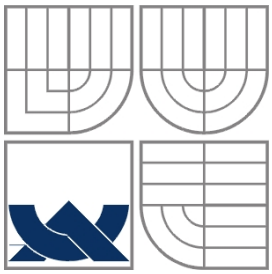
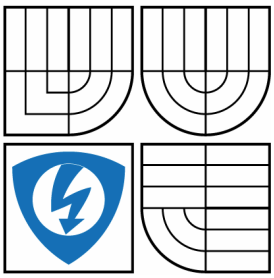


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INFORMAČNÍ SYSTÉM PRO PODPORU NÁVRHU KAMEROVÝCH SYSTÉMŮ

INFORMATION SYSTEM FOR SUPORT OF DESIGN OF CAMERA SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL HRONEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ MACHO, Ph.D.

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Pavel Hronek	***ID:***	70360
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Informační systém pro podporu návrhu kamerových systémů

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s databázovým systémem MySQL, jazyky PHP 5, Java a nástrojem pro vývoj webových aplikací Ajax.
2. Seznamte se s požadavky na informační systém pro podporu návrhu a konfiguraci kamerových systémů propojených bezdrátovou sítí.
3. Navrhněte databázový systém pro podporu návrhu kamerových systémů. Nakreslete příslušné ER diagramy. Uvažujte životní cykly.
4. Databázový systém implementujte a odlaďte.

DOPORUČENÁ LITERATURA:

[1] MASLAKOWSKI, Mark. Naučte se MySQL za 21 dní.1. vyd. Praha: Computer Press, 2001. ISBN 80-7226-448-8.
[2] Šimůnek Milan. SQL kompletní kapesní průvodce. 1. dotisk. Praha: Grada,1999.247 s.ISBN80-7169-692-7.

Termín zadání:	7.2.2011	***Termín odevzdání:***	23.5.2011

Vedoucí práce: Ing. Tomáš Macho, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací informačního systému pro firmu Patrokolos s.r.o., zabývající se bezdrátovými spoji pro kamerové a zabezpečovací systémy.

Informační systém umožňuje jeho uživatelům evidovat údaje o zákaznících, vložených do systému a přiřazovat jim nové zakázky, které si mohou samy navrhnout za pomoci integrovaného grafického rozhraní. Dále umožňuje pracovat s informacemi o vyrobených jednotkách. Na základě navrhnuté topologie zakázky umožní implementovaný systém sestavení konfiguračních skriptů pro OS Mikrotik, které pak do jednotky nahraje.

Informační systém je implementován v jazyce PHP 5 ve spolupráci s databázovým systémem MySQL 5.1.

Klíčová slova

Informační systém, MySQL, PHP, Ajax, JavaScript, návrh databáze, analýza IS, OS Mikrotik

Abstract

This thesis describes the design and implementation of information system for company Patrokolos s.r.o., dealing with wireless links for CCTV and security systems.

Information system enables its users to record data about custommers, entered into the system and assign them a new contract, they may suggest using an integrated graphical interface. It allows to work with information of produced units. Based on the topology of the proposed contract enabled the implemented system to build configuration skript for Mikrotik OS, which then loaded into the unit.

The information system is implemented in PHP 5 in cooperation with the MySQL database system 5.1.

Keywords

Information system, MySQL, PHP, Ajax, JavaScript, database design, analysis of IS, Mikrotik

Bibliografická citace:

HRONEK, P. *Informační systém pro podporu návrhu kamerových systémů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 64s. Vedoucí diplomové práce Ing. Tomáš Macho, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Informační systém pro podporu návrhu kamerových systémů 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 části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2011** ……
podpis autora

Poděkování

Děkuji vedoucím diplomové práce Ing. Tomáši Machovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **23. května 2011** ……
podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Tato diplomová práce se zabývá návrhem a realizací informačního systému pro podporu návrhu výrobků Camibox firmy Patrokolos s.r.o.. Camibox je stavebnicové řešení systému bezdrátového přenosu dat z IP CCTV kamer v bezlicenčním pásmu pro venkovní použití.

V současné době většinu práce z důvodu detailní znalosti technologie odvádí jednatel firmy. Při objednání zakázky pošle obchodní zástupce jednateli schématický náčrt zakázky na mapovém podkladu. Jednatel náčrt zkontroluje a překreslí ho do počítače, kde musí již specifikovat i typy použitých jednotek, jejich sériová čísla, spoje mezi těmito jednotkami aj. Poté zadá zaměstnanci ve výrobě kompletaci jednotek, potřebných k zakázce. Po vyrobení jednotek si jednatel nejdříve zkontroluje jejich funkčnost. Poté musí jednotku po jednotce manuálně nakonfigurovat, většinou pomocí grafického rozhraní WinBox OS Mikrotik. Po ukončení konfigurace všechny jednotky zapne a otestuje propustnost celé trasy. Poté jednotky vrací zpátky do výroby, kde jsou zabaleny do krabic a expedovány.

Realizovaný informační systém tento proces velmi zjednodušuje. Umožňuje obchodním zástupcům grafický návrh kompletního řešení bezdrátové sítě pro přenos obrazu pomocí jednotek Camibox. V porovnání se stávající situací tak vypadává zdlouhavé manuální překreslování zakázek. Systém umožňuje technikům vkládat do systému vyrobené jednotky, které pak může jednatel snadno přiřadit k jakékoliv zakázce. Jednateli také odpadá práce s konfigurováním jednotek. Tento informační systém totiž podle navrhnuté zakázky vygeneruje skripty, které nakonfigurují jednotky pro jednotlivé zakázky. Tuto činnost zde přebírá technik, který zvládne nakonfigurovat jednotky bez jakékoliv znalosti programování nebo operačního systému Mikrotik. Realizovaný informační systém umožňuje všem definovaným rolím náhled na aktuální stavy zakázek. Odpadá tak časově náročná emailová nebo telefonická komunikace. Tento informační systém usnadní práci celé struktuře firmy včetně jejího vedení.

První část práce se zabývá technologiemi, které zabezpečují funkčnost celého systému. Především se jedná o skriptovací jazyk PHP, databázový systém MySQL a skriptovací technologie použité na straně klienta – JavaScript a AJAX. Druhá část práce se věnuje seznámením se s požadavky firmy na tento informační systém. Ve třetí části je návrh a následný rozbor databázové části včetně popisu ER diagramů a hlavních životních cyklů. V poslední části je popis realizovaných procesů informačního systému a jejich následná analýza včetně rozboru pomocí DFD diagramů.

2. POUŽITÉ TECHNOLOGIE

Obchodní zástupci, kteří distribuují produkt Camibox, operují ve většině států Evropské unie. Z tohoto důvodu jsem zvolil informační systém, který bude komunikovat přes internet.

2.1 ZÁKLADNÍ POPIS KLIENTSKÝCH APLIKACÍ A INTERNETOVÉHO PROTOKOLU HTTP

Zvolil jsem IS implementovaný do webového rozhraní. Důvodem je jeho síla, která se týká kompatibility s různými OS a možnostmi přístupu do IS téměř na kterémkoli PC bez jakéhokoliv zásahu správce sítě.

2.1.1 Klientské aplikace

Aplikace, komunikující prostřednictvím internetu nebo jiné počítačové sítě, se dají rozdělit na dvě skupiny, na tzv. tenkého a tlustého klienta.

Tlustý klient je aplikace, která z větší části běží na počítači uživatele. Bývá napsána v některém ze standardních programovacích jazyků (C++, C#, atp.) a je klasicky nainstalována na PC jako každý standardní program. Ze serveru přenáší jen velmi málo dat. Mezi jeho výhody patří lepší programovatelnost (více možností, lepší tvorba vzhledu), bezpečnost a funkčnost i bez přístupu k síti. Na druhou stranu přináší vyšší nároky na HW a menší multiplatformost.

Další možností je použití tzv. tenkého klienta. V našem případě budeme tenkým klientem nazývat webovou aplikaci, což je aplikace, poskytovaná klientům z webového serveru pomocí počítačové sítě. Tenký klient zpracovává většinu datových a výpočetních úkonů na straně serveru, ke klientovi pak již posílá výslednou „obrazovku“ nejčastěji pomocí webového prohlížeče. Mezi výhody tenkých aplikací patří malá závislost na platformě, možnost využívat program bez instalace, snadná správa a záloha používaných dat. Naopak k nevýhodám můžeme přiřadit vyšší pořizovací cenu (s ohledem na problémy, které některé jazyky přinášejí) a vyšší nárok na datovou propustnost klient-server.

2.1.2 Protokol HTTP

HTTP je internetový protokol, původně určený pro výměnu hypertextových dokumentů ve formátu HTML. Nyní je však používán i pro přenos dalších informací, například může přenášet soubory, využívá XML a zpřístupňuje další protokoly. Jedná se o bezstavový protokol, takže neudržuje žádné informace o svých klientech. Transakce začíná tím, že přijme dotaz od klienta a končí tehdy, když na něj odpoví. Veškeré upřesňující informace musí mít uloženy klient. První řádek dotazu obsahuje cestu k

požadovanému dokumentu včetně použité metody, na druhém řádku začínají hlavičky. Pomocí nich může klient upřesnit dotaz nebo ho doplnit nepovinnými informacemi (např. jaký prohlížeč klient používá, jakou podporuje znakovou sadu, které informace je ochoten přijmout apod.). Odpověď začíná verzí HTTP a za ní je stavový kód, který nese informaci o tom, zdali byl dotaz úspěšný nebo ne. Na dalších řádcích, podobně jako u dotazu, mohou být umístěny hlavičky. Ty nesou doplňující informace ke klientovi, jako např. datum, jazyk, možnosti ukládání dočasných souborů apod.

Metod, které může klient použít je několik. Mezi základní patří metoda GET, kdy se veškerá data předávají jako součást URL. Používá se vždy, když klient přistupuje k dokumentu a neodeslal před tím formulář pomocí metody POST. Při použití metody POST se data odeslaná formulářem předávají v těle dotazu (za hlavičkami). Mezi metody, které se moc nepoužívají, patří např. metoda HEAD, která nevrací obsah stránky, ale pouze její hlavičku (používá se například pro zjištění poslední změny stránky) a metoda OPTIONS (používá se při dotazu na možnosti serveru). [8]

2.2 TECHNOLOGIE POUŽITÉ NA STRANĚ SERVERU

Server musí být schopen přijmout požadavek klienta, zpracovat ho a poslat mu zpět data, která po něm žádá. Základem je webový server, který naslouchá nejčastěji na portu 80 (v případě HTTP) a vyřizuje požadavky klienta. Dalšími základními službami, poskytovanými serverem, je uchovávání dat v databázi a aktivní skriptovací technologie, která umožňuje zpracovat požadavek klienta s využitím dat z databáze, provést složitější výpočty a výslednou statickou podobu dokumentu poslat pomocí webového serveru zpátky klientovi.

2.2.1 Webový server

Webový server je zodpovědný za vyřizování požadavků, které na server poslali klienti. Komunikace mezi klientem a serverem probíhá nejčastěji pomocí protokolu HTTP. Server vrací klientům buď předem připravená data (např. HTML soubory) nebo data, která se vytváří ze specifického dotazu klienta pomocí dat z databáze a některého ze skriptovacích jazyků. Mezi tři nejpoužívanější webové servery patří Apache HTTP Server, Internet Information Services a Sun Java System Web Server. [3]

Apache HTTP Server

Apache patří k nejrozšířenějším webovým serverům. Je šířen jako software s otevřenou licencí a jeho jádro je multiplatformní. Jeho vývoj začal v roce 1993 na Illinoiské Univerzitě v USA. V současné době ho vytváří a udržuje sdružení apache.org. Jeho instalace je poměrně jednoduchá, takže ji zvládne i méně zkušený uživatel. Existuje ve verzích pro většinu největších operačních systémů (Windows, Linx, Mac

OS apod.). Mezi jeho výhody patří stabilita, spolehlivost a flexibilnost. Proto na tomto HTTP serveru běží většina webových stránek celého světa. [3]

2.2.2 Databázový systém

Databáze je množina souvisejících dat, které jsou určitým způsobem setříděny a uloženy na nějakém paměťovém médiu. Vznikla jako náhrada za kartotéky, kde bylo velké množství informací a ve kterých se nedalo jednoduše hledat. *Základní myšlenkou databázové technologie je oddělení uživatelů systému od fyzických souborů s daty*[7].

Systémů, které se zabývají ukládáním dat do databází, je v současné době velké množství. Mezi nejznámější patří MySQL, jehož největší předností je jeho šíření zdarma. V komerční sféře patří k nejznámějším např. Oracle Database společnosti Oracle Corporation, SQL Server firmy Microsoft nebo Microsoft Office Access také firmy Microsoft. [7]

MySQL

MySQL je výkonný databázový relační systém s architekturou klient/server. Je dostatečně stabilní, bezpečný a má výborný poměr cena/výkon (pro jeho šíření zdarma a také pro jeho HW nenáročnost). Jádro databázového systému je napsáno v jazyce C a C++.

V roce 1995 byla vydaná první verze, o tři roky později byla vydaná verze pro MS Windows. S každou vydanou verzí se přidávaly funkce, které zlepšovaly tento databázový systém. Např. ve verzi 3.23 přibyla funkce replikace, ve verzi 4.1 poddotazy, ve verzi 5.0 pohledy atd. Od roku 2005 je k dispozici verze 5, nyní konkrétně 5.5, která umožňuje např. plánování úloh, zapisování historie operací do tabulek apod.

Mezi jeho výhody patří nezávislost na platformě a OS (aplikace klienta i serveru mohou běžet např. pod MS Windows, Apple, Linux), snadnost instalace, ODBC rozhraní umožňující v prostředí MS Windows spolupráci se spoustou programovacích jazyků, pro programování klientů se může použít široká škála jazyků (např. C, C++, Java, Perl, PHP), kompatibilita s SQL (jazyk, který se používá pro dotazování a aktualizaci dat v DB), triggery (příkazy SQL, které server spouští automaticky při určitých operacích) apod.

Mezi jeho nevýhody patří mimo jiné nemožnost řešení online záloh (tabulka se při záloze musí zamknout), nemožnost definování vlastních datových typů, omezení možnosti použití pro aplikace v reálném čase atd.

Databázový systém MySQL dnes používají desítky milionů serverů a konkurují tak komerčním databázovým systémům. [3] [4] [7]

Typy tabulek v MySQL

Zvláštností databáze MySQL je to, že pokud vytváříme novou tabulku, můžeme specifikovat jakého má být typu. V MySQL jsou tři nejdůležitější typy – MyISAM InnoDB a HEAP.

MyISAM je základním typem tabulek. Tabulky jsou stabilní a jednoduché na správu. Rozlišujeme dvě varianty těchto tabulek. MyISAM statický, který se používá v případě, že všechny sloupce tabulky mají pevně stanovenou velikost (přístup k této tabulce je velmi rychlý, ale nevýhodou je rezervace paměťového místa pro jednotlivé záznamy) a MyISAM dynamický, který se používá jen tehdy, pokud je v tabulce některý z datových typů VARCHAR, TEXT či BLOB a ty nemají předem definovanou velikost (přístup je pomalejší než u statické varianty, ale má menší nároky na paměť). O výběr se starat nemusíme, ten provede MySQL automaticky.

InnoDB je modernější alternativa MyISAM. Přináší nové funkce jako např. transakce, které umožňují několik logicky souvisejících příkazů, které se provedou všechny nebo ani jeden a pokud při provádění některého příkazu nastane chyba, budou výsledky všech příkazů odstraněny nebo zamykání na úrovni záznamů, které znamená to, že při provádění transakce nemusíme blokovat přístup k tabulce dalším uživatelům, ale uzamkneme jen jednotlivé záznamy. Mezi její nevýhody patří téměř dvojnásobné požadavky na paměť oproti tabulce MyISAM, složitější a pomalejší zjištění počtu záznamů v databázi pomocí příkazu COUNT atd.

HEAP je speciální typ tabulek, který data ukládá do RAM, nikoli na disk. Jsou značně omezené, např. nesmíme používat datový typ TEXT nebo BLOB. Jsou určeny pro rychlé zpracování malého množství dat. [3]

2.2.3 Skriptovací technologie

Skriptovací jazyky jsou skupinou jazyků, které se používají pro rychlé a jednoduché psaní programů. V těchto jazycích se většinou nemusíme starat o deklaraci proměnných (ve většině jazyků neexistuje typová kontrola, je tedy možné do čísla přiřadit řetězec), nemusíme se ani starat o to, zda nově inicializovaná proměnná bude mít skutečně nulovou hodnotu apod. Využívají také speciální typy asociativních polí (kde se jako index používá řetězec). Ke skriptovacím jazykům se řadí např. Perl, Python, PHP nebo Tcl.

PHP

Název tohoto skriptovacího jazyka, šířeného jako Open Source, je odvozen od slov Hypertext Preprocessor. Původně byl název odvozen ze slov Personal Home Page, ale byl později změněn z důvodu nedostatečného vystihnutí jazyka. Velká část jeho syntaxe pochází z jazyků C, Perl a Java. Jedná se tedy o skriptovací jazyk, to znamená, že pro spouštění skriptů je nutný externí program (tzv. interpreter), který jej vykoná.

Nejčastěji se tento jazyk používá na straně serveru při tvorbě dynamických webových aplikací. Mezi jeho výhody patří nezávislost na platformě či operačním systému a hlavně to, že je zdarma. Tento jazyk je masově rozšířený po celém světě a využívají ho desítky milionů serverů.

Vznik jazyka PHP sahá až do roku 1994, kdy jej jeho autor, Rasmus Lerdorf, použil na osobních stránkách jako určitou náhradu jazyka Perl. V roce 1995 uvolnil pro veřejnost verzi 2, aby urychlil vývoj a hledání chyb.
Zajímavostí v této verzi je implementace cyklu while. Ručně napsaná lexikální čtečka procházela skript a pokud narazila na slovo while, zapsala si jeho pozici v souboru do paměti. Na konci cyklu vyhledala čtečka uloženou pozici a celý cyklus byl znovu vykonán. O dva roky později došlo k uvolnění verze 3, ve které bylo přepsáno celé jádro programu. V roce 2000 vyšla verze 4. V poslední době se však začaly množit požadavky na objektově orientovaný přístup. S verzí 5 přišlo rozšíření o mnoho funkcí, umožňujících objektově orientované programování. Nyní je ve vývoji verze 5.3 společně s verzí 6, která přináší např. lepší podporu formátu Unicode.
Největšími přednostmi jazyka PHP je jeho snadná práce s databázovými systémy (např. MySQL, PostgreSQL, Oracle) a dobrá podpora protokolů (např. HTML a POP3), což mu umožňuje snadné vytváření aplikací.

Základním určením PHP je skriptování na straně serveru, které vytvoří dynamické webové stránky. Na podobném principu pracují jazyky jako ASP.NET či JavaServer Pages. Nejprve byl jazyk PHP určen pouze pro jednoduché skripty, ale s příchodem PHP 5 a rozvojem objektového programování se PHP spolu s databázovým systémem stará o chod celých webových systémů. [4]

2.3 TECHNOLOGIE POUŽITÉ NA STRANĚ KLIENTA

Základním programem na straně klienta je webový prohlížeč. Ten komunikuje se serverovou stranou a zpracovává přijatý (X)HTML kód, který pomocí CSS stylů zformátuje a zobrazí výslednou obrazovku uživateli. Mezi nejrozšířenější webové prohlížeče patří Internet Explorer (nyní ve verzi 8.0), Mozilla Firefox (nyní ve verzi 3.6.3) a Google Chrome (nyní ve verzi 11.0). Klientský počítač také využívá platformu JavaScript, která se stará o obsloužení akcí, které uživatel právě v danou chvíli udělá (např. stisk tlačítka, přejetí kurzoru nad obrázkem apod.). Další je poměrně nová technologie Ajax, vycházející z jazyku Java. Ta se stará také o obsloužení aktuálních akcí uživatele s tím rozdílem, že komunikuje se serverovou stranou a podle aktuální situace dynamicky vykresluje požadované informace. Obě tyto technologie využívají pro čtení nebo manipulaci s objekty DOM model.

2.3.1 XHTML

Značkovací jazyk XHTML byl původně vyvinut jako nástupce jazyku HTML, jehož vývoj byl ukončen v roce 2007 (avšak byla založena pracovní skupina, která má za úkol vyvinout novou verzi HTML). Jeho základ tvoří značkovací jazyk HTML a tzv. rozšiřitelný značkovací jazyk XML. Podpora HTML a XHTML je v současných prohlížečích na zhruba stejné úrovni. XHTML má tři verze – Strict (který se používá pro strukturovaný dokument bez formátovacích značek, předpokládá použití CSS), Transitional (používá se pro starší prohlížeče, které nepodporují CSS) a Frameset (umožňuje používat zastaralé značky pro rámce).

Předpona X znamená eXtensible, což je v překladu rozšiřitelný (pochází od XML). Ve skutečnosti ale jde o mnohem striktnější jazyk než je HTML. Jako příklad lze uvést ukončování nepárového tagu lomítkem, všechny tagy a atributy musí být psány malým písmenem (XHTML je oproti HTML citlivé na písmena) nebo uzavírání atributů tagů do uvozovek. [11]

2.3.2 CSS

CSS (neboli kaskádové styly) vytvářejí spolu s XHTML základ pro tvorbu výsledného vzhledu webové stránky. Hlavním účelem je oddělit samotný obsah stránky od jeho struktury. Kaskádové styly tvoří soubor pravidel, které v základu obsahují tzv. selektor a následnou deklarací pravidel. Selektor může určovat všechny typy vybraného tagu daného dokumentu, nebo s pomocí třídy a identifikátoru jednoznačně určit, o který tag se jedná. Výhod v používání CSS stylů je mnoho. Můžeme jednoznačně a přesně formátovat jakýkoliv text v dokumentu (např. barvu textu, jeho šířku, odsazení apod.), můžeme také např. vytvořit více stylů a ty pak náhodně nebo cíleně měnit (často se používá jiný styl např. na tisk dokumentu) a v neposlední řadě můžeme také poměrně snadno vytvářet vzhled webové prezentace. Zatím byly vydány tři verze kaskádových stylů (1, 2, 2.1) a nyní se pracuje na vývoji verze 3. [6]

2.3.1 DOM

DOM (Document Object Model) je W3C specifikace pro na platformě a jazyku nezávislý způsob přístupu k obsahu a struktuře dokumentu. [10] Je to vlastně způsob, kterým může Ajax, JavaScript nebo jiný programovací jazyk přistupovat či modifikovat hodnoty nebo strukturu HTML dokumentu. DOM představuje objektový model dokumentu. Definuje objekty pro reprezentaci a modifikaci dokumentů, chování atributů a vztahů mezi nimi. Na DOM lze nahlížet jako na stromovou strukturu dokumentu, která však takto nemusí být ve skutečnosti implementována. [11]

2.3.2 JavaScript

JavaScript je interpretovaný, objektově orientovaný programovací jazyk. Používá se zpravidla u webových prezentací pro zajištění dynamické reakce na uživatelovu akci (např. odeslaní formuláře, najetí nad obrázek apod.). Byl standardizován v roce 1997 a patří do rodiny jazyků C, C++ a Java. Používá se ve většině případů na klientské straně, z toho důvodu má ale také různá bezpečnostní omezení (např. nemožnost pracovat se soubory). Mezi jeho výhody patří multiplatformnost, díky které je rozšířený a používaný po celém světě. [9]

2.3.3 AJAX

Začátek používání technologie Ajax datujeme kolem roku 1998. Tehdy začaly tuto technologii používat některé aplikace, jako např. Microsoft Outlook Web Access. Ovšem až v roce 2005 byl vydán článek Jessem Jamesem Garrettem, který se o této technologii poprvé zmiňuje. Ajax znamená Asynchronous JavaScript and XML, jedná se tedy o více technologií v jedné. Ajax zpracovává aktuální dotaz klienta bez nutnosti přepsat stránku. Pomocí JavaScriptu komunikuje na pozadí se serverem a po přijmutí potřebných informací zobrazí požadovanou odpověď. Komunikace se serverem probíhá pomocí formátování dat na XML. Díky XHTML a CSS se s pomocí Ajaxu překresluje jen ta část stránky, kde chceme zobrazit odpověď. Ajax se využívá v širokém spektru aplikací, od jednoduchých nápověd (např. Google používá Ajax při napovídání hledaného slova, kdy při napsání pár písmen nabízí nejvyhledávanější výrazy, začínající těmito písmeny), až po složité on-line grafické editory, tabulkové procesory aj. Pro vývoj aplikací v jazyku Ajax můžeme použít klasické programy pro vývoj PHP nebo Javy (Macromedia Dreamweaver, Eclipse apod.). [5]

2.4 POPIS PRODUKTU CAMIBOX

Camibox je stavebnicové řešení systému bezdrátového přenosu dat z IP CCTV kamer, nebo jiných zabezpečovacích zařízení v bezlicenčním pásmu. Základ toho řešení stojí na produktu RouterBOARD, který se používá jako základní HW u mnoha ISP, používajících pro poslední míli WiFi pásma 2,4 a 5,5 Ghz. RouteBOARD je produktem Lotyšské společnosti Mikrotik a používá stejnojmenný operační systém.

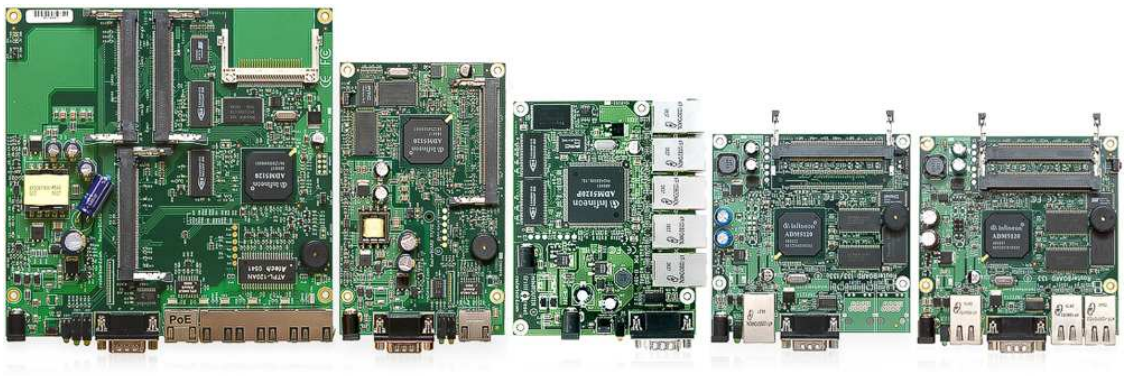
Jednotky Camibox jsou navrženy pro venkovní použití a používají PoE napájení (jednotka je napájena dvěma volnými páry ethernetového kabelu). Součástí jednotky je 18dB anténa, při větších vzdálenostech je možno pomocí externího RSMA konektoru připojit anténu s větším ziskem a menším vyzařovacím úhlem. K tomuto systému může být připojena nejen téměř jakákoliv IP kamera, ale systém může přenášet jakákoliv data, např. hlášení z čidel, přenos zvuku atp. Jednotky mají akustickou signalizaci síly signálu. Díky tomu se instalace celého systému obejde bez jakékoliv výpočetní techniky. Na zadní straně je vyveden BNC konektor pro připojení sluchátek právě pro

případné dosměrování spojů. Systém může být jednoduše začleněn do stávající počítačové sítě. Stavebnicové řešení umožňuje případně starší síť jednoduše rozšířit. Datový tok, přenášený pomocí těchto jednotek, může být reálně 30-160 MB, za předpokladu splnění požadavků na vyzářený výkon v daném státě. Přenášená data jsou šifrována tak, aby nemohlo dojít k jejich neoprávněnému úniku. [12]

Obrázek 2.1: Ilustrační obrázek produktu Camibox [12]

2.4.1 RouterBOARD

RouterBOARD (dále RB) je HW směrovač, který je primárně určen pro přenos dat v bezdrátových sítích. Vyrábí se v široké škále konfigurací, od jednoduchých malých, určených pro koncové odběratele, až po velké výkonné stroje, umožňující routování mezi jednotkami portů a datové toky v jednotkách GB. Základem RB je procesor (Atheros AR, Power PC, Mips), který bývá taktován od stovek MHz až po jednotky GHz (podle určení použití této desky). Součástí RB jsou LAN porty (1-10), mPCI sloty pro kartičky, umožňující bezdrátové vysílání (1-4), operační paměť (1-64MB) popř. sloty pro USB nebo CF kartu. Některé verze mají i slot pro SIM kartu, díky kterému je možné se k jednotce vzdáleně připojit. Výhodou této technologie je její cena, snadná rozšiřitelnost a možnosti konfigurace, která je dána OS Mikrotik. Základní konfigurace RB se provádí pomocí konzole a integrovaného portu RS 232, nebo pomocí ethernetového připojení.



Obrázek 2.2: Produkty RouterBOARD (RB 100 series) [www.routerboard.com]

2.4.2 OS Mikrotik

OS Mikrotik je založen na OS Linux. Na rozdíl od něj ale není šířen zdarma. Je nabízen v šesti licencích, které umožňují nasazení v různých odvětvích (domácnost, AP, hlavní směrovač apod.). Většinou se prodává jako předinstalovaný na produktu RouterBOARD (viz. kapitola 2.4.1), ale je možné ho zakoupit jako samotnou instalaci popř. na CF kartě a systém použít na běžném PC. OS je primárně určen pro síťové aplikace, proto jeho součástí nejsou žádné jiné rozšiřovací balíčky či moduly.

Systém umožňuje snadnou konfiguraci pomocí grafického rozhraní Winbox, určeného pro systém Windows. Možností konfigurace OS Mikrotik je nespočet. Od základního NATování pomocí masquarade, po routování včetně podpory protokolů OSPF, RIP a BGP, po prioritizaci síťového provozu pomocí QOS až po označování a třídění přicházejících nebo odcházejících paketů. Další výhodou je i to, že OS Mikrotik má definované API třídy pro různé programovací jazyky (C++, C#, PHP…). Po malé úpravě je tedy možné pomocí tohoto rozhraní komunikovat se vzdáleným systémem s OS Mikrotik.

OS Mikrotik disponuje vlastním skriptovacím jazykem, pomocí kterého je možné pracovat se základními proměnnými, cykly a podmínkami.

3. POŽADAVKY FIRMY PATROKOLOS

Firma Patrokolos, pro kterou je následující systém vyvíjen, byla založena roku 2002. Na začátku své existence se věnovala vývoji a výrobě antén a dalšího příslušenství pro WiFi. Nejprve to byly parabolické, všesměrové a Yagi antény pro pásmo 2,4 GHz, později bubnové a sektorové antény pro pásmo 5,5 GHz. Dále se firma specializovala na vývoj a prodej produktu Camibox pro bezdrátové propojení kamerových a zabezpečovacích systémů.

3.1 ZÁKLADNÍ POŽADAVKY NA IS

Základní funkcí IS je evidence dat o vyrobených jednotkách a jejich stavu. Technici, kteří se starají o výrobu, nejprve zkompletují výrobek a dají ho do krabice. S pomocí IS si poté vygenerují unikátní sériové číslo jednotky, které nalepí na krabici. Do IS také vloží informace o typu jednotky, nastavených přístupových jménech a heslech, počtu aktivních LAN a WLAN portů včetně jejich MAC adres a další upřesňující informace.

Další funkcí tohoto systému bude evidence zákazníků firmy a jejich obchodních zástupců. Ti budou moci do IS vkládat nové zákazníky a plánovat jim zakázky včetně rezervace vyrobených jednotek. Uživatel založí novou zakázku a pomocí grafického rozhraní si může kompletně naplánovat celou bezdrátovou síť, potřebnou pro kamerový systém. Základ stavebnicového systému tvoří jednotka Master, která je vždy v zakázce jen jedna a je na ni připojeno záznamové zařízení. Dále jsou zde jednotky client a slave. Obě slouží k připojení kamer s tím rozdílem, že jednotka slave umožňuje bezdrátové připojení další jednotky typu client nebo slave (jedná se o tzv. retranslační jednotku). Ke každé jednotce si uživatel definuje počet portů, potřebných pro připojení kamer a v případě jednotky slave i počet bezdrátových rozhraní. Ve výsledku bude mít kompletní schématický náčrt dané zakázky včetně výsledného počtu jednotek a počtu jejich portů.

Poslední funkcí tohoto systému bude kompletní konfigurace všech jednotek podle naplánované zakázky z předchozího bodu. IS podle informací o zakázce sestaví skripty pro konfiguraci RouterBOARDu tak, aby byly jednotky po zapnutí ihned funkční. Konfigurace pak bude jen otázkou pár minut a obsluha nemusí mít žádné znalosti OS Mikrotik, počítačových sítí nebo programování.

V systému by mělo existovat několik základních rolí (administrátor, výrobce, obchodní zástupce,…), které budou mít přesně definovány práva na jednotlivé funkce systému. V některých částech systému bude přístup k citlivým datům zákazníků, proto by kontroly přístupu k jednotlivým částem měly být dostatečně bezpečné. Při vkládání nových jednotek do systému musí být zkontrolováno, zda-li se zadaná

jednotka už v systému nenachází. Pro zadávání MAC adres LAN a WLAN portů to platí také (MAC adresa je unikátní). Grafické rozhraní, určené pro plánování nových zakázek by mělo umět on-line překreslovat požadovaná data, avšak pro začátek není kladen důraz na grafický vzhled. Konfigurace jednotek bude probíhat pomocí jejich připojení k serveru, umístěnému v místě výroby jednotek. Systém vyzve uživatele k připojení jednotky, kterou chce konfigurovat. Poté se na jednotku připojí a spustí na ní konfigurační skript pro připravenou zakázku.

4. NÁVRH DATABÁZOVÉHO SYSTÉMU

Pro IS jsem zvolil databázový systém MySQL z důvodu jeho velké rozšířenosti, snadné práce se skriptovacím jazykem PHP a jeho ceny (je zdarma). Databáze MySQL 5.1 nabízí možnost ukládání dat do tabulek typu MyISAM, HEAP nebo InnoDB. Vybral jsem ukládání dat do tabulek typu InnoDB z důvodu podpory transakcí, automatické kontroly omezení cizího klíče a automatického zotavení po havárii databáze. Databáze MySQL umožňuje kódování do mnoha typů znakových sad (utf8, cp1250, asci apod.). Zvolil jsem znakovou sadu utf8_general_ci z důvodu lepší podpory u cizojazyčných OS.

Nejprve jsem identifikoval všechny entity v systému a definoval jejich atributy a kandidáty na cizí klíče. Poté jsem pomocí programu MySQL Workbench vytvořil E-R diagram, který jsem rozdělil na několik, logicky souvisejících E-R diagramů. V příloze A je pak umístěn kompletní E-R diagram. Jako poslední jsem určil životní cykly navržených entit.

4.1 IDENTIFIKOVANÉ ENTITY

Entita je název pro objekt, který sdružuje data k sobě logicky náležející. V informačním systému je vždy před názvem entity prefix camibox_, a to z důvodu možného začlenění této databáze do nějaké již existující. Jména entit jsou zvolena podle dat, které obsahují. Podle požadavků na systém jsem identifikoval celkem 28 entit.

Název entity	Popis entity
camibox antennas	používané antény včetně jejich parametrů
camibox_bands	číselník frekvenčních pásem
camibox_cities	číselník měst včetně psč
camibox_config_countries	přiřazení frekvenčního pásma a povolených frekvencí k příslušnému státu
camibox_contracts	uskutečněné nebo plánované zakázky
camibox_countries	číselník států
camibox_custommers	informace o zákaznících
camibox_entity	číselník entit
camibox_maps	informace o mapách k jednotlivým zakázkám
camibox_modules	moduly v systému včetně jejich umístění
camibox_modules_groups	číselník skupin modulů
camibox_rights	vazební tabulka uživatelů a rolí
camibox_roles	číselník rolí
camibox_roles_modules	vazební tabulka rolí a modulů
camibox_scripts	skripty pro konfiguraci Camiboxu

Název entity	Popis entity
camibox_states	stavy jednotlivých entit
camibox_states_transition	přechody entit mezi jednotlivými stavy
camibox_units	informace o jednotkách Camibox
camibox_units_contracts	přiřazení jednotek k zakázkám
camibox_units_contracts_design	informace pro GUI
camibox_units_lans	detaily LAN portů vložených jednotek
camibox_units_types	číselník typů jednotek
camibox_units_types_master	číselník hlavních typů jednotek
camibox_units_wlans	detaily WLAN portů vložených jednotek
camibox_units_wlans_antennas	vazební tabulka WLAN portů a antén
camibox_units_wlans_connections	spojení WLAN portů různých jednotek
camibox_users	uživatelé systému
camibox_users_entity_roles	informace o vložených záznamech jednotlivých uživatelů

Tabulka 4.1: Seznam identifikovaných entit

4.2 E-R DIAGRAM ČÁST 1 – UŽIVATELÉ A PRÁVA

Tato část E-R diagramu se zabývá ukládáním dat o uživatelích, jejich rolích a právech na jednotlivé moduly. Všechny tabulky mají primární klíč s názvem atributu id, datovým typem INT s maximálním počtem číslic omezeným na deset a používají speciální funkci AUTO_INCREMENT, která každému nově vkládanému záznamu přiřadí hodnotu o jednu větší, než v předchozím záznamu. Všechny tabulky také obsahují atribut state, který určuje v jakém stavu se právě daný záznam nachází.

V tabulce camibox_users jsou uloženy jednotlivé záznamy o uživatelích IS. Především jde o jejich jméno, příjmení, email a telefon (atributy name, surname, email, mobil_phone). Pro textové řetězce v jednotlivých tabulkách je použit datový typ VARCHAR, délka textového řetězce je určena podle potřeby a přibližné velikosti vkládaného textu od 16 do 64 znaků. Hodnoty atributů login a password obsahují uživatelovy přihlašovací údaje. Hodnota atributu password je uložena zašifrovaná s pomocí algoritmu md5. Tato tabulka také vede záznamy o uživatelově posledním přístupu do systému (atributy last_login). Pro tyto atributy je použit datový typ DATETIME, což je kombinace data a času ve formátu rok-měsíc-den hodina:minuta:sekunda.

Pomocí cizího klíče id_contry (cizí klíč je vždy reprezentován názvem atributu s prefixem id_, používá datový typ INT a má stanoven maximální počet číslic na deset) a vazby N:1 je na tuto tabulku vázána tabulka camibox_countries, což je vlastně číselník států. To znamená, že ve stejném státě může být více uživatelů. Tabulka obsahuje

atribut country_name, jehož hodnota určuje celé jméno daného státu a atribut abbr, jehož hodnotou je určena mezinárodní zkratka daného státu.

Tabulka camibox_rights představuje vazební tabulku mezi uživateli IS a rolemi, které jsou v IS dostupné. Obsahuje cizí klíč id_user, kterým se odkazuje vazbou 1:N na tabulku camibox_users a cizí klíč id_role, kterým se odkazuje také vazbou 1:N na tabulku camibox_roles. To znamená, že jednomu uživateli může být přiřazeno více rolí a naopak stejné role mohou být přiřazeny různým uživatelům.

V tabulce camibox_roles je umístěn seznam všech rolí, které jsou v IS dostupné. Hodnota atributu role_name obsahuje název dané role.

Tabulka camibox_roles_modules představuje vazební tabulku mezi rolemi a moduly v IS. Obsahuje cizí klíč id_role, pomocí kterého je uskutečněna vazba 1:N s tabulkou camibox_roles a cizí klíč id_modul, pomocí kterého je realizována vazba 1:N s tabulkou camibox_modules. Tato tabulka umožňuje přiřadit k rolím příslušné moduly, na které má daná role právo.

V tabulce camibox_modules je seznam všech modulů (modul=skupina skriptů, pracujících se souvisejícími daty), dostupných v systému. Obsahuje atribut modul_name, který představuje unikátní jméno modulu, atribut modul_description, který představuje přesné umístění modulu v adresářové struktuře a atribut menu_view, v jehož hodnotě je uložena informace o zobrazení nebo skrytí daného modulu v menu (atributy, které používají číselné hodnoty mají datový typ INT, počet číslic je omezen podle potřeby na 1-10). Pomocí cizího klíče id_group se tato tabulka vazbou 1:N odkazuje na seznam názvů nadřazených modulů. To umožňuje přiřazení více modulů do související skupiny.

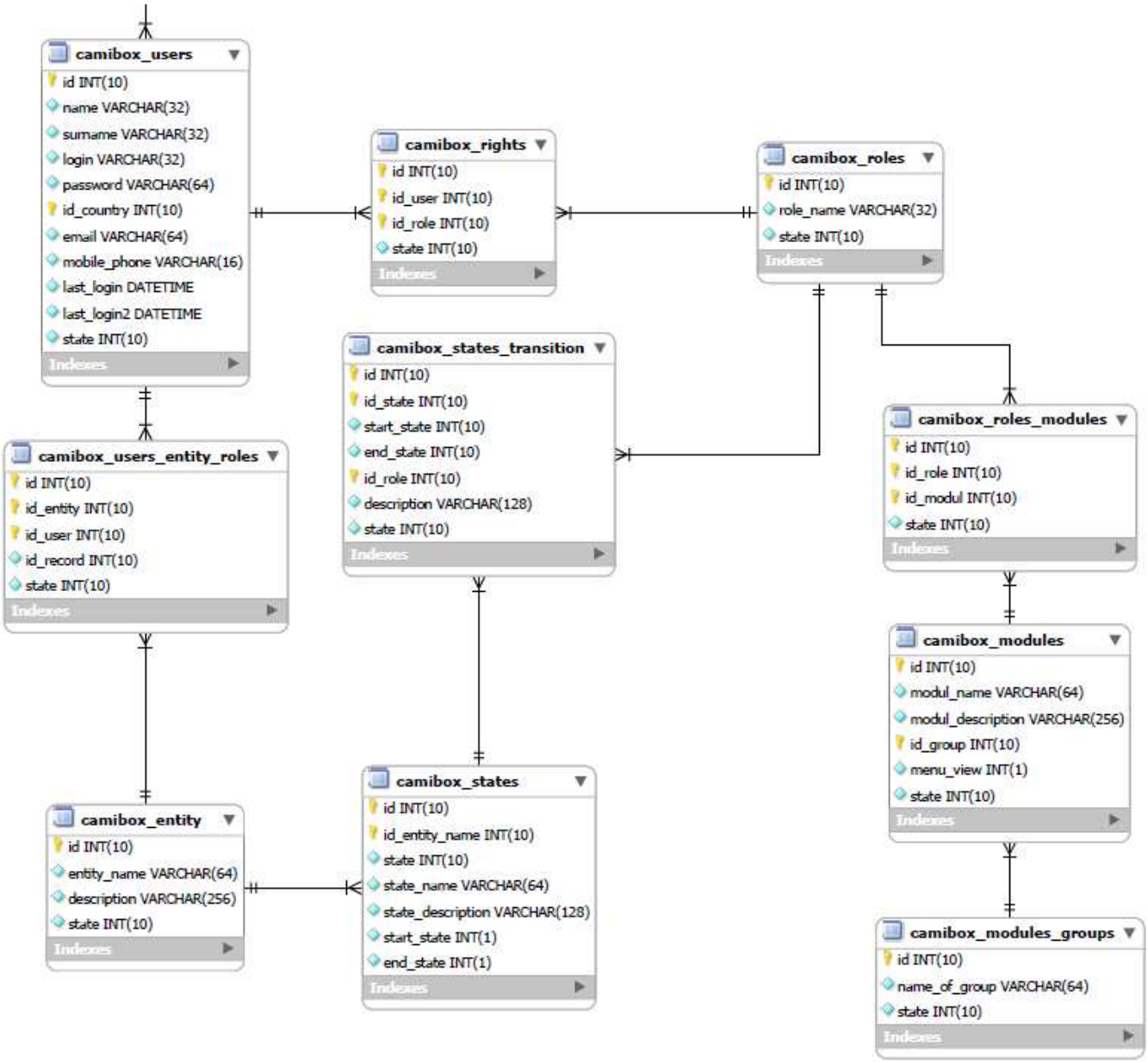
Tabulka camibox_users_entity_roles slouží k identifikaci vložených záznamů do databázového systému jednotlivými uživateli. Je svázána vazbou 1:N s tabulkou camibox_entity pomocí cizího klíče id_entity, což umožňuje odkazovat se na stejnou entitu z různých záznamů a také vazbou 1:N s tabulkou camibox_users pomocí cizího klíče id_user, což umožňuje přiřadit záznam danému uživateli. Dále obsahuje atribut id_record, jehož hodnota odkazuje na primární klíč záznamu z dané tabulky, určené atributem id_entity.

Tabulka camibox_states je číselník všech možných stavů záznamů v jednotlivých entitách. Obsahuje atributy state a state name, které spolu s id_entity_name jednoznačně identifikují daný stav záznamu. Hodnoty atributů start_state a end_state určují, zda se jedná o počáteční nebo koncový stav. Cizí klíč id_entity_name se vazbou 1:N odkazuje na tabulku camibox_entity, takže můžeme identifikovat více stavů jedné entity.

Tabulka camibox_entity tvoří číselník tabulek, dostupných v systému. Je tvořena atributem entity_name, což je unikátní jméno tabulky a atributem description, což je popis dané tabulky.

V tabulce camibox_states_transition jsou uloženy informace o jednotlivých přechodech mezi stavy a právy rolí na tyto přechody. Tato tabulka se pomocí cizího klíče id_state a vazby 1:N odkazuje na tabulku camibox_states a pomocí cizího klíče

id_role a vazby 1:N se zase odkazuje na tabulku camibox_roles, díky čemuž je možné jednoznačně určit, která role má právo převést záznam z jednoho stavu do druhého. Atributy start_state, end_state, id_state a id_role je jednoznačně určeno, o který stav se jedná a která role ho může převést do dalšího stavu.



Obrázek 4.1 ER diagram – uživatelé a jejich práva

4.3 E-R DIAGRAM ČÁST 2 – JEDNOTKY

Další část E-R diagramu je zaměřena na ukládání informací o jednotkách Camibox. Jako v předchozí části, i zde mají všechny tabulky primární klíč s názvem atributu id, datovým typem INT s maximálním počtem číslic omezeným na deset a používají speciální funkci AUTO_INCREMENT, která každému nově vkládanému záznamu přiřadí hodnotu o jednu větší, než v předchozím záznamu. Všechny tabulky také obsahují atribut state, který určuje v jakém stavu se právě daný záznam nachází.

Základní tabulkou je camibox_units, která obsahuje nejdůležitější údaje o vyrobených jednotkách, jako jejich sériové číslo, nastavené přístupové údaje a operační systém (atributy serial_number, login, password, os). I zde je pro tyto textové řetězce použit datový typ VARCHAR, délka textového řetězce je určena podle potřeby a přibližné velikosti vkládaného textu od 16 do 128 znaků. Pomoci cizího klíče id_unit_type (cizí klíč je i zde vždy reprezentován názvem atributu s prefixem id_, používá datový typ INT a má stanoven maximální počet číslic na deset) je tato tabulka svázána vazbou 1:N s tabulkou camibox_units_types. Tato vazba určuje, že v tabulce camibox_units může být více jednotek stejného typu.

Do tabulky camibox_units_lans, se ukládají informace o LAN portech vkládaných jednotek. Pomocí cizího klíče id_unit je vazbou 1:N svázána s tabulkou camibox_unit, což umožňuje ukládat do tabulky informace i o jednotkách, které mají více jak jeden LAN port. Tabulka obsahuje atribut mac_address, jehož hodnota určuje unikátní MAC adresu daného portu a atribut lan_number (číselné hodnoty zde používají datový typ INT, počet číslic je omezen podle potřeby na 1-10), jehož hodnota určuje pořadí portu, o který se u dané jednotky jedná.

Tabulka camibox_units_types shromažďuje informace o všech dostupných typech jednotek a počtu jejich portů. Obsahuje cizí klíč id_type, kterým se odkazuje vazbou 1:N na tabulku camibox_units_types_master. Díky této vazbě je možné mít v systému více jednotek stejného typu. Hodnota atributu mark nese hodnotu označení daného typu, hodnoty atributů lan_number, wlan_station_number a wlan_bridge number určují počet portů LAN, WLAN v režimu station nebo WLAN v režimu bridge, které daná jednotka obsahuje.

Tabulka camibox_units_types_master je vlastně číselník základních typů jednotek. Obsahuje atribut type_unit, jehož hodnota nese název hlavní skupiny typů jednotek.

Tabulka camibox_units_wlans_antennas obsahuje údaje o WLAN portech dané jednotky. Pomocí cizího klíče id_unit je vazbou 1:N svázána s tabulkou camibox_units, což umožňuje ukládat do tabulky informace i o jednotkách, které mají více jak jeden WLAN port. Obsahuje atribut type, jehož hodnota nese informaci o tom, zda-li se jedná o port typu client nebo o port typu bridge, atribut mac_address, jehož hodnota určuje unikátní MAC adresu daného portu a atribut wlan_number, jehož hodnota určuje pořadí portu, o který se u dané jednotky jedná.

Tabulka camibox_units_wlans_antennas je vazební tabulka mezi WLAN porty a jednotlivými anténami. Pomocí cizího klíče id_wlan je svázána vazbou 1:N s tabulkou camibox_units_wlans, což umožňuje přiřadit jednomu portu více antén. Dále je pomocí cizího klíče id_antenna je svázána vazbou 1:N s tabulkou camibox_antennas, což umožňuje, aby byla jedna anténa přiřazená více portům.

Tabulka camibox_antennas je číselník jednotlivých antén, které se dají k jednotce připojit. Hodnota atributu antenna_mark nese celý název antény, hodnota atributu diameter určuje průměr antény, hodnoty atributů gain_tx a gain_rx zisk antény v přijímacím nebo vysílacím směru.

Tabulka camibox_units_wlans_connections se pomocí dvou cizích klíčů id_source_wlan a id_destination_wlan odkazuje vazbou 1:1 na tabulku camibox_units_wlans. Toto spojení by se dalo realizovat i v tabulce camibox_units_wlans, ale toto řešení umožňuje snazší dotazy na propojení jednotlivých portů jednotek. Tabulka obsahuje atribut connection_lenght, jehož hodnota nese informaci o přibližné délce bezdrátového spoje.

Obrázek 4.2 ER diagram – jednotky a jejich porty

4.4 E-R DIAGRAM ČÁST 3 – ZÁKAZNÍCI A JEJICH ZAKÁZKY

Třetí část je zaměřena na data o zákaznících a jejich zakázkách. Jako v předchozích dvou částech, i zde mají všechny tabulky primární klíč s názvem atributu id, datovým typem INT s maximálním počtem číslic omezeným na deset a používají speciální funkci AUTO_INCREMENT, která každému nově vkládanému záznamu přiřadí hodnotu o jednu větší, než v předchozím záznamu. Všechny tabulky také obsahují atribut state, který určuje v jakém stavu se právě daný záznam nachází.

Základem je tabulka camibox_custommers, ve které jsou umístěna základní data charakterizující danou firmu. Atributy company_name, name, surname, street, email a phone určují název společnosti, jméno a příjmení zodpovědné osoby, adresa firmy, kontaktní email a mobilní telefon. I zde je pro tyto textové řetězce použit datový typ VARCHAR, délka textového řetězce je určena podle potřeby a přibližné velikosti vkládaného textu od 16 do 256 znaků. Hodnota atributu insert_date nese informaci o datu vložení zákazníka do systému (pro datumové a časové atributy je použit datový typ DATETIME, což je kombinace data a času ve formátu rok-měsíc-den hodina:minuta:sekunda). Cizí klíč id_city váže tabulku vazbou 1:N s tabulkou camibox_cities, takže více zákazníků může být ze stejného města a cizí klíč id_country váže tabulku vazbou 1:N s tabulkou camibox_countries, takže z jednoho státu může být více zákazníků.

Tabulka camibox_cities obsahuje číselník měst. Atribut city_name určuje jméno města, atribut post_code jeho poštovní směrovací číslo.

Tabulka camibox_countries obsahuje číselník států. Hodnota atributu country_name určuje celé jméno daného státu a hodnota atributu abbr mezinárodní zkratku daného státu.

Tabulka camibox_config_countries obsahuje seznam povolených pásem a scanlist v daném státě. Pomocí cizího klíče id_country je vazbou 1:N svázána s tabulkou camibox_countries, což znamená, že jeden stát může mít přiřazeno více frekvenčních rozsahů. Pomocí cizího klíče id_band je svázána vazbou 1:N s tabulkou camibox_bands, což určuje, že v jednom státě může být více frekvenčních pásem. Hodnota atributu frequency_scanlist nese hodnoty povolených frekvencí v daném státě, které jsou od sebe odděleny čárkou.

Tabulka camibox_bands je číselník používaných frekvenčních pásem. Hodnota atributu band nese název pásma včetně informace o jaké frekvenční pásmo se jedná.

Tabulka camibox_contracts obsahuje všechny zakázky, vytvořené v IS. Atributy action_name, wpa2_key a description určují jméno zakázky, wpa2 klíč pro zabezpečený přenos a stručný popis zakázky (číselné hodnoty zde používají datový typ INT, počet číslic je omezen podle potřeby na 1-10). Cizí klíč id_custommer realizuje vazbu 1:N s tabulkou camibox_custommers, a umožňuje tak přiřadit k jednomu zákazníkovi více zakázek.

Cizí klíč id_country_realization vytváří vazbu 1:N s tabulkou camibox_countries, umožňuje vytváření více zakázek v jedné zemi a určuje, jaké frekvenční pásmo bude pro danou zakázku použito.

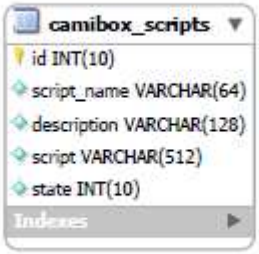
Vazební tabulka camibox_units_contracts přiřazuje jednotky k jednotlivým zakázkám. Cizí klíč id_contract realizuje vazbu 1:N s tabulkou camibox_contracts a určuje, že jednotka může být současně u více zakázek a cizí klíč id_unit realizuje vazbu 1:N s tabulkou camibox_units a určuje, že více jednotek může být přiřazeno u jedné zakázky.

Tabulka camibox_contracts_design obsahuje seznam všech vytvořených bodů v zakázkách a do nich vložených jednotek včetně jejich souřadnic v GUI. Cizí klíč id_contract tvoří vazbu 1:N s tabulkou camibox_contract a určuje, že v jedné zakázce může být v návrhu více bodů. Cizí klíč id_unit_type definuje vazbu 1:N s tabulkou camibox_units_types a určuje, že jeden typ jednotky může být použit ve více návrzích.

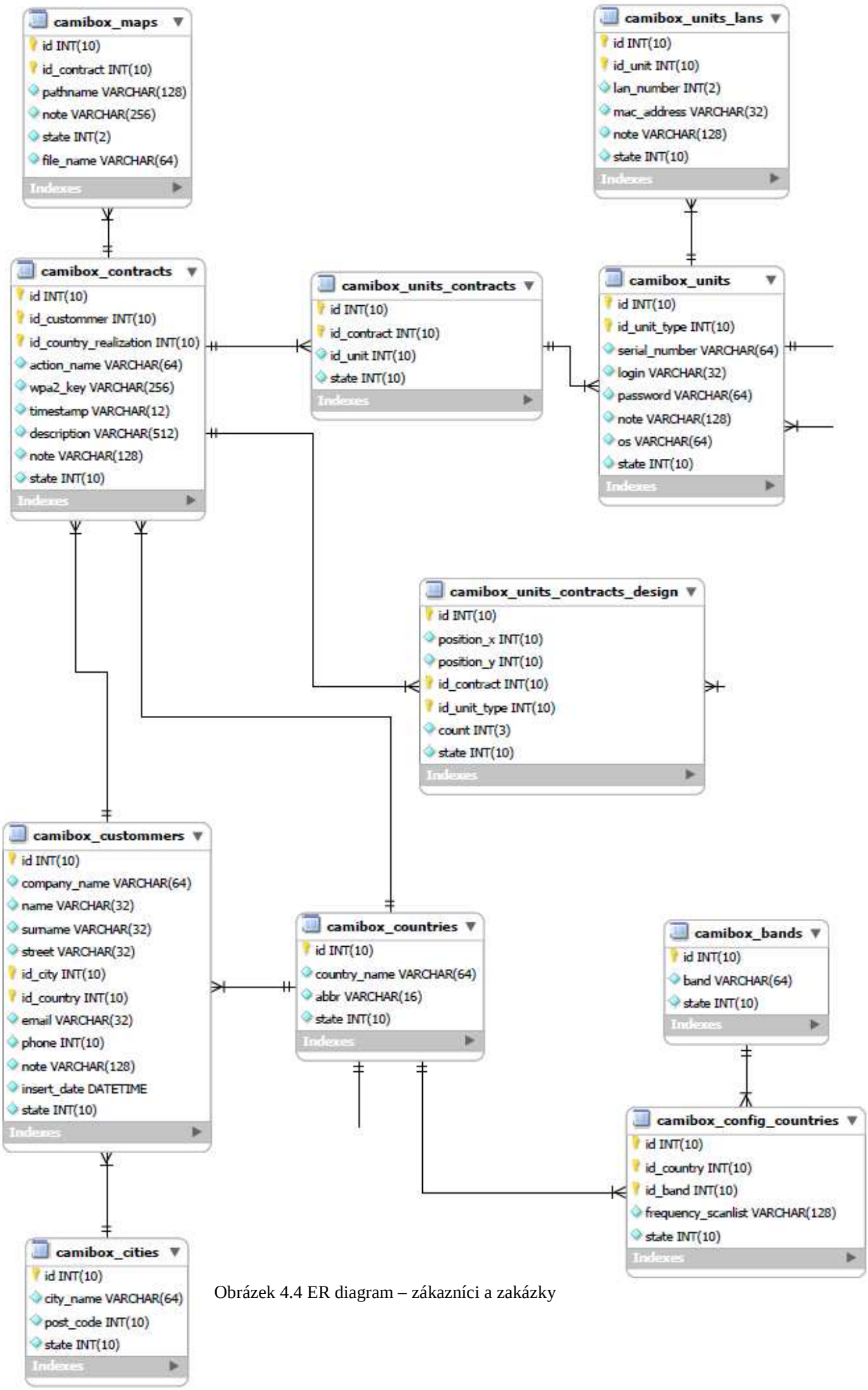
Tabulka camibox_maps obsahuje obrázky, vložené do GUI jako mapové podklady k jednotlivým zakázkám. Pomocí cizího klíče id_contract je svázána vazbou 1:N s tabulkou camibox contract a určuje, že jedna zakázka může mít více mapových podkladů. Hodnota atributu pathname určuje adresář, ve kterém se dané obrázky nachází, a hodnota atributu file_name nese přesný název souboru.

4.5 E-R DIAGRAM ČÁST 4 – TABULKA CAMIBOX_SCRIPTS

V této části je pouze jediná tabulka, které nemá žádný cizí klíč a ani se na ní žádný cizí klíč neodkazuje, takže není žádnou vazbou spojena s ostatními tabulkami. Jedná se o tabulku camibox_scripts, která obsahuje všechny dostupné skripty pro konfiguraci jednotek. Atribut script_name udává jedinečný název skriptu, atribut description jeho stručný popis. Hodnota atributu script určuje umístění skriptu v adresářové struktuře webu.



Obrázek 4.3 ER diagram – tabulka camibox_scripts



Obrázek 4.4 ER diagram – zákazníci a zakázky

4.6 ŽIVOTNÍ CYKLY

Životní cykly jsou důležitou součástí IS. Řídí přístup k jednotlivým záznamům v entitách, takže je IS chráněn proti nežádoucí modifikaci dat.

4.6.1 Definované role v informačním systému

Přechod mezi stavy životního cyklu je vždy určen jednou nebo více rolemi, které mohou danou transakci provést. Je tedy třeba definovat si role, které budou v IS figurovat.

1. Jednatel – bude mít přístup ke všem datům mimo administrace
2. Obchodní zástupce – bude mu umožněno přidávat nové zákazníky, vkládat a plánovat nové zakázky, atd.
3. Technik – bude mít přístup ke vkládání nových jednotek a ke konfiguraci
4. Administrátor – bude mít přístup ke všem číselníkům a nastavení IS

4.6.2 Nejdůležitější životní cykly

Ve většině entit mají záznamy pouze dva stavy, a to platný a neplatný záznam. V dalších je cyklus poměrně složitější (viz. následující odstavce).

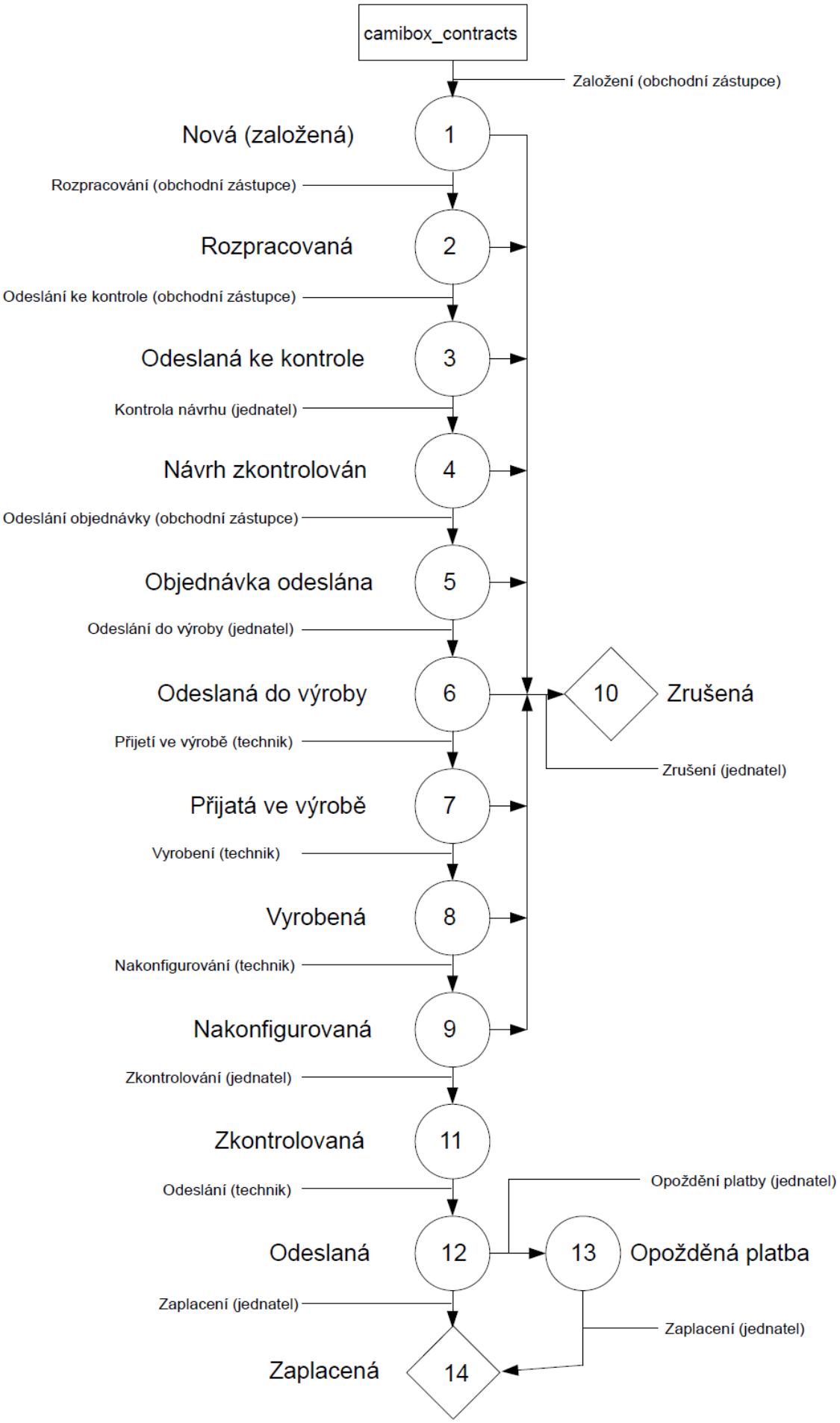
Životní cyklus záznamů v entitě camibox_contracts

Zakázky jsou jednou z nejdůležitějších položek v systému. Při náhledu na ní musíme jednoznačně zjistit, v jakém stavu se právě nachází. Při vytvoření je zakázka uvedena do stavu nová (založená). Do tohoto stavu ji může převést role obchodního zástupce za předpokladu, že je definován zákazník, pro kterého je zakázka navrhována a upřesňující údaje k ní (název akce, cílová země, předpokládaný rozsah zakázky apod.). Poté může role obchodního zástupce v grafickém rozhraní systému navrhovat topologii celé sítě. Pokud je definována alespoň jedna MASTER jednotka, je umožněn obchodnímu zástupci přesun zakázky do dalšího stavu – rozpracovaná. V tomto stavu je v IS uložena aktuální konfigurace zakázky a je možné dělat další úpravy. Pokud je zakázka již v konečně navrhnuté podobě, může jí role obchodního zástupce převést do dalšího stavu – odeslaná ke kontrole. V tomto stavu zakázku převezme role jednatele, který jí celou zkontroluje. Pokud najde nějaké nesrovnalosti, tak je opraví a posune zakázku do stavu návrh zkontrolován. Zde si ji opět převezme obchodní zástupce a pokud mu daná topologie vyhovuje, vyplní svůj identifikační klíč pro objednávky a

převede zakázku do stavu objednávka odeslána. Zde si ji opět přebírá role jednatele a převádí ji do stavu odeslaná do výroby, za předpokladu přijetí částky ze zálohové faktury, vystavené při objednání zakázky. Nyní si zakázku přebírá další role – technik. Ten opět zkontroluje všechny údaje a převede zakázku do stavu přijatá ve výrobě. Po tom, co technik zkompletuje všechny jednotky a zadá je do systému může zakázku poslat do dalšího stavu – vyrobená, za předpokladu, že všechny jednotky, které byly objednány pro tuto zakázku jsou ve stavu vyrobená. Nyní provede technik konfiguraci jednotek pomocí IS. Poté, co jsou všechny jednotky nakonfigurovány převede zakázku do stavu nakonfigurovaná, za předpokladu, že i všechny použité jednotky jsou ve stavu nakonfigurovaná. Nyní si zakázku přebírá jednatel, který si otestuje funkčnost celé zakázky a převede ji do stavu zkontrolovaná. Poté nastupuje zase role technika, který celou zakázku zabalí do krabic a odešle na některou z přepravních služeb. V tuto chvíli zakázku převede do stavu odeslaná. Poté je vystavena doplatková faktura. Pokud se zákazník s platbou této faktury opozdí, jednatel převede zakázku do stavu opožděná platba, a zákazníkovi je odeslán email s upozorněním. Po zaplacení částky jednatel zakázce nastaví konečný stav – zaplacená. Z jakéhokoliv stavu mimo posledních čtyř (zkontrolovaná, odeslaná, opožděná platba, zaplacená) může jednatel převést zakázku do konečného stavu zrušená, za předpokladu že uvede důvod zrušení zakázky.

Poč.stav	Konc. stav	Název přechodu	Role
-	1	Založení	Obchodní zástupce
1	2	Rozpracování	Obchodní zástupce
2	3	Odeslání ke kontrole	Obchodní zástupce
3	4	Kontrola návrhu	Jednatel
4	5	Odeslání objednávky	Obchodní zástupce
5	6	Odeslání do výroby	Jednatel
6	7	Přijetí ve výrobě	Technik
7	8	Vyrobení	Technik
8	9	Nakonfigurování	Technik
9	11	Zkontrolování	Jednatel
11	12	Odeslání	Technik
12	13	Opoždění platby	Jednatel
12	14	Zaplacení	Jednatel
1-9	10	Zrušení	Jednatel

Tabulka 4.2: Tabulka přechodů pro entitu camibox_contracts



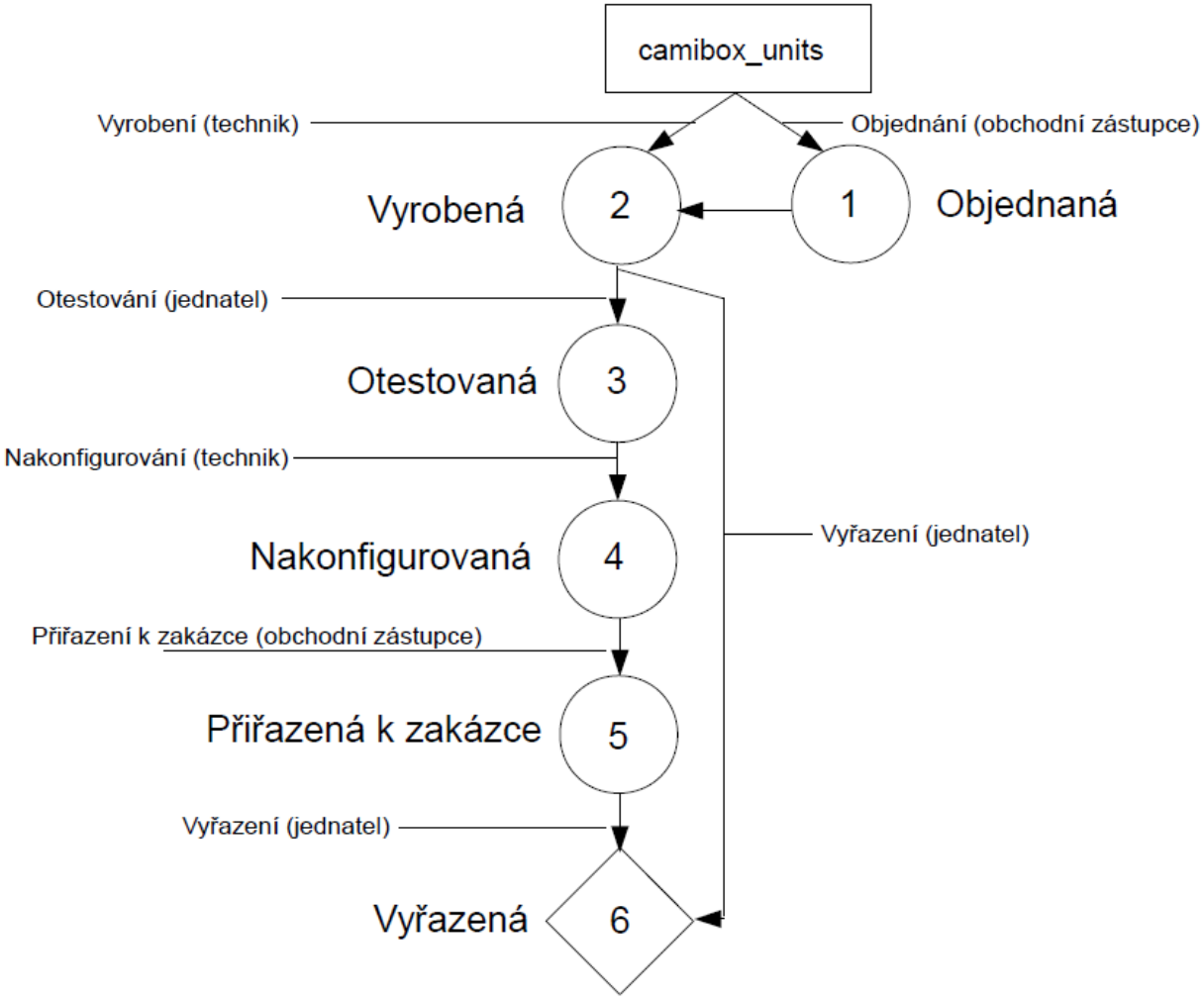
Obrázek 4.5 Životní cyklus záznamů v entitě camibox_contracts

Životní cyklus záznamů v entitě camibox_units

Jednotky jsou základem všech funkcí v informačním systému. Jejich cesta začíná tím, že je na skladu materiál pro jejich výrobu. Do počátečního stavu vyrobená ji může role technika převést za předpokladu, že je v systému zadáno sériové číslo této jednotky a je fyzicky evidována na skladu. Pokud není při objednání zakázky na skladě dostatečný počet jednotek, je jednotka obchodním zástupcem při objednávce zakázky převedena do počátečního stavu objednaná, a z tohoto stavu jí opět technik po vyrobení a umístění hotového kusu na sklad převede do stavu vyrobená. Poté ji role jednatele zkontroluje, a převede jednotku do stavu otestovaná. Stav jednotky postupně postupuje se stavem zakázky, k ní přiřazené. Pokud byla jednotka již připojena v GUI k nějaké zakázce a nakonfigurována, je technikem převedena to stavu nakonfigurovaná. Pokud je zakázka, kde je umístěna i příslušná jednotka expedována, je jí nastaven stav přiřazena k zakázce. Pokud se při otestování nebo v již expedované zakázce objeví nefunkční jednotka, je jí nastaven konečný stav vyřazená.

Poč.stav	Konc. stav	Název přechodu	Role
-	1	Objednání	Obchodní zástupce
-	2	Vyrobení	Technik
2	3	Otestování	Jednatel
3	4	Nakonfigurování	Technik
4	5	Přiřazení k zakázce	Obchodní zástupce
5, 2	6	Vyřazení	Jednatel

Tabulka 4.3: Tabulka přechodů pro entitu camibox_units



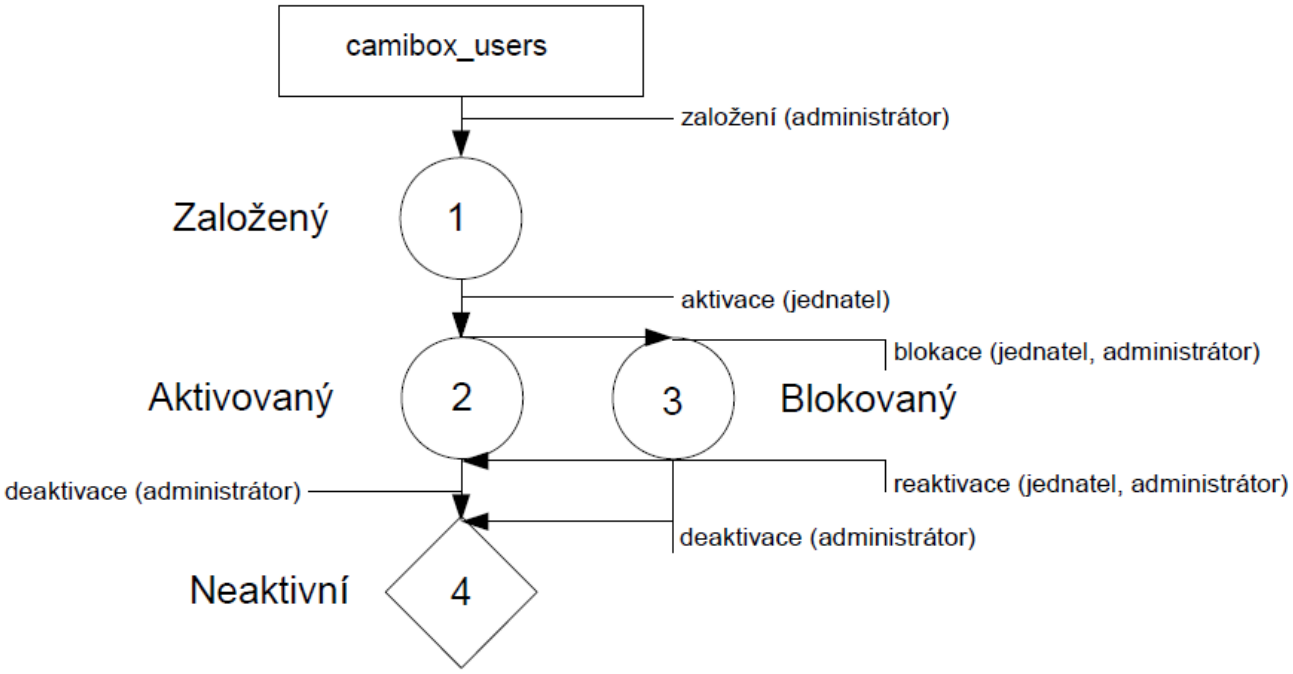
Obrázek 4.6 Životní cyklus záznamů v entitě camibox_units

Životní cyklus záznamů v entitě camibox_users

Při založení nového uživatele rolí administrátor se uživatel dostane do stavu založený. Poté čeká na odsouhlasení jednatelem, který zkontroluje všechny povinné údaje (jméno, příjmení, tel.číslo, email, stát, apod.), přiřadí uživateli jednu nebo více z možných rolí a posune ho do stavu aktivovaný. Pokud se stane, že uživateli je třeba nějakým způsobem zamezit působení v IS (např. z důvodu pokusu o průnik k důvěrným datům, nespolehlivému jednání apod.), je převeden jednatelem nebo administrátorem do stavu blokovaný. Pokud okolnosti, které ho přiměly k blokování pominou, může ho jednatel nebo administrátor znovu přesunout do stavu aktivní. Jestliže uživatel trvale ukončil spolupráci s firmou, je mu administrátorem nastaven konečný stav neaktivní.

Poč.stav	Konc. stav	Název přechodu	Role
-	1	Založení	Administrátor
1	2	Aktivace	Jednatel
2	3	Blokace	Jednatel, administrátor
3	2	Reaktivace	Jednatel, administrátor
2,3	4	Deaktivace	Administrátor

Tabulka 4.4: Tabulka přechodů pro entitu camibox_users



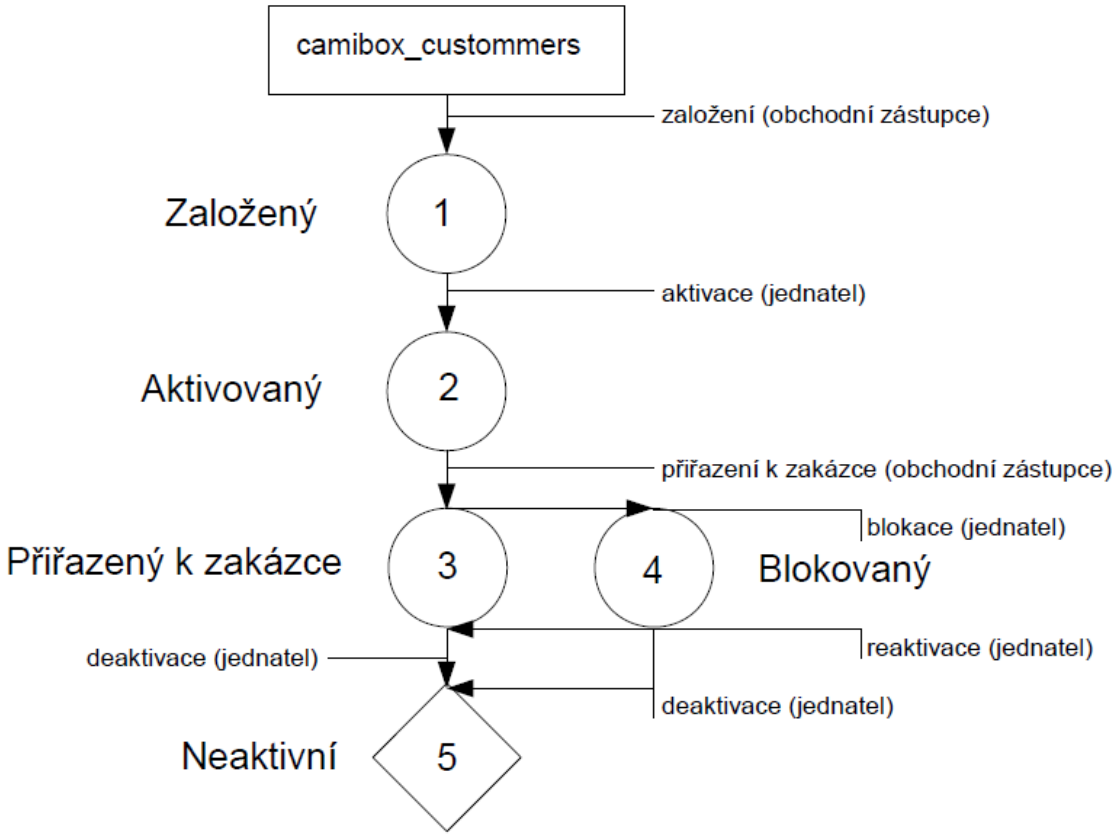
Obrázek 4.7 Životní cyklus záznamů v entitě camibox_users

Životní cyklus záznamů v entitě camibox_custommers

Do počátečního stavu – založený - zákazníka může převést role obchodního zástupce. Poté jednatel zkontroluje všechny povinné údaje (jméno, příjmení, tel.číslo, email, stát, apod.), a může zákazníka převést do stavu aktivovaný. V tuto chvíli je možné vytvořit zakázku, která bude patřit danému zákazníkovi. Po objednání zakázky (obchodním zástupcem) se stav zákazníka přesune do stavu přiřazený k zakázce. Pokud nastanou nějaké důvody pro zablokování zákazníka (např. špatná instalace jednotek apod.), posune ho role jednatele do stavu blokovaný. Pokud důvody pro blokování pominou, role jednatele ho opět může převést do stavu aktivovaný nebo přiřazený k zakázce. Do konečného stavu neaktivní ho může převést opět role jednatele, např. po zaniknutí firmy apod.

Poč.stav	Konc. stav	Název přechodu	Role
-	1	Založení	Obchodní zástupce
1	2	Aktivace	Jednatel
2	3	Přiřazení k zakázce	Obchodní zástupce
3	4	Blokace	Jednatel
4	3	Reaktivace	Jednatel
3	5	Deaktivace	Jednatel

Tabulka 4.4: Tabulka přechodů pro entitu camibox_custommers



Obrázek 4.8 Životní cyklus záznamů v entitě camibox_custommers

5. IMPLEMENTACE IS

Rozhodl jsem se pro vývoj systému ve skriptovacím jazyku PHP z důvodu jeho bezproblémové podpory databázového systému MySQL. Dále jsem se rozhodl pro vývoj některých částí systému v jazyku JavaScript, protože má nekonkurovatelnou podporu v reakcích aplikace na straně uživatele. Některé části systému jsou naprogramovány pomocí technologie Ajax, která přidává JavaScriptu neocenitelnou podporu v podobě možnosti výměny dat se serverem.

5.1 KONCEPCE A STRUKTURA IS

V informačním systému byly implementovány čtyři hlavní moduly – admin, contract, custommers a unit. Modulem je zde myšlena skupina skriptů, která obsluhuje jednu logicky související část informačního systému.

Modul admin obsahuje všechny obslužné funkce, které jsou třeba ke správě používaných číselníků a nastavení systému. Umožňuje celkovou správu systému.

Modul contract obsluhuje vše, co se týká zakázek. Umožňuje tedy zakázky do systému vkládat, modifikovat je, vyhledávat a třídit. Oprávněné role můžou posouvat zakázky po jejich životním cyklu podle předem definovaných pravidel. Modul také obsahuje grafické návrhové rozhraní, pomocí kterého je možné celou zakázku graficky navrhnout.

Modul custommers umožňuje vkládat nové zákazníky, editovat a třídit již vložené. Je zde dáván důraz na zabezpečení dat tak, aby se uživatel dostal jen k záznamům, které on sám vložil.

Modul unit obsahuje funkce, pomocí kterých je možné do systému vkládat nové jednotky, editovat je a třídit. Umí také podle vstupní zakázky vygenerovat skripty pro OS Mikrotik, které posléze nahraje do jednotky.

Každý modul je v základu reprezentován třemi soubory (skripty v jazyku PHP), a to InsertFormem, pomocí kterého se data vkládají do IS (je identifikován pomocí prefixu if_ před názvem souboru), ViewFormem, pomocí kterého se dají uložená data editovat (je identifikován pomocí prefixu vf_ před názvem souboru) a RecordListem, který dokáže pomocí integrovaného QueryFormu odfiltrovat informace, o které nemáme zájem a zobrazit tabulkový přehled požadované skupiny dat (je identifikován prefixem _rl před názvem souboru).

Moduly se nacházejí v adresářích, jejichž název je shodný s názvem daného modulu a jsou umístěny v kořenovém adresáři webu, kde je také umístěn adresář file, ve kterém jsou všechny globálně používané soubory. Jde o soubor style.css, ve kterém jsou definovány všechny kaskádové styly pro celý IS a o soubor head.php, ve kterém jsou informace, importované do hlaviček všech používaných skriptů. Dále jsou zde skripty v jazyce JavaScript, konkrétně menu.js, kde jsou funkce pro dynamické reakce menu, java_fc.js, kde jsou veškeré používané JavaScriptové funkce a ajax_fc.js, kde jsou

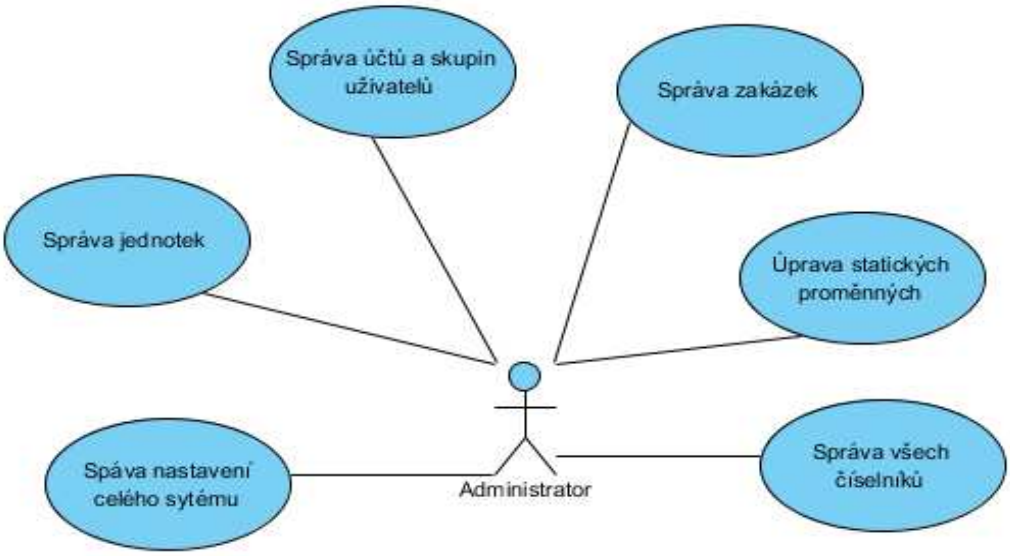
všechny funkce, používající metodu XMLHttpRequest technologie Ajax. V IS jsou definovány tři třídy v jazyku PHP, a to camiboxDb pro všechny funkce, které nějakým způsobem komunikují s databází a camiboxForm pro všechny funkce, používané ve formulářích a mikrotikAPI pro funkce, které zabezpečují vzdálenou komunikaci s OS Mikrotik.

5.2 USE CASE DIAGRAMY

Diagram případu užití poskytuje náhled na funkce, které bude mít daný uživatel (aktér) k dispozici. V tomto IS máme čtyři hlavní aktéry, jimiž jsou administrátor, obchodní zástupce, jednatel a technik.

Diagram případu užití – administrátor

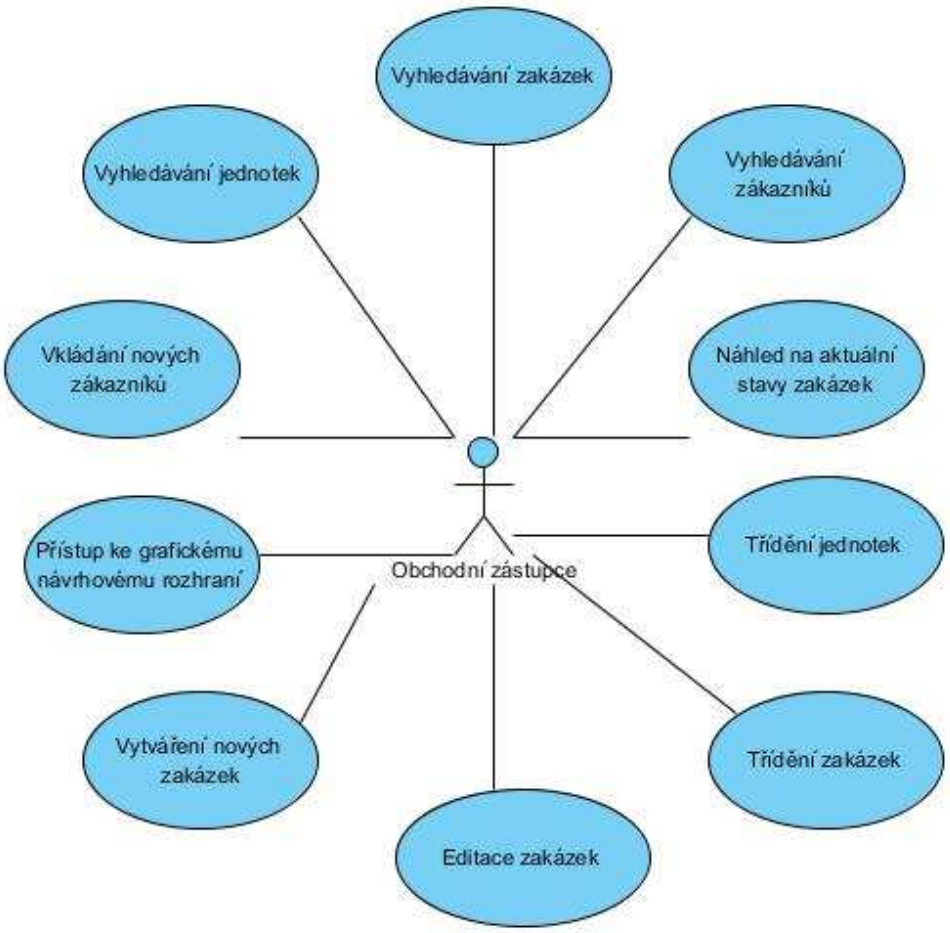
Administrátor bude mít přístup k celkové správě IS. Bude moct upravovat data v číselnících (za určitých podmínek) a vkládat do nich nové záznamy. Také může nastavovat statické proměnné, jako např. maximální velikost nahrávaných souborů, počet maximálně přihlášených uživatelů aj. Bude mít přístup ke správě uživatelských účtů, bude moct modifikovat data u jednotlivých účtů a přiřazovat jim patřičná oprávnění.



Obrázek 5.1 Diagram případu užití – administrátor

Diagram případu užití – obchodní zástupce

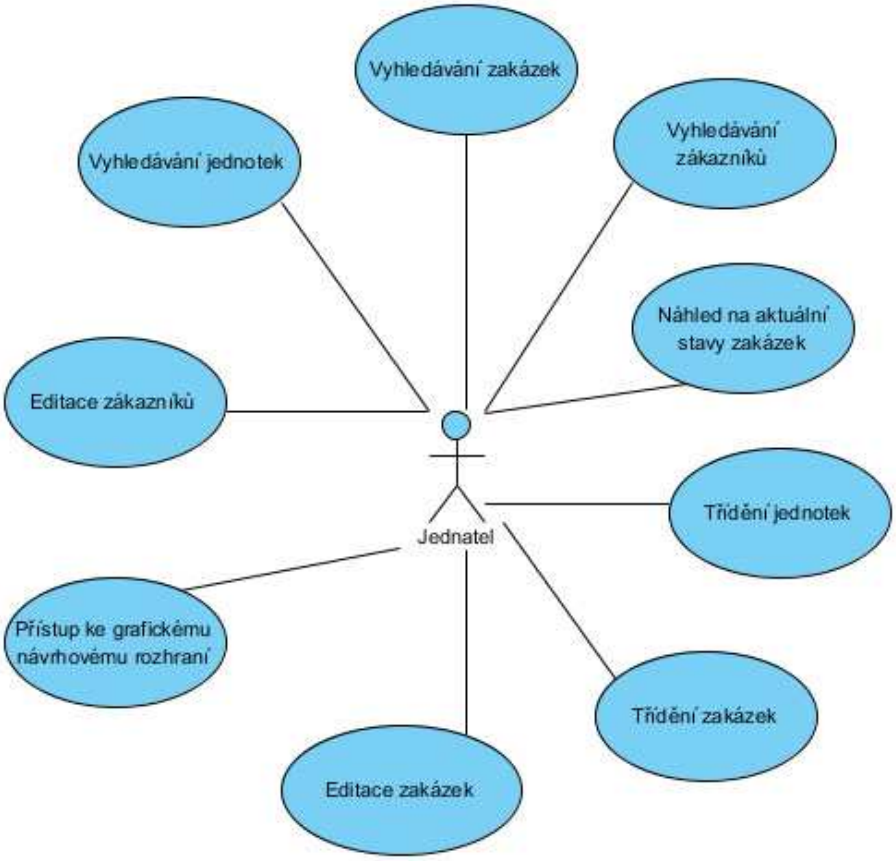
Obchodní zástupce bude mít přístup k vyhledávání jednotek, zakázek a zákazníků. Bude moct také třídit jednotlivé jednotky a zakázky. Bude mít přístup ke grafickému návrhovému rozhraní, takže bude moci zakládat nové zakázky a modifikovat je. Bude mu také umožněno vkládat do IS nové zákazníky.



Obrázek 5.2 Diagram případu užití – obchodní zástupce

Diagram případu užití – jednatel

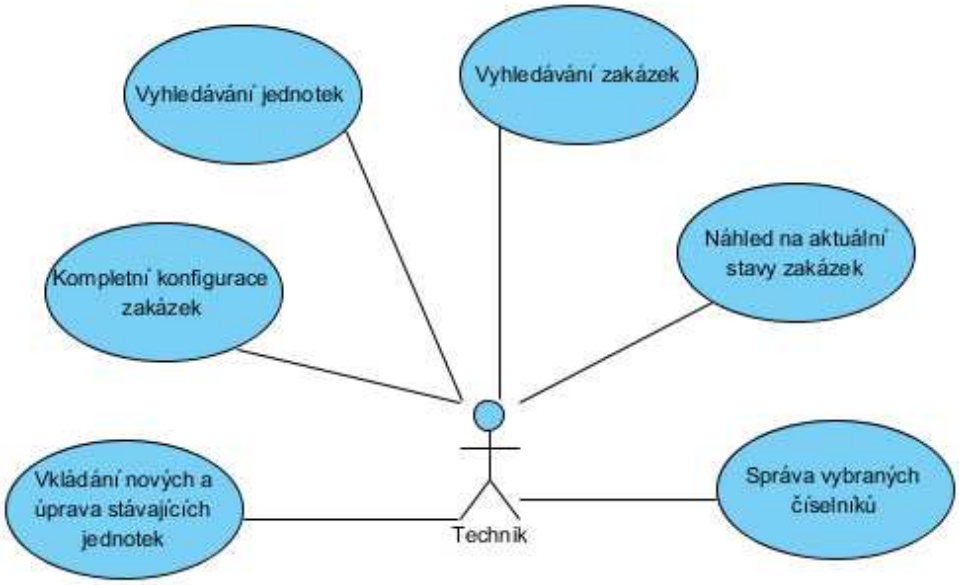
Jednatel bude mít podobná přístupová práva, jako obchodní zástupce. Bude moct vyhledávat jednotlivé jednotky, zakázky a zákazníky. Systém mu umožní třídit jednotky a zakázky podle kriterií a také bude moct zakázky a zákazníky editovat. Bude mít přístup ke grafickému návrhovému systému z důvodu opravy již odeslaných zakázek ke kontrole. Bude mít náhled na aktuální stavy všech zakázek.



Obrázek 5.3 Diagram případu užití – jednatel

Diagram případu užití – technik

Technik bude mít přístup pouze k základním funkcím systému. Bude mu umožněno vyhledávání jednotlivých zakázek a jednotek. Bude mít aktuální náhled na právě probíhající, jeho se týkající zakázky. Může do systému vkládat nové jednotky a modifikovat je. Bude mít přístup ke kompletnímu konfiguračnímu rozhraní celých zakázek.



Obrázek 5.4 Diagram případu užití – technik

5.3 DATA FLOW DIAGRAMY

V informačním systému jsou definovány 4 hlavní moduly. Každý modul bude v základu tvořen InsertFormem pro vkládání dat, ViewFormem pro jejich editaci a RecordListem s integrovaným QueryFormem pro jejich třídění a náhled. Při jejich návrhu byly vytvořeny DFD diagramy, které modelují příslušné datové a informační toky mezi nimi a příslušnými entitami. DFD diagramy jsou umístěny v přílohách B1-B4.

5.3.1 DFD – modul admin

Modul admin (jehož DFD diagram je umístěn v příloze B.1) bude primárně určen ke správě všech číselníků v systému. Základem tohoto modulu je proces if_admin.php, který se stará o vkládání nových dat do tabulek. Provádí operace INSERT (vkládání dat do databáze) a SELECT (dotaz na databázi, zda-li nově vkládaný záznam již neexistuje) s entitami camibox_bands (vkládání nových frekvenčních pásem), camibox_cities (vkládání nových měst), camibox_config_contries (vkládání nových povolených pásem v daných státech), camibox_rights (vkládání nových práv uživatelů), camibox_roles_modules (vkládání nových práv rolí na jednotlivé moduly), camibox_scripts (vkládání nových konfiguračních skriptů), camibox_antennas (vkládání nových antén), camibox_units_types (vkládání nových typů jednotek), camibox_units_types_master (vkládání nových hlavních typů jednotek) a camibox_users (vkládání nových uživatelů). Může být volán buď přímo z menu IS, nebo z procesu rl_admin.php.

Po vložení záznamu do databáze volá proces if_admin.php další proces, a to vf_admin.php. Ten se stará o zobrazování detailů uložených dat a umožňuje editovat vybrané atributy jednotlivých entit, které jsou totožné s těmi, které používá proces if_admin.php. Provádí operace SELECT (dotaz na požadovaná data) a UPDATE (aktualizace modifikovaných dat).

Další proces, který je v tomto modulu k dispozici je qf_admin.php. Je volán vždy, když žádáme o přístup k procesu rl_admin.php. Jeho úkolem je filtrace dat a to tak, aby se nám zobrazili požadované informace v přiměřeném množství záznamů. Provádí pouze operace SELECT (dotaz na uživatele) do tabulky camibox_users.

Posledním procesem je rl_admin.php. Je volán vždy nadřazeným procesem qf_admin.php. Jeho výstupem je seznam požadovaných záznamů. Provádí pouze operace SELECT (dotaz na požadované informace) s tabulkami, které jsou totožné s těmi, které používají procesy if_admin.php a vf_admin.php.

5.3.2 DFD – modul contract

Účelem modulu contract (jehož DFD diagram je umístěn v příloze B.2) bude správa zakázek, evidovaných v systému a jejich grafický návrh. Základem tohoto modulu je proces if_contract.php, který se stará o vkládání nových dat do tabulek. Provádí operace INSERT (vkládání dat do databáze) s entitami camibox_contracts (vkládání nových zakázek) a camibox_users_entity_roles (přiřazení vkládaného záznamu k příslušnému uživateli) a operace SELECT (vyžádání si dat ze souvisejících tabulek) s entitami camibox_countries (přiřazení zakázky do jednoho z nabízených států) a camibox_custommers (přiřazení zakázky k jednomu z nabízených zákazníků). Může být volán buď přímo z menu IS, nebo z procesu rl_contract.php.

Po vložení záznamu do databáze volá proces if_contract.php další proces, a to vf_contract.php. Ten se stará o zobrazování detailů uložených zakázek a umožňuje editovat vybrané atributy. Provádí operaci SELECT (dotaz na požadovaná data) entit camibox_countries, camibox_custommers a camibox_users_entity_roles a operaci UPDATE (aktualizace modifikovaných dat) entity camibox_contracts.

Další proces, který je v tomto modulu k dispozici je qf_contract.php. Je volán vždy, když žádáme o přístup k procesu rl_contract.php (například z procesu vf_contract.php). Jeho úkolem je filtrace dat a to tak, aby se nám zobrazily pouze požadované informace v přiměřeném množství záznamů. Provádí pouze operace SELECT (dotaz na požadované výběrové prvky filtru) entit camibox_countries, camibox_custommers, camibox_states a camibox_users_entity_roles.

Dalším procesem je rl_contract.php. Je volán vždy nadřazeným procesem qf_contract.php a jeho výstupem je seznam požadovaných záznamů. Provádí pouze operace SELECT (dotaz na požadované informace) s entitami camibox_custommers, camibox_contracts a camibox_states.

Poslední proces, používaný v tomto modulu je project_contract.php. Ten se stará o obsluhu funkcí a dat pro návrhy jednotlivých zakázek. Je možné ho spustit z procesu vf_contract.php. Provádí příkazy SELECT pro entity camibox_units_types_master (kontrola počtu jednotlivých typů jednotek v návrhu), camibox_units (dotazy na volné jednotky, které se dají použít na aktuálně navrhnutou zakázku), camibox_unit_types (dotaz na jednotlivé typy jednotek v systému, které se dají do návrhu vložit), camibox_units_wlan_connections (dotaz na spojení jednotlivých WLAN portů u navrhnuté zakázky), camibox_maps (dotaz na mapový podklad pro danou zakázku), camibox_units_contract_design (dotaz na body, které již byly vložené do návrhu zakázky) a camibox_users_entity_roles (dotaz, zda-li si požadované záznamy vyžádala role, která je do systému vložila). Dále provádí příkaz INSERT pro tabulky camibox_units_wlan_connections (vložení nového spojení WLAN portů jednotek v návrhu), camibox_maps (vložení nového mapového podkladu pro navrhovanou zakázku), camibox_units_contracts (přiřazení konkrétních jednotek k navrhované zakázce) a camibox_units_contract_design (vložení informací o aktuálně umístěném bodu v navrhované zakázce). Proces provádí také příkaz UPDATE, ale pouze s tabulkou

camibox_units_contract_design (aktualizovaní informace o pozici přemístěného bodu v návrhu zakázky).

5.3.3 DFD – modul custommers

Modul custommers (jehož DFD diagram je umístěn v příloze B.3) obsahuje procesy, které obsluhují data k jednotlivým zákazníkům. Základem tohoto modulu je proces if_custommers.php, který se stará o vkládání nových dat do tabulek. Provádí operace INSERT s entitami camibox_custommers (vložení informací o právě vkládaném zákazníku) a camibox_users_entity_roles (definování, který uživatel zákazníka do systému vložil). Dále provádí operace SELECT s entitami camibox_cities (načtení číselníku měst) a camibox_countries (načtení číselníku států). Může být volán buď přímo z menu IS, nebo z procesu rl_custommers.php.

Po vložení záznamu do databáze volá proces if_custommers.php další proces, a to vf_custommers.php. Ten se stará o zobrazování detailů uložených dat a umožňuje editovat vybrané atributy jednotlivých záznamů o zákaznících. Provádí operace SELECT s entitami camibox_cities (dotaz na město, kde daná firma sídlí), camibox_countries (dotaz na stát, ve kterém firma operuje), camibox_custommers (dotaz na detaily o zobrazované firmě) a camibox_users_entity_roles (dotaz, zda-li uživatel, který si žádá o zobrazení informací opravdu vložil tento záznam do systému). Dále provádí operaci UPDATE pro entitu camibox_custommers (uložení změn, které uživatel u daného zákazníka provedl).

Další proces v tomto je qf_custommers.php. Jeho úkolem je filtrace dat a to tak, aby se nám zobrazily požadované informace v přiměřeném množství záznamů. Provádí pouze operace SELECT do tabulek camibox_countries (dotaz na seznam všech států), camibox_cities (dotaz na seznam všech měst), camibox_states (dotaz na stavy, které jsou k dispozici pro záznamy jednotlivých zákazníků) a camibox_users_entit_roles (omezení zobrazovaných zákazníků jen na ty, které uživatel do systému sám vložil).

Posledním procesem je rl_custommers.php. Je volán vždy nadřazeným procesem qf_custommers.php. Jeho výstupem je tabulkový seznam požadovaných záznamů. Provádí pouze operace SELECT (dotaz na požadované informace) s tabulkami camibox_states, camibox_custommers a camibox_users_entity_roles.

5.3.4 DFD – modul unit

Modul unit (jehož DFD diagram je umístěn v příloze B.4) obsahuje procesy pro vkládání, editace, náhledy a konfigurace jednotek. Základem tohoto modulu je proces if_unit.php, který se stará o vkládání nových dat do tabulek. Provádí operace INSERT (vkládání dat do databáze) s entitami camibox_units (vkládání nových jednotek), camibox_units_lans (vkládání MAC adres a dalších informací o LAN portech definované jednotky) a camibox_units_wlans (vkládání MAC adres a dalších informací o WLAN portech definované jednotky). Dále provádí operace SELECT (vyžádání si dat ze souvisejících tabulek) s entitami camibox_units_types (zobrazení číselníku všech

typů jednotek, které jsou v systému k dispozici). Může být volán buď přímo z menu IS, nebo z dalšího procesu rl_unit.php.

Po vložení záznamu do databáze volá proces if_unit.php další proces, a to vf_unit.php. Ten se stará o zobrazování detailů uložených jednotek a umožňuje editovat vybrané atributy. Provádí operaci SELECT (dotaz na požadovaná data) entit camibox_units_wlans (dotaz na MAC adresy jednotlivých WLAN portů), camibox_units_lans (dotaz na MAC adresy jednotlivých LAN portů), camibox_units (dotaz na detailní informace o dané jednotce) a camibox_units_types (dotaz na číselník všech definovaných typů jednotek). Dále provádí operaci UPDATE (aktualizace modifikovaných dat) entit camibox_units_lans (aktualizace informací o LAN portech), camibox_units_wlans (aktualizace informací o WLAN portech) a camibox_units (aktualizace informací o vlastní jednotce) .

Další proces, který je v tomto modulu k dispozici je qf_unit.php. Je volán vždy, když žádáme o přístup k procesu rl_unit.php (například z procesu vf_unit.php). Jeho úkolem je filtrace dat a to tak, aby se nám zobrazily pouze požadované informace v přiměřeném množství záznamů. Provádí pouze operace SELECT (dotaz na požadované číselníky) entit camibox_units_types (číselník všech používaných typů jednotek), camibox_units_types_master (číselník hlavních typů jednotek) a camibox_states (seznam všech stavů, do kterých se mohou záznamy o jednotkách dostat).

Dalším procesem je rl_unit.php. Je volán vždy nadřazeným procesem qf_unit.php a jeho výstupem je seznam požadovaných záznamů. Provádí pouze operace SELECT (dotaz na požadované informace) pro entity camibox_states, camibox_units a camibox_units_types.

Poslední proces, definovaný v tomto modulu je config_unit.php. Ten umožňuje konfiguraci jednotek podle navržené zakázky v modulu contract. Je možné ho spustit z procesu rl_unit.php. Provádí příkazy SELECT na entity camibox_config_countries (zjištění povolených frekvencí v daném státě), camibox_units_contracts (zjištění jednotek, přiřazených k dané zakázce), camibox_contract (získání detailů o dané zakázce), camibox_scripts (zjištění skriptu, který má být použit ke konfiguraci), camibox_units_wlans (zjištění počtu WLAN portů vybrané jednotky a jejich MAC adres), camibox_units_lans (zjištění počtu LAN portů vybrané jednotky a jejich MAC adres) a camibox_units (zjištění detailů k vybrané jednotce). Proces provádí pouze jeden příkaz UPDATE pro tabulku camibox_units (změna stavu dané jednotky).

5.4 HLAVNÍ FUNKCE A MODULY APLIKACE

V aplikaci byly naprogramovány čtyři hlavní moduly – admin, custommers, contract a unit. Základem všech modulů jsou skripty s prefixem if_ (InsertForm), které se starají o vkládání nových dat, s prefixem vf_ (ViewForm), které se starají o zobrazování uložených dat a jejich modifikaci a s prefixem rl_ (RecordList), které se starají o zobrazení určité skupiny souvisejících záznamů. Skripty se však svým provedením v jednotlivých modulech liší (viz. kapitoly 5.4.4-5.4.8).

Dále byly vyvinuty dvě třídy, a to třída camiboxDb, která se stará o obsluhu funkcí, vyžadující komunikaci s databází a camiboxForm, která se stará o obsluhu, funkčnost a kontrolu všech formulářových prvků. Dále byla použita třída mikrotikAPI, pomocí které je možné navázat vzdálené spojení s deskami RouterBOARD (zde s jednotkou Camibox).

Při vývoji aplikace byl kladen důraz na zabezpečení jednotlivých modulů, a to nejen z hlediska bezpečnosti, ale i z hlediska přístupu k jednotlivým informacím. Oprávněnost přístupu do modulů je kontrolována na začátku každého skriptu. Navíc zde existují i mechanismy, které zabraňují definovaným uživatelům nahlížet či modifikovat data jiných uživatelů.

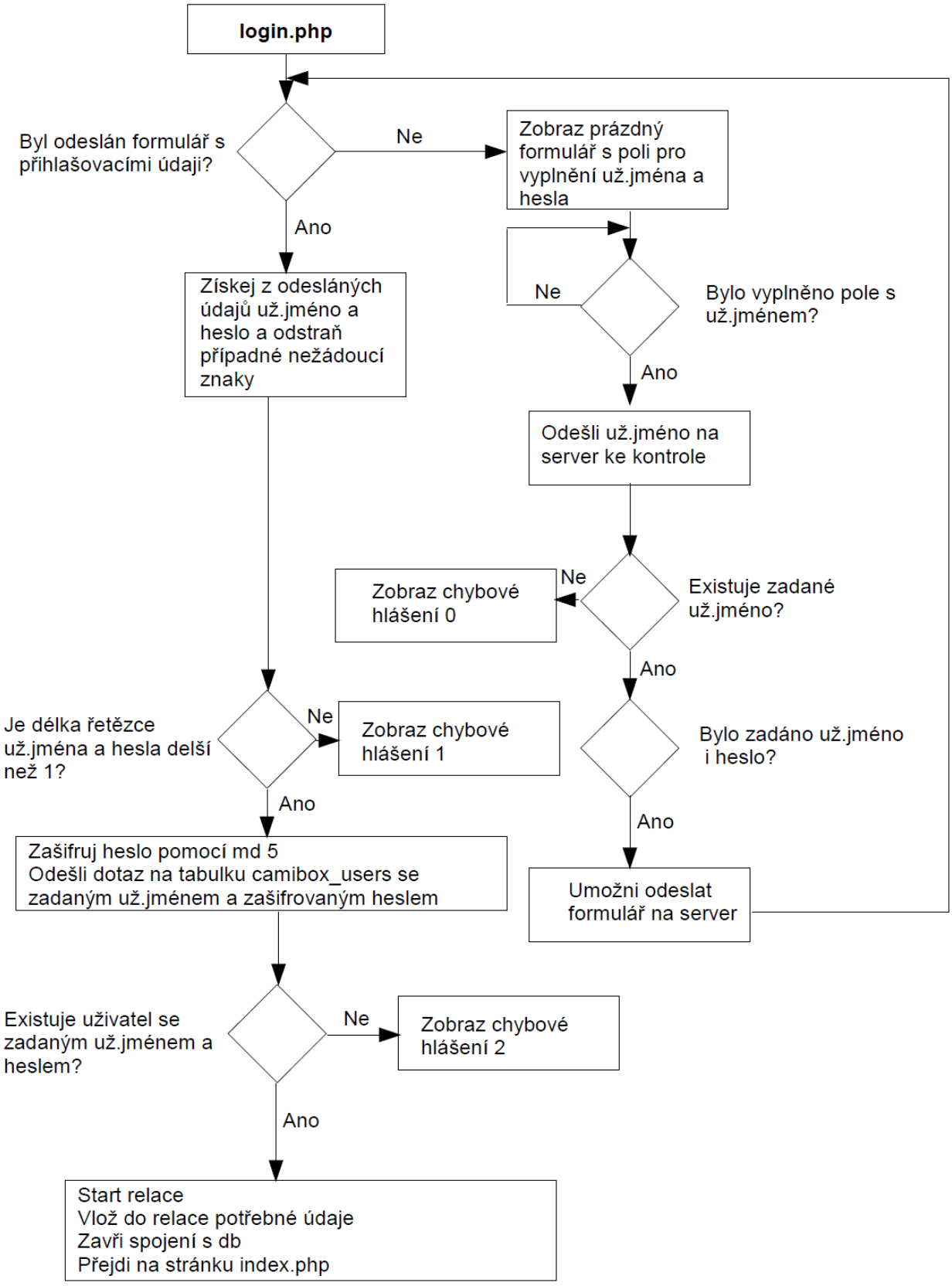
5.4.1 Zabezpečení a autentizace uživatelů

Autentizace uživatelů je uskutečněna ve skriptu login.php. Uživatel je identifikován unikátním uživatelským jménem a svým heslem (heslo je zašifrováno pomocí funkce md5), pomocí nichž je proveden dotaz na databázi a je zjištěno, zda má daný uživatel právo přístupu do IS.

Systém po zadání uživatelského jména ihned zkontroluje, zda-li daný uživatel v systému existuje. Pokud zadané uživatelské jméno není v tabulce uživatelů nalezeno, je ještě před vyplněním hesla a odesláním formuláře upozorněn, že daný uživatel neexistuje.

Pokud uživatel nemá právo přístupu do IS, nebo útočník zkouší hádat heslo ke zjištěnému uživatelskému jménu, zobrazí se mu hláška o špatně zadaných údajích. Pokud jsou zadané údaje správné, tak systém nastaví údaje do Session a pustí uživatele dále. Do relace jsou uloženy údaje o uživateli a údaje dále potřebné v systému (např. id_uživatele, jméno a příjmení a také čas, po kterou bude daná relace platná).

V každém modulu, ještě před tím, než je cokoli odesláno, je kontrolována platnost relace. Pokud její platnost již vypršela, je uživatel přesměrován zpět na stránku login.php, kde se může znovu přihlásit a založit novou relaci. Každý modul, který v IS existuje, je zabezpečen proti neoprávněnému přístupu. Pomocí funkce check_rights() je poslán dotaz na databázi s id uživatele a je kontrolováno, zda-li má na tento modul uživatel právo. Pokud ano, modul se mu zobrazí, pokud ne, zobrazí se mu jen hlavička stránky a chybové hlášení o tom, že na daný modul nemá právo.



Obrázek 5.5 Vývojový diagram funkce pro autentizaci a relaci

5.4.2 Třída camiboxDb

Tato třída definuje všechny funkce, které nějakým způsobem komunikují s databází. Třída pracuje s rozhraním mysqli, které je objektově orientované a spolupracuje s PHP od verze 5. Konstruktor této třídy provede připojení k databázi pomocí definovaného uživatelského jména a hesla, v případě neúspěchu vypíše hlášení o neúspěšném pokusu o připojení. Destruktor této třídy ukončuje spojení s databází.

Funkce publish_array_of_object($sql)

Funkce provádí dotaz na databázi a vrací pole hodnot podle příkazu SQL. Parametr string $sql nese celý SQL příkaz, který se má provést. Funkce vrací pole s výsledkem dotazu. V případě chyby vypíše varování a vrací false. Nejprve provede dotaz na databázi. Pokud je dotaz v pořádku a vrátí nějaká data podle zadaného příkazu, uloží je do proměnné $result. Pokud ne, vypíše chybu a vrací false. Dále testuje počet řádků ve výsledku. Pokud je v proměnné nějaký řádek, tak do pole $result_array ukládá záznamy jako objekty až do doby, než ve výsledku není žádný záznam. K záznamům pak můžeme přistupovat pomocí příkazu $result_array->nazev_sloupce.

Funkce view_menu($id_user)

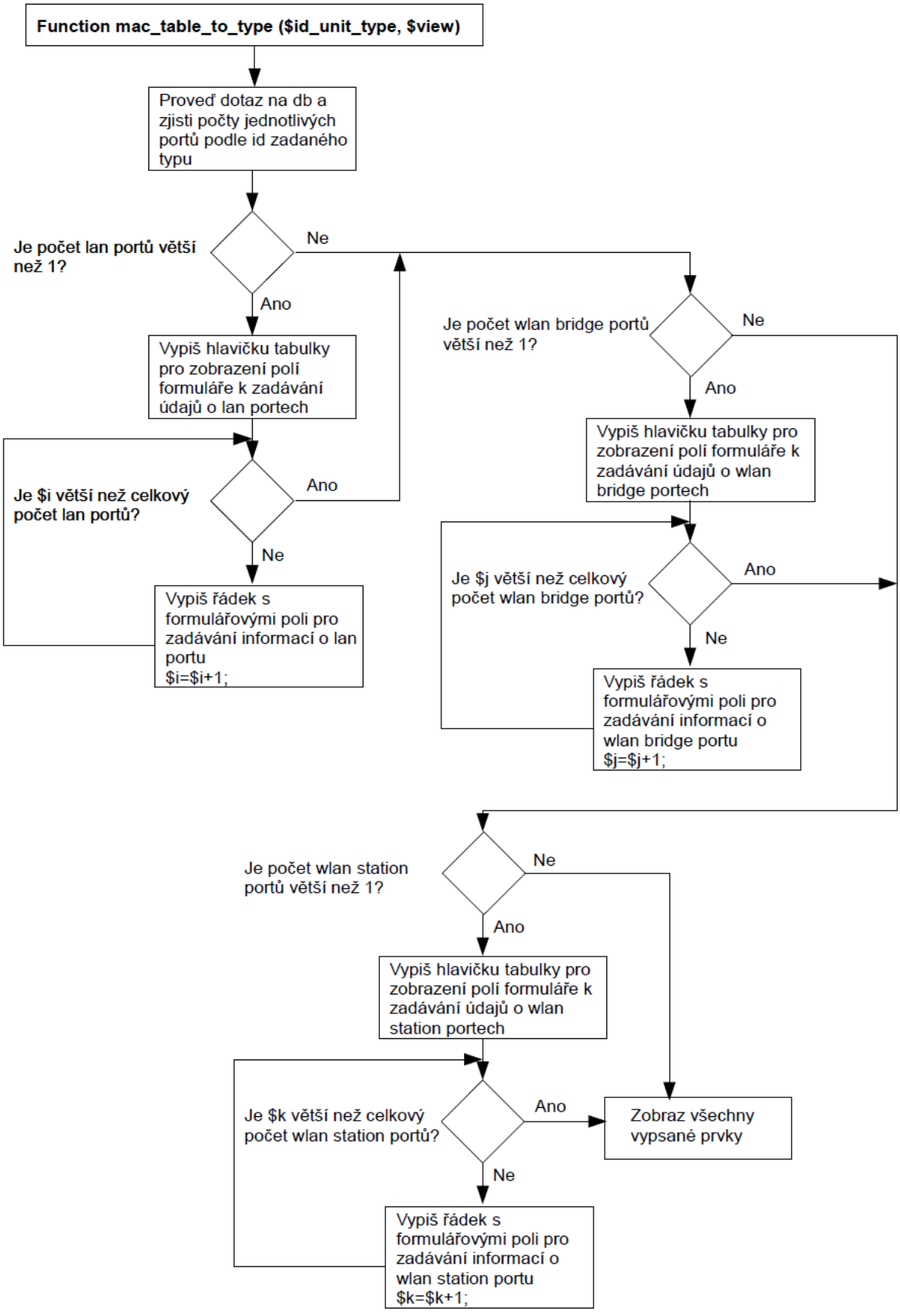
Funkce vytváří strukturu menu podle toho, na které moduly má daný uživatel právo. Funkce si nejprve ověří podle vstupního parametru int $id_user, na které role má uživatel právo. Poté zjistí počet položek v hlavní struktuře menu a vypíše první položku. V cyklu pak znovu ověřuje, na které moduly má uživatel právo a vypisuje je do druhé úrovně menu. K položkám přidává odkazy na metody JavaScript, volané pomocí parametru „onclick“, zabezpečující dynamické reakce celého menu.

Funkce check_rights($modul, $id_user)

Funkce je volána na začátku každého spouštěného skriptu a provádí kontrolu práv na daný modul. V případě, že uživatel nemá na modul oprávnění, tak mu zobrazí chybové hlášení. Parametr string $modul nese celý název modulu. Parametr int $id_user nese informaci o id_uzivatele pro kontrolu oprávnění. Funkce provede dotaz na tabulku camibox_roles_modules a camibox_rights s parametrem názvu modulu a id uživatele, předaných funkci. Pokud funkce vrátí nějaký záznam, je vše v pořádku. Jinak vrací upozornění o chybějícím oprávnění.

Funkce mac_table_to_type($id_unit_type, $view)

Funkce vytváří tabulku s formulářovými poli k vyplnění MAC adres a dalších doplňujících údajů. Funkce si podle parametru int $id_unit_type uloží počty portů jednotlivých rozhraní (WLAN client, WLAN bridge a LAN). Poté v cyklu vypisuje jednotlivé řádky, jimž přiřazuje unikátní identifikátory podle aktuálního pořadí portu.



Obrázek 5.6 Vývojový diagram funkce mac_table_to_type

Funkce check_rights_result($modul, $id_user, $action, $path, $sql)

Funkce, která provede dotaz na databázi a vrátí řádek tabulky do výpisu, kde je odkaz s textem podle parametru. Pokud uživatel na daný modul nemá právo, bude odkaz neaktivní. Parametr string $modul určuje, na který modul je odkazováno. Parametr int $id_user je id aktuálního uživatele. Parametr string $path je aktuální cesta k souboru. Parametr string $sql je SQL příkaz pro výpis aktuálního počtu záznamu, ke kterému se pod odkazem dostaneme. Parametr int $action - pokud je 0, vypíše se odkaz s názvem zobrazit, pokud 1 vypíše se odkaz s id, pokud 2 vypíše se odkaz s textem smazat, pokud 3 vypíše se vyřadit, pokud 4 tak se vypíše aktivovat, pokud 5 tak se vypíše deaktivovat, a pokud 6 tak blokovat.

Funkce view_port_unit($id_type)

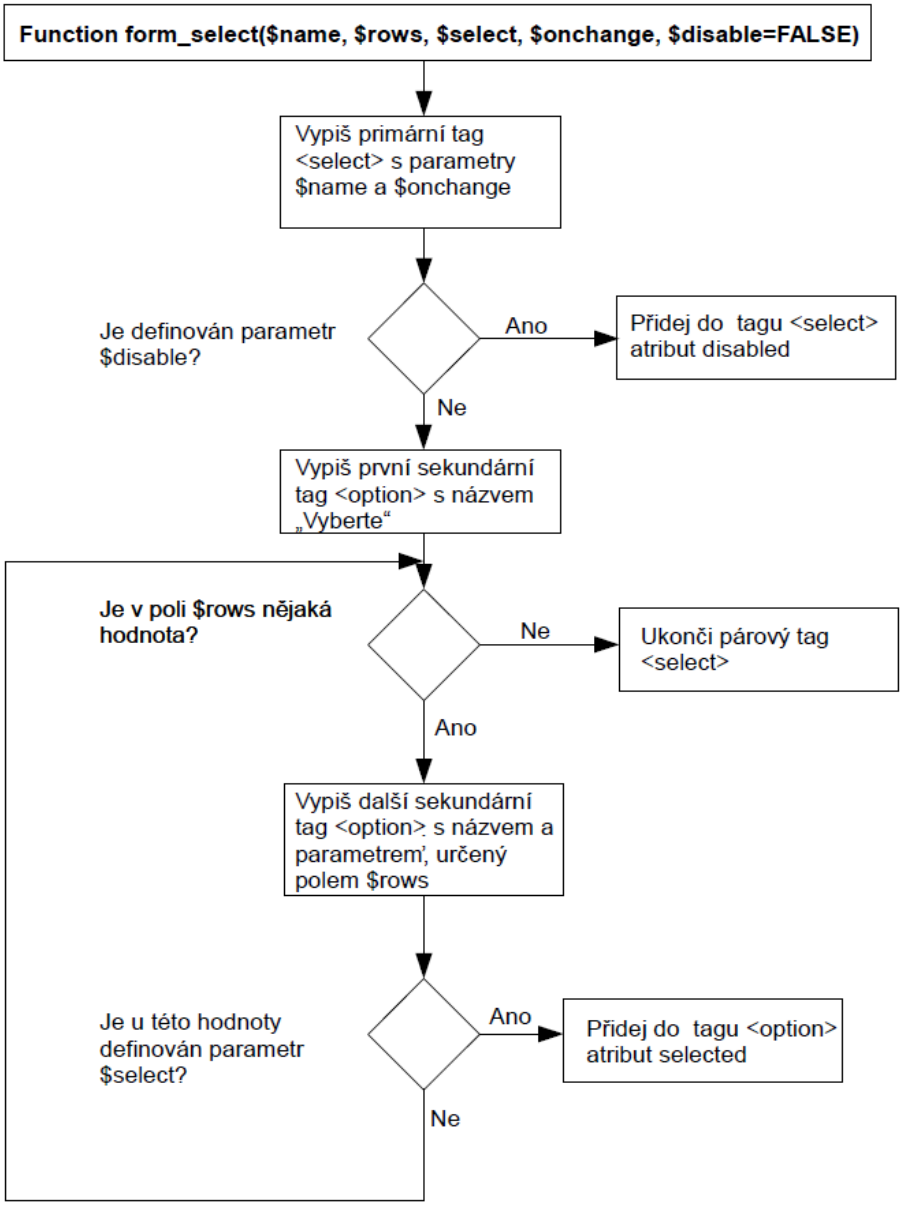
Funkce vypisuje počty jednotlivých LAN, WLAN – bridge a WLAN - station portů zadaného typu jednotky, jehož id je vstupním parametrem této funkce. Parametr int $id_type určuje id typu jednotky, o který máme zájem. Funkce provede dotaz na tabulku camibox_units_types s podmínkou, kde id záznamu se rovná vstupnímu parametru této funkce.

5.4.3 Třída camiboxForm

Tato třída definuje všechny funkce, které se používají pro práci s formuláři a jejich prvky. Konstruktor této třídy zabezpečuje základní funkce, nutné pro správné fungování formuláře, jako například definování metody přenosu dat.

Funkce form_select($name, $rows, $select, $onchange, $disable=FALSE)

Funkce, která vloží do formuláře seznam hodnot v rolovacím seznamu (prvek select). Parametr string $name určuje název pole, parametr array $rows nese hodnoty, které budou zobrazeny v seznamu, parametr string $select určuje vybranou hodnotu, parametr bool $disable určuje, zda bude seznam skrytý nebo editovatelný, parametr string $onchange určuje název skriptu, který je volán v případě vybrání hodnoty uživatelem. Funkce nejprve vypíše základní tag selectu s informacemi o názvu, akci při změně a viditelnosti. Poté prochází pole s hodnotami selectu a zobrazuje podřízený tag option s názvem řádku seznamu a jeho hodnotou. Nakonec funkce uzavře nadřízený párový tag select.



Obrázek 5.7 Vývojový diagram funkce form_select

Funkce filter($query, $arg)

Funkce pro filtraci pomocí více podmínek. Parametr array $dotaz nese hodnoty testovaných záznamů, parametr array $arg nese názvy sloupců, které budou v tabulce podmínkovány. Funkce nejprve spočítá počet záznamů v poli $query. Poté v cyklu dává dohromady celou podmínku a to tak, že nejprve vloží podmínku s klíčovým slovem WHERE, v dalších cyklech pak přidává podmínky pomocí klíčového slova AND. Funkce vrací string, ve kterém je celá podmínka filtru pro SQL dotaz

Funkce validate_data($formdata, $value, $text)

Funkce, která provede kontrolu údajů v odeslaném formuláři, které jsou přijaté polem $formdata a v případě, že některá položka není vyplněna, tak zavolá funkci error report. Parametr array $formdata nese celé pole testovaných hodnot, parametr array

$value nese názvy klíčů pole $formdata a parametr $text nese případnou chybovou hlášku, zobrazovanou při zjištění nevyplněného pole. Funkce nejprve spočte počty prvků v poli $formdata a $value, čímž zkontroluje správný počet prvků ve vstupních polích. Poté v cyklu kontroluje obsah jednotlivých prvků v poli $formdata. Pokud nějaký prvek není vyplněný, vypíše chybovou hlášku a vrací false.

Funkce insert_data($formdata, $entity, $value, $column)

Funkce, která zabezpečuje vkládání dat do databáze podle hodnot, zadaných ve formuláři. Parametr array $formdata je pole všech hodnot, které byly formulářem odeslány, parametr string $entity určuje tabulku, do které se mají uvedená data vložit, parametr array $value nese názvy klíčů pole $formdata a parametr arry $column názvy atributů tabulky, do které se mají údaje vkládat. Funkce nejprve spočítá počet prvků v polích $formdata a $value, čímž si ověří správnost vstupních parametrů. Poté v cyklu prochází všechny atributy pole $column a sestavuje z nich část příkazu INSERT. V dalším cyklu prochází všechny zadané hodnoty, odeslané formuláři a sestavuje druhou část příkazu. Nakonec je celý příkaz pomocí funkce query z třídy camiboxDb odeslán na server. Pokud bylo vložení dat úspěšné, vrátí funkce true, jinak vrací false.

Funkce view_data($sql, $value, $column)

Funkce umožňuje prohlížení již uložených dat. Parametr string $sql nese celý SQL příkaz SELECT, pomocí kterého vybíráme hodnoty, o které máme zájem, parametr array $value nese pole názvů klíčů dalšího pole $formdata, parametr array $column názvy atributů, které chceme z dané tabulky zobrazit. Funkce nejprve spočítá počet prvků v polích $value a $column, čímž si zjistí, zda si odpovídají velikosti polí, a tudíž jestli jsou vstupní parametry správně zadány. Poté provede pomocí funkce publish_array_of_object z třídy camiboxDb dotaz $sql na databázi a prvky, které funkce vrátí uloží do pole. Poté toto pole prochází a přiřazuje jednotlivým prvkům v poli $formdata odpovídající hodnoty, vrácené databází. Funkce vrací pole hodnot $formdata.

Funkce save_data($formdata, $entity, $value, $column)

Funkce, která umožňuje uložit modifikovaná data z formuláře. Parametr array $formdata nese pole hodnot z formuláře, parametr string $entity název tabulky, ve které se mají uvedené údaje aktualizovat, parametr $value, který nese názvy klíčů pole $formdata a parametr array $column, který nese názvy atributů tabulky, ve kterých se mají poslaná data aktualizovat. Funkce nejprve porovná počty prvků v polích $value a $column, čímž zjistí, zda je počet vstupních parametrů správný. Poté v cyklu sestavuje výsledný SQL příkaz pomocí zadaných hodnot z parametrů $column, $value a $formdata. Nakonec sestaví celý SQL příkaz UPDATE a pomocí funkce query třídy camiboxDb odešle požadavek na databázi. Pokud žádaná aktualizace proběhla, vrací true, jinak vrací false.

Funkce function link_button ($modul, $id_user, $param, $txt, $link)

Funkce pro zobrazení tlačítka, které bude odkazovat na jiný modul včetně ověření, zda na něj má daný uživatel právo. Pokud ne tak se mu tlačítko vůbec nezobrazí. Parametr string $modul určuje kompletní název modulu pro ověření přístupu, parametr int $id_user nese hodnotu o tom, který uživatel chce tlačítko zobrazit a taktéž k ověření práv, parametr int $param určuje v případě více možných vzhledů tlačítek o jaké tlačítko půjde, parametr string $txt nese hodnotu textu, který bude zobrazen v tlačítku a parametr string $link určuje aktuální cestu k odkazovanému modulu. Funkce nejprve provede dotaz na databázi, zda-li má uživatel se zadaným id právo na daný modul. Pokud ano, tak vytvoří tag input type=“button“ pro tlačítko a přiřadí mu parametry, které byly funkci předány.

5.4.4 Třída mikrotik API

Protože bylo nutné v implementovaném Informačním systému vytvořit komunikaci mezi aplikací a deskou RouterBOARD, využil jsem třídu mikrotikAPI, napsanou v jazyku PHP, kterou firma Mikrotik nabízí zdarma na svých stránkách.

Třída ve své hlavičce definuje statické proměnné, které jsou využity při samotné komunikaci. Jde o proměnnou $attempts, která určuje počet pokusů o připojení, proměnnou $delay, která určuje zpoždění jednotlivých pokusů o připojení v sekundách, proměnnou $port která určuje číslo portu, na kterém naslouchá Mikrotik API (standartně port 8728) a proměnnou $timeout, která určuje čas v sekundách při pokusu o připojení nebo čtení hodnot, kdy bude vyhlášeno překročení časového limitu, kdy jednotka neodpověděla.

function connect($ip, $login, $password)

Funkce, která zabezpečuje vlastní připojení k API rozhraní OS Mikrotik. .Parametr string $ip nese celou IP adresu vzdáleného zařízení, parametr $login uživatelské jméno a parametr $password heslo pro autentizaci připojení na vzdálené jednotce. Funkce se nejprve pomocí metody fsocketopen() s parametry zadané IP adresy a čísla portu pokusí otevřít komunikační socket. Pokud se mu to podaří, posílá pomocí funkce write() příkaz /login a čeká na odpověď pomocí funkce read(). Pokud mu vzdálený systém nepošle zpátky odpověď „done“, znamená to, že vyžaduje autentizaci pomocí uživatelského jména a hesla. Funkce tedy pomocí write() pošle systému zadané už.jméno z parametru funkce a heslo, které před tím ještě zašifruje algoritmem md5. Pokud se spojení povedlo (tzn. odpověď od systému je „done“), tak je nastaven příznak connected na true. Tento celý proces je v cyklu, jehož počet odpovídá maximálnímu počtu pokusů o připojení.

function read($parse)

Funkce slouží k přijímání odpovědí od vzdáleného připojeného systému. Má pouze jeden parametr bool $parse, který určuje, zda-li se má odpověď rozdělovat, nebo má být poslána jako jeden textový řetězec. Funkce nejprve přečte začátek dat, které chce přijmout. Pokud je nastaven první přijímaný bit, provede bitový posun o čtyři bity doleva a znovu čte i pro druhý a třetí kus zprávy. Poté pokračuje na další byte. Pokud je délka přijímaných dat větší než nula, tak čte v nekonečné smyčce data, posílaná přes socket a ukládá je do pole $RESPONSE. Dokud vzdálený systém nepošle „done“, provádí funkce cyklické čtení. Po přečtení všech dat, za předpokladu že je nastaven parametr $parse, provede rozdělení odpovědi. Funkce vrací pole $response, ve kterém je celá přijmutá odpověď.

function write($command, $param2)

Funkce, které slouží pro zápis příkazů do vzdáleného systému. Parametr string $command, nese celý příkaz, který se má provést, parametr bool/int $param2 typ příkazu. Funkce nejprve zkontroluje zadaný příkaz, který rozdělí do pole $data po jednotlivých řádcích. Poté z každého záznamu v poli odstraní případné mezery a pomocí funkce fwrite() postupně pošle každý příkaz vzdálenému systému. Pří úspěšném poslání všech příkazů vrací true, jinak vrací false.

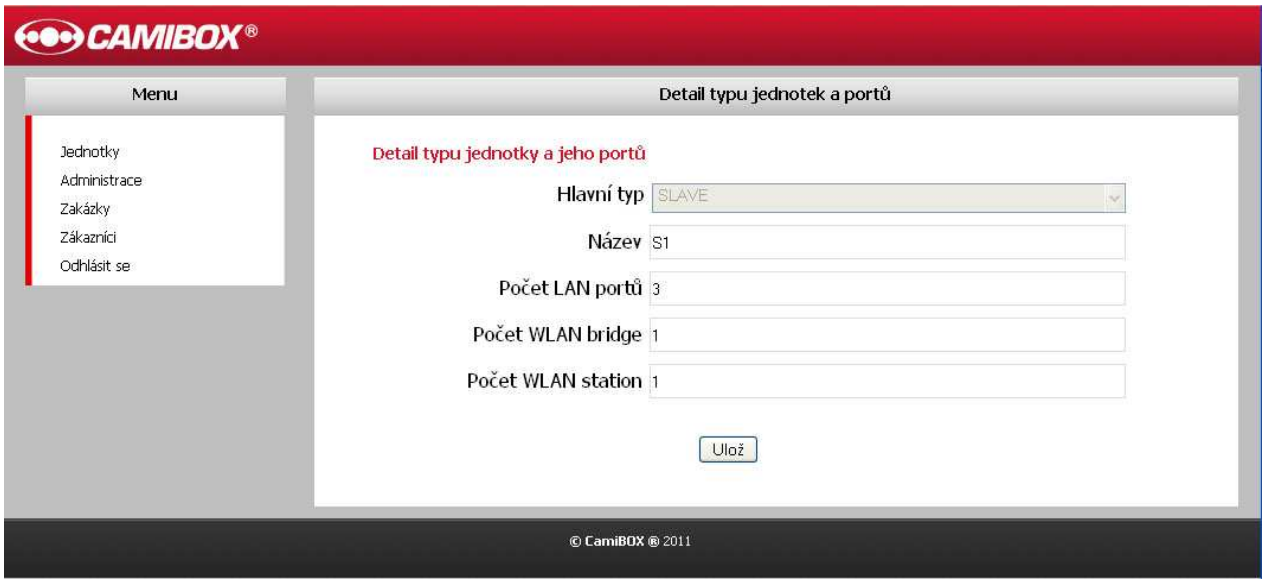
5.4.5 Modul admin

Modul admin je určen pro administrátora aplikace, kterému poskytuje administrační rozhraní. To mu umožňuje možnost úpravy jednotlivých číselníků a parametrů systému. Modul umožňuje vkládat, editovat a mazat záznamy v tabulkách camibox_antennas, camibox_bands, camibox_cities, camibox_config_contries, camibox_rights, camibox_roles_modules, camibox_scripts, camibox_lans_types a camibox_units_types_master. Každý skript v modulu je chráněn pomocí funkce checklogin() před neoprávněným přístupem. Systém umožňuje přidělit správu číselníků i jiným rolím, než jen administrátorovi (např. roli obchodního zástupce pro případ, kdy by chtěl do systému vložit zákazníka ze státu, který ještě není v číselníku států definován). Všechny skripty v modulu admin využívají funkce tříd camiboxDb a camiboxForm. Dále využívají skripty menu.js pro dynamické funkce menu a java_fc.js pro všechny dynamické kontroly vyplňovaných formulářů.

Základním skriptem v tomto systému je if_admin.php. Umožňuje vkládat nové záznamy do tabulek, uvedených v předchozím odstavci. Využívá funkce třídy camiboxForm k zobrazení polí formuláře, které potřebujeme k získání dat od uživatele. Pomocí funkce form_label() zobrazuje popisky jednotlivých formulářových prvků, pomocí funkce form_text_input() zobrazuje jednořádkové pole pro zadávání textu a pomocí funkce form_select() zobrazuje výběrový seznam prvků podle zadaných parametrů. Po odeslání formuláře na server pomocí funkce validate_data() ověří všechny povinné položky. Pokud nějaká položka nebyla vyplněna, zobrazí chybové

hlášení, jinak volá funkci insert_data(), která vloží příslušné záznamy do databáze. Pokud bylo vložení údajů do tabulky úspěšné, přesměruje se stránka pomocí modifikace hlavičky souboru na stránku vf_admin.php, která uložená data zobrazí.

Druhý skript v tomto modulu je již zmíněný vf_admin.php. Ten se stará o zobrazování uložených informací z administračních tabulek a také o ukládání případných modifikovaných dat. Skript využívá tytéž funkce z třídy camiboxForm, jako if_admin.php, tj. form_label(), form_text_input() a form_select(). Při volání skriptu je nejprve zkontrolována a ošetřena hodnota vstupního parametru tak, aby nemohlo dojít k tzv. útoku SQL INJECTION, který spočívá v neoprávněné modifikaci SQL příkazu. Poté je zavolána funkce view_data(), která přiřadí příslušným formulářovým prvkům jejich hodnoty, které pak zobrazí. Pokud jsou data modifikována, tj. skript vf_admin.php volá sám sebe, je zavolána funkce validate_data() která zkontroluje, jestli jsou všechna povinná pole vyplněna. Pokud všechna povinná pole vyplněna nejsou, vypíše chybovou hlášku a modifikovaná data neuloží. Pokud vyplněna jsou, zavolá funkci save_data(), která uloží všechna modifikovaná data a poté znovu volá funkci view_data().



Obrázek 5.8 Náhled na obrazovku skriptu vf_admin.php

Používaný skript rl_admin.php umožňuje vytvořit seznam všech souvisejících dat v daném číselníku. Po zavolání tohoto skriptu je vytvořen SQL dotaz pro požadované typy dat a je použita funkce publish_array_object_limit() třídy camiboxDb, která přijme data, zaslaná databázovým serverem a umístí je do pole. Skript pak toto pole postupně prochází a vyžádaná data zobrazuje do tabulky. Z tohoto skriptu je možné záznamy mazat. Pomocí navigace je také možné se dostat na detail vybraného záznamu, popř. vložit nový záznam.

Obrázek 5.9 Náhled na obrazovku skriptu rl_admin.php

5.4.6 Modul custommers

Modul slouží pro vkládání, editaci a přehled zákazníků. Pracuje se záznamy v tabulkách camibox_custommers, camibox_cities, camibox_countries, camibox_states a camibox_users_entity_roles. Každý skript v modulu, stejně jako u modulu admin, je chráněn pomocí funkce checklogin() před neoprávněným přístupem. Tento modul využívají především role obchodního zástupce a jednatele. Všechny skripty v tomto modulu využívají funkce tříd camiboxDb a camiboxForm. Dále využívají skripty menu.js pro dynamické funkce menu a java_fc.js pro všechny dynamické kontroly vyplňovaných formulářů. V tomto modulu je dáván veliký důraz na to, aby se uživatelům zobrazila jen ta data, která do systému sami vložili. Je totiž nežádoucí, aby obchodní zástupci měli přístup k údajům jiných obchodních zástupců. Funkčnost skriptů je v zásadě totožná s těmi, které používá modul admin.

Základním skriptem v tomto modulu je if_custommers.php. Umožňuje vkládat nové záznamy do tabulky camibox_custommers. Využívá funkce třídy camiboxForm k zobrazení polí formuláře, které potřebujeme pro získání dat od uživatele, konkrétně form_label(), form_text_input(), form_select() (funkce jsou totožné s funkcemi, používanými modulem admin viz kapitola 5.4.4). Dále také využívá funkci form_error(), která v případě chybně vyplněného pole zobrazí upozornění právě pod toto pole. Po odeslání formuláře na server se pomocí funkce validate_data() ověří všechny povinné položky, v případě úspěšného ověření je zavolána funkce insert_data(), která vloží příslušné záznamy do tabulky camibox_custommers. Pokud bylo vložení údajů do tabulky úspěšné, je provedeno přesměrování na stránku vf_custommers.php.

Další skript v tomto modulu je vf_custommers.php. Ten se stará o zobrazování uložených informací o zákaznících a také o ukládání modifikovaných dat. Skript využívá tytéž funkce třídy camiboxForm, jako if_custommers.php, tj. form_label(), form_text_input() a form_select(). Při volání této stánky ze skriptů if_custommers.php nebo rl_custommers.php, je nejprve zkontrolována a ošetřena hodnota vstupního parametru. Poté je zavolána funkce view_data(), která příslušná data zobrazí do připraveného formuláře. Pokud jsou data modifikována, tak skript vf_custommers.php volá sám sebe a je zavolána funkce validate_data() která zkontroluje správnost vyplněných polí a zavolá funkci save_data, která všechna modifikovaná data uloží.

Dalším skriptem, který je zde využit je rl_custommers.php, který umožňuje vytvořit tabulkový přehled vložených zákazníků. Po zavolání tohoto skriptu je vytvořen SQL dotaz a je použita funkce publish_array_object_limit() třídy camiboxDb, která přijme data, zaslaná databázovým serverem a uloží je do pole. Skript pak toho pole postupně prochází a data zobrazuje do tabulky.

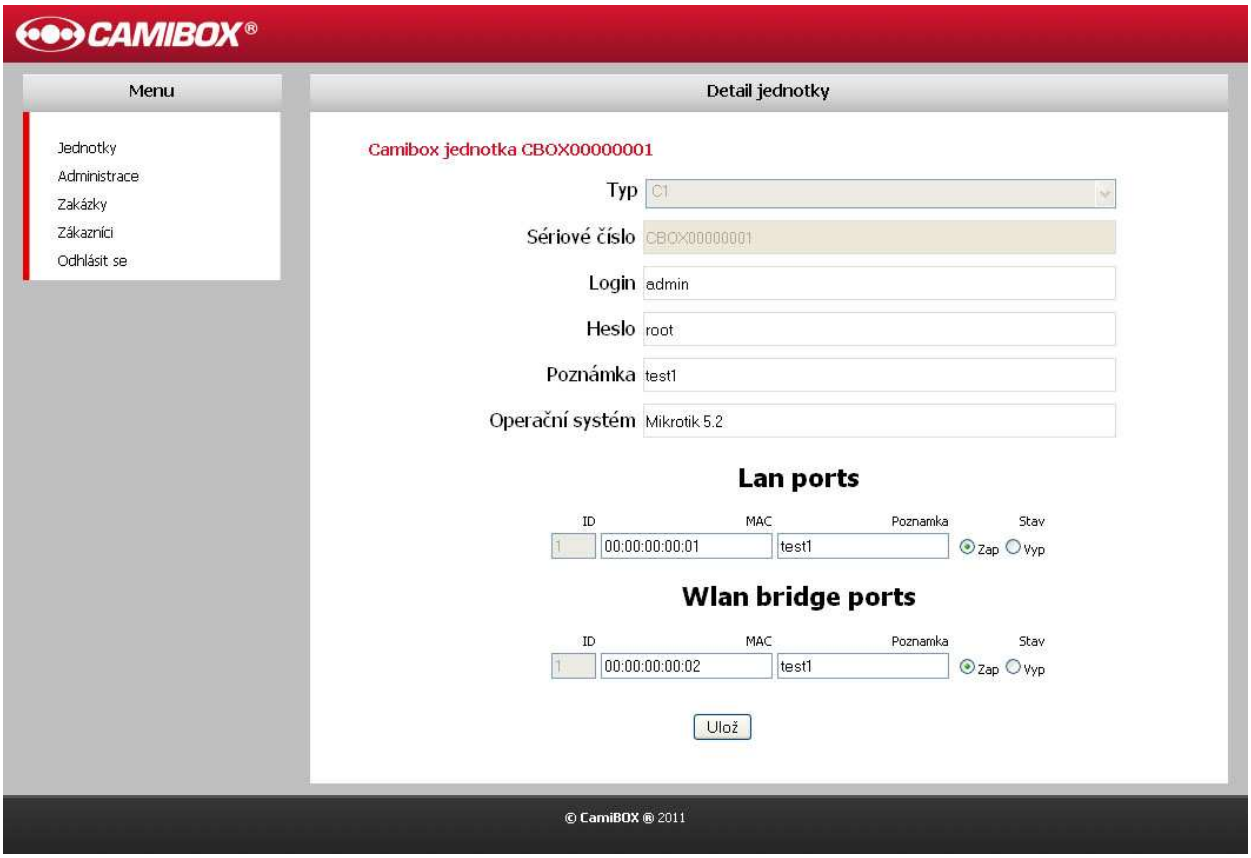
5.4.7 Modul unit

Tento modul se stará o vkládání, editaci, přehled a konfiguraci jednotek. Využívá ho především role technika, který sleduje objednávky jednotlivých jednotek, a postup jejich výroby zaznamenává do systému, takže mají všechny role přehled, kdy asi tam může být daná zakázka vyřízená. Každý skript v modulu, stejně jako u modulu admin a custommers, je chráněn pomocí funkce checklogin() před neoprávněným přístupem. Všechny skripty v tomto modulu využívají funkce tříd camiboxDb a camiboxForm. Dále využívají skripty menu.js pro dynamické funkce menu, java_fc.js pro všechny dynamické kontroly vyplňovaných formulářů a ajax_fc.js pro všechny dynamické reakce, které si pro svojí správnou funkčnost potřebují vyměnit data se serverem. Základní skripty jsou v zásadě totožné s těmi, které používají moduly admin a custommers, proto už zde nebudou tak detailně popisovány.

Stejně tak jako v předchozích modulech (viz. kapitoly 5.4.4 a 5.4.5), i zde je skript if_unit.php, který se stará o vkládání nových dat do systému. Používá funkce form_label(), form_text_input() a form_select() třídy camiboxForm. Na rozdíl od předcházejících modulů jsou zde dva rozdíly. Zaprvé, skript pomocí funkce get_prefix() zkontroluje poslední sériové číslo jednotky, které bylo do systému vloženo a vygeneruje nové sériové číslo, které předvyplní do příslušného formulářového prvku. Uživatel se tak již nemusí starat o to, zda-li je sériové číslo správně vyplněno. Další rozdíl je použití technologie Ajax pro zobrazení tabulky, ve které jsou prvky pro vyplnění detailů jednotlivých portů jednotek. Tato funkce je volána pomocí atributu „onchange“ při vybrání typu jednotky z rozbalovacího seznamu.

Po úspěšném vložení nového záznamu je provedeno přesměrování na skript vf_unit.php, na který se dá dostat i ze skriptu rl_unit.php. Funkčně je tento skript totožný s těmi v předcházejících modulech. Zobrazuje uložené informace a ukládá ty, které byly změněny. Pracuje s tabulkami camibox_units, camibox_units_lans a camibox_units_wlans.

Skript rl_unit.php umožňuje filtrovat jednotky pomocí znalosti typu zařízení, typu jednotky nebo jejího stavu. Přehledně zobrazuje seznam dostupných jednotek včetně informace o jejich stavech.



Obrázek 5.10 Náhled na obrazovku skriptu vf_unit.php

Posledním skriptem, používaným v modulu unit je config_unit.php. Ten umožňuje konfiguraci jednotek podle zakázky, navržené v modulu contract. Při jeho volání se spustí funkce check_unit(), která vrátí číslo zakázky, ve které je daná jednotka přiřazena. Uživateli je zobrazen seznam všech jednotek v zakázce a tlačítko na konfiguraci celé zakázky. Po stisku tohoto tlačítka je zavolána funkce config_unit(), která nejprve vypíše upozornění, ve kterém žádá o připojení jednotky k serveru pomocí UTP kabelu. Po odkliknutí hlášení je zavolána funkce connect() třídy mikrotikAPI a pomocí defaultní IP adresy jednotky se na ní připojí. V případě úspěchu je uživateli vypsáno hlášení o úspěšném připojení a je zavolána funkce make_script(), která upraví skript pro konfiguraci jednotky (popsaný v kapitole 5.5.2) podle aktuální zakázky a aktuálně konfigurované jednotky. Po upravení skriptu je uživateli vypsáno hlášení o začátku konfigurace, a je zavolána funkce write() třídy mikrotikAPI, která nakopíruje konfigurační skript do jednotky. Poté je pomocí této funkce zaslán jednotce příkaz k restartu, po jehož provedení se spustí nakopírovaný konfigurační skript. Funkce dále otestuje dostupnost nakonfigurované jednotky, vypíše uživateli hlášení o úspěšném nakonfigurování a změní stav jednotky. Po konfiguraci všech jednotek změní stav zakázky na nakonfigurovaná.

5.4.8 Modul contract

Modul contract patří v tomto IS k těm nejdůležitějším. Umožňuje vkládat, editovat, zobrazovat a navrhovat zakázky. Jako v předchozích modulech (viz. kapitoly 5.4.4 - 5.4.6), i zde jsou všechny skripty chráněny pomocí funkce checklogin() před neoprávněným přístupem a využívají funkcí tříd camiboxDb, camiboxForm a také skripty menu.js, java_fc.js a ajax_fc.js.

Funkčnost základních skriptů if_contract.php, vf_contract.php a rl_contract.php je totožná s těmi, které jsou v ostatních modulech, proto zde už nebudou popisovány.

Ze skriptu vf_contract.php je možné se dostat do skriptu project_contact.php, který umožňuje grafický návrh celé zakázky. V tomto skriptu je pod menu, nabízejícím navigaci po IS, zobrazeno ještě menu pro grafický návrhový systém (dále GUI). Na začátku tohoto skriptu je kontrola a ošetření vstupního parametru. Poté je volána Ajaxová funkce getDivData(), která pošle id_zakázky skriptu getCountStart.php. Ten provede nejprve dotaz na tabulku camibox_maps a zjistí, zda-li k návrhu nebyl uložen mapový podklad. Pokud ano, zavolá funkci viewMap(), které předá informace o názvu a umístění mapy. Ta pak mapový podklad zobrazí v návrhovém okně. Dále provede dotaz na tabulku camibox_units_contracts a zjistí, zda-li už byly do návrhu vloženy nějaké body. Pokud ne, vrátí prázdnou návrhovou plochu. Pokud ano, vyžádá si informace o typu vložené jednotky a jeho umístění v návrhovém okně a zavolá JavaScriptovou funkci makeDivContract(), které tyto informace předá. Ta vytvoří nový bod v návrhovém okně, kterému přiřadí příslušné id podle aktuálního počtu jednotek v daném návrhu a posune ho do místa, jehož souřadnice byly této funkci předány. Takto cyklicky zobrazuje všechny body, které byly v zakázce již navrženy.

Po této úvodní inicializaci může uživatel pokračovat v návrhu. V menu pro GUI má na výběr z několika položek. Pokud stiskne volbu „Vložit mapový podklad“, je zavolána funkce openWindow(), která otevírá nový skript popup_upload.php. Ten používá funkce pro upload souborů. Zobrazí formulářový prvek, s pomocí kterého můžeme vybrat soubor na lokálním počítači. Po stisku tlačítka „Nahraj“ je volána funkce upload(), která nahraje příslušný soubor na server a zapíše o něm informace do tabulky camibox_maps. Po úspěšném nahrání souboru je volána funkce close(), která nově otevřené okno zavře a funkce changeMaps(), která zobrazí nahraný soubor jako podklad do návrhového okna.

Dále je na výběr položka „Vložit novou jednotku“. Po jejím stisku je volána funkce makeVisible(), která zobrazí okno se seznamem všech typů jednotek, které jsou v systému k dispozici. Po výběru jednotky je pomocí parametru „onchange“ volána Ajaxová funkce get_info_div(). Ta nejprve ukryje zobrazené okno pro výběr jednotky a poté změní kurzor šipky na zaměřovací kříž. Pomocí Ajaxové funkce get_data() zavolá skript getSaveInfo.php, kterému předá informace o vkládané jednotce a čísle zakázky. Ten poté vloží do tabulky camibox_units_contracts_design nový záznam s identifikací dané zakázky, typem vkládané jednotky a pořadovým číslem vkládaného bloku. Dále funkce get_info_div() čeká na stisk tlačítka uvnitř návrhové plochy, kam chceme daný

bod vložit. Po stisku tlačítka volá funkci getPosition(), která pomocí metody event() zjistí aktuální souřadnice polohy kurzoru myši a znovu pomocí funkce get_data() zavolá skript getSaveInfo.php, pomocí kterého se zaktualizuje záznam v tabulce camibox_units_contract_design o pozici vloženého bodu. Pomocí Ajaxové funkce get_java_data() se dále zavolá skript getCount.php, který daný bod zobrazí.

Zobrazené body ilustrují jednotlivé jednotky, umisťované do zakázky. V horní části bodu je tmavý pruh, kde je zobrazen název typu jednotky. Po kliknutí na tento pruh je zavolána funkce findObjectID(), která zjišťuje rodičovský prvek v DOM stromu objektu, na který bylo kliknuto. Poté volá funkci drag(), které předává odkaz na rodičovský prvek a pomocí které je možno s bodem v návrhovém okně hýbat. Po přemístění objektu je vždy zavolán skript getSaveInfo.php, který uloží pozici přemístěného bodu.

V menu GUI jsou dále na výběr položky, které posouvají zakázkou po svém životním cyklu (např. volbou Zkontrolovat zakázku ji může jednatel posunout do stavu zakázka zkontrolována, apod.).

Obrázek 5.11 Náhled na obrazovku GUI project_contract.php

5.5 KONFIGURAČNÍ SKRIPT

Systém umožňuje automatickou konfiguraci jednotek Camibox. Komunikace se vzdáleným systémem je zajištěna pomocí funkcí connect(), read() a write() třídy mikrotikAPI. Byl vytvořen konfigurační skript pro OS Mikrotik, který je pro každou zakázku individuálně upraven pomocí funkce make_script(). Tato funkce komunikuje s tabulkami camibox_config_countries, camibox_contracts, camibox_custommers, camibox_units, camibox_units_contracts, camibox_units_types, camibox_units_lans, camibox_units_wlans a camibox_units_wlans_connections.

Na začátku konfiguračního skriptu je nejprve nastaveno identifikační číslo jednotky příkazem „set-name“. Název jednotky je složen z prefixu CamiBox, id dané zakázky, typ jednotky a nakonec sériové číslo jednotky. Poté je nastaven wpa2 klíč pomocí příkazu „add name=WPA2 authentication-types=wpa2-psk“ pro zabezpečenou komunikaci mezi jednotkami, který nastaví klíč podle údaje z tabulky camibox_contracts.

Dále jsou nastaveny všechna bezdrátová rozhraní pomocí příkazu „set wlan“. Je nastaveno jméno rozhraní (příkaz „name“), podle typu (LAN, WLAN) a pořadového čísla rozhraní, SSID (příkaz „ssid“), které se skládá s prefixu CamiBOX, typu jednotky, sériového čísla a názvu rozhraní, scan-list(příkaz „scan-list“) podle scan-listu z tabulky camibox_config_countries, země, ve které bude jednotka umístěna (příkaz „country“), omezení výkonu bezdrátového rozhraní (příkaz „antena-gain“), frekvenční vysílací pásmo (příkaz „band“) a frekvence, na které bude vysíláno (příkaz frequency). Tyto nastavení se postupně provedou pro všechna bezdrátová rozhraní.

Dále se mezi jednotkami nastaví nstreme (příkaz „interface wireless nstreme“), což je protokol OS Mikrotik, který dokáže lépe využít celý vysílací kanál. Zde nastavíme pro každé bezdrátové rozhraní (příkaz set WLAN) povolení protokolu nstreme (příkaz „enable-nstreme“) a maximální velikost jednotlivých rámců (příkaz „framer-limit“).

Vytvoříme dvě bridgové skupiny (příkaz „interface bridge“), kdy jedna se stará o bridgování dat z kamer (LAN portů) a druhá se stará o bridgování přijímaných a vysílaných dat (WLAN protů) (příkaz „add name“).

Dále nastavíme jména LAN portům (příkazy „interface ethernet“ a „name“). Poté přiřadíme jednotlivé rozhraní do bridgových skupin (příkaz „interface bridge port“). LAN porty do jedné skupiny a WLAN porty do druhé skupiny (příkazy „add bridge“ a „interface“). V systému dále nastavíme buď DHCP server na jednotce master (příkazy „ip pool“, „ip dhcp-server“, „set pool“) nebo DHCP klient na jednotkách client či slave (příkazy „ip dhcp-client“, „interface“ a „host_name“).

Jako poslední nastavíme ntp client pro synchronizaci času (příkaz „systém ntp client“, „primary-ntp“ a „secondary-ntp“).

Poté můžeme tento skript nahrát do připojené jednotky.

6. ZÁVĚR

Cílem této diplomové práce bylo navrhnout a implementovat informační systém s grafickým návrhovým rozhraním pro firmu Patrokolos s.r.o., která se zabývá vývojem a prodejem produktu Camibox, který umožňuje vytvořit bezdrátové datové spoje pro kamerové a zabezpečovací systémy.

V informačním systému byly definovány čtyři uživatelské role – administrátor, obchodní zástupce, technik a jednatel. Pomocí těchto rolí a implementovaných metod umožňuje řízený přístup k jednotlivým položkám a funkcím informačního systému.

Informační systém umožňuje jeho správci pracovat se všemi číselníky, které jsou v systému k dispozici a také nastavovat jeho parametry. Dále umožňuje uživatelům systému pracovat s daty, které byly získány od zákazníků firmy a používat tyto data dále v návrhu zakázek. Oprávněným rolím je umožněno vkládat do systému nové zakázky, editovat je a za pomoci implementovaného návrhového rozhraní i graficky navrhovat. Informační systém je určen i pro výrobní sektor firmy k ukládání informací o jednotlivých zakázkách. Dovoluje technikovi přístup ke konfiguračnímu rozhraní, které zajišťuje sestavení konfiguračních skriptů pro OS Mikrotik podle vybrané zakázky a umožňuje jejich nahrání pomocí lokálního serveru do jednotek Camibox.

Pro všechny používané části programu byly vytvořeny funkce pro zabezpečení proti neoprávněnému přístupu. Všechna vstupní data procházejí ověřovacími funkcemi, čímž je systém zabezpečen proti útoku typu SQL-injection. Byly implementovány funkce, které zabezpečují citlivá data, uložená v databázovém systému proti nahlížení či modifikaci neoprávněnými uživateli.

Pro informační systém byl použit databázový systém MySQL 5 a jeho funkce jsou naprogramovány v jazyce PHP 5. Některé funkce, využívající technologii Ajax, byly naprogramovány v jazyce JavaScript. Díky těmto použitým technologiím, které jsou šířeny zdarma se značně snížila cena vytvořeného informačního systému. Vývoj systému probíhal ve vývojovém prostředí programu Macromedia Dreamweaver 2004.

Informační systém je pro testovací účely dostupný na internetové adrese www.sps-fekt.cz/camibox s uživatelským jménem admin a heslem hronek87. Vytvořený testovací účet poskytuje uživateli všechny role, které jsou v systému dostupné.

Systém splňuje požadavky firmy Patrokolos s.r.o. a v průběhu července roku 2011 bude spuštěn jeho testovací provoz.

7. LITERATURA

[1] DOSTÁLEK, Libor - KABELOVÁ, Alena. *Velký průvodce protokoly TCP/IP a systémem DNS*. 2.akt Brno : Computer Press, 2008. ISBN 978-80-251-2236-5

[2] PRASAD, Nand – PRASAD, Neeli. *802.11 WLANs and IP Networking*.1.vyd.USA : Artech House, 2005. ISBN 1-58053-789-8

[3] KOFLER, Michael. *Mistrovství v MySQL 5. 1*. Překlad Jan Svoboda, Ondřej Báše, Jaroslav Černý. 1.vyd.Brno : Computer Press 2007. ISBN 978-80-251-1502-02

[4] GUTMANS, Andi – BAKKEN, Stig, RETHANS, Derick. *Mistrovství v PHP 5*. Překlad Bohdan Kostka. 2.vyd.Brno : Computer Press 2008. ISBN 978-80-251-1519-0

[5] HOLZNER, Steven. *Mistrovství v AJAXu*. Překlad Jakub Zemánek. 1.vyd.Brno : Computer Press 2007. ISBN 978-80-251-1850-4

[6] CROFT, Jeff – LLOYD, Ian – RUBIN, Dan. *Mistrovství v CSS*. Překlad Josef Bábík. 1.vyd.Brno : Computer Press 2007. ISBN 80-86497-60-7

[7] VOSTROVSKÝ, Václav – MERUNKA, Vojtěch. *Databázové systémy*. Katedra informatiky PEF ČZU 1999.

[8] CONOLLY, Thomas – BEGG, Carolyn – HOLOWCZAK, Richard. *Databáze*. Překlad Vilém Gutfreund. 1.vyd.Brno : Computer Press 2009. ISBN 978-80-251-2328-7

[9] ODELL, Den. *JavaScript*. Překlad Ondřej Baše. 1.vyd.Brno : Computer Press 2010. ISBN 978-80-251-2733-9

[10] ANSLESON, Ryan – SCHUTTA, T.Nathaniel. *Ajax*. Překlad Jakub Zemánek. 1.vyd. Brno : Computer Press 2007. ISBN 80-251-1285-3

[11] TEAGUE, Jason Cranford. *DHTML a CSS*. Překlad Jan Gregor. 1.vyd.Praha : SoftPress 2005. ISBN 80-86497-77-1

[12] *Produktový list produktu Camibox*. České Budějovice : Patrokolos s.r.o. 2011. URL < http://www.camibox.eu/download/CamiBOX_Presentation_CZ.pdf>

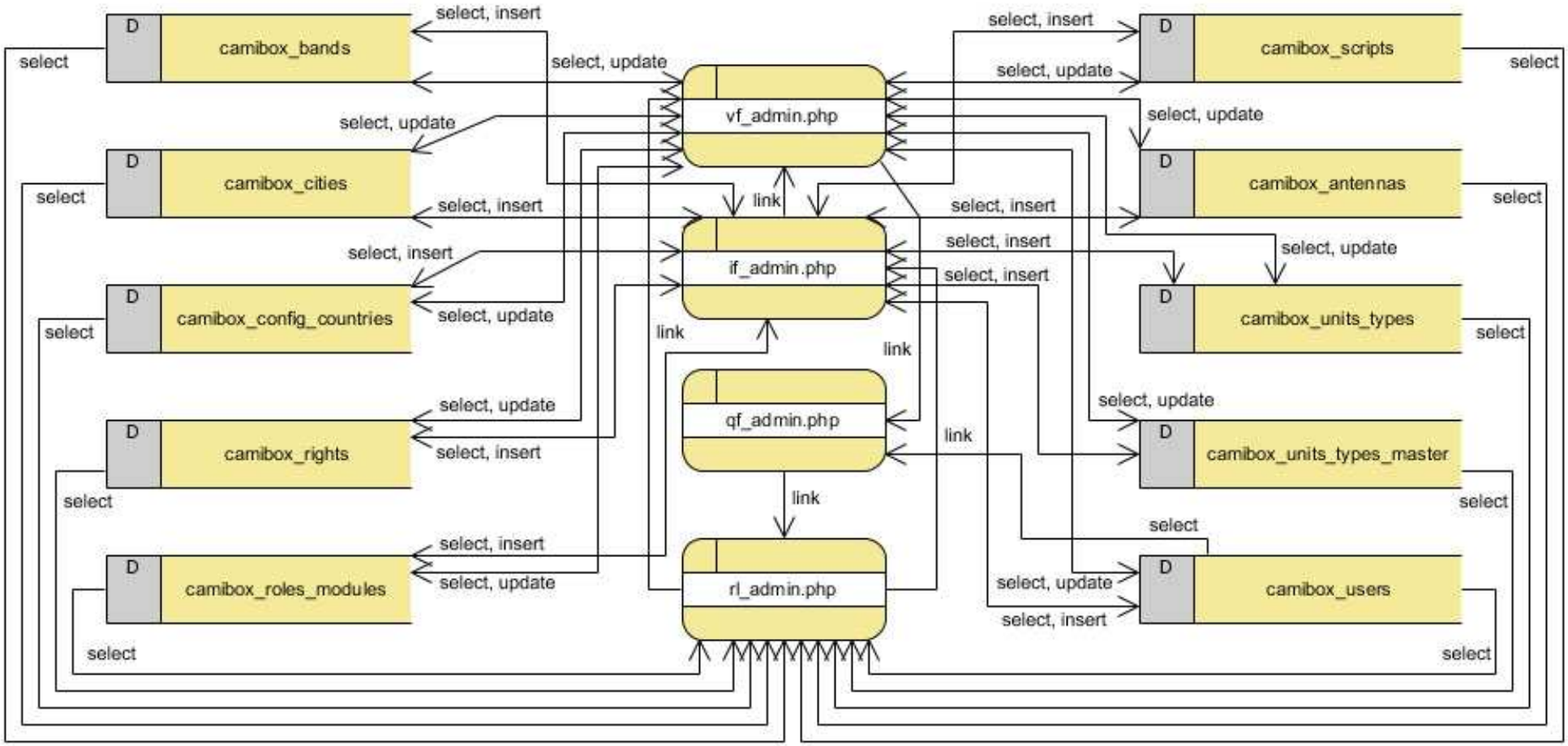
8. SEZNAM ZKRATEK

API Application Proggramming Interface – rozhraní pro programování aplikací
CCTV Closed Circuit Television – uzavřený televizní okruh
CSS Cascading Style Sheets – kaskádové styly
E-R Entity-Relationship – entita-relace
GPL General Public Licence – všeobecná veřejná licence GNU
GUI Graphical User Interface – grafické uživatelské rozhraní
HTML HyperText Markup Language – hypertextový značkovací jazyk
HTTP Hypertext Transfer Protocol – protokol původně určený pro výměnu hypertextových dokumentů
HW Hardware – technické vybavení počítače
IP Internet Protocol - protokol komunikace v počítačové síti
IS Information systém - informační systém
ISP Internet service provider – poskytovatel internet. připojení
LAN Local Area Network – lokální síť
MAC Media Access Control – fyzická adresa síťového zařízení
ODBC Open DataBase Connectivity – systém pro propojení s databází
OS Operating System – operační systém
PHP Hypertext Preprocessor – hypertextový preprocesor
SQL Structured Query Language – strukturovaný jazyk pro dotazy na relační databáze
URL Uniform Resource Locator – doménová adresa serveru
WLAN Wirelesss Local Area Network – bezdrátová lokální síť
XHTML Extensible Hypertext Markup Language – rozšiřitelný značkovací hypertextový jazyk
XML Extensible Markup Language – rozšiřitelný značkovací jazyk

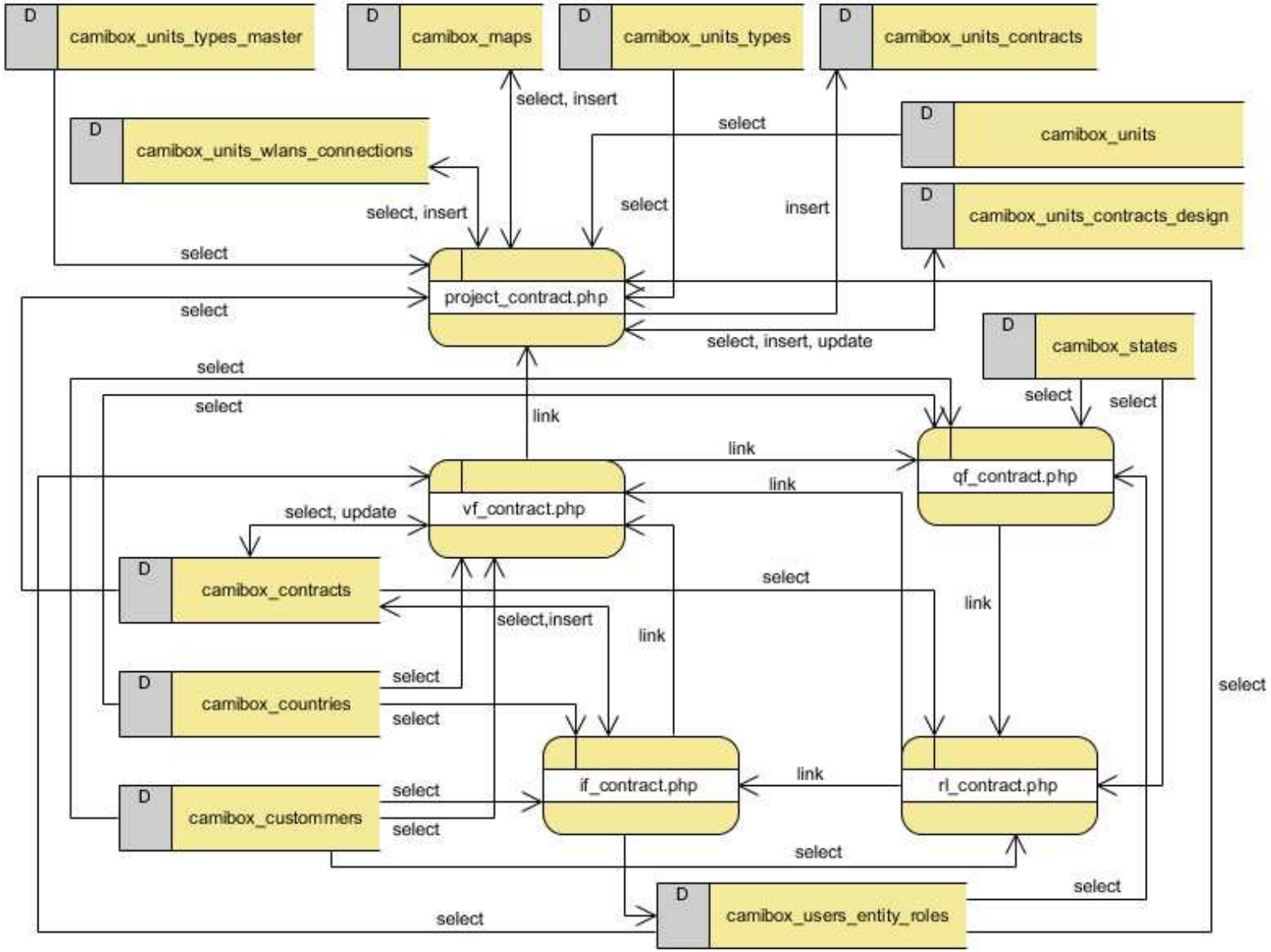
9. SEZNAM PŘÍLOH

Příloha A ER diagram
Příloha B DFD diagramy
Příloha B.1 DFD diagram – modul admin
Příloha B.2 DFD diagram – modul contract
Příloha B.3 DFD diagram – modul custommers
Příloha B.4 DFD diagram – modul unit
Příloha C CD-ROM se zdrojovými kódy a vypracovanou diplomovou prací

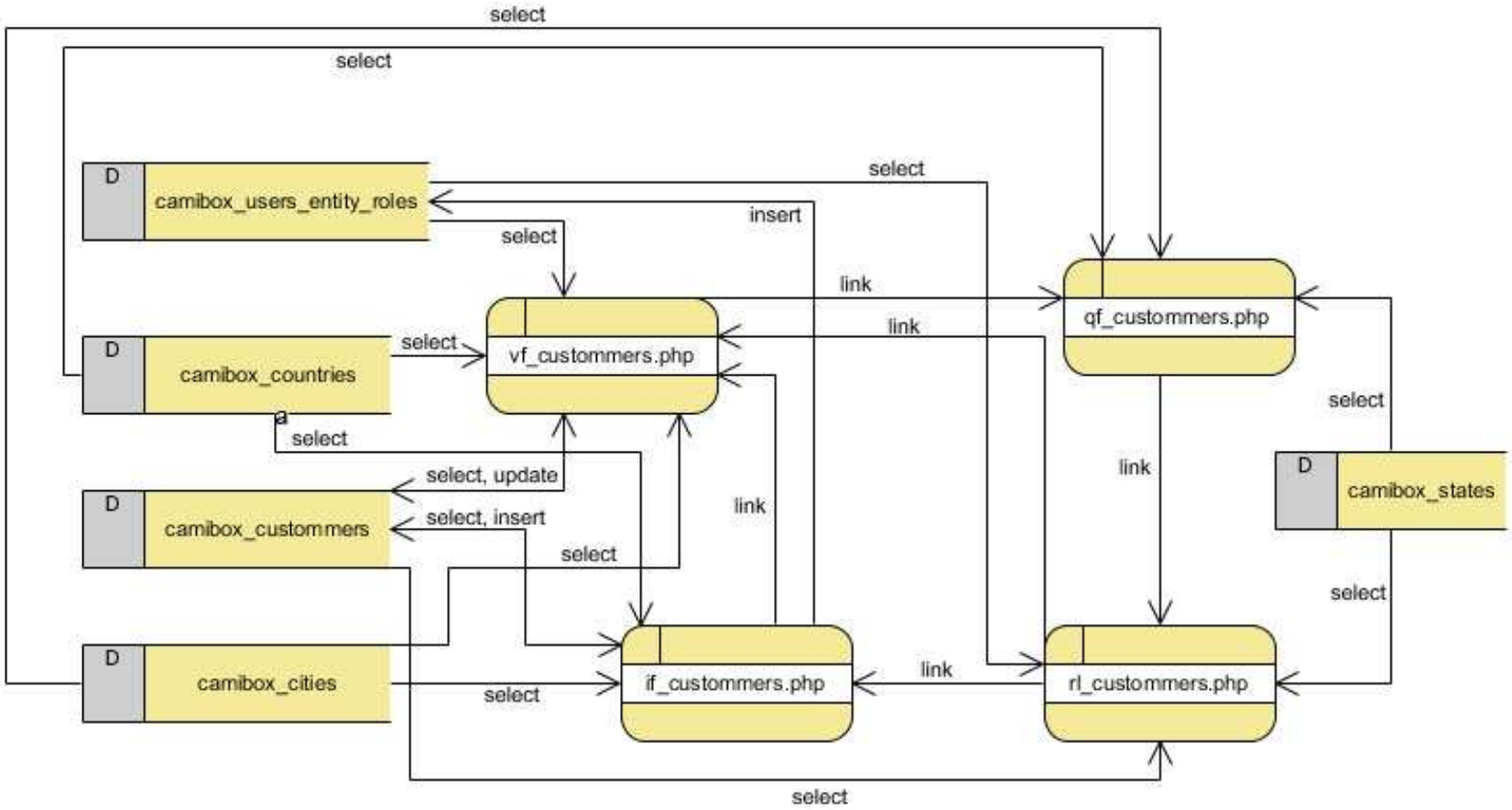
Příloha B.1 - DFD diagram – modul admin



Příloha B.2 - DFD diagram – modul contract



Příloha B.3 - DFD diagram – modul custommers



Příloha B.4 - DFD diagram – modul unit