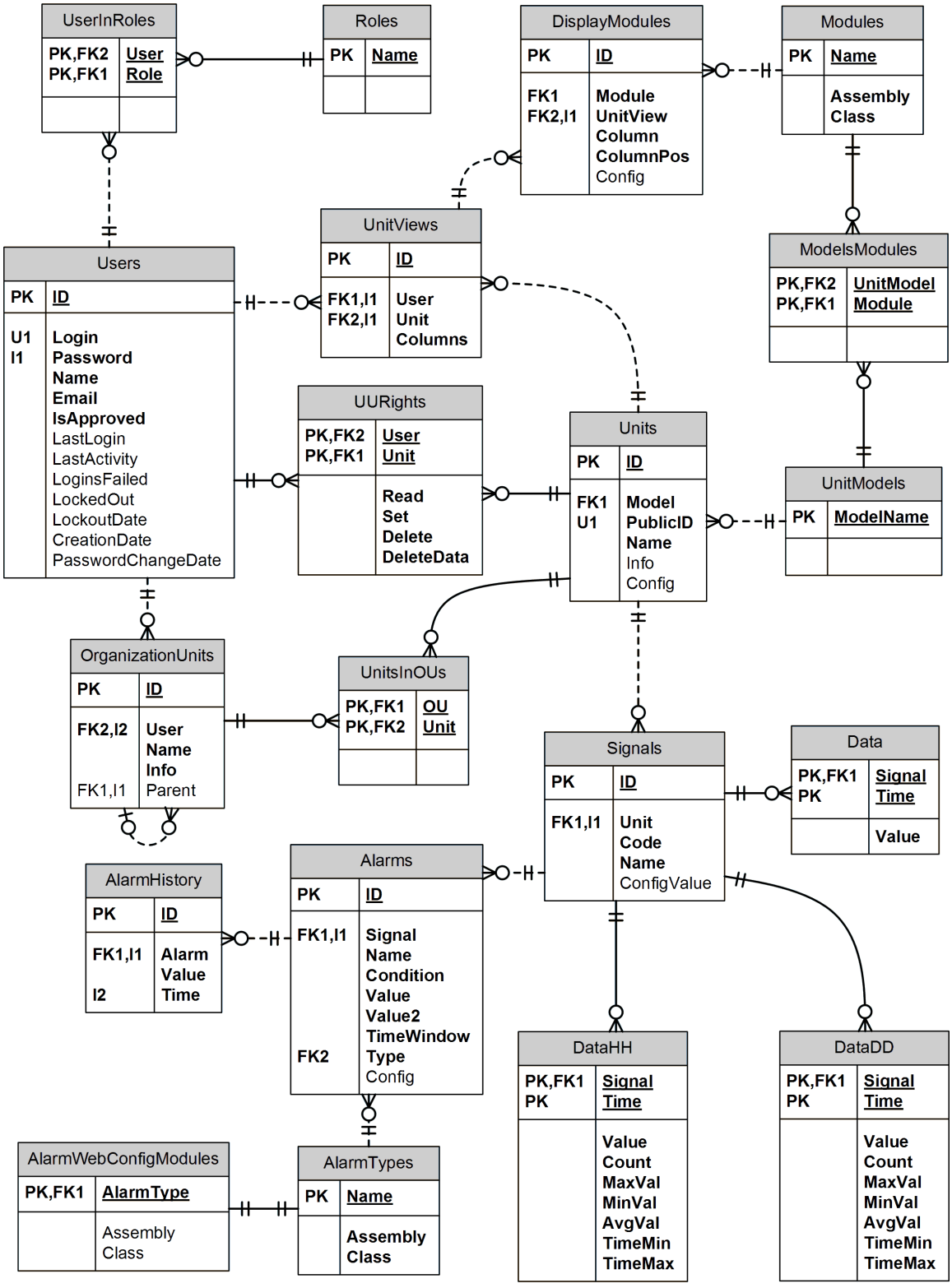
Při návrhu systému byl vytvořen konceptuální model, který slouží pro popis systému. Umožňuje lepší pochopení vztahů mezi částmi navrhovaného systému. Je tvořen ER diagramem.

Konceptuální model ve formě ER diagramu je na obrázku 3.1. Vztahy typu M:N mezi entitními množinami byly převedeny na dva vztahy 1:N a pomocnou spojovací entitní množinu. Jedná se o tyto entitní množiny: `UserInRoles`, `UnitsInOus` a `UnitModels`. V diagramu jsou vyznačeny indexy (položky s označením *In*). Uvedené doplňující vlastnosti transformují ER diagram na diagram použitelný pro návrh databáze.

3.2 Kontext

Aby byl systém v budoucnosti snadněji upravitelný, byla snaha vytvořit třídy co nejvíce nezávislé na konkrétních implementacích jiných tříd. Proto třídy implementující chovaní modelované entitní množiny implementují rozhraní odpovídající dané entitní množině. Veškerá komunikace mezi objekty těchto tříd probíhá jen přes daná rozhraní. Tento přístup přináší jisté nevýhody. Např. když objekt potřebuje vytvořit instanci jiného objektu, tak neví jak to provést, protože nezná jeho třídu. Z tohoto a dalších důvodů obsahují objekty systému referenci na objekt nazývaný kontext. Kontext by měl implementovat rozhraní `ICtmsContext`.

Kontext obsahuje reference na podsystémy systému. Jeho účelem je poskytování těchto referencí. Každý objekt a podsystém aplikační vrstvy je vybaven referencí na kontext, proto



Obrázek 3.1: ER. diagram

může komunikovat s ostatnímy podsystémy systému.

Rozdělení systému na podsystémy a komunikace pouze přes definovaná rozhraní zajišťuje lepší rozšiřitelnost v budoucnu např. v případě, že by různá použití systému vyžadovala různé chování podsystémů.

Následující podkapitoly popisují objekty vrácené vlastnostmi kontextu.

3.2.1 Uživatel

Vlastnost kontextu `CurrentUser` představuje aktuálního uživatele systému, Díky tomu lze získat objekty závislé na aktuálním uživateli. Při požadavku na konkrétní objekt (viz. 3.2.4) slouží pro vyhodnocení oprávněnosti získání požadovaného objektu.

3.2.2 Továrna objektů

Vlastost kontextu `Factory` vrací istanci třídy s rozhraním `ICtmsFactory`. Její účel je vytvářet nové objekty systému. Proto jako jediná v systému zná konkrétní třídy objektů. Pro daný účel obsahuje sadu továrních metod (návrhový vzor Tovární metoda viz. [5]). Každému objektu systému po jeho vytvoření nastavuje svůj kontext. Díky tomu všechny objekty ve zpracovávaném požadavku sdílejí stejný kontext.

3.2.3 Přístup k datové vrstvě

Vlastnost kontextu `DataProvider` vrací objekt s rozhraním `ICtmsDataProvider`. Tento objekt představuje z pohledu aplikační vrstvy vrstvu datovou. Je využíván pokud je potřeba získat popř. uložit objekt z/do databáze.

3.2.4 Podsystémy pro získávání objektů

Řada objektů, nebo prvky uživatelského rozhraní mohou vyžadovat získání konkrétního objektu (myšleno jako objekt systému, inicializovaný na základě identifikátoru, nebo jiných atributů). Protože objekty neznají konkrétní třídy obsahující statické metody navracející požadované objekty, obsahují podsystémy kontextu řadu metod, navracející požadovaný objekt.

Tyto podsystémy jsou získatelné vlastnostmi kontextu `UserManager`, `UnitManager`, `AlarmManager`, `DataManager` a `ModuleManager`. Implementují jim příslušná rozhraní. Každý podsystém má na starost vrácení požadovaných konkrétních objektů, nebo sadu objektů, popř. funkce dle jejich zaměření.

Další funkcí těchto podsystémů je zajištění existence vždy jen jedné instance objektu s určitým identifikátorem. Důvodem je, aby všechny objekty, kterým přísluší daný objekt, se odkazovaly na společný objekt a ne několik jeho kopií. Tímto se předejde vykonání zbytečných dotazů do databáze.Pro zajištění této funkcionality slouží registrační metody. Objekt při své inicializaci zavolá registrační metodu, ta uloží jeho identifikátor a referenci do struktury pro to určené. V implementaci se používají hashovací tabulky. Při

následném dalším požadavku pro získání objektu subsystém zjistí, zda se v hashovací tabulce nalézá záznam, kde klíč představuje identifikátor objektu. Pokud ano, je požadovaný objekt navrácen, v opačném případě se musí načíst z databáze.

Dalším úkolem podsystémů je ověření, zda uživatel má práva pro přístup k požadovanému objektu. Pokud práva nemá, tak požadovaný objekt není vrácen.

Příklad získání uživatele s identifikátorem `User_ID` je popsán sekvenčním diagramem na obr. 3.2. V diagramu je znázorněna registrace objektu a opětovný dotaz na objekt se stejným identifikátorm. Hashovací tabulku představuje objekt `UserCache`. Kontrola oprávnění byla pro zjednodušení vynechána.

3.2.5 Zaznamenávání chyb

Systém se skládá z různých podsystémů, který využívají řadu různých modulů. Každý podsystém nebo modul může generovat různé chyby např. z důvodu špatné konfigurace. Aby měl administrátor systému přehled o takových chybách, je do systému zabudována podpora pro jednotné zaznamenávání chyb. Zajišťuje ji podsystém definovaný rozhraním `IErrorLog`.

3.2.6 Konfigurace specifické pro prostředí

Podsystémy, jako jsou akce alarmů, mohou požadovat konfigurace společné pro instance jejich využití (v tomto případě Alarmy). Tyto konfigurace mohou být rozdílné pro konkrétní prostředí – např. pokud je akce alarmu spouštěna komunikační službou, nebo pokud je spouštěna webovou službou ve funkci komunikačního modulu, umístěnou na jiném počítači.

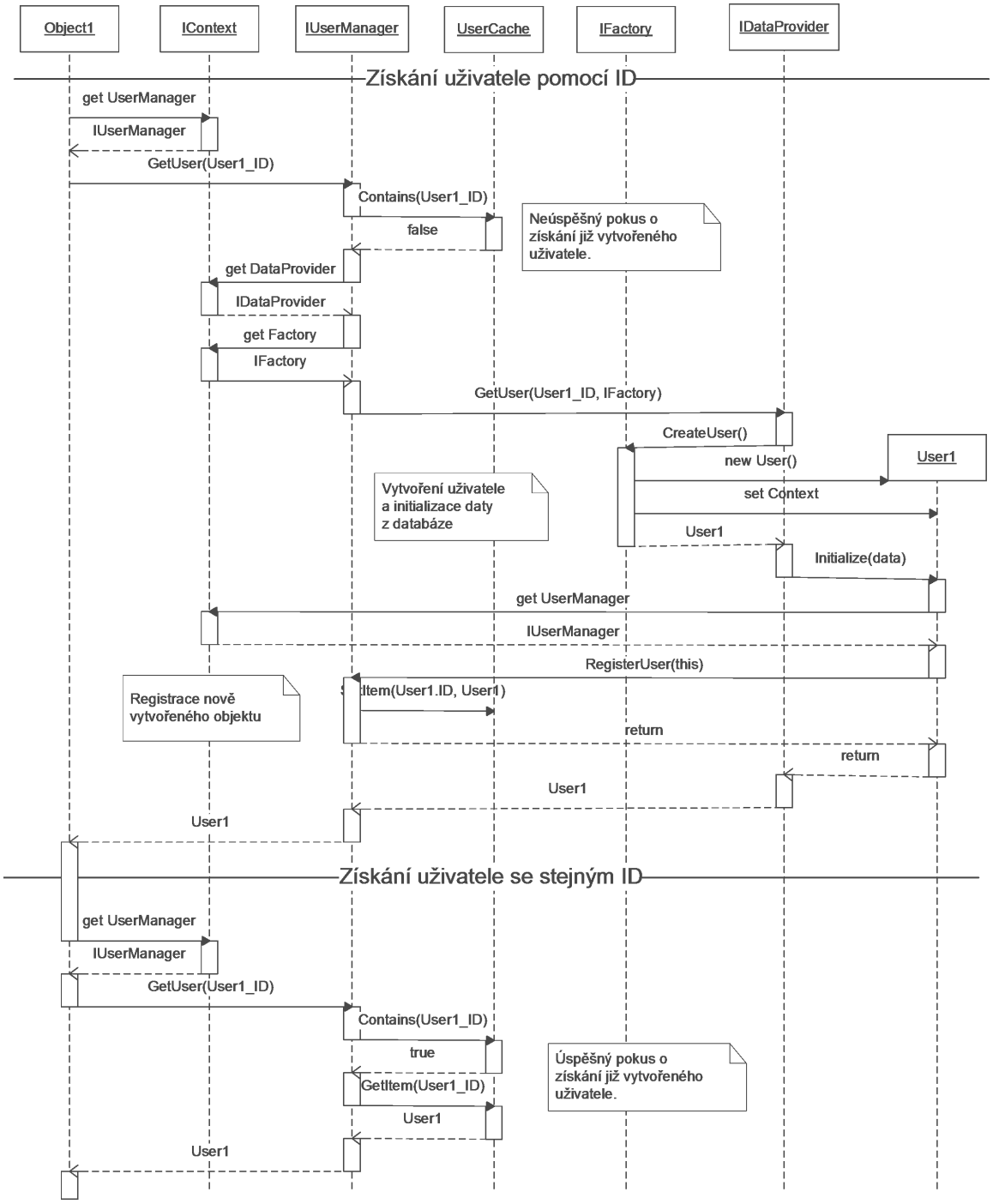
Proto kontext obsahuje vlastnost `Configuration` (rozhraní `IConfiguration`), která poskytuje získání konfiguračních hodnot pro dané prostředí. Hodnoty jsou ve formátu dvojice klíč – hodnota, klíč definuje název konfigurační proměnné a hodnota její obsah. Pro jednoduché konfigurace by tato funkcionalita měla dostačovat. V případech, kdy jsou požadována data se složitou strukturou, lze tento systém využít pro uložení názvu souboru, který obsahuje požadovanou strukturu dat.

3.3 Jednotky

Jednotky tvoří abstraktní model reálných vzdálených zařízení. V konceptuálním modelu je představuje entitní množina `Units`. V systému jsou definovány rozhraním `IUnit`.

Aby byla jednotka indentifikovatelná při získávání dat od fyzické jednotky, musí mít fyzická jednotka jedinečný identifikátor, který uvede v komunikaci se systémem. Tento identifikátor tvoří vlastnost `PublicId`.

Přístup k jednotkám je omezen na uživatele s příslušnými právy. Je to modelované entitní množinou `UURights`. Práva mohou být: čtení, modifikace, mazání jednotky a mazání



Obrázek 3.2: Příklad získání objektu – uživatele

dat. Oprávnění pro jednotku dále řídí oprávnění pro přístup k objektům patřící pod jednotku tj. k jejím signálům a jím souvisejícím alarmům a datům.

Jednotky mohou být různých typů od různých výrobců. V závislosti na typu jednotky může být požadovanáno nastavení, jehož strukturu nelze předem předvídat. Proto jed-

notka obsahuje vlastnost `Model` určující typ jednotky a vlastnost `Config` pro uložení nastavení o neznámé struktuře. `Config` je realizován jako řetězec obsahující xml dokument s požadovanou strukturou. K jeho nastavením by měl sloužit zobrazovací modul (viz. 5.2.5) k tomu určený. XML schéma dokumentu není pevně stanoveno z důvodu výše zmíněných různých požadavků na strukturu dat. Komunikaci s jednotkou by měl zajišťovat jen jeden typ komunikační služby. Ta by si měla ověřit, zda načtený dokument vyhovuje XML schématu, který dokáže zpracovat. Konfigurační modul by měl tedy ukládat konfiguraci ve formátu vyhovujícím XML schématu komunikační služby.

3.4 Signály

Signály reprezentují měřicí prvky jednotky např. teplotní čidlo, strážkoměr, detektor pohybu. Definuje je rozhraní `ISignal`, v konceptuálním modelu entitní množina `Signals`. Každý signál je svázán s právě jednou jednotkou. Při přijímání dat se identifikuje v rámci jednotky unikátním jménem, které nemusí být unikátní v rámci celého systému - 2 jednotky mohou mít stejné názvy signálů. K signálu může být přiřazena množina alarmů.

Pokud signál reprezentuje měřící prvek schopný rekonfigurace na základě číselné hodnoty, tak lze nastavit požadovanou hodnotu ve vlastnosti `ConfigValue`. Její hodnota by měla být vybrána a poslána jednotce při prvním požadavku o rekonfiguraci. Pokud typ jednotky vyžaduje, aby již poslaná rekonfigurační hodnota nebyla poslána znova, obecně pokud již není potřeba, lze ji nastavit na hodnotu `Double.NaN` značící stav `nenastaveno`. V databázi je to řešeno hodnotou `null`.

3.5 Alarmy

Představují akce vyvolané splněním podmínky při příjímání nových dat. Jsou definovány rozhraním `IAlarm`, v konceptuálním modelu entitní množinou `Alarms`. Každý alarm je přiřazen právě k jednomu signálu, jehož příchozí data kontroluje. Při splnění podmínky spustí definovanou akci. Systém uchovává historii vyvolaných akcí (entitní množina `AlarmHistory`) pro pozdější využití.

Množina dostupných akcí tvoří modulární systém. Každá akce představuje jeden modul. Moduly by měly být implementované jako třídy s rozhraním `IAlarmAction`. Množina modulů použitých v systému představuje entitní množina `AlarmTypes`. Ta uchovává informace o tom, která třída v jaké assembly implementuje modul s daným jménem.

Každý modul může vyžadovat konfigurační hodnoty rozdílné struktury od ostatních modulů. Aby bylo zaručeno dostatečně univerzální ukládání konfigurace, obsahuje rozhraní `IAlarms` vlastnost `Config`, která představuje řetězec obsahující dokument xml. Modul tento řetězec může zpracovat dle jeho preferencí např. použít xml deserializaci, nebo zpracovat pomocí XmlReaderu.

3.6 Organizační jednotky

I když se předpokládá, že systém bude využíván uživateli spravující malý počet jednotek, je do systemů zabudována podpora pro organizaci jednotek do organizačních jednotek (dále jen OU, kvůli možné záměně s jednotkou ve smylu vzdáleného zařízení).

Cíl OU je zpřehlednit uživatelské prostředí při velkém počtu jednotek. Vytvářejí strukturu, kterou je možné přirovnat k adresářím na disku, kde adresáře tvoří OU a soubory představují jednotky. OU tvoří kontajner pro jiné OU a jednotky. Každá OU je svázána právě s jedním uživatelem. OU může obsahovat libovolné množství jiných OU a jednotek. Počet rodičovských OU, tj. OU, která obsahuje vybranou OU, je omezen na jednu nebo žádnou. OU, která nemá rodičovskou OU se nazývá kořenová. Vytvářejí nejvyšší úroveň OU, kterou lze přirovnat k adresářům v adresáři `/` v OS typu Linux.

Přestože jednotka může být sdílena více uživateli, OU mají vždy vztah mezi jedním uživatelem a jednou jednotkou. Neexistuje možnost, aby dva uživatelé sdíleli stejnou OU. Tím se mírně zjednodušší systém – nemusí se řešit konflikty při různých uživatelských oprávnění na jednotky obsažené v potenciálně sdílené OU.

Předchozí text lze shrnout do tvrzení, že OU představují mechanismus, nezávislý na na zbytku systému, kterým si uživatel organizuje jednotky.

3.7 Data

Data představují zaslané naměřené hodnoty. Dělí se na změřená (entitní množina `Data`), hodinové agregace (entitní množina `DataHH`) a denní agregace (entitní množina `DataDD`). V systému je definuje rozhraní `ICtmsData` popř. `ICtmsAggregatedData`.

Data očekávaná systémem mají povahu čísel s plovoucí řádovou čárkou získané měřením analogové veličiny nebo exaktních hodnot, daných výčtem stavů, získaných ze zabezpečovacích aj. zařízení.

Systém pro práci s vlastními hodnotami využívá typ `Double`. Tento typ umožňuje uložit největší rozsah hodnot s plovoucí řádovou čárkou nabízený standardními numerickými typy *.NET Frameworku* a zároveň je dostatečně přesný pro malé exaktní hodnoty. Práci s vektory lze řešit využitím více signálů, kde každý signál představuje jeden rozměr vektoru.

Agregovaná data obsahují statistické informace pro agregovaný interval naměřených dat. Zavádějí do systému redundaci, ale jejich vyhodnocování je rychlejší. Jsou použitelné např. při analýze dat za větší časové období, kdy není potřeba velká hustota informací jako v případě naměřených dat.

O příjem dat se starají komunikační moduly viz. 5.1.

Kapitola 4

Datová vrstva

Převádí data z datového zdroje na abstraktní objekty aplikační vrstvy. Provádí veškeré operace s datovým zdrojem. Skládá se z poskytovatele dat a datového zdroje.

4.1 Poskytovatel dat

Je třída, která tvoří prostředníka mezi aplikační vrstvou a datovým zdrojem definována rozhraním `ICtmsDataProvider`.

V aktuální verzi toto rozhraní implementuje třída `CtmsMsSqlDataProvider`, která může jako datový zdroj využít *MS SQL Server 2005* popř. novější kompatibilní verzi. Pro získání dat využívá uložené procedury.

Při převádění dat na objekty aplikační vrstvy je potřeba objekty vytvořit a inicializovat. Jelikož poskytovatel dat ví jaké rozhraní by měl mít daný objekt, ale neví jaké je třídy, musí mu aplikační vrstva poskytnou objekt schopný vytvořit požadované objekty. Předává mu proto v parametru objekt s rozhraním `ICtmsFactory` popsaný v 3.2.2, který obsahuje požadované tovární metody. Ukázka vytvoření a inicializace objektu aplikační vrstvy je součástí ukázky získání objektu uživatele na obr. 3.2.

Dalšímy účely poskytovatele dat je vkládaní, aktualizace a mazání dat v databázi. Slouží pro to metody, které přijímají jako parametr objekt aplikační vrstvy. Z vlastností předaného objektu poskytovatel dat vytvoří parametry pro uloženou proceduru vykonávající požadovanou činnost. Nakonec proceduru vykoná a navrátí úspěšnost operace.

4.2 Návrh databáze

Datový zdroj představuje úložiště dat systému. V aktuální verzi je použit *Microsoft Sql Server 2005*. Schéma databáze odpovídá ER diagramu na obr. 3.1. Názvy entitních množin odpovídají názvům tabulek. Výjimku tvoří množiny `AlarmTypes` a `AlarmWebConfigModules`, které jsou ve vztahu 1:1, proto mohou být součástní jedné tabulky. V diagramu jsou vyznačeny primární klíče (PK), cizí klíče (FK) a indexy (I).

4.2.1 Uložené procedury

Systém využívá pro přístup k datům uložené procedury místo klasických dotazů. Z hlediska výkonu tento přístup je výhodnější, protože SŘBD procedury jednou zkompiluje a poté si tzv. prováděcí plán (execution plan) drží v cache. Při použití již zkompilované procedury je další dotaz rychlejší.

Využití uložených procedur přináší další výhody:

- Snadnější správa dotazů – oddělení SQL kódu od C# kódu
- Využitelnost v jiných aplikacích v případě potřeby
- Správa uživatelských oprávnění v případě potřeby
- Zmenšení objemu přenášených dat – přenáší se pouze název procedury a parametry, ne celý dotaz, který může narůst do značné velikosti.

Zdroj: [2]

4.2.2 Indexy a primární klíče

Výkon databáze je značně ovlivňován správným použitím indexů. Při vytváření tabulky se implicitně nad primárním klíčem vytvoří clusterovaný index. Ve většině tabulek byl jen jeden kandidátní klíč, který byl využit jako primární. V případě tabulek `Users` a `Units` byl místo kandidátního klíče `Login` popř. `PublicID` zvolen za primární klíč sloupec `ID`, který je typu `int`, jehož hodnota je automaticky generována pomocí vlastnosti identity. K tomuto kroku vedlo především to, že na obě tabulky se odkazují 4 jiné tabulky, kde by, při použití typu nvarchar, indexy a sloupce cízích klíčů zabíraly o mnoho větší místo na disku.

Indexy na sloupcích, které nejsou primárním klíčem, vyznačené na obr. 3.1, byly vytvořeny na základě analýzy uložených procedur. Hlavními kritérii bylo využití sloupců ve WHERE části dotazu a očekávané využití dotazů.

4.2.3 Datové typy

Při návrhu tabulek byly použity nejmenší vhodné datové typy pro sloupce. V případě řetězců je využit typ `nvarchar`, který umožňuje uložit UNICODE znaky. Konfigurační data modulů v podobě řetězce obsahující xml dokument se ukládají do sloupců s nastaveným typem `xml`. Při použití tohoto typu jsou ukládaná data kontrolována, zda se jedná o validní xml dokument, nebo jeho část. Ověřování proti XML schématu je možné, ale nebylo využito protože XML schéma konfiguračních dat se může u každého modulu lišit.

Kapitola 5

Prezentační vrstva

Tvoří vrstvu systému starající se o komunikaci s vnějším okolím. Komunikační technologie lze rozdělit dle druhu protistrany na:

- Webové uživatelské rozhraní – stará se o komunikaci s uživatelem
- Webové služby – komunikují s jinými systémy jako jsou cizí webové stránky nebo jednotky
- Komunikační moduly – komunikují s jednotkami specifickým protokolem

Podrobněji jsou popsány v příslušných kapitolách.

5.1 Komunikační moduly

5.1.1 Účel

Se systémem mohou komunikovat jednotky od různých výrobců s různými komunikačními protokoly. Z toho důvodu byl navržen systém komunikačních modulů, kde každý modul obsluhuje určitý protokol.

5.1.2 Architektura

O správu modulů se stará Widows služba nazvaná *Ctms Communication Service*. Úkolem služby je spustit všechny nakonfigurované moduly a dle změny stavu je patřičně ovládat tj. pozastavit, ukončit, nebo opětovně spustit. Jinak řečeno modul kopíruje stav služby. Aby spuštěný modul neblokoval vlákno služby, měl by modul po inicializaci vytvořit vlastní pracovní vlákno.

Modul představuje serverovou aplikaci, která poslouchá na určitém portu a obsluhuje příchozí požadavky.

Pokud by jednotka chtěla použít jako komunikční protokol http popř. https je možné využít pro zpracování požadavku již existující webový server např. *Internet Information*

Services. Tímto řešením lze ušetřit prostředky pro vývoj serveru a zaměřit se přímo na vlastní komunikaci.

5.1.3 Zpracování požadavků

Při obsluze požadavků modul využívá knihovny Ctms Engine. První krokem by mělo být vytvoření kontextu s uživatelem typu služba. Další chování je závislé zcela na potřebách modulu.

Obecný model komunikace se skládá z:

1. Zpracování příchozího požadavku.
2. Uložení přijatých dat.
3. Vyhodnocení přijatých dat a případné spuštění akcí příslušných alarmů.
4. Získání rekonfiguračních dat signálů jednotky.
5. Odeslání rekonfiguračních dat a uzavření komunikace.

Při zanedbání režijní komunikace např. pro vytvoření zabezpečeného spojení, lze tento model uskutečnit posláním jednoho požadavku a následné odpovědi. Je proto vhodný pro webové služby.

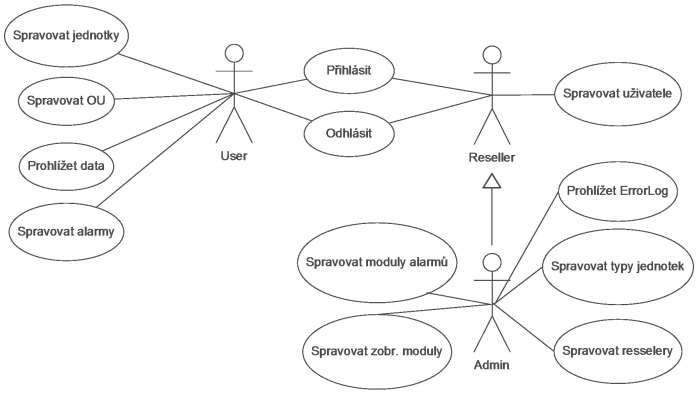
5.1.4 Komunikační modul pro jednotky řady CSAIO84

Jednotky typu CSAIO84 mají v sobě zabudovaný modul, který umožňuje komunikaci pomocí protokolu http. Proto byla pro komunikaci, místo vlastního modulu, vytvořena webová služba ASP.NET.

5.2 Webové uživatelské rozhraní

Webové uživatelské rozhraní slouží pro konfiguraci jednotek, signálů, alarmů, uživatelů a pro zobrazení dat prostřednictvím webových stránek.

Webové rozhraní se skládá z MasterPage [3] a ostatních stránek. MasterPage tvoří šablonu, která definuje uživatelské prvky společné pro všechny prvky, styl který bude použit pro rozvržení a místo (`PlaceHolder`), kam se vykreslí obsah ostatních stránek. Ostatní stránky z pravidla obsahují formulář s konkretním cílem např. přidání/editace/zobrazení detailů jednotky, uživatele, signálu, alarmu nebo zobrazení seznamu jednotek, přehled systému pro uživatele apod.



Obrázek 5.1: Use case diagram

5.2.1 Uživatelé

V uživatelském rozhraní se uživatelé dělí na: administrátory, resellery a uživatele. Jejich oprávnění znázorňuje use case diagram na obr. 5.1.

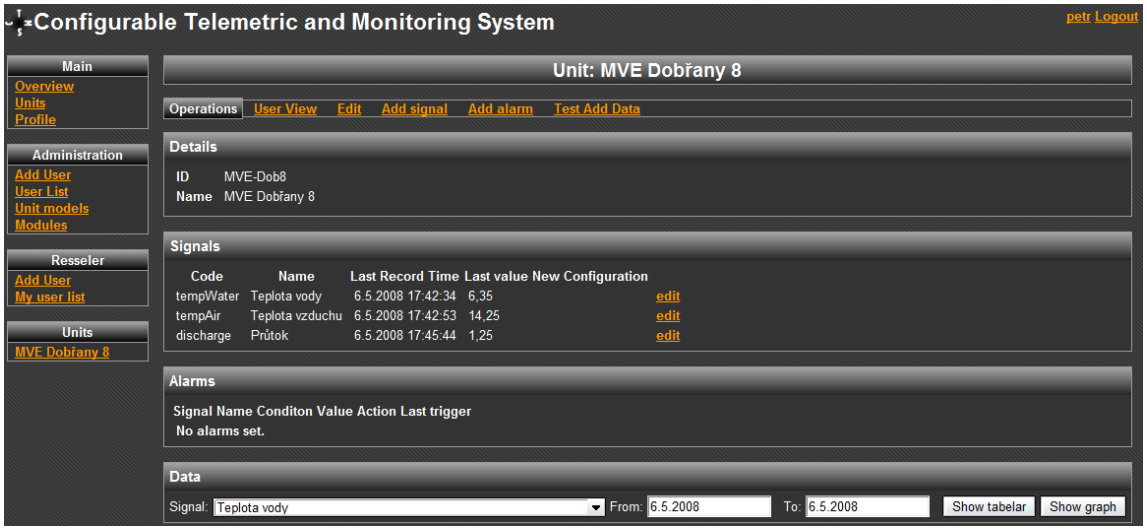
Z diagramu je vidět, že pro práci se systémem je nutné přihlášení. ASP.NET používá pro účely autentizace a autorizace uživatelů tzv. *Membership API*. To umožňuje využít na stránkách předpřipravené ovládací prvky tvořící rozhraní pro přihlášení, odhlášení, zobrazení přihlášeného uživatele, kontrolu přístupu dle rolí ap. Pro získání dat z databáze používá *Membership API Membership* a *Role providery* viz. [4]. Jedná se o abstraktní třídy poskytující rozhraní a základní funkcionalitu pro vlastní providery pracující se specifickým uložištěm. Pro získání informací o uživatelích systému byly vytvořeny vlastní providery – `CtmsMembershipProvider` a `CtmsRoleProvider`. Tvoří most mezi *Membership API* a aplikační vrstvou systému.

5.2.2 Kontext stránky

Aby ovládací prvky stránky mohly pracovat se systémem, je nutné vytvořit kontext. Ten se vytváří v *Load* fázi stránky. Uživatel kontextu je nastaven na aktuálně přihlášeného uživatele.

5.2.3 Rozvržení a vzhled

Rozvržení i vzhled je řízeno kaskádovými styly. Je využit dvousloupcový layout s horním panelem. V levém sloupci se nachází ovládací menu a seznam jednotek pro rychlý výběr. V horním panelu se nachází informace o přihlášeném uživateli a název systému s logem. Pravý panel tvoří kontainer pro obsah jednotlivých stránek. Ukázka rozvržení je na obr. 5.2



Obrázek 5.2: Ukázka webového uživatelského rozhraní – detaily jednotky

5.2.4 Kontrola uživatelských vstupů

Většina stránek obsahuje formulář, do kterého uživatel vyplňuje potřebné informace. Aby se zaručilo, že aplikační kód bude pracovat s daty v požadovaném formátu, využívá uživatelské rozhraní dvojí validace vstupů. První validace probíhá u klienta ve webovém prohlížeči pomocí JavaScriptu. Tímto se zaručí, že klient odešle správná data a zároveň se ušetří komunikace server-klient v případě, že by uživatel zadal špatný vstup a musel by zadání opakovat. Druhá validace probíhá na serveru. Její cíl je předejít podvržení nesprávných dat.

Dalším kontrolním mechanismem v systému je vlastní kontrola oprávnění na daný objekt vzhledem k uživateli kontextu (3.2.1) při získávaní objektu (3.2.4). Pokud by záškodník podvrhl dotaz na objekt, na který nemá právo, tak mu tento systém zabrání v možnosti provádět neoprávněné operace.

5.2.5 Uživatelem rozvržitelné zobrazení

Webové rozhraní obsahuje systém zobrazovacích modulů. Jedná se o možnost definování rozvržení zobrazovacích modulů pro jednotku.

Zobrazovací moduly mohou být různého charakteru. Pravděpodobně nejčastějším druhem modulů bude modul pro prezentování dat jako je např. tabulka hodnot nebo graf. Jiným druhem jsou konfigurační moduly určené pro specifickou konfiguraci jednotek nebo signálů dle typu jednotky.

Při rozhodování jaký typ rozvržení modulů zvolit bylo vybráno rozvržení do sloupců definované kaskádovými styly. Oproti rozvržení tabulkou umožňuje mít rozdílně vysoké řádky, respektive moduly jsou ve sloupcích vkládány pod sebe nezávisle na ostatních sloupcích – nejedná se tedy čistě o řádky jako takové. Další možností rozvržení je absolutní pozicování

na stránce. To však bylo zavrženo, protože při použití systému z ruzných počítačů s rozdílně velkým rozlišením resp. velikostí plochy pro vykreslení webové stránky, může docházet k umístění modulu v nezobrazené části. Univerzální řešení tohoto problému pro všechny uživatele neexistuje.

Systém modulů je modelován entitními množinami:

- `UnitViews`: pohled pro konkretního uživatele a jednotku, obsahuje počet zobrazených sloupců.
- `DisplayModules`: zobrazený modul. Obsahuje pozici (sloupec, pozice ve sloupci) typ modulu a konfigurační údaje.
- `Modules`: představuje samotný modul. Pro vytvoření instance obsahuje název třídy a assembly ve které je obsažena.
- `ModelsModules`: všechny moduly nemusí umět spolupracovat se všemi typy jednotek. ModelsModules udává pro které jednotky je daný modul použitelný.

Třídy zobrazovacích modulů by měly implementovat rozhraní `ICtmsWebModule`. Díky tomu získají přístup k jejich konfiguračním údajům a ke kontextu. Přes kontext mohou provádět změny v systému a získávat data.

5.2.6 Konfigurace alarmů

Alarmy využivají modulární systém akcí, kde každý modul může mít různou strukturu konfiguračních dat. Aby byly konfigurovatelné ve webovém rozhraní musí být pro každý modul dostupný ovládací prvek, který umožní daný modul nakonfigurovat a konfiguraci uložit. Pro tento účel vznikly konfigurační moduly alarmů, které jsou s vlastními moduly alarmů v relaci 1:1. Popisuje je entitní množina `AlarmWebConfigModules`. Konfigurační moduly alarmů by měly implementovat rozhraní `IAlarmWebConfigModule`. Hlavními vlastnostmi konfiguračních modulů je vlastnost `Control` vracející *ASP.NET Control* představující konfigurační rozhraní a vlastnost `Alarm`, která poskytne modulu konfigurovaný alarm.

5.3 Webové služby

Systém obsahuje řadu webových služeb. Ty slouží pro poskytování dat cizím systémům nebo pro komunikaci s jednotkami, které dokáží komunikovat prostřednictvím http/https protokolu.

Webové služby nabízené systémem jsou implementovány pomocí technologie *ASP.NET Web Services*. Webové metody služeb jsou psané tak, aby přenášely co nejvíc potřebných informací v co nejméně požadavcích. Vede to k zmenšení objemu přenášených dat o režijní data komunikačního protokolu. Tohoto chování se dosáhne zvolením vhodně strukturovaných

parametrů a návratových typů metod. Ideálním případem je provedení požadovaných operací a vrácení požadovaných hodnot během jednoho požadavku a následné odpovědi.

K přenosu dat se využívá protokol SOAP.

5.3.1 Zabezpečení

Pro omezení přístupu k datům by požadavek měl probíhat pomocí zabezpečeného protokolu https. Požadavek by měl obsahovat autentizační údaje(jméno, heslo) uživatele a to buď v SOAP hlavičce nebo v datech v závislosti na metodě.

Pokud klient nedokáže komunikovat pomocí protokolu https, situace se zhorší protože není vhodné přenášet nezašifrované autentizační údaje, pro hrozící zneužití např. pro přihlášení na webového rozhraní. Klientem neschopným této komunkace může být například vzdálené zařízení obsahující nedostatečně výkoný procesor. Částečným řešení tohoto problému je jednoduché symetrické kryptování a využití vygenerovaného jedinečného identifikátoru jako náhrady jména a hesla. Tento identifikátor by sloužil jen pro přihlášení ke kontrétní službě a v případě jeho prozrazení by nehrozilo jeho zneužití pro přihlášení do webového rozhraní.

Kapitola 6

Závěr

Cílem bakalářské práce bylo vytvořit základ modulárního systému sloužící pro telemetrické a zabezpečovací účely. Cíle bylo dosaženo implementací třívrstvého systému skládajícího se z databáze, knihovny tvořící aplikační logiku a umožňující spolupráci s databází a webového uživatelského rozhraní obsahující sadu webových služeb.

Aby byla umožněna možnost komunikace systému s různými typy jednotek, byla navržena komunikační služba s možností rozšíření moduly pro potřebný typ komunikačního protokolu.

Pro potřeby zabezpečení byl navrhnut a implementován modulární systém alarmů vykonávající akce v závislosti na přijatých datech. Aby mohla být ověřena jeho funkčnost, byl implementován testovací modul akce alarmu.

Implementované webové uživatelské rozhraní umožňuje pro administrátora administraci zobrazovacích modulů, modulů alarmů, uživatelů a modelů jednotek. Pro uživatele nabízí možnost konfigurace jednotek, signálů a alarmů. Pro ověření funkčnosti uživatelského rozvržení byl vytvořen testovací modul.

Zahrnuté webové služby poskytují základní metody pro přístup k datům a možnost nezabezpečené komunikace s jednotkou řady CSAIO84.

Při této úrovni implementace je systém schopný zpracovávat data od jednotek řady CSAIO84 a získané data zobrazit. Aby byl systém použitelný v reálném provozu, je vhodné ho doplnit o zobrazovací moduly nabízející přívětivější zobrazení dat a využitelné moduly akcí alarmů.

6.1 Možnosti rozšíření

Vzhledem k modulární povaze systému může být práce rozšířena celou řadou modulů poskytující lepší možnosti zobrazení dat, konfigurace jednotek nebo doplnění akcí alarmů - např. zasílání mailů nebo zpráv sms.

Jinou možností je tvorba nových podsystémů. Pro telemetrické účely by bylo vhodné vytovřit službu, která by by byla schopná získat data z jednotek tím, že by si je výžádala -

opačný přístup než v implementovaném řešení. Tento přístup by byl ale jen možný pro veřejně dostupné jednotky (tj. ty s veřejnou IP adresou). Bylo by to využitelný např. pro sběr dat z ethernetových teploměrů, který často nabízejí data prostřednictvím webových služeb.

Další možností rozšíření by mohlo být zprostředkování dálkového řízení přes web. Mohlo by toho být docíleno vytvořením ovládacího modulu stažitelného z webu technologií Flash nebo Silveright. Takový modul by musel řešit stejné problémy s připojením jako služba s aktivním získávaním dat. Komunikace ovládacího modulu by mohla probíhat zprostředkovaně přes web. server, nebo přímo s cílovým řízeným zařízením. V určitých případech by mohl zpracovávat video nebo zvuk.

Pro účely pronajímání služeb systému by systém měl být rošířen o účtovací systém, který by definoval, které služby a v jakém množství si uživatel zaplatil a může využívat.

Literatura

[1] Český statistický úřad: Využívání informačních a komunikačních technologií v domácnostech a mezi jednotlivci v roce 2007. [Online], [cit. 2008-05-10].
URL <http://www.czso.cz/csu/2007edicniplan.nsf/p/9701-07>

[2] Chapple, M.: SQL Server Stored Procedures — About.com:Databases. [Online], [cit. 2008-05-05].
URL <http://databases.about.com/od/sqlserver/a/storedprocedure.htm>

[3] Microsoft: ASP.NET Master Pages Overview — MSDN. [Online], [cit. 2008-05-05].
URL <http://msdn.microsoft.com/en-us/library/wtxbf3hh.aspx>

[4] Microsoft: Microsoft ASP.NET 2.0 Providers: Introduction — MSDN. [Online], [cit. 2008-05-05].
URL <http://msdn.microsoft.com/en-us/library/aa478948.aspx>

[5] Pecinovský, R.: *Návrhové vzory*. Computer Press, 2007, iSBN 978-80-251-1582-4.

[6] Wikipedia: Multitier architecture — Wikipedia, The Free Encyclopedia. [Online], [rev. 2008-23-04], [cit. 2008-05-05].
URL <http://en.wikipedia.org/w/index.php?title=Multitier_architecture&oldid=207669640>

Seznam příloh

1. DVD se zdrojovými kódy, textem práce a pokyny pro instalaci