VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

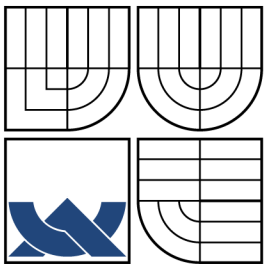
WEBOVÉ ROZHRANÍ PRO MONITORING SENZOROVÉHO POLE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

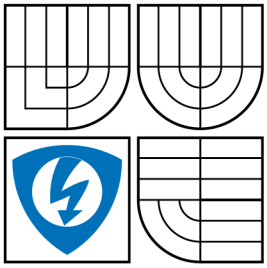
AUTOR PRÁCE
AUTHOR

BC. PAVEL VAJSAR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÉ ROZHRANÍ PRO MONITORING SENZOROVÉHO POLE

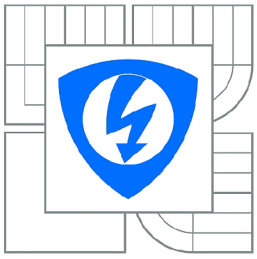
WEB INTERFACE FOR SENSOR NETWORK MONITORING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. PAVEL VAJSAR
AUTHOR

VEDOUCÍ PRÁCE — ING. PATRIK MORÁVEK
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Pavel Vajsar	***ID:***	74901
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Webové rozhraní pro monitoring senzorového pole

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a vytvořit webovou aplikaci, která umožní monitorování libovolné senzorové sítě. Práce bude obsahovat studium problematiky aplikačního nasazení bezdrátových senzorových sítí a analýzu parametrů, které je nutné sledovat pro věrné a efektivní monitorování senzorové sítě. Navržená aplikace bude zahrnovat návrh a vytvoření databáze, kde budou uloženy všechny údaje potřebné pro efektivní zobrazení sítě. Zde bude kladen důraz na univerzálnost databáze pro různé typy aplikací. Důležitým požadavkem aplikace je její široká přístupnost, dostupnost a přehledost snímaných parametrů sítě s uživatelsky přívětivým ovládáním.

DOPORUČENÁ LITERATURA:

[1] Li, Yingshu, Thai, My T., Wu, Weili, Wireless Sensor Networks and Applications, Signals and Communication Technology, 2008, p. 444, ISBN: 978-0-387-49591-0
[2] Mohammad Hammoudeh, Robert Newman, Sarah Mount, An Approach to Data Extraction and Visualisation for Wireless Sensor Networks. 2009 Eighth International Conference on Networks, pp.156-161, 2009

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Patrik Morávek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem této práce je navrhnout a vytvořit aplikaci, která umožní sledování (monitorování) bezdrátové senzorové sítě. Základem projektu je návrh databáze, která je schopna uchovávat data získaná z bezdrátové senzorové sítě. Hlavním požadavkem při jejím návrhu je univerzálnost, která plyne z požadavku na použití databáze pro maximální počet realizací bezdrátových senzorových sítí. Samotná aplikace je navržena tak, aby byla co nejvíce univerzální a modulární, kde jednotlivé moduly poskytují služby pro monitorování. Aplikace je dále skrze navržený a implementovaný konektor schopna monitorovat reálnou bezdrátovou senzorovou síť. Použitými technologiemi pro implementaci jsou J2EE (na straně serveru) a Adobe Flex/Air (na straně klienta).

KLÍČOVÁ SLOVA

Bezdrátová senzorová síť, Java, Adobe Flex/Air, BlazeDS, Hibernate, Spring

ABSTRACT

The aim of this work is to design and create an application that will allow monitoring of wireless sensor networks. The basis of the project is to design a database that is capable of storing data acquired from wireless sensor networks. The main requirement for the design is versatility, which flows from the requirement to use the database for broad range of applications implementing wireless sensor networks. The application itself is designed to be the most versatile and modular, which means that modules provide specific monitoring services. The application is able to monitor the real wireless sensor network through designed and implemented connector. For implementation J2EE (server side) and Adobe Flex/Air (client side) technologies are used.

KEYWORDS

Wireless sensor network, Java, Adobe Flex/Air, BlazeDS, Hibernate, Spring

VAJSAR P. *WEBOVÉ ROZHRANÍ PRO MONITORING SENZOROVÉHO POLE*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2010. 60 s., 20 s příloh. Vedoucí práce Ing. Patrik Morávek.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „WEBOVÉ ROZHRANÍ PRO MONITORING SENZOROVÉHO POLE“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Poděkování

Děkuji vedoucímu práce Ing. Patriku Morávkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne .....

.....
podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této práce je nejprve uvést základní teoretické poznatky o bezdrátových senzorových sítích WSN (bezdrátová senzorová síť – Wireless Sensor Network). Dále se zabývat strukturou těchto sítí, možnými oblastmi nasazení a výhodami, které přináší nasazení bezdrátové senzorové sítě v různých oblastech (průmysl, životní prostředí atd.).

Úvodní teoretický výklad dává přehled o tom, co mají všechny senzorové sítě společné a které parametry je potřeba sledovat a brát v úvahu pro úspěšné nasazení a následný provoz sítě. Tyto poznatky dále slouží jako základ pro návrh databázového úložiště, které bude sdružovat data potřebná jak pro dohled nad sítí, tak i pro její řízení a obsluhu. Jelikož použití bezdrátových senzorových sítí je velice různorodé, tak klíčovou vlastností takto navrženého databázového úložiště je jeho univerzálnost. Tyto požadavky s sebou dále přinášejí i negativní vlastnosti tohoto databázového úložiště a to jeho rozsáhlost a tím i složitost.

Navržená aplikace následně data z úložiště načítá a zobrazuje vhodnou podobou uživateli, který s aplikací pracuje. Aplikace byla navrhována tak, aby byla co nejvíce univerzální. Její hlavní částí je tedy "jádro", které nahrává jednotlivé moduly (části). Tyto moduly poskytují prostředky pro monitorování bezdrátové senzorové sítě a jsou mezi sebou vzájemně nezávislé. Výhodou je, že případná chyba v jednom z modulů neovlivní ostatní moduly. Moduly jsou do paměti nahrávány až v případě potřeby, což snižuje paměťovou náročnost aplikace. Celá aplikace disponuje pěti moduly - modul mapy, uživatele, grafického zobrazení, informací o senzorech a modul pro monitorování reálné senzorové sítě.

Jak je již patrné z posledního jmenovaného modulu, tak aplikace je schopna monitorovat reálnou senzorovou síť. Pro tuto možnost bylo nutné navrhnout a implementovat konektor, který je schopen přijímat data ze sítě, zpracovat je a uložit do databázového úložiště, ze kterého jsou následně načítána do aplikace.

Mohlo by být žádoucí monitorovat síť, přímo v místě jejího nasazení, proto byla do aplikace následně zavedena podpora technologie Adobe Air, kterou podporují různá mobilní kapesní zařízení. Aplikace následně komunikuje se serverovou částí aplikace prostřednictvím mobilní sítě. Výhodou je i to, že je aplikace nahrána v mobilním zařízení a není nutné ji při každém spuštění stahovat ze serveru, jako v případě přístupu k aplikaci prostřednictvím internetového prohlížeče.

1 BEZDRÁTOVÁ SENZOROVÁ SÍŤ

Senzorová síť je svou strukturou podobná klasické bezdrátové síti s tím, že jednotlivé koncové body netvoří stanice (počítače ovládané člověkem), nýbrž senzory rozmístěné v dané lokalitě. Tyto senzory mají předem definovanou činnost, kterou je v omezené míře možné modifikovat. Senzor ve svém okolí sleduje určené veličiny (fyzické veličiny - teplota, tlak, osvětlení, chemické látky v ovzduší), které průběžně odesílá ke zpracování.

1.1 Architektura WSN, výměna a zpracování dat

Bezdrátová senzorová síť je rozdělena do několika částí, viz. obr. 1.1. Na nejzákladnější úrovni to jsou samotné senzory, které jsou mezi sebou propojeny. Tyto senzory mohou být dále rozděleny do dílčích skupin, které lze nazývat jako senzorové pole.

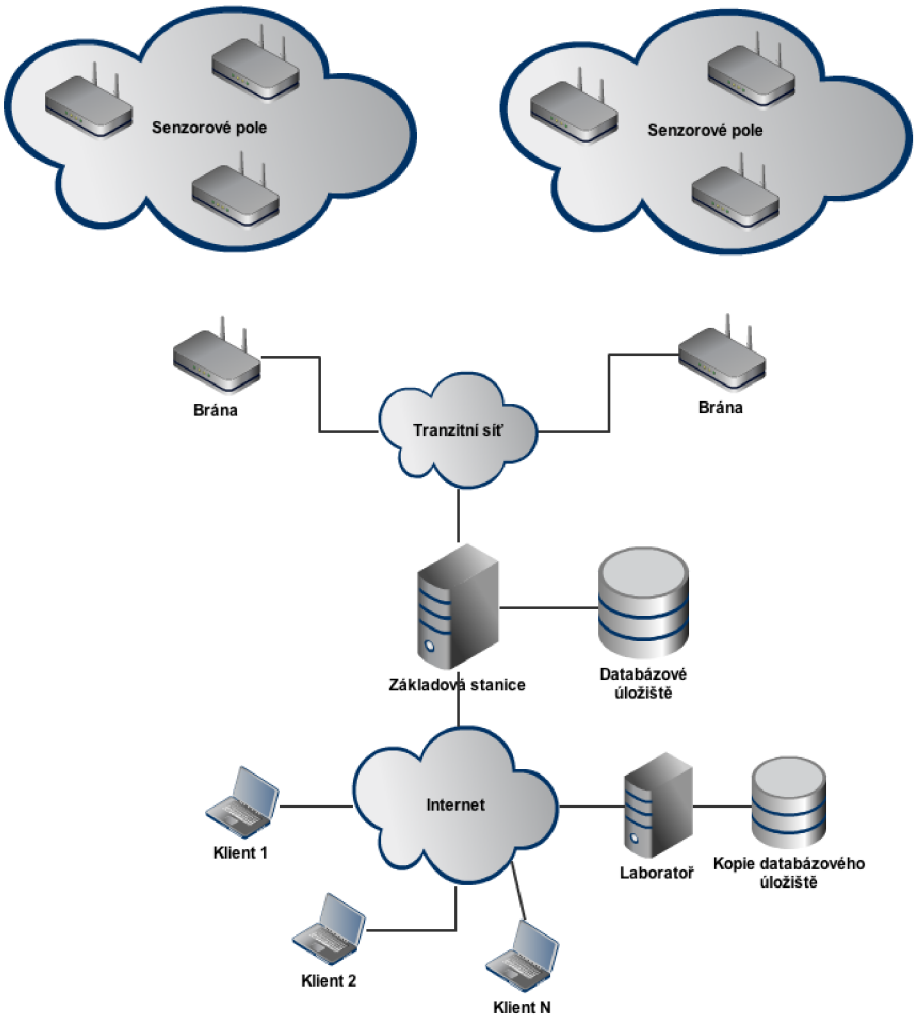
Jednotlivá senzorová pole jsou připojena k bráně. Připojení může být realizováno prostřednictvím jednoho nebo více uzlů. Bránu lze přirovnat k routeru v klasických počítačových sítích. Brána představuje rozhraní mezi senzorovým polem a lokální tranzitní sítí, která slouží k přenosu dat ze senzorů k základové stanici. Tranzitní síť může být také bezdrátová. V místě základové stanice poté dochází k uložení dat získaných ze senzorů do databázového úložiště. Základová stanice se může (ale nemusí) nacházet přímo v místě bezdrátové senzorové sítě. Následně je tato základová stanice připojena prostřednictvím WAN sítě k síti Internet.

Správa senzorové sítě tudíž nemusí být v místě nasazení senzorové sítě, ale může probíhat odkudkoliv. To znamená, že bezdrátové senzorové pole může být např. na Slovensku a správa řízení a dohledu se může nacházet v České republice - veškerá komunikace následně probíhá prostřednictvím sítě Internet a základové stanice v blízkosti bezdrátové senzorové sítě.

Další možností komunikace se senzorovou sítí je komunikace prostřednictvím PDA (přenosných zařízení) - tato komunikace může probíhat přímo v místě sledování a uživatel má zároveň k dispozici jak ty nejnovější údaje získávané bezdrátovou senzorovou sítí, tak i přehled o jejím stavu - monitoring dané sítě. Uživatel si sám v přenosném zařízení může vybrat, který z parametrů sítě chce monitorovat.

Dále je žádoucí provádět zálohu naměřených dat uložených v místě základové stanice. Možným způsobem je replikace (kopie) databázové úložiště do databázového úložiště v místě sledování.

Celá architektura by měla řešit i možnosti výpadku energie. Z toho důvodu je vhodné, aby každá z vrstev v hierarchii bezdrátové senzorové sítě (senzory, senzorové brány, základové stanice) měli vlastní perzistentní úložiště, do kterého by se data ukládala a k jejich zpracovávání by docházelo až po obnovení zdroje energie.



Obrázek 1.1: Architektura bezdrátové senzorové sítě

1.2 Senzor - uzel bezdrátové senzorové sítě

Senzor je složen z přijímače/vysílače pro možnost obousměrné komunikace. Dále obsahuje mikrokontrolér pro zpracování dat a přidruženou senzorovou desku, která obsahuje senzory. Tato deska může být vyměnitelná, takže oblast sledování senzoru záleží na konkrétní připojené senzorové desce.

Další důležitou částí senzoru je zdroj energie. Zde je žádoucí, aby senzor měl co nejvíce dostupné energie (kapacita zdroje) a aby také svou dostupnou energii využíval co nejhospodárněji. Na spotřebu energie má vliv jak použitý software (operační systém senzoru), tak i jeho hardware.

Z pohledu operačního systému to znamená, že je například možné regulovat vysílací výkon, nebo sdružovat měřená data a vyslat je poté najednou (agregace dat) a nebo uvést senzor např. do stavu offline, když není momentálně nutné, aby senzor měřil a posílal data.

V použitém hardwarovém vybavení je žádoucí použít součástky (části senzoru) s co nejnižší spotřebou energie.

1.3 Oblasti nasazení

Nasazení bezdrátové senzorové sítě je ve většině případů podmíněno potřebou dohlížet a sledovat určitou oblast. Výhodou nasazení takovéto sítě je, že není nutná stálá přítomnost člověka v oblasti sledování. To je v některých případech velice žádoucí a nebo i hlavním důvodem pro nasazení takovéto sítě. WSN (bezdrátová senzorová síť – Wireless Sensor Network) může být v příslušné realizaci nasazena téměř v každé oblasti, a to v oblasti životního prostředí, průmyslu a například i v armádě. Konkrétní implementace mohou být následující:

- **Sledování kvality vody v čističce odpadních vod** - v tomto případě je kontrolována kvalita vody, která prošla čističkou. V případě, že senzor zjistí zvýšenou přítomnost některých nežádoucích látek, tak jsou o tomto stavu ihned informování pracovníci příslušné čističky a obratem mohou podniknout další kroky k nápravě problému.

- **Sledování populace živočišného druhu** - tato implementace již byla realizována na ostrově Great Duck Island ležícím na východním pobřeží Spojených států pro monitorování populace buřňáčka bělavého. Hlavní důvody pro nasazení bezdrátové senzorové sítě byly dva. A to potřeba eliminovat přítomnost člověka na monitorovaném území z důvodu vlivu na chování sledovaného druhu. Druhým důvodem byla špatná dostupnost monitorovaného území. Ostrov není obydlen a je obtížně dostupný, proto stálá přítomnost člověka by byla velice obtížná až nemožná [1] .

2 DATABÁZOVÉ ÚLOŽIŠTĚ

Aby bylo databázové úložiště maximálně univerzální bylo nutné brát v úvahu maximální počet možných realizací bezdrátových senzorových sítí. Z jednotlivých realizací vyčlenit parametry pro sledování a tyto parametry poté vhodně spojit a pro dobré pochopení převést do grafické podoby - ERD (Diagram struktury databáze – Entity Relationship Diagram)

2.1 Sledované parametry

Sledované parametry lze pro lepší pochopení rozdělit do několika skupin.

1. Senzor (uzel) - je základní jednotkou celé senzorové sítě, proto je nutné mít o uzlu co nejvíce informací
    - stav senzoru (online, offline, poškozený, kolik může odeslat zpráv za určitou časovou jednotku)
    - rozšířenou autoritu (cluster head)
    - polohu
    - typ a výrobce
    - hardwarovou adresu
    - stav baterie
    - typ přidružené senzorové desky
    - přítomnost v klastru
    - přítomnost ve skupině
    - jaké může posílat senzor zprávy
    - jaké příkazy mohou být senzoru posílány
    - kolik informací může poslat v jednom paketu
    - jaké veličiny může měřit (závisí na přidružené senzorové desce)
    - zda je uzel mobilní (jako rychlostí po jaké oblasti se může pohybovat)
    - jaké senzory jsou jeho sousedy
2. Linka mezi senzory - informace o přenosové lince mezi jednotlivými senzory
    - jaké senzory jsou k dané lince připojeny
    - stav linky

- šířka pásma
- chybovost
- útlum linky

3. Prostředí - blíže specifikuje místo, kde je senzor umístěn
   - typ prostředí (nepřístupné)
   - popis tohoto prostředí (povrch, podnebí)
4. Poloha - souvisí jak se senzory, tak s vymezením určité oblasti, ve které se může nacházet více senzorů. Poloha nějakého prvku je definována třemi parametry, které vymezují, kde se prvek nachází:
   - oblast
   - budova
   - místnost

   Tyto parametry jsou pak dále specifikovány souřadnicemi, u kterých je kladen důraz na to:
   - aby souřadnice mohly být libovolného typu (zeměpisné, GPS, relativní)
   - aby byly použitelné pro uzly, oblasti, budovy
5. Skupina - sdružuje několik senzorů do jedné skupiny. Využito při posílání hromadných zpráv. Zprávu tedy obdrží celá skupina, tudíž všechny uzly náležící dané skupině
6. Kluster - podobné skupině s tím rozdílem, že kluster sdružuje sobě si blízké uzly v rámci senzorového pole

2.2 Popis databázového úložiště

Všechny uvedené parametry je tedy nutné efektivně ukládat. Jelikož je navržená struktura poměrně rozsáhlá, a tím i složitá je popis celého databázového úložiště rozdělena do několika částí. V této kapitole je popsán význam tabulek v databázovém úložišti. Popis jednotlivých atributů v tabulkách je uveden v příloze C, stejně jako ERD (Diagram struktury databáze – Entity Relationship Diagram) celé databázové struktury. Navržená struktura databázového úložiště bude implementována na MySQL serveru v podobě relační databáze.

- AREA - senzor nebo celé senzorové pole může ležet v určité geografické oblasti. Pro definici takové oblastí slouží tato tabulka.
- AREA_POINT - každá oblast je vyznačena body, podle kterých lze určit polohu a rozlohu oblasti. Možné typy bodů (souřadnic) jsou GPS souřadnice, zeměpisné souřadnice a kartezské souřadnice v rovinách X, Y, Z. V případě kartézských souřadnic musí být znám počátek, který je nutné definovat v atributu START_POINT_XYZ.
- TYPE_COORDINATION - spíše informativní charakter o tom, jaká soustava souřadnic byla použita pro vymezení oblasti a polohy senzoru (uzlu).
- COUNTRY - odkazované v tab. AREA pro upřesnění v jaké zemi (státu) leží daná oblast.
- BUILD - senzor nebo i celé senzorové pole může ležet pouze v prostoru budovy (budov), tabulka může nést i informaci o oblasti, ve které budova leží.
- ROOM - senzor nebo i celé senzorové pole může ležet pouze v prostoru místnosti (místností), tabulka může nést informaci o místnosti (např. její číslo) a dále referenci na budovu a skrze budovu zjistit i oblast, ve které se budova s místností nacházejí.
- BACKUP - informace o zálohování (replikace) celého databázového úložiště. Lze zpětně zjistit kdy a kým byla provedena poslední záloha databázové úložiště.
- CATEGORY - spíše všeobecný charakter. Určuje kategorie zpráv, dat, příkazů, které můžou být v senzorové síti dostupné. Např. kategorie systémových zpráv, nebo kategorie zpráv měření.
- SUBCATEGORY - upřesňuje charakter zpráv, dat, příkazů. Např. v hlavní kategorie může být dáno, že se jedná o kategorii zpráv měření a podkategorie tuto zprávu zpřesní na to, že se jedná o měření teploty.
- CLUSTER - senzorové pole lze rozdělit na dílčí bloky senzorů, které leží blízko sebe. Rozdíl od skupiny je v tom, že kluster sdružuje senzory blízké si svou geografickou polohou. Toto v případě skupiny platit nemusí.
- COMMAND - senzory mohou podporovat různé typy příkazů. Může se jednat o příkazy zjišťující (např. ping, kterým lze zjistit dostupnost senzoru), nebo řídící (např. změnit stav senzoru).

- DATA - měřené hodnoty posílané senzory se ukládají do této tabulky. Současně s měřenou hodnotou se ukládá i datum a čas měření a přes cizí klíč (ID_CATEGORY) kategorie a následně podkategorie získaných dat.
- ENVIRONMENT - senzory a senzorová pole mohou být umístěny v různých typech prostředí. V určitých případech je nutné mít o tomto prostředí nějaké informace (se zvýšenou teplotou, utajené, vysoká nadmořská výška, atd.).
- EXTENDED_AUTHORITY - každý senzor (uzel) může mít nastavenou rozšířenou pravomoc. Např. hierchická agregace - senzory na úrovni klusteru posílají měřená data právě tomuto uzlu a ten následně všechna tato data pošle do základové stanice.
- GROUP - senzory mohou být sdružovány do skupin, pro lepší správu a kontrolu nad senzory. Např. pro posílání hromadných zpráv pro rekonfiguraci, nebo požadavku na zjištění stavu. Zprávu tedy lze poslat skupině a tudíž jí obdrží všechny senzory náležící dané skupině.
- LINE - sledování stavu linky a jejích základních parametrů.
- MOBILITY - senzory v rámci senzorové sítě nemusí být pouze fixní (nehybné). V tomto případě je dobré znát informace o pohybu senzoru.
- MOBILITY_AREA - tabulka realizující vazbu N:M mezi typem mobility a oblastmi, po kterých může být daná mobilita realizována. To znamená, že lze udržovat informace o tom, po jakých oblastech se může senzor pohybovat v případě, že je mobilní.
- MOBILITY_BUILD - tabulka realizující vazbu N:M mezi typem mobility a budovami, po kterých může být daná mobilita realizována. To znamená, že lze udržovat informace o tom, v rámci jakých budov se může senzor pohybovat v případě, že je mobilní.
- MOBILITY_ROOM - tabulka realizující vazbu N:M mezi typem mobility a místnostmi, po kterých může být daná mobilita realizována. To znamená, že lze udržovat informace o tom, po jakých místnostech se může senzor pohybovat v případě, že je mobilní.
- NEIGHBOUR - každý senzor má ve svém okolí sousedy, které je dobré znát. Např. přestane-li nějaký senzor fungovat, tak lze skrze tuto tabulku zjistit jeho nejbližšího souseda a v omezené míře přesunout funkčnost porušeného senzoru na tento sousední senzor.

- NODE - senzor (uzel) je nejzákladnějším prvkem celé senzorové sítě. Informací, které je nutné monitorovat je celá řada a většina z nich je defonována přes cizí klíče. Co je ale specifické a nutné znát u každého uzlu je jeho název, hardwarová adresa, časové pásmo (senzor může ležet v jiném časovém pásmu než dohled nad senzorovou sítí) a množství dostupné energie (stav baterie).

- NODE_COMMAND - tabulka realizující vazbu N:M. Uchovává informace o jednotlivých uzlem a příkazech, které uzel podporuje a které mu lze zaslat.

- NODE_GROUP - tabulka realizující vazbu N:M, která spojuje senzory (uzly) s jejich skupinou. Je možné přidat jeden uzel do více skupin a v jedné skupině může být více uzlů.

- NODE_LINE - tabulka realizující vazbu N:M mezi senzory a linkami. Je tedy možné udržovat informace o tom, ke kterým linkám je senzor připojen a jaké senzory jsou připojeny k dané lince.

- NODE_MESSAGE -zprávy zasílané senzory. Informují o stavu senzoru a probíhajících operacích. Mohou mít pouze informativní charakter, nebo na ně lze reagovat příkazy.

- POSITION - pozice uzlu může být definována různými typy souřadnicového systému (GPS souřadnice, zeměpisné souřadnice, kartezské (relativní souřadnice)).

- PRODUCER - výrobci senzorů a senzorových desek, které jsou použity v senzorové síti.

- PRODUCT_TYPE - výrobce může vyrábět více senzorů, nebo více senzorových desek.

- PRODUCT_TYPE_PROPERTY - tabulka realizující vazbu N:M mezi produktem a jeho vlastnostmi. Lze tedy k danému produktu zjistit jeho dostupné vlastnosti a na základě vlastnosti si zjistit, které produkty touto vlastností disponují.

- PROPERTY - každý produkt v senzorové síti je specifikován svými parametry. Tyto parametry jsou definovány svým názvem a třemi hodnotami, které mohou nabývat libovolných hodnot. V případě senzoru měřícího určitou fyzikální veličinu to může být minimální hodnota, maximální hodnota a rozsah. Počet hodnot dané vlastnosti bude možné v budoucnu přidávat dle potřeby senzorové sítě.

- SENSORE_BOARD - senzor může využívat pro snímání hodnot senzorovou desku, na které jsou umístěny příslušné senzory. Tato deska může být odnímatelná a výměnná. Výměnou této desky lze měnit účel (snímané hodnoty) senzoru.

- SENSORE_BOARD_PROPERTY - tabulka realizující vazbu N:M. Senzorová deska může obsahovat více senzorů a tím pádem i více vlastností. Naopak definovaná vlastnost může být přidružena více senzorovým deskám.

- STATE - senzor (uzel) se může nacházet v různých stavech definující jeho činnost a periodicitu odesílání měřených dat. Např. funkce senzoru je v pořádku, má dostatek energie a data posílá každou minutu. Naopak může být nastaven do režimu úspory energie, kdy posílá data jednou za týden, nebo v případě závady na senzoru může být nastaven stav poruchy.

- USER - aplikace pro monitoring bezdrátové sítě bude dostupná pouze po ověření identity uživatele, proto je nutné v databázi uchovávat informace o uživatelých, kteří mohou s aplikací pracovat.

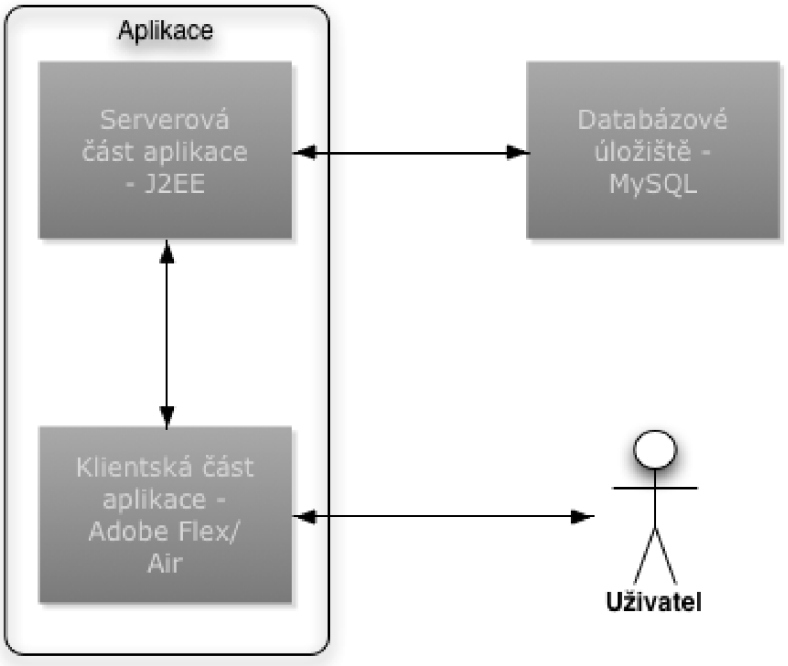
- AUTHORITIES - popis jednotlivých dostupných rolí v aplikaci pro monitoring senzorové sítě. Role určují práva aktuálně přihlášeného uživatele.

- USER_LOG - z důvodu bezpečnosti je vhodné uchovávat informace o času a datu, kdy se uživatel do systému přihlásil a kdy odhlásil. Dále je zde ještě zaznamenávána IP adresa, ze které byl realizován přístup do systému.

- CROSSBOW_PACKET - ukládání dat jednotlivých paketů přijatých z reálné senzorové sítě

3 NÁVRH APLIKACE PRO MONITORING BEZDRÁTOVÉ SENZOROVÉ SÍTĚ

Samotné databázové úložiště není pro monitoring bezdrátové senzorové sítě postačují a nutné dále navrhnout a implementovat aplikaci, které bude schopna s navrženým databázovým úložištěm pracovat. Aplikace bude rozdělena na 2 částí, viz 3.1.



Obrázek 3.1: Komunikační struktura

3.1 Požadavky

Před začátkem vývoje aplikace je nutné stanovit si základní požadavky, které musí aplikace nutně splňovat. Jedná se spíše o obecné požadavky, od kterých se bude odvíjet výběr nástrojů pro implementaci a způsob nasazení celé aplikace.

1. Minimální softwarové požadavky na vybavení klienta
2. Minimální hardwarové požadavky na vybavení klienta
3. Snadná správa a údržba
4. Stabilita

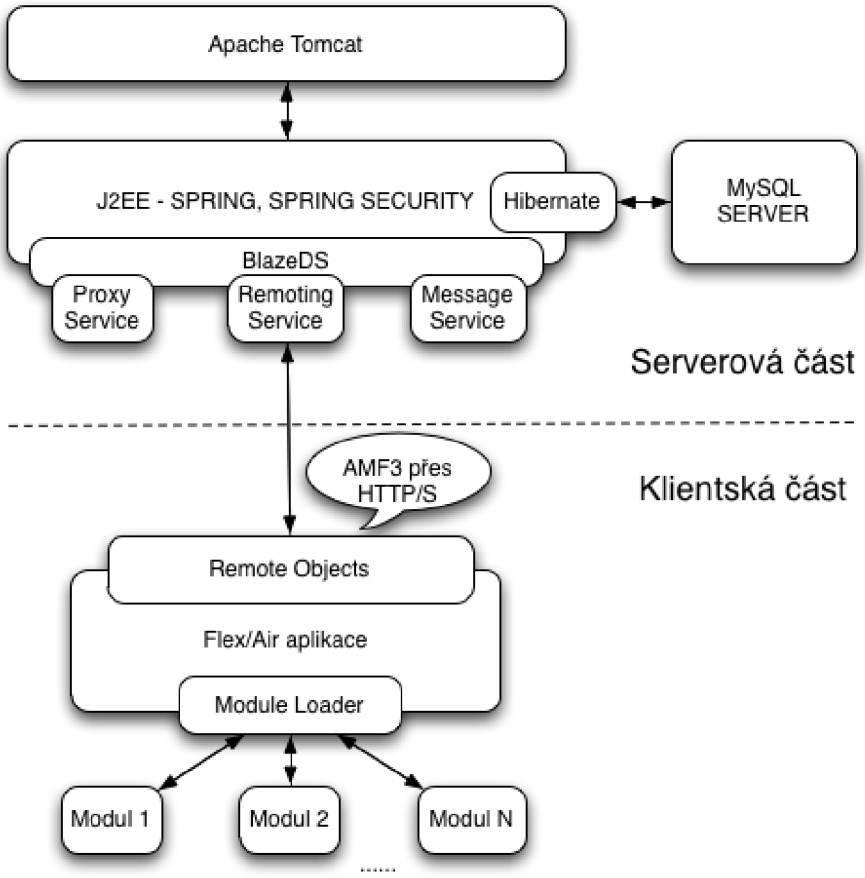
5. Dostupnost
6. Kvalitní uživatelské rozhraní
7. Jednoduchost
8. Cena

3.2 Implementace

Z výše uvedených požadavků vyplývá, že je vhodné danou aplikaci navrhnou a realizovat jako aplikaci webovou. Tím by velkou měrou byla splněna podmínka 1, 2 a 5. Klientovi pro práci s aplikací postačí pouze webový prohlížeč. Nebude potřeba žádná instalace ani žádný dodatečný software. Aplikace bude nasazena na serveru, který bude neustále v provozu (až na výjimky) a tudíž dostupný prostřednictvím sítě Internet.

Pro implementaci bude na straně serveru nasazena platforma Java a rámce Hibernate [7], Spring [10], Spring Security [11]. Je to z důvodu rozsáhlých možností této platformy, které usnadní celý vývoj aplikace. Jako uživatelské rozhraní byla zvolena technologie Adobe Flex [26], která aplikaci poskytne jednoduché uživatelské rozhraní s kvalitním designem a podporou mnoha nástrojů pro vizualizaci dat získaných z bezdrátové senzorové sítě (grafy, tabulky, mapy). Celá takto vytvořená aplikace bude nasazena na webový server Apache Tomcat [13] s podporou platformy Java.

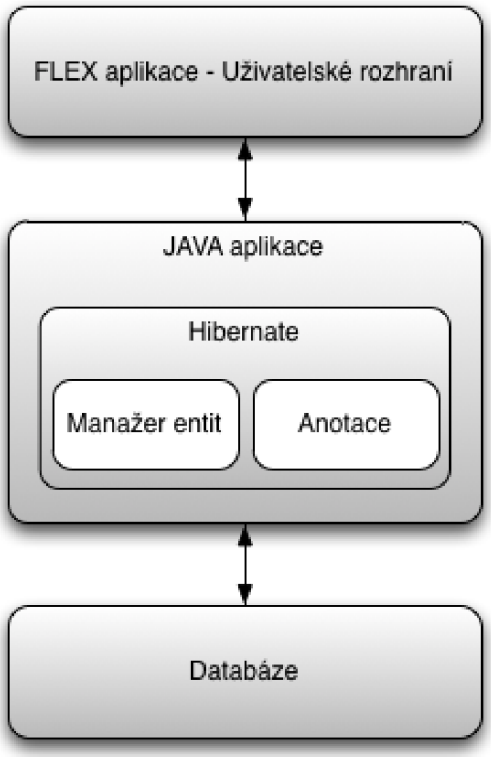
Struktura aplikace z pohledu použitých nástrojů je patrná na obr. 3.2. Jelikož jsou části klient-server z pohledu aplikace maximálně odděleny, tak bylo zapotřebí vytvořit univerzální komunikační rozhraní mezi klientem a serverem. K tomuto účelu byl použit nástroj BlazeDS [9], který toto rozhraní poskytuje. Tento nástroj, zjednodušeně řečeno, umožňuje volání služeb na serveru z klientské aplikace a výměnu dat mezi nimi. Samotná klientská část aplikace byla vytvořena tak, aby byla co nejvíce modulární a přizpůsobitelná. Hlavní částí je tedy "jádro", které provádí operace načítání funkčních modulů. Až tyto moduly poskytují služby pro monitoring bezdrátové senzorové sítě. Moduly mohou být následně upravovány pro potřeby konkrétní sítě, nebo mohou být vytvářeny moduly nové. A to moduly, které buď přímo souvisí s monitoringem sítě nebo pro doplňkové funkce.



Obrázek 3.2: Struktura aplikace

3.2.1 Rámec Hibernate

Rámec Hibernate slouží pro objektově-relační mapování.To znamená, že v prostředí platformy Java pracujeme s entitami z relační databáze jako s objekty přímo v Javě a přímo v aplikaci. Mezi těmito objekty je možné vytvořit stejné vazby jako jsou mezi jednotlivými entitami v databázi. Načítání referencí přes cizí klíče poté provádí rámec Hibernate. Z důvodu, že implementované databázové úložiště je značně rozsáhlé s mnoha vzájemnými vazbami, tak právě tyto vlastnosti rámce Hibernate přináší do aplikace určité zjednodušení a hlavně zachování integrity dat. V případě, že by se SQL dotazy skrze více entit a více cizích klíčů musely tvořit ručně, tak by jistě aplikace byla náchylnější na chyby způsobené nesprávností vytvořených dotazů.



Obrázek 3.3: Pozice rámce Hibernate v aplikaci

3.2.2 Adobe Flex jako uživatelské rozhraní

Adobe Flex [26] je technologie, která úzce souvisí s technologií Adobe Flash. V případě Flexu se zjednodušeně jedná o Flash bez časové osy. Jedná se čistě o klientskou technologii. To znamená, že její implementace je možná pouze na straně klienta. V případě naší aplikace tedy jako uživatelské rozhraní. Jedná se v podstatě opět o rámec, který pro svou činnost využívá kombinaci MXML [4] (značkovací jazyk založený na XML) a ActionScriptu [3].

3.3 Adresářová struktura projektu

Jelikož je celý projekt koncipován jako klient/server, tak se toto rozdělení projevilo i při vývoji aplikace. Celý projekt je tedy rozdělen na dva dílčí projekty, kde jeden z nich serverovou část a druhý část klientskou.

- *wsn-server* - obsahuje objektově-relační model úložiště, poskytuje služby klientu a řeší zabezpečení aplikace

- *wsn-flex* - uživatelské rozhraní aplikace, získává data ze serveru a ve vhodné podobě tyto data interpretuje.

Při vývoji aplikace bylo použito několik rámců a nástrojů (Spring, Spring Security, BlazeDS, Hibernate, SpringFlex Integration). Tyto nástroje jsou do projektu integrovány prostřednictvím dodatečných .jar knihoven a konfiguračních souborů. Seznam všech potřebných knihoven je uveden v příloze. Bez těchto knihoven není běh aplikace možný. Seznam potřebných knihoven je uveden v příloze B.

Každý z těchto projektů má svou specifickou a přesně definovanou strukturu. Jedna se zejména o umístění konfiguračních souboru a souborů obsahující zdrojové kódy. Níže je uveden popis nejdůležitějších složek, balíčků a konfiguračních souborů z obou projektů.

1. *wsn-server*

    - *src* - obsahuje veškeré zdrojové kódy aplikace, které jsou následně uspořádány do balíčků.
    - *src/cz/feec/xvajsa01/model* - objekty mapované na databázové tabulky. Každé tabulce v databázi (až na výjimky) odpovídá jeden objekt, který je na tuto tabulku mapován. Každý objekt přestavuje samostatnou Java třídu.
    - *src/cz/feec/xvajsa01/services* - webové služby, každá služba = samostatná Java třída.
    - *src/cz/feec/xvajsa01/session* - nastavení sezení mezi databázovým serverem a Hibernate prostřednictvím rámce Spring.
    - *src/cz/feec/xvajsa01/test* - třídy určené pro testování webových služeb.
    - *WebContent* - zkompilované soubory .swf a .html soubory aplikace, které tvoří uživatelské rozhraní.
    - *WebContent/WEB-INF* - konfigurační soubory.
    - *WebContent/WEB-INF/lib* - knihovny rámců a další knihovny potřebné pro běh aplikace.
    - *WebContent/WEB-INF/flex* - konfigurační soubory nástroje BlazeDS. Nastavují komunikaci mezi J2EE serverem a klientskou částí aplikace.
    - *WebContent/WEB-INF/web.xml* - globální konfigurační soubor projektu - nastavení uvítací stránky a chybových stránek (Error 404, 403) a filtrů.
    - *WebContent/WEB-INF/applicationContext.xml* - nastavení připojení k databázovému serveru, použitého dialektu databáze a cesty k objektům pro objektově relační mapování.

- *WebContent/WEB-INF/applicationContext-security.xml* - konfigurace zabezpečení aplikace.

2. *wsn-flex*

   - *bin-debug* - binární (zkompilované soubory) projektu. Tato složka není plně využita, protože projekt je nastaven tak, že výchozí složka pro tyto soubory je v projektu *wsn-server/WebContent*. Při testování aplikace není tedy nutné zkompilované soubory klientské části kopírovat do hlavního projektu - vývojové prostředí toto dělá automaticky.
   - *icons* - ikony použité v aplikaci.
   - *libs* - knihovny použité v klientské části aplikace (Google Maps SDK [6], AlivePDF [24]).
   - *src* - zdrojové kódy.
   - *src/Modules* - jednotlivé moduly aplikace, které jsou dynamicky načítány.
   - *src/Style/app.css* - definice stylu aplikace.
   - *src/Style/app.mxml* - "jádro" klientské aplikace. Definice základního rozvržení a dynamické načítání modulů.
   - *src/Style/appLogin.mxml* - ověření přístupových informací (jméno, heslo) a přesměrování do hlavní aplikace.
   - *src/Style/appLogout.mxml* - odhlášení z aplikace.

3.4 Důležitá nastavení

Převážná většina nejrůznějších nastavení byla provedena v dílčím projektu *wsn-server*. Jedná se zejména o konfiguraci použitých rámců (mapování filtrů) a nástrojů (BlazeDS). Tyto nastavení jsou nutná zejména ke správné integraci a tím i funkci těchto rámců a nástrojů. Níže jsou uvedena nastavení, která se již týkají aplikace.

```
<context:component-scan base-package="cz.feec.xvajsa01.services" />
```

Říká rámci Spring, kde má hledat objekty (Java třídy), které jsou vytvořeny tak, aby poskytovaly webové služby. Takto načtené služby je následně možné volat vzdáleně. Z výše uvedeného příkladu nastavení plyne, že rámec Spring bude prohledávat pouze balík *cz.feec.xvajsa01.services*.

```
<bean id="myDataSource" class="org.apache.commons.dbcp.BasicDataSource"
     destroy-method="close">
```

```
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://Localhost:3306/wsn"/>
    <property name="username" value="xvajsa01"/>
    <property name="password" value="wsn01"/>
</bean>
```

Nastavení cesty a přístupu k databázovému úložišti. Z použitého ovladače k databázovému serveru je patrné, že je použit databázový server MySQL [12]. Přístup k tomuto serveru je umožněn prostřednictvím lokální smyčky na portu 3306 a název databáze je *wsn*.

```
<bean id="mySessionFactory" class="org.springframework.orm.hibernate3.
annotation.AnnotationSessionFactoryBean">
<property name="dataSource" ref="myDataSource" />
<property name="annotatedClasses">
      <list>
      <value>cz.feec.xvajsa01.model.Users</value>
      <value>cz.feec.xvajsa01.model.Authorities</value>
      ...
      </list>
</property>
      <property name="hibernateProperties">
      <props>
      <prop key="hibernate.dialect">
            cz.feec.xvajsa01.model.CustomMysqlDialect
      </prop>
      <prop key="hibernate.show_sql">true</prop>
      <prop key="hibernate.hbm2ddl.auto">update</prop>
      </props>
      </property>
</bean>
```

Toto nastavení vytvoří "sezení" mezi rámcem Hibernate a databázovým serverem. V aplikaci může existovat pouze jedno takového sezení. Dále nastavení určuje strukturu celého úložiště prostřednictvím párových značek *list* a *value*. Hodnoty uvnitř značky *value* jsou cesty k jednotlivým objektům, které jsou mapovány na databázové tabulky = každá položka *value* představuje v databázi jednu vlastní tabulku.

Vlastností *hibernate.dialect* říkáme rámci Hibernate jaký typ databázového serveru využíváme. V našem případě je to MySQL server a použitý dialekt je "vlastní" dialekt. Ve skutečnosti se jedná o mírně upravený *MySQL5InnoDBDialect* dialekt.

Jestliže bychom se rozhodli změnit typ databázového úložiště například na PostgesSQL, tak je nutné změnit dialekt i ovladač pro přístup k databázovému serveru, viz výše.

Důležitým prvkem tohoto nastavení je vlastnost *hibernate.hbm2ddl.auto*, která má hodnotu *update*. To znamená, že jestliže se při kompilaci aplikace v databázi nenachází tabulka, na kterou má být mapován objekt ze seznamu, tak je tato tabulka automaticky vytvořena. Struktura této tabulky odpovídá definici objektu. Naopak, jestliže tabulka existuje, ale má např. odlišnou strukturu, než ta co je uvedena v definici objektu, tak je struktura tabulky dodatečně upravena. Je-li tedy v objektu definován sloupec, který v tabulce není, tak je tabulka o tento sloupec rozšířena. Jestliže v tabulce sloupec existuje, ale má jiné vlastnosti, tak naopak ponechán beze změny. To je z důvodu, že tabulka může obsahovat již nějaké záznamy a změnou vlastností sloupce by byla data znehodnocena a tím by mohlo dojít k narušení integrity dat. V případě rozšíření tabulky o nový sloupec zůstanou všechna data zachována. Možné hodnoty vlastnosti *hibernate.hbm2ddl.auto* jsou:

- *create* - odstraní původní data a vytvoří novou prázdnou strukturu definovanou jednotlivými objekty.
- *create-drop* - stejné jako create, ale na konci sezení celou databázi odstraní. Lze použít v případě potřeby pouze dočasné databázové struktury.
- *update* - aktualizuje databázovou strukturu.
- *validate* - pouze ověří, neprovádí žádné změny.

Tato vlastnost je vhodná zejména při nasazení aplikace na nový databázový server. Není nutné prostřednictvím SQL skriptů nejprve importovat strukturu databázového úložiště. Rámec Hibernate a vlastnost *hibernate.hbm2ddl.auto* tuto strukturu vytvoří za nás.

```
<beans:bean id="authDataSource"
      class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
     <beans:property name="driverClassName" value="com.mysql.jdbc.Driver"/>
     <beans:property name="url" value="jdbc:mysql://Localhost:3306/test"/>
     <beans:property name="username" value="xvajsa01"/>
     <beans:property name="password" value="wsn01"/>
</beans:bean>
```

Pro zabezpečení aplikace byl použit rámec Spring Security [11] a toto nastavení říká rámci, kde má ověřit zadané přístupové informace pro přístup do aplikace. Jedná se o podobnou definici, jako pro databázové úložiště.

```
<authentication-provider>
     <password-encoder hash="sha-256"/>
     <jdbc-user-service data-source-ref="authDataSource"/>
</authentication-provider>
```

Užitečné nastavení, které eliminuje nutnost programovat funkci pro získání otisku (hashe) z hesla a následné ověřování otisku s otiskem v databázovém úložišti. Toto nastavení vytvoří automaticky otisk z hesla zadaného při pokusu o přístup do aplikace s otiskem v databázi. Pro vytvoření otisku je použit algoritmus sha-256. Číslo 256 znamená, že otisk (hash) bude mít délku 256 bitů. Tato délka se stejná, pro jakkoliv dlouhý vstupní řetězec [22].

```
<form-login login-page="/appLogin.html"/>
<intercept-url pattern="/appLogin.html"
      access="IS_AUTHENTICATED_ANONYMOUSLY" />
<intercept-url pattern="/appLogout.html"
      access="IS_AUTHENTICATED_ANONYMOUSLY" />
<intercept-url pattern="/app.html"
      access="ROLE_USER" />
```

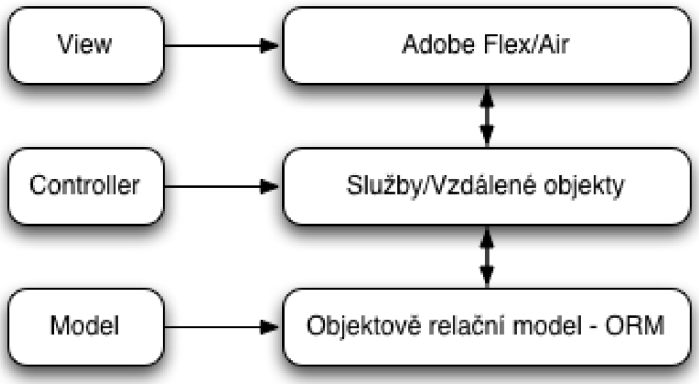
Tato nastavení řídí přístup k jednotlivým částem aplikace. Každé z těchto nastavení definuje stránku a práva, která jsou potřebná pro přístup k této stránce. Ke stránkám *appLogin.html* a *appLogout.html* je povolen přístup bez jakékoliv role. Pro přístup ke stránce *app.html* je nutná role *ROLE_ USER*. Jedná se totiž o samotnou aplikaci pro monitorování senzorové sítě, do které není povolen anonymní přístup. Ostatní dvě stránky slouží pro ověření uživatele a bezpečné odhlášení z aplikace.

Jestliže je v databázovém úložišti definovaná struktura tabulek *AUTHORITIES* a *USERS*, tak jak vyžaduje rámec Spring Security, tak za nás provede ověření role uživatele automaticky. To znamená, že při pokusu o přístup do aplikace zadá uživatel své uživatelské jméno a heslo, tyto údaje jsou dále předány rámci Spring Security prostřednictvím *authentication-provider* a následně dojde k ověření, zda se daný uživatel nachází v databázi a zda souhlasí zadané heslo (otisk) s heslem (otiskem) v databázi. Jestliže ano, tak následně dojde k načtení uživatelských rolí k zadanému uživatelskému jménu a jestliže uživatel disponuje rolí, která je pro přístup na požadovanou stránku nutná, tak mu je následně umožněn přístup.

Jestliže by se uživatel pokusil přímo přistoupit na stránku *app.html* bez toho aniž by předem zadal své přístupové informace, tak automaticky dojde k přesměrování na *appLogin.html*.

4 OBJEKTOVĚ RELAČNÍ MODEL - ORM

Jestliže se na aplikaci podíváme z pohledu tří vrstvového modelu (M - Model, V - View, C - Controller), tak objektově relační model tvoří nejnižší vrstvu, viz 4.1 . Model je tvořen jednotlivými objekty, které jsou tzv. perzistentní a uchovávají v sobě informace (data). Jedná se zejména o data, získaná z databázového úložiště. Až na výjimky představuje objekt v modelu tabulku v databázi. Při práci s těmito objekty v prostředí platformy Java se tedy do objektů načítají data z databázové úložiště a následně se s tabulkou a příslušnými daty pracuje jako s klasickými Java objekty obsahující data [7], [17].

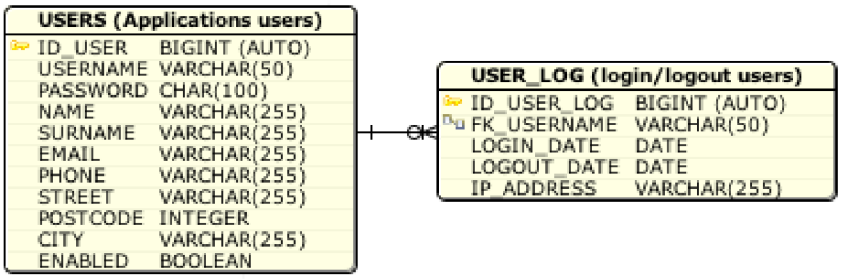


Obrázek 4.1: Vrstvový model

O naplnění objektů daty se stará druhá vrstva modelu, která obsahuje tzv. *Služby*. Jsou to objekty, které obsahují metody pro načítání/ukládání/aktualizování dat mezi databází a objektově relačním modelem. Jestliže tedy načtená data v objektu v aplikaci změníme a následně objekt předáme metodě ke zpracování, tak tato metoda promítne změnu do databáze. V prostředí aplikační logiky je tedy práce s databází transparentní.

Jelikož v databázovém úložišti jsou definovány mezi jednotlivými tabulkami vazby (relace) a tabulky jsou vzájemně závislé, tak je vhodně tyto relace definovat i mezi objekty v objektově relačním modelu. Jedná se o vazby:

- 1:1 - jednomu záznamu v první tabulce odpovídá jeden záznam ve druhé tabulce
- 1:N - jednomu záznamu v první tabulce odpovídá více záznamů ve druhé tabulce
- M:N - více záznamům v první tabulce odpovídá více záznamům ve druhé tabulce.



Obrázek 4.2: Relace 1:N

Na výše uvedeném obrázku 4.2 je patrná vazba 1:N. V tabulce *USERS* jsou uloženy informace o jednotlivých uživatelích aplikace. V tabulce *USER_LOG* pak informace o přístupech jednotlivých uživatelů do aplikace (datum a čas přihlášení/odhlášení a IP adresa). Ke každému uživateli může v tabulce *USER_LOG* připadat více odpovídajících položek. Vazba je realizována prostřednictvím cizího klíče *FK_USERNAME*, který říká, kterému uživateli daný záznam patří.

Obě tyto tabulky představují v objektově relačním modelu aplikace dva různé objekty (třídy). Objekt je definován jako klasická třída doplněna o anotace. Jednotlivé vlastnosti třídy představují jednotlivé sloupce v tabulce

- *@Entity* - říká o třídě, že bude mapována na databázovou tabulku.
- *@Table(name = "USERS")* - definuje, na jakou tabulku bude třída mapována a v případě, že tabulka v databázi neexistuje, tak vytvoří tabulku s názvem uvedeným v parametru.
- *@GeneratedValue(strategy=GenerationType.AUTO)* - hodnota sloupce bude generována automaticky, když nebude zadána ručně.
- *@Id* - definuje sloupec, jako primární klíč tabulky.
- *@OneToMany* - definice vazby 1:N. Znamená, že k jednomu záznamu z tabulky *USERS* lze načíst více záznamu z tabulky *USER_LOG*. Proto definice vlastnosti nad kterou je tato anotace uvedena je typu množina (*Set*).
- *@ManyToOne* - definice vazby N:1. Znamená, že k jednomu záznamu z tabulky *USER_LOG* lze načíst odpovídající záznam z tabulky *USER*. Definice vlastnosti nad kterou je tato anotace uvedena je objekt typu (*Users*).
- *@JoinColumn* - definuje cizí klíč, prostřednictvím, kterého bude vazba realizována.

V reálném použití to tedy funguje tak, že když dojde v aplikaci k načtení uživatele z databáze, tak vlastnost *userlog*, která je typu množina, bude obsahovat všechny odpovídající záznamy z *USER_LOG*. A jelikož je definice vazby i v tabulce *USER_LOG*, tak to pravidlo platí o opačně. Jestliže dojde v aplikaci k načtení záznamu z *USER_LOG*, tak vlastnost *users* bude obsahovat uživatele, kterému daný záznam patří. Jedná se o tzv. obousměrnou vazbu 1:N. Není tedy potřeba tvořit žádné SQL dotazy přes více tabulek a rámec Hibernate to vše udělá za nás. Nezbytné pro použití uvedených anotací je mít součástí projektu všechny potřebné knihovny uvedené v příloze B

```
@Entity
@Table(name = "USERS")
public class Users implements Serializable {
      private static final long serialVersionUID = -4814719013497551615L;

      @GeneratedValue(strategy=GenerationType.AUTO)
      @Column(name = "ID_USER")
      private Integer id;

      @Id
      @Column(name="USERNAME")
      private String username;

      @Column(name="PASSWORD")
      private String password;

      @Column(name="ENABLED")
      private Boolean enabled;

      @Column(name="NAME")
      private String name;

      @Column(name="SURNAME")
      private String surname;

      @Column(name="EMAIL")
      private String email;

      @Column(name="PHONE")
```

```
	private String phone;

	@Column(name="STREET")
	private String street;

	@Column(name="POSTCODE")
	private Integer postcode;

	@Column(name="CITY")
	private String city;

	@OneToMany (cascade = {CascadeType.ALL}, fetch = FetchType.EAGER)
	@JoinColumn (name = "FK_USERNAME")
	private Set<UserLog> userLog;

	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	.... get/set metody jednotlivých vlastností třídy

@Entity
@Table(name = "USER_LOG")
public class UserLog implements Serializable {
	private static final long serialVersionUID = -4814719013497551615L;

	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	@Column(name="ID_USER_LOG")
	private Integer id;

	@Column(name="LOGIN_DATE")
	private Date loginDate;

	@Column(name="LOGOUT_DATE")
	private Date logoutDate;
```

```
@Column(name="IP_ADDRESS")
private String ipAddress;

@ManyToOne
@JoinColumn (name="FK_USERNAME",
      nullable = true, updatable = false, insertable = false)
private Users user;

public Users getUser() {
      return user;
}
public void setUser(Users user) {
      this.user = user;
}
.... get/set metody jednotlivých vlastností třídy
```

5 WEBOVÉ SLUŽBY A VZDÁLENÉ OBJEKTY

Webové služby a vzdálené objekty představují z pohledu tří vrstvového modelu aplikace prostřední vrstvu, viz 4.1. Jejich úkolem je na vyžádání načítat data z databáze do objektů objektově relačního modelu a dále tyto objekty předávat přes AMF kanál uživatelské části aplikace, nebo naopak získávat data z uživatelské části aplikace a ukládat do databázového úložiště.

Veškeré tyto služby jsou definovány v balíku *cz.feec.xvajsa01.services*, kde každé databázové tabulce odpovídá jedna služba. Služba má opět strukturu klasické třídy doplněnou o dodatečné anotace. Nezbytné pro použití uvedených anotací je mít součástí projektu všechny potřebné knihovny uvedené v příloze B. Vzdálené služby umožňuje definovat rámec Spring [8], [10], [19].

Uvedené služba obsahuje tři metody :

- *getAllUsers()* - vrátí seznam všech uživatelů z databáze.
- *updateUser(Users user)* - provede aktualizaci uživatele v databázi na základě vstupního parametru.
- *getUserByName(String username)* - vratí pouze uživatele, jejichž uživatelské jméno se schoduje s hodnotou vstupního parametru.

Význam jednotlivých anotací je následující:

- *@RemotingDestination* - dovolí klientské aplikaci zavolat tuto třídu, jako vzdálený objekt.
- *@Service("usersService")* - parametr udává, pod jakým názvem, se bude k tomuto objektu přistupovat.
- *@RemotingInclude* - anotace použita při definici metod, umožní vzdálené volání metody.

```
@RemotingDestination
@Service("usersService")
public class UsersService extends MySessionFactory {

      @RemotingInclude
      public List<Users> getAllUsers() {
            return hibernateTemplate.findByExample(new Users());
      }
```

```
	@RemotingInclude
	public void updateUser(Users user) {
		hibernateTemplate.merge(user);
	}

	@RemotingInclude
	public List<Users> getUserByName(String username) {
		List<Users> user = hibernateTemplate.find(
			"from Users WHERE username=?", username);
		return user;
	}
}
```

5.1 Volání vzdáleného objektu z prostředí Adobe Flex

Výše uvedený příklad vytvoří na straně serveru službu, kterou je následně nutné prostřednictvím technologie Adobe Flex zpracovat. Prostřednictvím těchto služeb dochází k získávání dat ze serveru (databáze) a posílání dat směrem od uživatele na server a následné ukládání do databáze.

Pro správnou funkci je nutné nejprve definovat AMF kanál [18], po kterém komunikace probíhá a objekt, který bude odkazovat na danou službu na serveru. Dále jsou spolu s definicí tohoto objektu definovány metody pro obsluhu (parametry *result* a *fault*). Metoda uvedená v parametru *result* se provede v případě, že objekt je úspěšně zavolán, naopak metoda uvedená v parametru *fault* se provede v případě, že se vyskytne nějaká chyba během zpracovávaní. Více informací k tomuto tématu lze nalézt [15], [16].

```
<s:AMFChannel id="myamf" uri="messagebroker/amf" />

<s:ChannelSet id="channelSet" channels="{[myamf]}" />

<s:RemoteObject id="usersService" destination="usersService"
		channelSet="{channelSet}"
		result="resultHandlerUser(event)"
		fault="faultHandler(event)" />
```

Přístup k jednotlivým metodám vzdálené služby se provede prostřednictvím tečkové notace. Chceme-li tedy v prostředí Adobe Flex zavolat metodu *getAllUsers()*

ze služby *usersService*, tak to provedeme prostřednictvím id parametru, objektu odkazujícího se na vzdálenou službu:

```
usersService.getAllUsers()
```

Na základě toho příkazu dojde k vykonání těla metody uvedené v parametru *result* nebo *fault*, objektu odkazujícího se na vzdálenou službu.

5.2 Serializace datových typů

Aplikace je postavena na dvou technologiích, kde na straně serveru je použita technologie J2EE a straně klienta Adobe Flex. Při komunikaci mezi těmito technologiemi dochází k výměně dat, která mají určitý datový typ. Jelikož obě uvedené technologie používají různé datové typy a datové typy dodatečně definované, tak je pro jejich vzájemnou komunikaci nezbytná serializace, neboli překlad datových typů [21].

Překlad základních datový typů v prostředí obou platforem je zajištěn automaticky prostřednictvím nástroje BlazeDS. Problém nastává při definici vlastního datového typu, ať už na straně Javy nebo Flexu. S takovým typem si už samotný BlazeDS neporadí a je nutné mu s překladem pomoci.

Jestliže se podíváme na metodu *updateUser(Users user)* ze vzdálené služby, tak jako vstupní parametr vyžaduje objekt typu *Users*. Tento typ je definován na straně serveru v objektově relačním modelu aplikace, Adobe Flex tudíž tento datový typ nezná. Serializace na úrovni uživatelsky definovaného datového typů mezi Javou a Flexem se provádí tak, že se v prostředí obou platforem vytvoří stejný objekt se stejným názvem, stejnou strukturou a stejným pojmenováním a datovým typem jednotlivých vlastností. Dále se v prostředí Flexu tomuto objektu definuje, jakému objektu na straně Javy odpovídá a tudíž i do jakého typu objektu bude serializován (přeložen).

```
[Bindable]
[RemoteClass(alias="cz.feec.xvajsa01.model.Users")]
public class Users
{
      public function Users()
      {
      }

      private var _id:int;
      private var _username:String;
      private var _password:String;
```

```
private var _enabled:Boolean;
private var _name:String;
private var _surname:String;
private var _email:String;
private var _phone:String;
private var _street:String;
private var _postcode:int;
private var _city:String;
private var _userLog:ArrayCollection = new ArrayCollection();

public function get userLog():ArrayCollection
{
      return _userLog;
}

public function set userLog(value:ArrayCollection):void
{
      _userLog = value;
}
.... get/set metody jednotlivých vlastností třídy
```

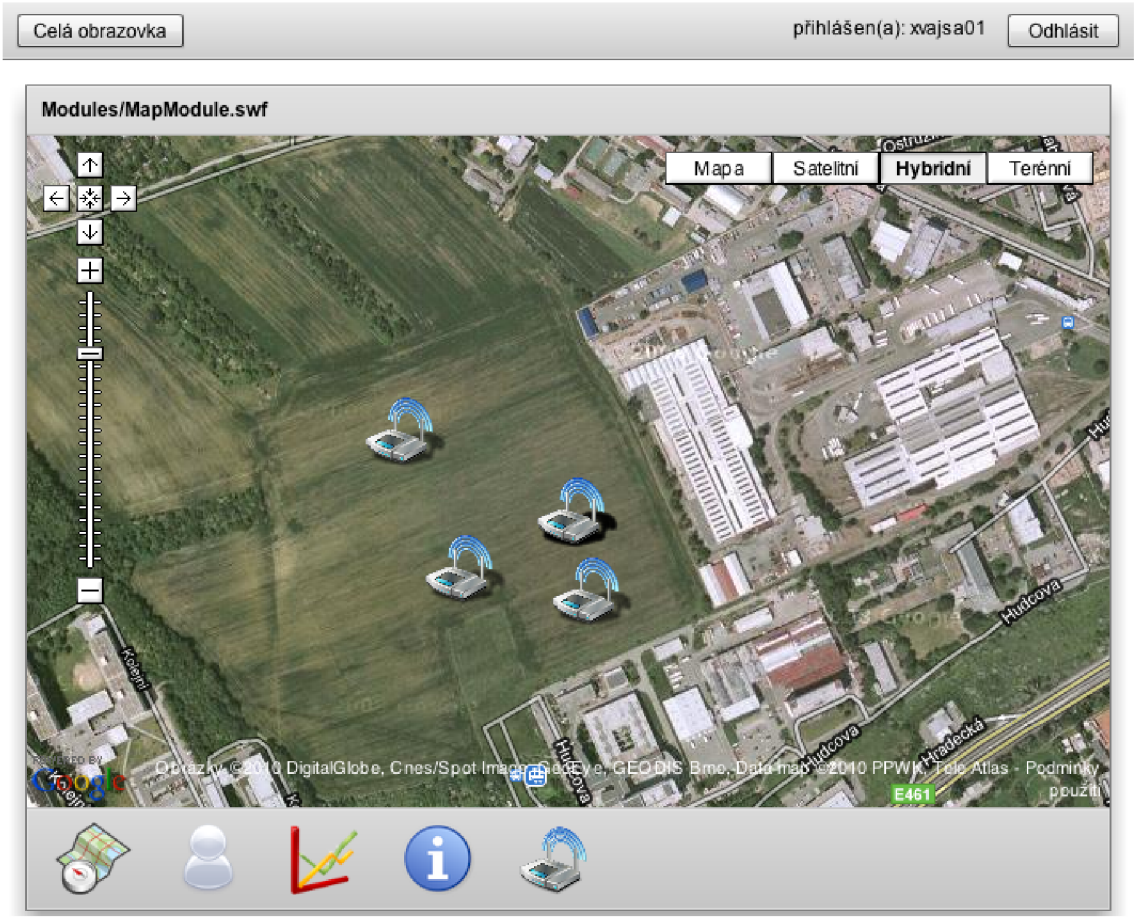
Takto definovaná třída ve Flexu, je totožná s třídou *cz.feec.xvajsa01.model.Users* na straně serveru. Parametr *RemoteClass* před definicí třídy udává, na jaký datový typ má být tento objekt serializován. Následně je možné ve Flexu vytvořit nový objekt typu *Users*, do tohoto objektu načíst data získaná ze serveru, nebo vložit data nová a použít objekt jako vstupní parametr metody *updateUser(Users user)*.

```
var user:Users = new Users();
user.username = "username";
user.name = "jmeno";
....
usersService.updateUser(user);
```

6 UŽIVATELSKÉ ROZHRANÍ APLIKACE

Uživatelské rozhraní aplikace, viz 6.1 je tvořeno několika částmi. V horní části obrazovku je to panel s informací o přihlášeném uživateli a tlačítky pro odhlášení z aplikace a pro zobrazení přes celou obrazovku.

Hlavní částí celého rozhraní je panel, který obsahuje aktuálně načtený modul. V hlavičce tohoto panelu je informace o názvu a cestě k tomuto modulu. Panel dále ve své spodní části obsahuje lištu ikon. Každá z těchto ikon představuje jeden modul a po kliknutí myší na tuto ikonu dojde k načtení příslušného modulu do panelu a paměti aplikace. Při výběru jiného modulu, je původní modul z pamětí odstraněn.



Obrázek 6.1: Uživatelské rozhraní aplikace

7 MODULY

Z obrázku 3.2 a jeho popisu 3.2 plyne, že navržená aplikace má za cíl být co nejvíce modulární a přizpůsobitelná pro maximální počet požadavků. Proto je klientská část aplikace rozdělena na dílčí části, neboli moduly. Jednotlivé moduly jsou mezi sebou nezávislé a poskytují svou vlastní funkcionalitu.Každý z těchto modulů představuje vlastní soubor s jeho zdrojovým kódem.

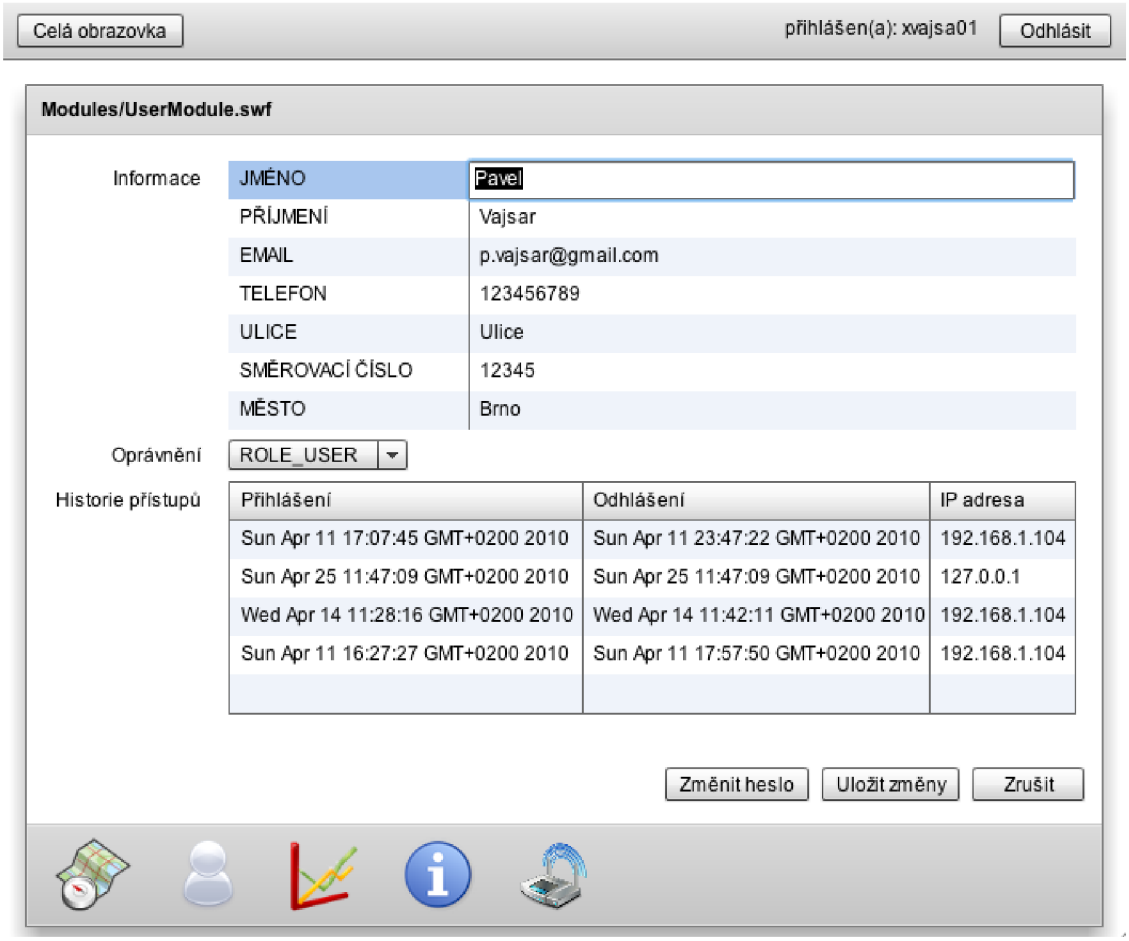
Výhodou tohoto řešení je snadnější a přehlednější úprava stávající funkce jednotlivých modulů a již zmíněná rozšířitelnost. Dále lze za výhodu považovat i to, že případná chyba v některém z modulů nezpůsobí pád celé aplikace, ale pouze modulu, ve kterém se chyba vyskytuje. Ostatní funkce aplikace zůstanou zachovány.

Jednotlivé moduly jsou nahrávány až za běhu aplikace a počet současně nahraných modulů v paměti je pouze jeden. To znamená, že při nahrání modulu, je původní modul odstraněn z paměti. Tím dochází k minimalizaci využité paměti aplikace a i k rychlejšímu zpracovávání požadavků na méně rychlých hardwarových platformách.

Samotné načítání modulů je prováděno za pomoci komponenty *ModuleLoader*. Tato komponenta je součástí vývojové platformy Flexu a nabízí široké možnosti využití, [20]. Jednotlivé moduly aplikace se nachází ve *wsn-flex/src/Modules*. Při vývoji modulu lze postupovat stejně, jako při vývoji klasické MXML aplikace s tím rozdílem, že uvozující tag je *mx:Module* namísto tagu *mx:Application*.

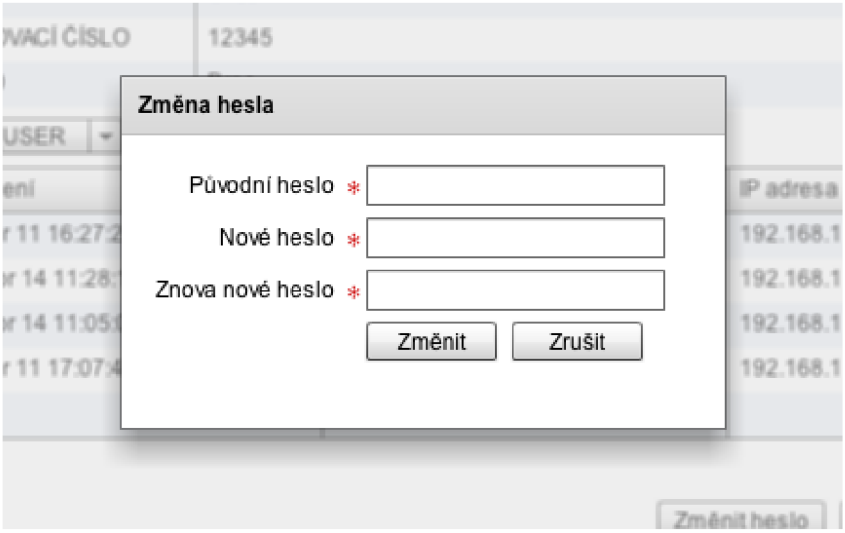
7.1 Modul uživatele

Prostřednictvím tohoto modulu lze zobrazit detailní informace o aktuálně přihlášeném uživateli, jemu přidělená oprávnění a historii přístupů do aplikace, viz 7.1. Detailní informace o uživateli lze jednoduše editovat. Po kliknutí na hodnotu, která má být editována se zobrazí kurzor a hodnotu je možné upravit. Klikem na tlačítko *Uložit změny* dojde k odeslání hodnot na server a k uložení do databázového úložiště. Jestliže tlačítko nebude stisknuto, tak se žádné změny v databázovém úložišti neprojeví.



Obrázek 7.1: Modul uživatele

Dále v tomto modulu mohou uživatelé měnit své heslo pro přístup do aplikace. Po kliknutí myší na tlačítko *Změnit heslo* dojde ke ztmavení stránky a v popředí se zobrazí panel se třemi položkami, viz 7.2. Pro úspěšné provedení změny hesla je nutné znát heslo původní a následně zadat dvakrát heslo nové. Jestliže původní heslo je shodné se zadaným. Respektive jestliže otisk hesla se shoduje s otiskem v databázi a obě vložená nová hesla se shodují, tak dojde ke změně hesla a uživatel se následně může do aplikace přihlásit se svým novým heslem.

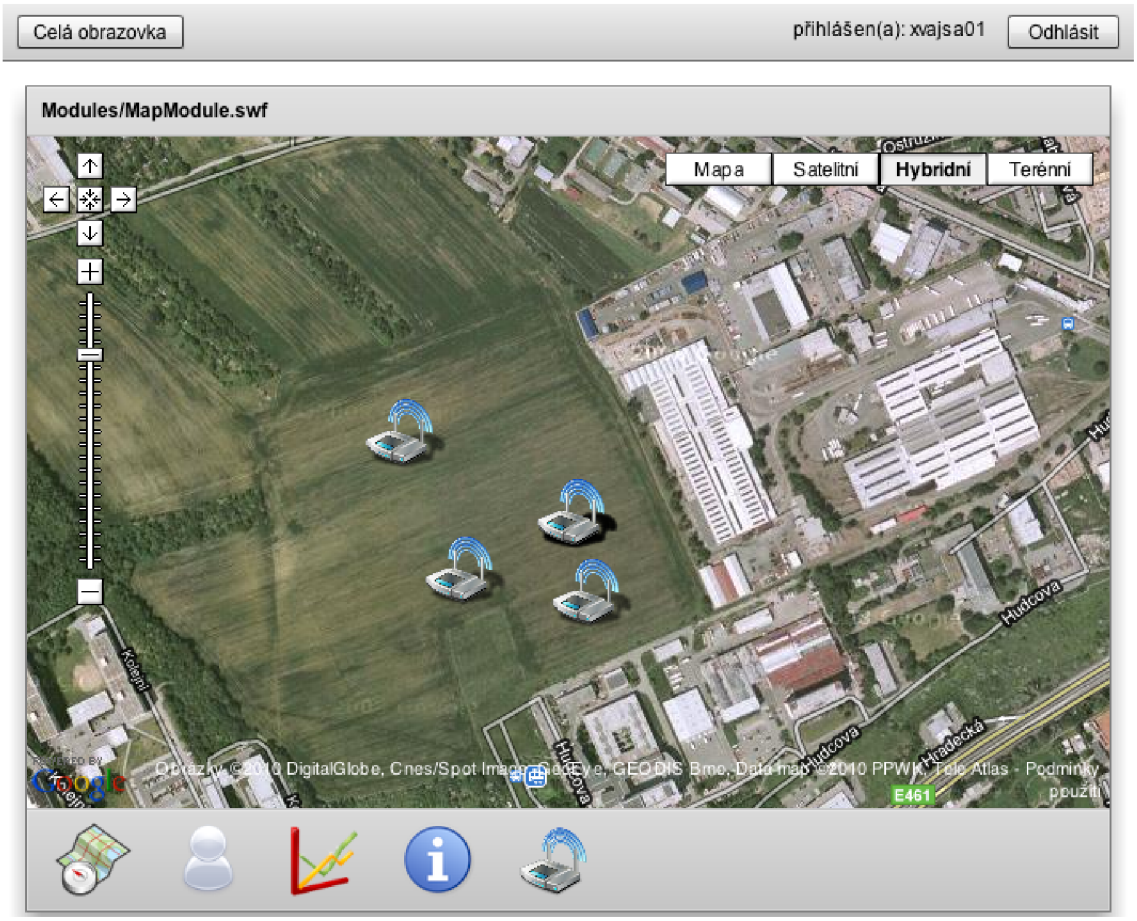


Obrázek 7.2: Změna hesla

Po stisknutí tlačítka *Zrušit* dojde k zavření okna a návratu k informacím o uživateli. Žádná změna hesla se neprovede.

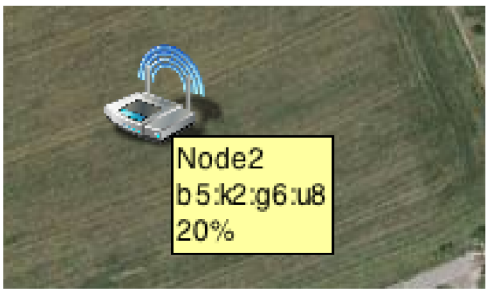
7.2 Modul mapy

Modul mapy byl vytvořen tak, aby zprostředkovával uživateli informace o přesné poloze jednotlivých uzlů. Jako mapové podklady byly použity Google Maps. Práce s Google mapou následně probíhá za pomoci Google Maps SDK, což je knihovna, kterou je nutné importovat do projektu [6].



Obrázek 7.3: Modul mapy

Po najetí kurzoru myši na příslušný senzor na mapě dojde k zobrazení základních informací o senzoru, viz 7.4. Jestliže kurzor následně opustí ikonu, tak se okno s informacemi o senzoru automaticky ztratí. Mezi základní zobrazené informace patří název senzoru, jeho hardwarová adresa a stav baterie. Pro získání dalších informací o senzoru lze využit modul senzoru 7.4.



Obrázek 7.4: Detail senzoru

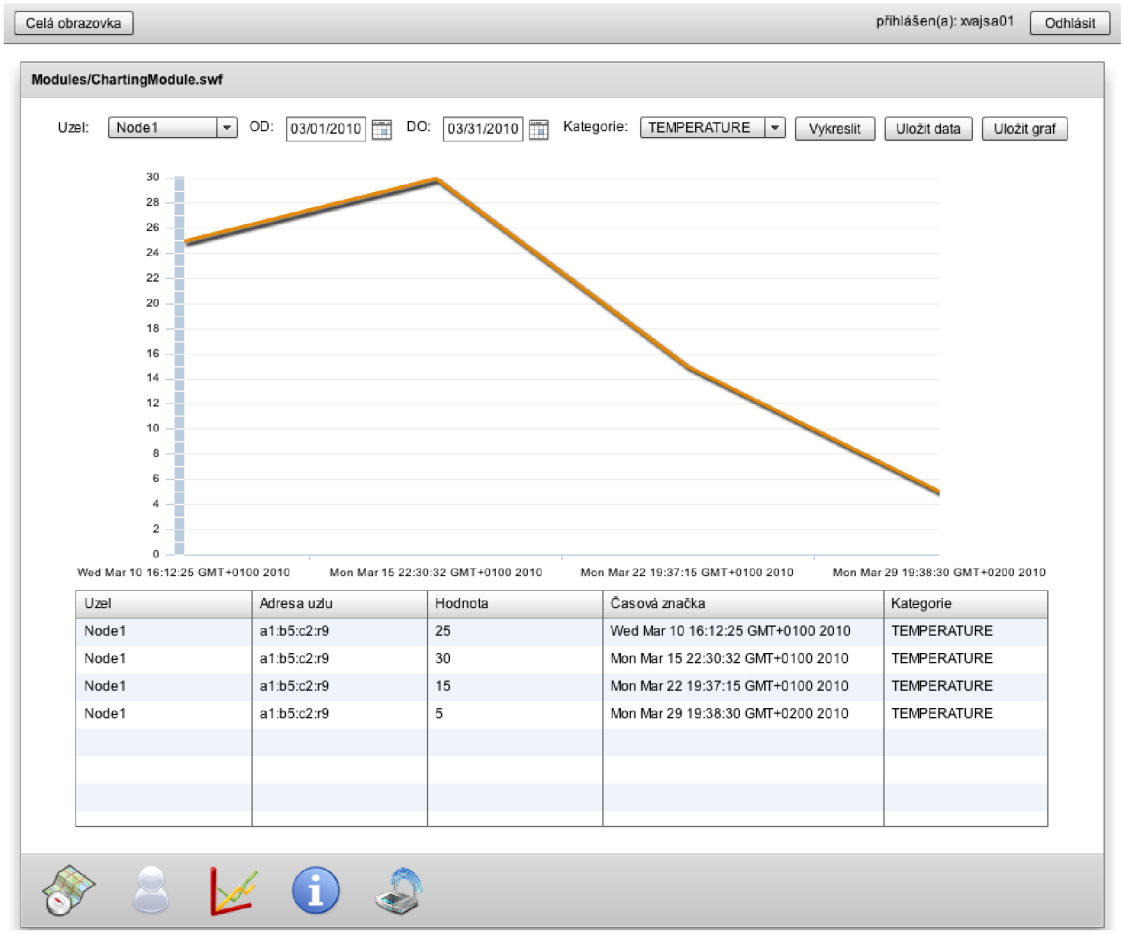
Dále mapa obsahuje ovládací prvky, kterými lze přibližovat/oddalovat pohled, pohybovat se po mapě a měnit typ mapy mezi klasickou, satelitní, hybridní a terenní.

7.3 Modul grafického zobrazení

Získaná data ze senzorové sítě je žádoucí v některých případech zobrazit v podobě grafu a sledovat tak změnu určité monitorované hodnoty například v závislosti na čase. Modul grafického zobrazení nejprve načte všechny uzly z databáze. Následně je nutné z těchto uzlů vybrat ten, jehož změřené hodnoty chceme zobrazit. Výběrem uzlu se dále automaticky načtou z databáze příslušné kategorie hodnot, které uzel schopen měřit. Výběrem jedné z těchto kategorií dojde k nahrání hodnot z databáze, viz 7.5.

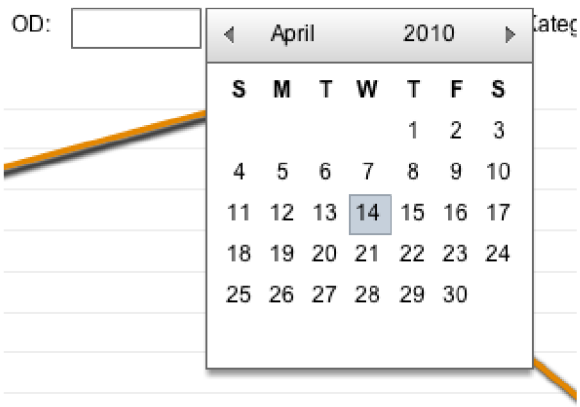
Po výběru uzlu (senzoru) a kategorie dat, je možné kliknutím na tlačítko *Vykreslit*, zobrazit hodnoty v podobě grafu. Graf bude následně sestaven z hodnot uvedených v tabulce pod grafem. Dále lze specifikovat období, které chceme sledovat. Kliknutím na ikonu kalendáře vedle polí *Od:* a *Do:* dojde k zobrazení kalendáře, prostřednictvím kterého lze vybrat počáteční a koncové datum sledovaného období. Následným kliknutím na tlačítko vykreslit dojde k zobrazení hodnot v grafu pouze pro vybrané období.

Pro další práci s daty mimo aplikaci lze stiskem tlačítka *Uložit data* naměřené hodnoty exportovat do souboru formátu CSV, který lze otevřít jako tabulku v některém z tabulkových procesorů. Struktura formátu CSV je taková, že jednotlivá pole (sloupce) jsou odděleny středníkem a konec řádku znakem $\backslash n$. Pro uložení grafu do souboru typu PDF lze stiknout tlačítko *Uložit graf*. Pro ukládání grafu do formátu PDF byla využita knihovna AlivePDF [24].



Uzel	Adresa uzlu	Hodnota	Časová značka	Kategorie
Node1	a1:b5:c2:r9	25	Wed Mar 10 16:12:25 GMT+0100 2010	TEMPERATURE
Node1	a1:b5:c2:r9	30	Mon Mar 15 22:30:32 GMT+0100 2010	TEMPERATURE
Node1	a1:b5:c2:r9	15	Mon Mar 22 19:37:15 GMT+0100 2010	TEMPERATURE
Node1	a1:b5:c2:r9	5	Mon Mar 29 19:38:30 GMT+0200 2010	TEMPERATURE

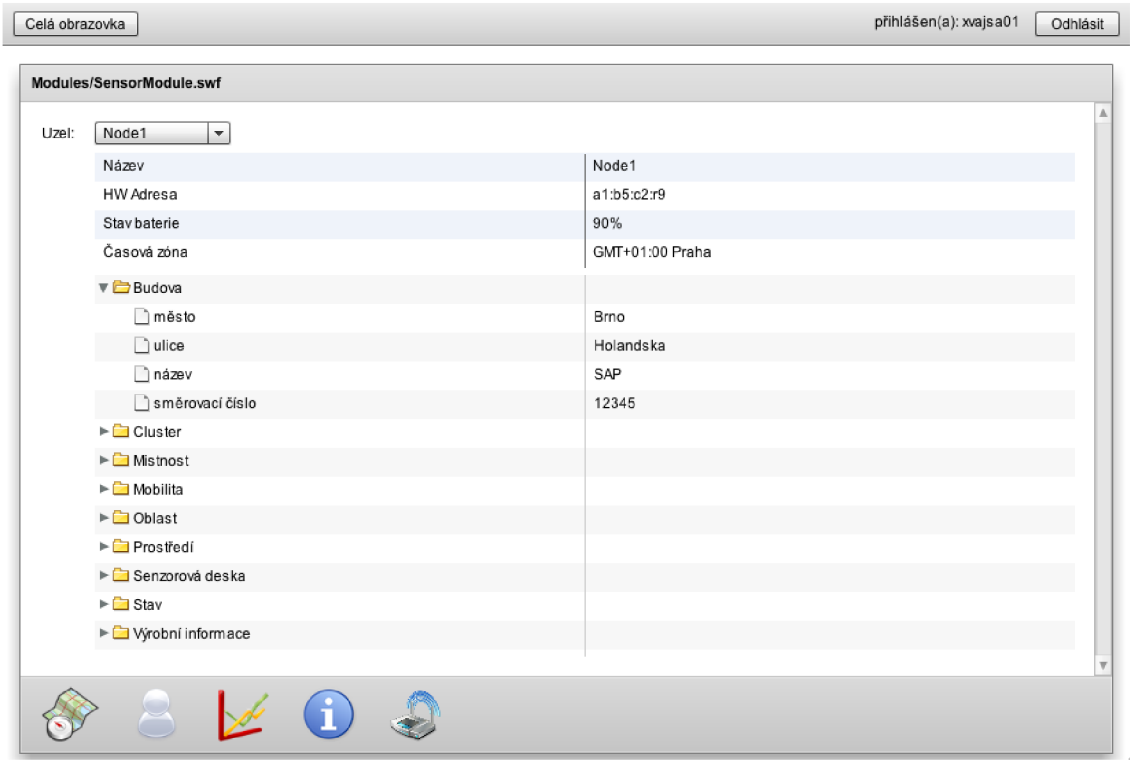
Obrázek 7.5: Modul grafického zobrazení



Obrázek 7.6: Specifikace sledovaného období

7.4 Modul senzorů

Modul senzorů slouží k zobrazování detailních informací o jednotlivých uzlech (senzorech). Po spuštění modulu se načtou názvy všech dostupných uzlů. Výběrem některého z nich dojde k detailnímu zobrazení informací o vybraném uzlu. Jednotlivé spolu-související informace jsou sdruženy do skupin (složek). Tyto složky jsou pojmenovány podle toho, které informace sdružují. Kliknutím na šipku vedle složky dojde k zobrazení všech vlastností a hodnot z dané skupiny, viz 7.7



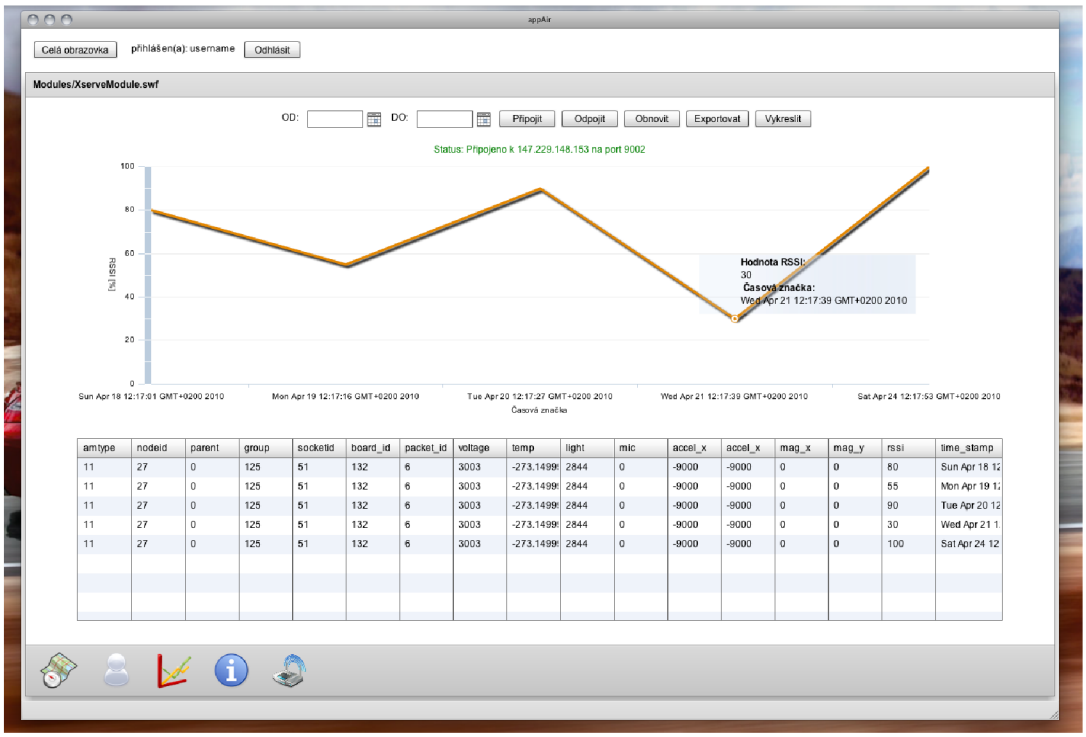
Obrázek 7.7: Modul senzorů

8 ADOBE AIR

Technologie Adobe Air [25] je obdobou technologie Adobe Flex [26] s tím rozdílem, že výsledná aplikace běží mimo okno internetového prohlížeče. Vývoj na platformě Adobe Air probíhá opět kombinací značkovacího jazyka MXML [4] a ActionScriptu [3].

Technologie Adobe Air byla v projektu použita z důvodů jak její nezávislosti na platformě (stejně jako Adobe Flex) a hlavně kvůli tomu, že je podporována i množstvím mobilních zařízení. Vytvořenou aplikaci lze spouštět na operačních systémech Mac OS X, Linux, Windows ale i na zařízeních např. s iPhone OS, Android a Windows Mobile. Podpora mobilních zařízení dále rozšiřuje univerzálnost celého řešení.

Aplikaci je tedy možné nahrát do přenosného mobilního zařízení a senzorovou síť monitorovat přímo v místě nasazení sítě. Aplikace je instalována lokálně a není tedy nutné ji při spuštění stahovat ze vzdáleného serveru, jako v případě přístupu k aplikaci prostřednictvím internetového prohlížeče. K charakteru rychlosti datových přenosů mobilních sítí lze tuto možnost považovat za velký přínos.



Obrázek 8.1: Okno AIR aplikace

Na obrázku 8.1 je vidět aplikace, která běží mimo okno internetového prohlížeče a to přímo v prostředí operačního systému. Dále na obrázku 8.2 je vidět ikona aplikace v panelu.

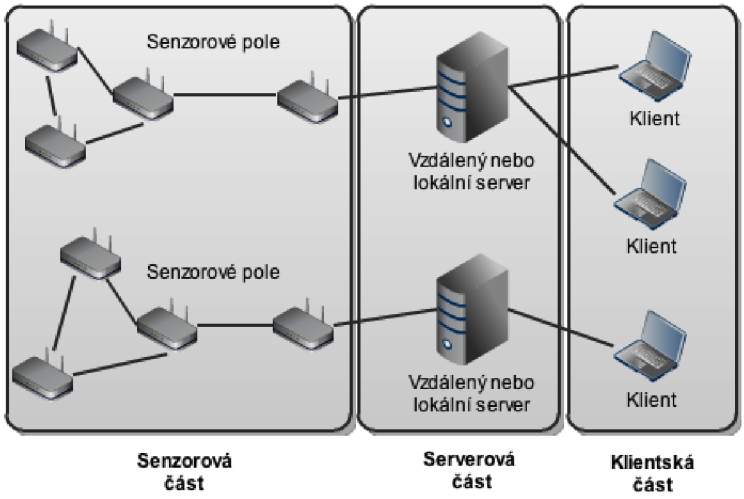
Obrázek 8.2: Ikona aplikace v panelu

Důležité je ještě zdůraznit to, že každá platforma pro mobilní zařízení má svá specifika a před nasazením aplikace na takovéto zařízení je nutné tyto specifika brát v úvahu a aplikaci dodatečně přizpůsobit danému zařízení. Řeč může být například o dotekových displejích těchto zařízení a podpora gest a multidoteků. Samotná platforma Adobe Air tyto vlastnosti podporuje a do budoucna se plánuje jejich integrace do aplikace.

9 APLIKACE A REÁLNÁ SENZOROVÁ SÍŤ

Jedním z cílů celého projektu bylo i to, aby byl opravdu použitelný v praxi pro monitorování reálných senzorových sítí. Jednou takovou experimentální sítí disponuje Ústav telekomunikací Fakulty elektrotechniky a komunikačních technologií, kde byla aplikace testována. V podstatě se nejedná pouze o síť, ale o celé řešení od společnosti Crossbow Technology [23], které se skládá z několika částí, které spolu vzájemně komunikují a jsou nezbytné pro chod celé sítě a jejího monitorování. Části jsou následující:

- Klientská část - poskytuje uživateli informace o síti a skrze tuto část je možné síť konfigurovat.
- Serverová část - překlad dat ze senzorové sítě do potřebných formátů, vyrovnávací paměť pro data zasílaná sítí, tvoří most mezi senzorovou a klientskou částí.
- Senzorová část - software běžící na každém z uzlů senzorové sítě.

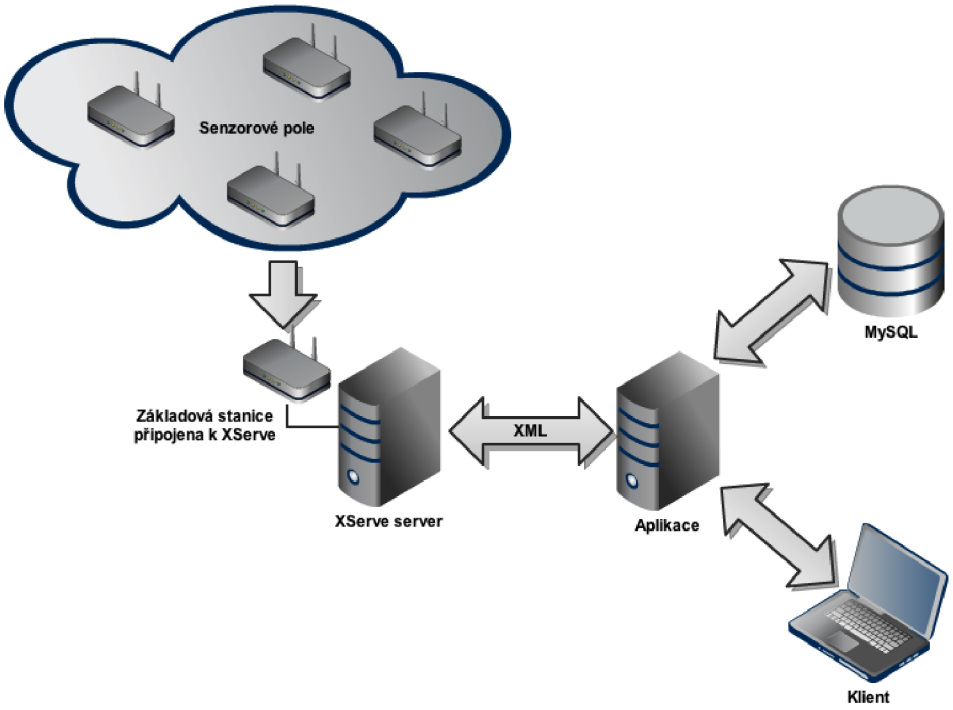


Obrázek 9.1: Řešení od společnosti Crossbow z pohledu jednotlivých částí

Na obrázku 9.1 je patrné celé řešení z pohledu jednotlivých části. Na jednotlivých částech, je instalován příslušný software, který se stará o činnost daného zařízení. V senzorové části je to *XMesh*, což je v podstatě operační systém senzoru. V serverové části je to nástroj *XOtap* a *XServe*, který přebírá data od senzorů a s těmito daty provádí příslušné operace. To jaké operace s daty provádí záleží na konkrétním nastavení *XServe*. Může data od senzorů např. posílat a rovnou ukládat do databáze, ukládat data do XML souboru, nebo posílat XML stream dat na svou IP adresu a příslušný port.

9.1 Nastavení XServe

XServe tvořící serverovou část řešení od Crossbow Technology je část, na kterou je aplikace pro monitoring napojena, viz 9.1. Pro komunikaci bezdrátové senzorové sítě a vyvíjené aplikace byl zvolen formát XML. *XServe* je tedy nakonfigurován tak, aby data získaná od jednotlivých senzorů, posílal na svou IP adresu a nastavený port, viz 9.2.



Obrázek 9.2: Napojení aplikace na reálnou senzorovou síť

Na obrázku 9.2 je znázorněna komunikační struktura mezi bezdrátovou senzorovou sítí, aplikací a klientem, který k aplikaci přistupuje. Základová stanice má podobu senzoru bezdrátové senzorové sítě a je připojena pevným spojem k počítači, na kterém je instalována aplikace *XServe*. Tento počítač je připojen do sítě Internet a disponuje veřejnou IP adresou. *XServe* je následně nakonfigurován tak, aby odesílal data od jednotlivých senzorů na veřejnou IP adresu počítače a na port 9002. Data jsou posílána po jednotlivých paketech a každý paket je před odesláním konvertován do formátu XML. Aplikace následně může být nasazena na jiném serveru a následně prostřednictvím sítě Internet tento stream XML dat odebírat. Pro odebírání streamu bylo zapotřebí navrhnout a implementovat konektor, který je schopen odebírat a zpracovávat XML data, více o konektoru v kap. 9.2. Konektor následně provádí ukládání přijatých dat ze senzorové sítě do navrženého databázového úložiště. Klient následně skrze aplikaci může s těmito daty pracovat.

9.2 Konektor

Úkolem konektoru je přebírání dat od serveru, ke kterému je připojena základová stanice. Na vyžádání konektor vytvoří vlákno se soket spojením, kterým se připojí k veřejné IP adrese a portu serveru, na kterém je spuštěna a nakonfigurována služba *XServe*. Tato služba do vytvořeného soket spojení zasílá data ze senzorové sítě a konektor v případě detekce posílání dat tyto data zachytává a nahrává je do vstupní vyrovnávací paměti. Data jsou zachytávána po jednotlivých blocích, kde každý z bloků představuje jeden paket od jednoho senzoru v XML podobě. Konektor přijatá data následně rozloží na jednotlivé vlastnosti a jim přidružené hodnoty a ukládá je do databázového úložiště.

Vstupní vyrovnávací paměť je důležitá protože senzorová síť může pakety posílat v časových intervalech v jednotkách milisekund a mohlo by se stát, že aplikace nebude schopna data zpracovávat v reálném čase. Tím by mohlo docházet ke ztrátě některých paketů. Existence vstupní vyrovnávací pamětí tento problém eliminuje.

Konektor je definován ve třídě s názvem *CrossbowConnector.java*, která se nachází v balíku *cz.feec.xvajsa01.crossbow* a jako vzdálenou službu lze konektor volat pod názvem *crossbowConnector*.

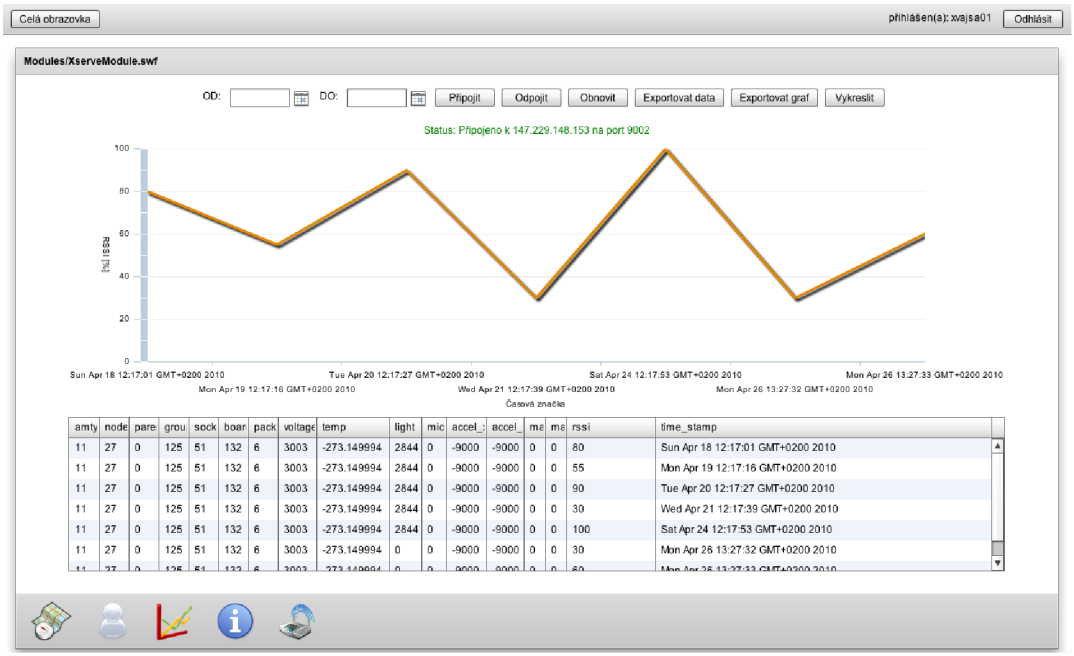
```
<?xml version="1.0" ?>
<MotePacket>
   <ParsedDataElement>
      <Name>nodeid</Name>
      <ConvertedValue>1</ConvertedValue>
   </ParsedDataElement>
   <ParsedDataElement>
      <Name>temp</Name>
      <ConvertedValue>3.097866</ConvertedValue>
   </ParsedDataElement>
   ...
</MotePacket>
```

Obrázek 9.3: Ukázka struktury XML paketu

Na ukázce 9.3 je patrný příklad formátu XML paketu, které konektor odebírá na příslušné IP adrese a portu. Celý paket začíná XML hlavičkou a následně začátkem paketu uvozeného tagem *MotePacket*. Jednotlivé vlastnosti jsou uvedeny uvnitř tagu *Name* a hodnoty příslušící k těmto vlastnostem jsou uvedeny v tagu *ConvertedValue*. Z uvedené ukázky tedy vyplývá, že senzor změřil teplotu o hodnotě 3.097866°C.

9.3 Modul aplikace pro monitorování reálné WSN

Po návrhu a implementaci konektoru na reálnou bezdrátovou síť byl vytvořen modul, který umožňuje ovládání tohoto konektoru, tzn. spuštění/zastavení zachytávání paketů. Dále umožňuje zobrazovat získaná data a na základě těchto dat vytvářet graf hodnot RSSI (síla signálu senzoru) v závislosti na čase.

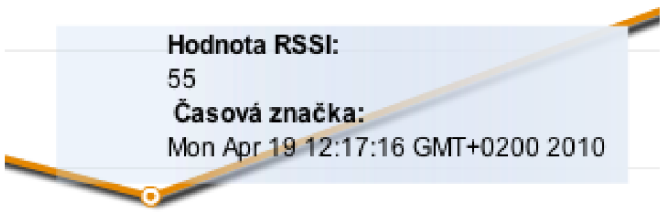


Obrázek 9.4: Modul pro monitorování reálné WSN

V hlavní části modulu se nacházejí tlačítka pro spuštění jednotlivých akcí a ovládání modulu. Význam jednotlivých tlačítek je následující:

- *OD:* - po kliknutí na ikonu kalendáře vedle tohoto pole dojde k zobrazení kalendáře, kde lze následně vybrat datum od kterého se mají zobrazit hodnoty při následném vykreslování grafu.
- *DO:* - stejná funkce jako předchozí tlačítko s tím rozdílem, že jeho výběrem definujeme datum, do kterého má být graf vykreslen.
- *Připojit* - tlačítko pro ovládání konektoru, po jeho stisku dojde k vytvoření soket spojení na příslušný server, následně odebíraná data jsou ukládána do databázového úložiště. Socket spojení je navázáno v novém vlákně, proto je možné v aplikaci provádět další úkony aniž by se spojení přerušilo (např. načíst a pracovat v jiném modulu). Indikace spojení je patrná z lišty pod tlačítky, kde je zobrazen stav připojení.

- *Odpojit* - Ukončí spojení se serverem.
- *Obnovit* - během aktivního spojení se serverem se ukládala data do databázového úložiště. Stiskem tohoto tlačítka se provede jejich aktualizace (načtení) do uživatelského rozhraní aplikace, tzn. všechna data z databáze se načtou do tabulky umístěné pod grafem, viz obrázek 9.4.
- *Exportovat data* - v některých případech je žádoucí naměřená data jednoduchou formou získat z aplikace pro možnost pozdější analýzy nebo úpravy. Stiskem toho tlačítka se vyvolá výzva pro uložení dat z tabulky pod grafem do souboru typu CSV. Tento typ souboru lze následně otevřít v některém z tabulkový procesorů jako je např. MS Office Excel, OpenOffice Calc.
- *Exportovat graf* - pro pozdější analýzu si lze uložit i průběh právě zobrazeného grafu. Stiskem tohoto tlačítka dojde k zobrazení výzvy pro uložení grafu s popisy jeho os do souboru formátu PDF. Pro ukládání obsahu do formátu PDF byla použita knihovna AlivePDF [24] pro prostředí Adobe Flex.
- *Vykreslit* - po stisku tohoto tlačítka dojde k vykreslení grafu, jestliže není specifikován interval, ve kterém se mají hodnoty zobrazit, tak se vytvoří graf ze všech hodnot, které jsou uvedeny v tabulce pod grafem.



Obrázek 9.5: Detail hodnot grafu

Po vykreslení grafu a následném přiblížení kurzoru myši k průběhu grafu dojde k zobrazení vyskakujícího okénka s příslušnými hodnotami osy X a osy Y včetně popisu os, viz obrázek 9.5. Po přemístění kurzoru od grafu toto okénko automaticky zmizí.

Zdrojové kódy tohoto modulu jsou umístěny v klientské části aplikace *wsn-flex/src/Modules* v souboru *XserveModule.mxml*.

10 ZÁVĚR

Cílem práce nebylo navrhnout pouze uživatelské rozhraní pro monitorování, ale celé řešení pro monitorování maximálního počtu bezdrátových senzorových sítí. Tento požadavek byl brán v úvahu i při výběru použitých technologií. Serverová část aplikace může tedy být nasazena na serverech založených na platformách Linux, Windows i Mac OS X a naopak i přístup ke klientské části aplikace a práce s ní může probíhat opět z jakékoliv již jmenované platformy.

Jelikož problematika a nasazení bezdrátových sítí je velice různorodá a realizací takových sítí existuje mnoho, tak byl kladen důraz na to, aby výsledné řešení bylo přizpůsobitelné dané síti. Z toho plynul požadavek na modulárnost aplikace, kde jednotlivé moduly lze upravovat pro potřeby konkrétní sítě a následně i vytvářet moduly nové a dodatečné je integrovat do aplikace a tím rozšiřovat její funkcionalitu. Dále se univerzálnost promítla i v použitých technologiích při vývoji. Databázové úložiště je přepsáno do objektově relačního modelu což umožňuje použití téměř jakéhokoliv typu databázového úložiště s minimálním zásahem do zdrojového kódu aplikace. Aplikace si následně z objektově relačního modelu vytvoří strukturu relační databáze přímo na databázovém serveru.

Závěrečná část práce je věnována nasazení aplikace pro monitorování reálné senzorové sítě. Zde bylo nutné vytvořit konektor, který bude schopen se připojit k této síti, získávat ze sítě data, tyto data zpracovávat a následně je ukládat do navrženého databázového úložiště. Pro práci s těmito daty bylo následně nutné vytvořit modul, který uložená data z databáze ve vhodné podobě zobrazí a umožní uživateli s daty pracovat. Pro další analýzu a práci s daty umožňuje modul zobrazená data exportovat do souboru CSV. Graf lze exportovat do souboru PDF.

Tímto ale vývoj aplikace nekončí a dále se počítá s vylepšením podpory mobilních zařízení, vylepšení modulu pro měření úrovně signálu v závislosti na vzdálenosti jednotlivých uzlů od základové stanice, měření doby životnosti uzlu a podporou dalších funkcí.

REFERENCE

[1] MAINWARING, Alan *Wireless Sensor Networks for Habitat Monitoring. WSNA : 02* [online]. 2002 [cit. 2009-11-30]. Dostupné z URL: <http://www.polastre.com/papers/wsna02.pdf>.

[2] Wikipedia *Wireless sensor network* [online]. 2009 [cit. 2009-11-30]. Dostupné z URL: <http://en.wikipedia.org/wiki/Wireless_sensor_network>.

[3] ActionScript *ActionScript* [online]. 2000, 2009 [cit. 2009-12-02]. Dostupné z URL: <http://www.actionscript.org>.

[4] Adobe *Coding with MXML and ActionScript* [online]. 2009 [cit. 2009-12-02]. Dostupné z URL: <http://www.adobe.com/devnet/flex/quickstart/coding_with_mxml_and_actionscript>.

[5] Google *Google Mapy* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://maps.google.cz/>.

[6] Google *Google Code : Google Maps API for Flash* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://code.google.com/intl/cs/apis/maps/documentation/flash/>.

[7] *JBoss Community : Hibernate* [online]. 2006 [cit. 2010-04-15]. Dostupné z URL: <http://www.hibernate.org/>.

[8] *SpringSource. Spring BlazeDS Integration Resources* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://www.springsource.org/spring-flex>.

[9] *Adobe. Adobe Open Source : BlazeDS* [online]. 2008, 2009-10-20 [cit. 2010-04-15]. Dostupné z URL: <http://opensource.adobe.com/wiki/display/blazeds/BlazeDS/>.

[10] *Spring Source : Spring Documentation* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://www.springsource.org/documentation>.

[11] *Spring Source : Spring Security* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://static.springsource.org/spring-security/site/index.html>.

[12] *MySQL : Documentation* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://dev.mysql.com/doc/>.

[13] *Apache Tomcat 6.0 : User Guide* [online]. 1999-2010 [cit. 2010-04-15]. Dostupné z URL: <http://tomcat.apache.org/tomcat-6.0-doc/index.html>.

[14] *SCHUMACHER, Maik . Flex Developer Center : Adobe Flex, BlazeDS, and Hibernate JPA on Tomcat and MySQL — Part 1: Putting it all together in a simple demo application* [online]. 2008 [cit. 2010-04-15]. Dostupné z URL: <http://www.adobe.com/devnet/flex/articles/flex_hibernate.html>.

[15] *WARD, James. James Ward – RIA Cowboy : Flash Builder 4 Data Wizards with Java / Spring* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://www.jamesward.com/2010/01/11/flash-builder-4-data-wizards-with-java-spring/>.

[16] *WARD, James. James Ward – RIA Cowboy : Flex 4, Java, Spring, and Hibernate in Flash Builder 4* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://www.jamesward.com/2010/01/17/flex-4-and-java-spring-hibernate-in-flash-builder-4/>.

[17] *Spring Framework : Object Relational Mapping (ORM) data access* [online]. 2009 [cit. 2010-04-15]. Dostupné z URL: <http://static.springsource.org/spring/docs/2.5.x/reference/orm.html>.

[18] *WARD, James; ROSE, Jon. Flex & Spring Integration* [online]. Maynard:DZone, 2009 [cit. 2010-04-15]. Dostupné z URL: <http://refcardz.dzone.com/refcardz/flex-spring-integration>.

[19] *GRELLE, Jeremy. Spring BlazeDS Integration : Reference Guide* [online]. 2010 [cit. 2010-04-15]. Dostupné z URL: <http://static.springsource.org/spring-flex/docs/1.0.x/reference/html/index.html>.

[20] *Creating Modular Applications. Adobe : Matthew J. Horn,* [online]. 2007. 50 s. Dostupné z URL: <http://blogs.adobe.com/flexdoc/modules/>.

[21] *Adobe. Converting data from ActionScript to Java : Serializing between ActionScript and Java.* [online]. 2009. 50 s. Dostupné z URL: <http://livedocs.adobe.com/blazeds/1/blazeds_devguide/help.html?content=serialize_data_2.html>.

[22] *Hašovací funkce* [online]. 2010 [cit. 2010-05-10]. Wikipedie Dostupné z URL: <http://cs.wikipedia.org/wiki/Hašovací_funkce>.

[23] *Crossbow Technology. Wireless Sensor Networks* [online]. 2009 [cit. 2010-05-10]. Crossbow Dostupné z URL: <http://www.xbow.com/Home/wHomePage.aspx>.

[24] *ALIVEPDF 0.1.5 [RC] NEW API'S* [online]. 2009 [cit. 2010-05-10]. ALIVEPDF Dostupné z URL: <http://alivepdf.bytearray.org/>.

[25] *Adobe Systems Incorporated. Adobe AIR Technologies* [online]. 2010 [cit. 2010-05-10]. Dostupné z URL: <http://labs.adobe.com/technologies/air/>.

[26] *Adobe Systems Incorporated. What is Flex?* [online]. 2010 [cit. 2010-05-10]. Dostupné z URL: <http://www.adobe.com/products/flex/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

WSN bezdrátová senzorová síť – Wireless Sensor Network

UI uživatelské rozhraní – User Interface

PN senzorové pole – Patch Network

ERD Diagram struktury databáze – Entity Relationship Diagram

FK Cizí klíč – Foreign Key

PK Primární klíč – Primary Key

AS ActionScript

MXML Minimal XML

SEZNAM PŘÍLOH

A BEZDRÁTOVÁ SENZOROVÁ SÍŤ

A.1 DATABÁZOVÉ ÚLOŽIŠTĚ

NODE		
FK	**Atribut**	**Popis**
PK	ID_NODE	primární klíč tabulky
FK	ID_STATE	reference na stav, ve kterém se senzor nachází (online, offline)
FK	ID_AREA	reference na oblast, ve které se senzor nachází
FK	ID_BUILD	reference na budovu, ve které se senzor nachází
FK	ID_ROOM	reference místnost, ve které se senzor nachází
FK	ID_MOBILITY	jestliže je uzel mobilní, tak je v tomto atributu uvedeno, jaký typ mobility podporuje
FK	ID_EXTENDED_AUTHORITY	reference na typ rozšířené autority, kterou má uzel přiřazenou
FK	ID_PRODUCT_TYPE	reference na typ senzoru
FK	ID_ENVIRONMENT	popis prostředí, ve kterém se senzor nachází
FK	ID_SENSORE_BOARD	typ přidružené senzorové desky
FK	ID_CLUSTER	reference na cluster, ke kterému senzor náleží
-	NAME	název (jméno) senzoru
-	HW_ADDRESS	hardwarová adresa senzoru
-	POWER	množství dostupné energie (stav akumulátoru)
-	TIMEZONE	časové pásmo, ve kterém se senzor nachází

Tabulka A.1: NODE

STATE		
FK	**Atribut**	**Popis**
PK	ID_STATE	primární klíč tabulky
-	NAME	reference na stav, ve kterém se senzor nachází (online, offline)
-	DESCRIPTION	reference se oblast, ve které se senzor nachází
-	DATA_SENDING_FREQ	informace o tom jak často senzor zasílá data směrem k bráně

Tabulka A.2: STATE

AREA		
FK	**Atribut**	**Popis**
PK	ID_AREA	primární klíč tabulky
FK	ID_COUNTRY	reference na název země (státu) na jehož území daná oblast leží
FK	ID_ENVIRONMENT	reference specifikující danou oblast (údaje o typu krajiny)
FK	ID_TYPE	reference na typ použitých souřadnic
-	NAME	název oblasti
-	PROVINCE	název kraje (okresu) kde se oblast rozkládá

Tabulka A.3: AREA

BUILD		
FK	**Atribut**	**Popis**
PK	ID_BUILD	primární klíč tabulky
FK	ID_AREA	informace o budově, ve které se místnost nachází
-	NAME	název oblasti
-	STREET	ulice, jako adresa budovy
-	POSTCODE	směrovací číslo vztahující se k adrese
-	CITY	město ve kterém budova leží

Tabulka A.4: BUILD

ROOM		
FK	**Atribut**	**Popis**
PK	ID_ROOM	primární klíč tabulky
FK	ID_BUILD	informace o budově, ve které se místnost nachází
-	NAME	název místnosti
-	FLOOR	poschodí, na kterém místnost leží v rámci budovy
-	NUMBER	číslo místnosti

Tabulka A.5: ROOM

MOBILITY		
FK	**Atribut**	**Popis**
PK	ID_MOBILITY	primární klíč tabulky
-	NAME	název
-	SPEED	rychlost jakou se senzor může pohybovat
-	DESCRIPTION	popis

Tabulka A.6: MOBILITY

MOBILITY_AREA		
FK	**Atribut**	**Popis**
PK	ID_MOBILITY_AREA	primární klíč tabulky
FK	ID_MOBILITY	reference na příslušnou mobilitu
FK	ID_AREA	reference na oblast, pro kterou mobilita platí

Tabulka A.7: MOBILITY_AREA

MOBILITY_BUILD		
FK	**Atribut**	**Popis**
PK	ID_MOBILITY_BUILD	primární klíč tabulky
FK	ID_MOBILITY	reference na příslušnou mobilitu
FK	ID_BUILD	reference na budovu, pro kterou mobilita platí

Tabulka A.8: MOBILITY_BUILD

MOBILITY_ROOM		
FK	**Atribut**	**Popis**
PK	ID_MOBILITY_ROOM	primární klíč tabulky
FK	ID_MOBILITY	reference na příslušnou mobilitu
FK	ID_ROOM	reference na místnost, pro kterou mobilita platí

Tabulka A.9: MOBILITY_ROOM

EXTENDED_AUTHORITY		
FK	**Atribut**	**Popis**
PK	ID_EXTENDED_AUTHORITY	primární klíč tabulky
-	NAME	název
-	DESCRIPTION	popis

Tabulka A.10: EXTENDED_AUTHORITY

PRODUCT_TYPE		
FK	**Atribut**	**Popis**
PK	ID_PRODUCT_TYPE	primární klíč tabulky
FK	ID_PRODUCER	odkaz na informace o výrobci
FK	ID_SENSORE_BOARD	odkaz na typ senzorové desky přidružené k senzoru
-	NAME	název produktu

Tabulka A.11: PRODUCT_TYPE

PRODUCT_TYPE_PROPERTY		
FK	**Atribut**	**Popis**
PK	ID_PRODUCT_TYPE_PROPERTY	primární klíč tabulky
FK	ID_PRODUCT_TYPE	reference na typ produktu
FK	ID_PROPERTY	reference na vlastnosti (parametry) tohoto produktu

Tabulka A.12: PRODUCT_TYPE_PROPERTY

ENVIRONMENT		
FK	**Atribut**	**Popis**
PK	ID_ENVIRONMENT	primární klíč tabulky
-	NAME	název typu prostředí
-	DESCRIPTION	popis prostředí

Tabulka A.13: ENVIRONMENT

SENSORE_BOARD		
FK	**Atribut**	**Popis**
PK	ID_SENSORE_BOARD	primární klíč tabulky
-	SENSORE_COUNT	počet senzorů umístěných na senzorové desce
-	DESCRIPTION	popis senzorové desky

Tabulka A.14: SENSORE_BOARD

SENSORE_BOARD_PROPERTY		
FK	**Atribut**	**Popis**
PK	ID_SENSORE_BOARD_PROPERTY	primární klíč tabulky
FK	ID_PROPERTY	ukazatel na vlastnosti senzorové desky
FK	ID_SENSORE_BOARD	ukazatel na senzorovou desku

Tabulka A.15: SENSORE_BOAR_PROPERTY

COUNTRY		
FK	**Atribut**	**Popis**
PK	ID_COUNTRY	primární klíč tabulky
-	NAME	název země
-	COUNTRY_CODE	kód země

Tabulka A.16: COUNTRY

PRODUCER		
FK	**Atribut**	**Popis**
PK	ID_PRODUCER	primární klíč tabulky
-	NAME	název země
-	COUNTRY	zěmé, kde má výrobce sídlo
-	STREET	ulice
-	POSTCODE	směrovací číslo
-	CITY	město

Tabulka A.17: PRODUCER

PROPERTY		
FK	**Atribut**	**Popis**
PK	ID_PROPERTY	primární klíč tabulky
-	PROPERTY_NAME	název vlastnosti
-	VALUE1	hodnota 1 vlastnosti (např. minimální hodnota rozsahu)
-	VALUE2	hodnota 2 vlastnosti (např. maximální hodnota rozsahu)
-	VALUE3	hodnota 3 vlastnosti (např. přesnost)
-	DESCRIPTION	popis dané vlastnosti

Tabulka A.18: PROPERTY

NODEGROUP		
FK	**Atribut**	**Popis**
PK	ID_NODEGROUP	primární klíč tabulky
-	NAME	název skupiny
-	DESCRIPTION	popis

Tabulka A.19: NODEGROUP

NODE_GROUP		
FK	**Atribut**	**Popis**
PK	ID_NODE_GROUP	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_NODEGROUP	reference na skupinu, ke které uzel patří

Tabulka A.20: NODE_GROUP

LINE		
FK	**Atribut**	**Popis**
PK	ID_LINE	primární klíč tabulky
-	NAME	název
-	BANDWIDTH	šířka pásma
-	ATTENUATION	útlum linky
-	POWER	úroveň signálu
-	BER	chybovost linky
-	STATUS	stav linky
-	DESCRIPTION	popis

Tabulka A.21: LINE

NODE_LINE		
FK	**Atribut**	**Popis**
PK	ID_NODE_LINE	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_LINE	reference na linku, ke které je uzel připojen

Tabulka A.22: NODE_LINE

POSITION		
FK	**Atribut**	**Popis**
PK	ID_POSITION	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_TYPE_COORDINATION	reference na informace o typu použitých souřadnic (GPS, zeměpisné, atd.)
-	GPS	GPS souřadnice
-	X	x souřadnice
-	Y	y souřadnice
-	Z	z souřadnice
-	GEOGRAPHIC	zeměpisné souřadnice
-	START_POINT_XYZ	výchozí bod pro relativní (kartezké) souřadnice

Tabulka A.23: POSITION

TYPE_COORDINATION		
FK	**Atribut**	**Popis**
PK	ID_TYPE_COORDINATION	primární klíč tabulky
-	NAME	název typu
-	DESCRIPTION	popis typu souřadnic

Tabulka A.24: TYPE_COORDINATION

NEIGHBOUR		
FK	**Atribut**	**Popis**
PK	ID_ NEIGHBOUR	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_NEIGHBOUR_NODE	reference na sousední uzel
-	DISTANCE	vzdálenost mezi uzly
-	DIRECT_ CONNECT	indikátor přímého spojení dvou uzlů

Tabulka A.25: NEIGHBOUR

CLUSTER		
FK	**Atribut**	**Popis**
PK	ID_ CLUSTER	primární klíč tabulky
FK	CLUSTER_HEAD_NODE	reference na řídící uzel klusteru
-	NAME	název klusteru
-	DESCRIPTION	popis

Tabulka A.26: CLUSTER

DATA		
FK	**Atribut**	**Popis**
PK	ID_ DATA	primární klíč tabulky
FK	ID_NODE	reference na uzel, který data měřil
FK	ID_SUBCATEGORY	kategorie dat (měřená hodnota, systémové informace, atd.)
-	VALUE	změřená hodnota
-	DATE_MEASURE	datum měření
-	TIME_MEASURE	čas měření

Tabulka A.27: DATA

SUBCATEGORY		
FK	**Atribut**	**Popis**
PK	ID_SUBCATEGORY	primární klíč tabulky
FK	ID_CATEGORY	reference na podkategorii dat (teplota, tlak, ping, echo)
-	NAME	název
-	DESCRIPTION	popis

Tabulka A.28: SUBCATEGORY

CATEGORY		
FK	**Atribut**	**Popis**
PK	ID_CATEGORY	primární klíč tabulky
-	NAME	název
-	DESCRIPTION	popis

Tabulka A.29: CATEGORY

COMMAND		
FK	**Atribut**	**Popis**
PK	ID COMMAND	primární klíč tabulky
FK	ID_SUBCATEGORY	kategorie příkazu
-	COMMAND	vlastní příkaz
-	NAME	název
-	DESCRIPTION	popis
-	RETURN_VALUE	návratová hodnota příkazu

Tabulka A.30: COMMAND

NODE_COMMAND		
FK	**Atribut**	**Popis**
PK	ID NODE_COMMAND	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_COMMAND	reference na příkaz, který může poslán uzlu

Tabulka A.31: NODE_COMMAND

NODE_MESSAGE		
FK	**Atribut**	**Popis**
PK	ID_NODE_MESSAGE	primární klíč tabulky
FK	ID_NODE	reference na uzel, který zprávu odeslal
FK	ID_SUBCATEGORY	kategorie příchozí zprávy
FK	ID_NODEGROUP	reference na skupinu, která zprávu odeslala
-	MESSAGE	obsah zprávy
-	DESCRIPTION	popis

Tabulka A.32: NODE_MESSAGE

AREA_POINT		
FK	**Atribut**	**Popis**
PK	ID_AREA_POINT	primární klíč tabulky
FK	ID_NODE	reference na uzel
FK	ID_TYPE_COORDINATION	reference na informace o typu použitých souřadnic (GPS, zeměpisné, atd.)
-	GPS	GPS souřadnice
-	X	x souřadnice
-	Y	y souřadnice
-	Z	z souřadnice
-	GEOGRAPHIC	zeměpisné souřadnice
-	START_POINT_XYZ	výchozí bod pro relativní (kartezké) souřadnice

Tabulka A.33: AREA_POINT

USER		
FK	**Atribut**	**Popis**
PK	ID_ USER	primární klíč tabulky
-	USERNAME	přihlašovací jméno
-	PASSWORD	heslo pro přístup uživatele
-	NAME	jméno
-	SURNAME	příjmení
-	EMAIL	emailová adresa
-	PHONE	telefon
-	STREET	ulice
-	POSTCODE	směrovací číslo
-	CITY	město

Tabulka A.34: USER

AUTHORITIES		
FK	**Atribut**	**Popis**
PK	ID_ AUTHORITIES	primární klíč tabulky
FK	USERNAME	reference na uživatele
-	AUTHORITY	název role
-	NOTE	poznámka k dané roli

Tabulka A.35: AUTHORITIES

USER_LOG		
FK	**Atribut**	**Popis**
PK	ID_USER_LOG	primární klíč tabulky
FK	ID_USER	reference na uživatele
-	LOGIN_DATE	datum přihlášení
-	LOGIN_TIME	čas přihlášení
-	LOGOUT_DATE	datum odhlášení
-	LOGOUT_TIME	čas odhlášení
-	IP_ADDRESS	IP adresa, ze které byl uživatel přihlášen

Tabulka A.36: USER_LOG

BACKUP		
FK	**Atribut**	**Popis**
PK	ID_ BACKUP	primární klíč tabulky
FK	ID_USER	reference na uživatele, který provedl zálohu
-	BACKUP_NAME	název zálohy (stanovený formát)
-	BACKUP_DATE	datum zálohy
-	DESCRIPTION	popis

Tabulka A.37: BACKUP

CROSSBOW_PACKET		
FK	**Atribut**	**Popis**
PK	ID_ CROSSBOW_PACKET	primární klíč tabulky
-	AMTYPE	amtype
-	NODE_ID	identifikátor senzoru
-	PARENT	identifikátor rodiče
-	GROUP_ID	identifikátor skupiny
-	SOCKET_ID	identifikátor soketu
-	BOARD_ID	identifikátor desky
-	PACKET_ID	identifikátor paketu
-	VOLTAGE	stav baterie
-	TEMP	teplota v okolí senzoru
-	LIGHT	osvětlení v okolí senzoru
-	MIC	hladina zvuku v okolí senzoru
-	ACCEL_X	poloha X
-	ACCEL_Y	poloha Y
-	MAG_X	poloha X
-	MAG_Y	poloha Y
-	RSSI	síla signálu senzoru
-	TIME_STAMP	časová značka paketu

Tabulka A.38: CROSSBOW_PACKET

B POUŽITÉ JAR A SWC KNIHOVNY

1. JAR knihovny

    - antlr-3.2.jar
    - antlr-runtime-3.2.jar
    - aopalliance-1.0.jar
    - asm-3.2.jar
    - aspectjrt-1.6.5.jar
    - aspectjweaver-1.6.5.jar
    - backport-util-concurrent.jar
    - cfgatewayadapter.jar
    - cglib-2.2.jar
    - commons-codec-1.3.jar
    - commons-collections-3.2.jar
    - commons-dbcp-1.2.2.jar
    - commons-httpclient-3.0.1.jar
    - commons-logging-1.1.1.jar
    - commons-pool-1.5.4.jar
    - dom4j-1.6.1.jar
    - ehcache-1.6.2.jar
    - flex-messaging-common.jar
    - flex-messaging-core.jar
    - flex-messaging-opt.jar
    - flex-messaging-proxy.jar
    - flex-messaging-remoting.jar
    - flex-rds-server.jar
    - h2-1.0.71.jar
    - hessian-3.0.1.jar
    - hibernate-annotations.jar
    - hibernate-commons-annotations.jar
    - hibernate-core.jar

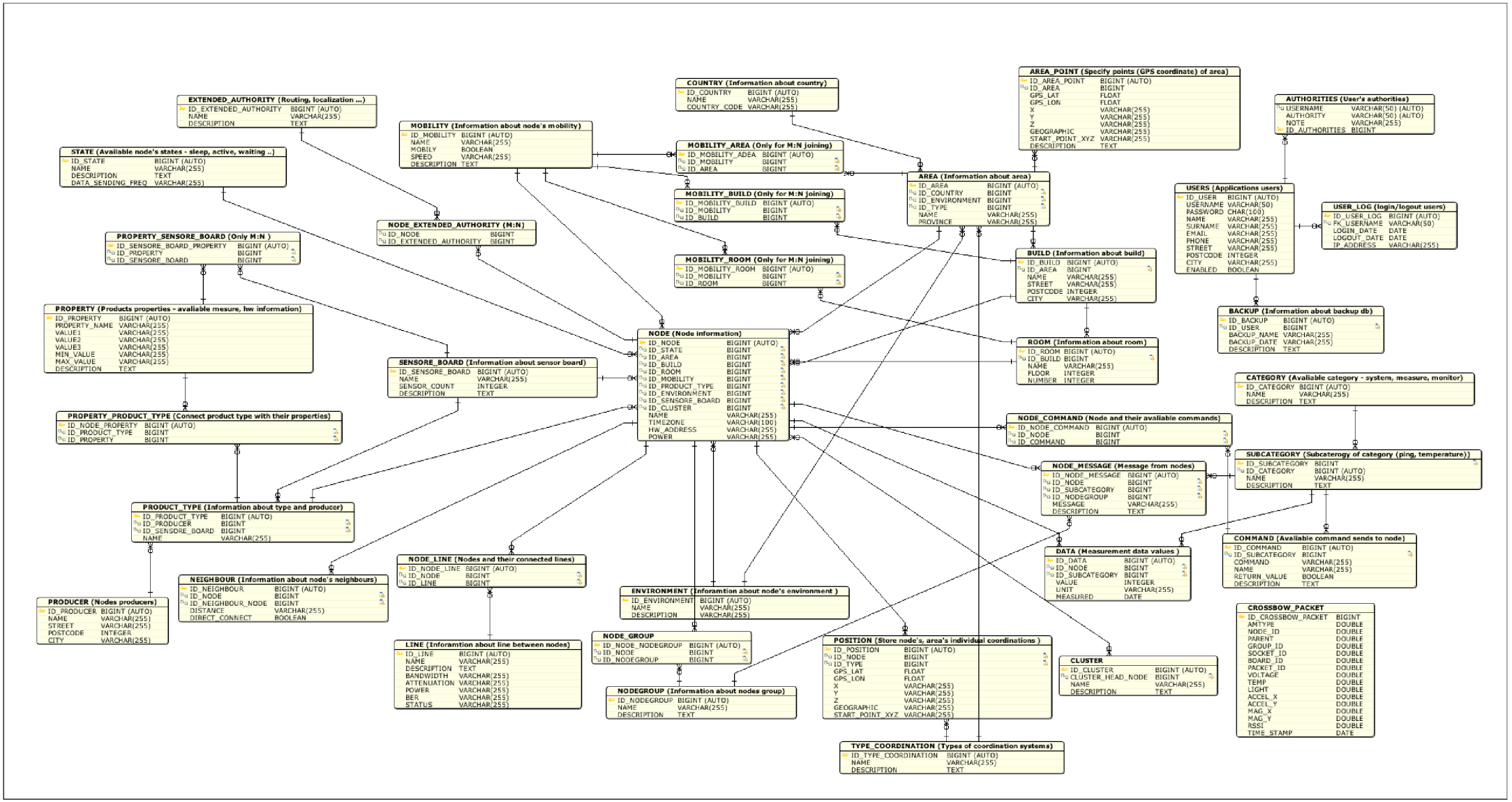
- jackson-core-asl-1.4.0.jar
- jackson-mapper-asl-1.4.0.jar
- javassist-3.9.0.GA.jar
- jstl-1.1.2.jar
- jta-1.1.jar
- junit-4.4.jar
- log4j-1.2.15.jar
- mysql-connector-java-5.1.10-bin.jar
- org.springframework.aop-3.0.0.RELEASE.jar
- org.springframework.asm-3.0.0.RELEASE.jar
- org.springframework.aspects-3.0.0.RELEASE.jar
- org.springframework.beans-3.0.0.RELEASE.jar
- org.springframework.context-3.0.0.RELEASE.jar
- org.springframework.context.support-3.0.0.RELEASE.jar
- org.springframework.core-3.0.0.RELEASE.jar
- org.springframework.expression-3.0.0.RELEASE.jar
- org.springframework.flex-1.5.0.CI-343.jar
- org.springframework.instrument-3.0.0.RELEASE.jar
- org.springframework.instrument.tomcat-3.0.0.RELEASE.jar
- org.springframework.jdbc-3.0.0.RELEASE.jar
- org.springframework.jms-3.0.0.RELEASE.jar
- org.springframework.orm-3.0.0.RELEASE.jar
- org.springframework.oxm-3.0.0.RELEASE.jar
- org.springframework.spring-library-3.0.0.RELEASE.libd
- org.springframework.test-3.0.0.RELEASE.jar
- org.springframework.transaction-3.0.0.RELEASE.jar
- org.springframework.web-3.0.0.RELEASE.jar
- org.springframework.web.portlet-3.0.0.RELEASE.jar
- org.springframework.web.servlet-3.0.0.RELEASE.jar
- org.springframework.web.struts-3.0.0.RELEASE.jar
- persistence.jar

- slf4j-api-1.5.8.jar
- slf4j-log4j12-1.5.8.jar
- spring-security-acl-2.0.5.RELEASE-sources.jar
- spring-security-acl-2.0.5.RELEASE.jar
- spring-security-cas-client-2.0.5.RELEASE-sources.jar
- spring-security-cas-client-2.0.5.RELEASE.jar
- spring-security-core-2.0.5.RELEASE-sources.jar
- spring-security-core-2.0.5.RELEASE.jar
- spring-security-core-tiger-2.0.5.RELEASE-sources.jar
- spring-security-core-tiger-2.0.5.RELEASE.jar
- spring-security-ntlm-2.0.5.RELEASE-sources.jar
- spring-security-ntlm-2.0.5.RELEASE.jar
- spring-security-openid-2.0.5.RELEASE-sources.jar
- spring-security-openid-2.0.5.RELEASE.jar
- spring-security-portlet-2.0.5.RELEASE-sources.jar
- spring-security-portlet-2.0.5.RELEASE.jar
- spring-security-samples-contacts-2.0.5.RELEASE-sources.jar
- spring-security-samples-contacts-2.0.5.RELEASE.war
- spring-security-samples-tutorial-2.0.5.RELEASE-sources.jar
- spring-security-samples-tutorial-2.0.5.RELEASE.war
- spring-security-taglibs-2.0.5.RELEASE-sources.jar
- spring-security-taglibs-2.0.5.RELEASE.jar
- standard-1.1.2.jar
- xalan.jar

2. SWC knihovny

- as3crypto.swc
- map_flex_1_18.swc
- AlivePDF.swc

C ER DIAGRAM



B. 1: Diagram struktury databáze