VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Jiří Kozlovský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INFORMAČNÍ SYSTÉM PRO EVIDENCI ZAŘÍZENÍ POČÍTAČOVÉ SÍTĚ FEKT
INFORMATION SYSTEM FOR NETWORK DEVICES DOCUMENTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jiří Kozlovský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Macho, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Jiří Kozlovský ***ID:*** 151649
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Informační systém pro evidenci zařízení počítačové sítě FEKT

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s požadavky na evidenci prvků počítačové sítě FEKT a jejich komponent (switchů, WiFi AP, VLAN, uživatelských skupin, WiFi účtů, Webů).
2. Proveďte systémovou analýzu, navrhněte datový a procesní model systému, řešte životní cykly důležitých entit.
3. Vytvořte koncepci informačního systému, který by umožňoval prostřednictví WWW rozhraní evidovat prvky počítačové sítě. Řešte problematiku oprávnění přístupu k systému včetně rolí jednotlivých uživatelů.
4. Informační systém implementujte a odlaďte.
5. Ověřte funkčnost informačního systému a vyhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] Šimůnek, M. SQL kompletní kapesní průvodce. 1. dotisk. Praha: Grada, 1999. ISBN80-7169-692-7.

[2] Brázda J. PHP 5 začínáme programovat. Praha: Grada Publishing a.s., 2006. 244 s. ISBN 80-247-1146-X.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 29.5.2017

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

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá vytvořením a realizací konceptu informačního systému pro evidenci zařízení počítačové sítě Fakulty elektrotechniky a komunikačních technologií Vysokého učení technického v Brně. Cílem práce je provést systémovou analýzu, navrhnout procesní i datový model informačního systému, řešit problematiku životních cyklů evidovaných entit a oprávnění přístupu k systému včetně rolí jednotlivých uživatelů. Pak systém implementovat, odladit a ověřit jeho funkčnost.

KLÍČOVÁ SLOVA

informační systém, evidence síťových zařízení, volba programovacího jazyka, volba databázového úložiště, uživatelské role, oprávnění rolí, procesní model, datový model, ER diagramy, systémová analýza

ABSTRACT

This study is focused on creating and realisation of the concept of information system for network devices documentation at The Faculty of Electrical Engineering and Communication at Brno University of Technology. The aim of the study is to create system analysis, create process and data model, to solve the problematics of the documented entities lifecycles, solve problems of system access, including user roles. Then to implement the system, debug it and verify it's functionality.

KEYWORDS

information system, documenting network devices, programming language choice, database storage choice, user roles, role permissions, process model, data model, ER diagrams, system analysis

KOZLOVSKÝ, Jiří *Informační systém pro evidenci zařízení počítačové sítě FEKT*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2017. 56 s. Vedoucí práce byl Ing. Tomáš Macho , Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Informační systém pro evidenci zařízení počítačové sítě FEKT“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Děkuji svému vedoucímu bakalářské práce, panu Ing. Tomáši Machovi, Ph.D. , za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Děkuji také panu Ing. Radovanu Holkovi, CSc. za tipy k rozšíření funkcionalit systému a úvod do tvorby informačních systémů.

Rovněž patří můj dík rodině za podporu při studiu a vytvoření potřebného zázemí. Děkuji také přítelkyni za její velkou trpělivost se mnou při psaní této bakalářské práce.

Brno . . . . . .

. . . . . .

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Cílem této bakalářské práce je implementovat IS (informační systém), který by umožňoval prostřednictvím webového rozhraní evidovat prvky počítačové sítě FEKT (Fakulta elektrotechniky a komunikačních technologií). K tomu je zapotřebí provést systémovou analýzu a posléze navrhnout procesní a datový model. Dále je potřeba zvolit programovací jazyk, databázové úložiště, vývojové prostředí a aplikační server. Posléze je nutné navrhnout grafické webové rozhraní, realizovat datový model, naprogramovat systém dle procesního modelu, pak jej odladit a zdokumentovat.

Datový model byl navržen pomocí ER modelů (entitně vztahový model - entity relationhip model) v programu MySQL Workbench[1]. Pro modelování procesního modelu byl použit online nástroj Lucidchart[2]. UML diagramy byly vytvořeny open-source nástrojem PlantUML[3].

V práci se dále zabývám problematikou uživatelských rolí a jejich oprávnění, obnovou zapomenutého hesla, strukturou databáze, vazbami jednotlivých evidovaných entit, normalizací a zobecněním datového modelu do formy, která umožňuje libovolně přidávat další třídy entit pro evidenci či jakkoliv měnit jejich vazby nebo vlastní parametry entit správnou konfigurací.

Také se zde zabývám životními cykly entit, jejich napojením na oprávnění uživatelů či architekturou systému. Malou část také věnuji instalaci a úvodní konfiguraci informačního systému.

[1] Oficiální stránky MySQL Workbench: `http://www.mysql.com/products/workbench/`
[2] Oficiální stránky Lucidchart: `https://www.lucidchart.com`
[3] Oficiální stránky PlantUML: `http://plantuml.com/`

1 SYSTÉMOVÁ ANALÝZA

Informační systém vzniká za účelem usnadnění administrace síťových prvků a zlepšení přehlednosti aktuálních instalací na různých místech fakulty.

1.1 Evidované třídy entit

Mezi evidované prvky se řadí servery a jejich komponenty, switche a jejich komponenty, Wi-Fi AP (Wi-Fi přístupový bod - Wi-Fi access point), nastavení VLAN (virtuální LAN), Wi-Fi účty, weby a uživatelské skupiny.

1.1.1 Server

Server je zařízení, jež zejména zajišťuje přístup k informačním zdrojům uloženým na jeho datových discích. Může ale poskytovat i jiné služby, jako například poskytovat výpočetní výkon po síti. Každému serveru se přiřazuje několik prvků.

Jedná se především o prvky charakterizující výkon serveru, jako je velikost operační paměti RAM (random access memory), výkon a počet jader jednotlivých procesorů, typy datových disků a systémového disku, jejich velikost.

Dalšími prvky, jsou základní deska, řadič disku, typ RAID (vícenásobné diskové pole nezávislých disků - redundant array of independent disks), typy provozovaných služeb označovaných pomocí čísla SAP (protokol pro vyhledávání a oznamování služeb v síti - service advertising protocol) a v neposlední řadě také výrobce, dodavatel a umístění serveru, název serveru určený administrátorem, typ serveru či napajecí zdroj serveru.

1.1.2 Switch

Switch je zařízení, které mezi sebou propojuje síťová zařízení, jež spolu chtějí komunikovat. Na rozdíl od tzv. hubu směruje pakety jen do těch směrů, do kterých je to potřeba [1].

Informace o switchi, které se mají ukládat, jsou jeho název, typ, stručný popis, výrobce, umístění a sériové číslo.

Komponentami switche jsou větráky, napájecí zdroje, moduly či tzv. transceivery (síťový prvek, který je vysílačem a přijímačem zároveň - z angl. spojení transmitter a receiver).

1.1.3 Wi-Fi AP

Wi-Fi AP slouží k vytvoření přístupové brány do místní virtuální sítě (VLAN) a také do internetu. Jeden přístupový bod může vytvořit maximálně jednu Wi-Fi síť.

Mnohdy však bude plnit úkol tzv. přemostění, což nastane v případech, kdy administrátor chce pokrýt několik budov stejnou sítí. V tomto režimu bude AP pouze přeposílat packety na hlavní AP, kde bude probíhat fyzické spojení uživatele Wi-Fi s VLAN a internetem.

1.1.4 VLAN

VLAN je ve své podstatě distribuovaný program [2]. Nejedná se tedy o fyzické zařízení. Jeho účelem je virtuálně propojit jeden a více serverů či klientů dohromady tak, aby mohli mezi sebou přímo komunikovat bez použití internetu či jiné sítě [2].

Komunikace probíhá pomocí adresování internetového protokolu (IP address). Každé zařízení komunikující pomocí IP s ostatními zařízeními uvnitř VLAN má v síti přiřazenu alespoň jednu unikátní IP adresu. Bez toho by totiž nebylo možné uskutečnit komunikaci internetovým protokolem.

Může se jednat buďto o IP adresu verze 4, která má podobu `ddd.ddd.ddd.ddd`, kde každé `d` představuje decimiální číslici [3], nebo o IP adresu verze 6, která má podobu `hhhh:hhhh:hhhh:hhhh:hhhh:hhhh`, kde každé `h` představuje hexadecimální číslici [4].

1.1.5 Uživatelské skupiny

Uživatelské skupiny zajišťují evidenci veškerých UNIXových skupin na jednotlivých serverech. Skládají se z názvu skupiny a jejího numerického identifikátoru.

Informační systém by měl evidovat také členství uživatelů serverů v těchto skupinách.

1.1.6 WiFi účty

WiFi účty slouží ke shromáždění informací o dočasných WiFi účtech určených pro hosty sítě, kteří nemají záznam v univerzitním autorizačním serveru nebo o trvalých WiFi účtech sloužících k údržbě sítě. Obsahují tedy přihlašovací jméno a heslo do dané WiFi sítě.

Uložené informace by měly obsahovat i další metadata, jako je seznam umístění kde lze tyto WiFi účty používat či maximální počet jejich připojení.

1.1.7 Webové stránky

Evidují informace o veřejných i neveřejných webových stránkách fakulty. Především tedy název serveru, na kterém jsou provozovány, ústav fakulty, se kterým jsou spjaty, a URL adresu, kde lze tyto stránky nalézt.

Navíc má každá taková webová stránka přiděleného správce, jehož je také potřeba evidovat. U něj stačí jméno, příjmení, ústav, kterému přísluší, a osobní číslo.

1.1.8 Ostatní evidované entity

Je ještě mnoho entit a parametrů, které jsem nezmínil, a přesto je budeme evidovat. Nebyly dosud zmíněny, neboť jich je velké množství a výše uvedený seznam entit a parametrů k evidenci má poskytovat základní náhled na data, jež jsou předmětem evidence.

Každou entitu a jí příslušný parametr k evidenci lze vyčíst z logického ER diagramu, který byl vytvořen jako součást systémové analýzy, viz příloha s obrázkem C.1 (Logický ER diagram IS)

1.2 Uživatelé IS

Pro jakoukoliv operaci v IS (kromě obnovy hesla a přihlášení) je potřeba být přihlášený. Pokusí-li se uživatel navštívit adresu, kterou mohou vidět pouze přihlášení uživatelé a sám uživatel přihlášený není, zobrazí se mu přihlašovací obrazovka.

Po úspěšném přihlášení se uživatel přesměruje na stránku, na kterou se pokusil přistoupit, když ještě nebyl přihlášený. Pokud žádná taková stránka není, je přesměrován na výchozí stránku jeho nejvyšší role.

Pro správu uživatelů a jejich práv nebude použit žádný autoritativní server. Tím by mohl být například univerzitní LDAP (The Lightweight Directory Access Protocol) server s již vytvořenými účty, na něž by se navázala patřičná oprávnění. Veškeré informace o uživatelích a jejich rolích budou tedy uloženy v databázi informačního systému.

Přihlašovat se uživatelé budou pomocí emailové adresy a hesla, které si zvolí sami, či jej uživateli přiřadí administrátor. Uživatel si pak samozřejmě bude moci své heslo změnit.

V IS nikdy nebudou uložena samotná hesla. Vždy se uloží jen jejich tzv. hash (volně přeloženo jako změť) použitím hashovací funkce `bcrypt`, jejíž bezpečnost je velmi vysoká, neboť umožňuje se zvyšujícími se výpočetními výkony zvyšovat i obtížnost prolomení hashe inkrementací počtu iterací [5].

1.2.1 Uživatelské role a oprávnění

Je žádoucí, aby jakákoliv operace v IS byla ověřena vůči oprávnění spjatému s některou z uživatelových rolí.

Role tedy existují jako jakýsi předpis, který vyjmenovává, jaká oprávnění jsou spolu s touto rolí uživateli přidělena.

Veškerá oprávnění, která informační systém dokáže rozlišit, jsou:

- Vytvořit uživatele
- Vytvořit třídu entit
- Vytvořit typ entity
- Vytvořit entitu
- Vytvořit výrobce
- Vytvořit dodavatele
- Vytvořit umístění
- Změnit libovolného uživatele
- Změnit třídu entit
- Změnit typ entity
- Změnit entitu
- Změnit výrobce
- Změnit dodavatele
- Změnit umístění
- Zobrazit libovolného uživatele
- Zobrazit třídu entit
- Zobrazit typ entity
- Zobrazit entitu
- Zobrazit výrobce
- Zobrazit dodavatele
- Zobrazit umístění
- Změna stavu z *neaktivní* na *aktivní*
- Změna stavu z *aktivní* na *neaktivní*
- Změna stavu z *aktivní* na *v opravě*
- Změna stavu z *aktivní* na *odepsaný*
- Změna stavu z *v opravě* na *aktivní*
- Manipulovat s rolemi ostatních uživatelů

Důvodem, proč neexistuje oprávnění manipulovat s rolemi libovolných uživatelů, je skutečnost, že by se tak například poslední administrátor mohl uzamknout mimo administrativní pozici a žádný administrátor, který by mu nazpět roli přiřadil, by

už v systému nebyl. Tento uzamčený, bývalý administrátor, by pak musel požádat správce databáze, aby mu přiřadil administrativní roli přímo v ní.

V IS se budou rozlišovat tři základní role: `Administrátor`, `Správce zařízení` a `Auditor`.

Administrátor

Role administrátora zahrnuje veškerá oprávnění.

Správce zařízení

Role správce zařízení zahrnuje oprávnění role auditora a tato další oprávnění:

- Vytvořit typ entity
- Vytvořit entitu
- Vytvořit výrobce
- Vytvořit dodavatele
- Vytvořit umístění
- Změnit typ entity
- Změnit entitu
- Změnit výrobce
- Změnit dodavatele
- Změnit umístění
- Změna stavu z *neaktivní* na *aktivní*
- Změna stavu z *aktivní* na *neaktivní*

Auditor

Role auditora uživateli umožňuje omezené množství operací:

- Zobrazit třídu entit
- Zobrazit typ entity
- Zobrazit entitu
- Zobrazit výrobce
- Zobrazit dodavatele
- Zobrazit umístění

2 VOLBA PROGRAMOVACÍHO JAZYKA A DATABÁZOVÉHO ÚLOŽIŠTĚ

Vzhledem k povaze úkolu je potřeba zvolit takový programovací jazyk, který umožní tvorbu webového rozhraní a který zároveň bude schopen manipulovat se zvoleným databázovým systémem [6].

Zvolený jazyk také musí být interpretovatelný webovým serverem komunikujícím s internetovým prohlížečem uživatele [6]. Některé jazyky však obsahují svůj vlastní webový server jako jejich součást. Příkladem může být `Flask` v Pythonu [7]. Webovými servery jsou například `Apache httpd`, `Nginx`, `Jetty`, `Apache Tomcat`, `Node.js` a mnoho dalších [1].

Tyto podmínky splňují jazyky Python [7], Java [8], Perl [9], Node [10], PHP [11], .NET [12], Ruby [13], C++ (CppCMS [2]) a další [3]

Ze široké škály jazyků, které splňují uvedené podmínky, jsem vybral PHP. Důvodem je popularita a tradice jazyka [15], existence mnoha frameworků [16], časová nenáročnost vývoje, spolehlivé a snadné propojení s relační MySQL databází [11] a také nabyté zkušenosti s vývojem v PHP. Za databázové úložiště je tedy zvolena relační databáze MySQL.

Pro maximální efektivitu práce a udržitelnost vývoje bude pro vývoj v PHP použit Laravel framework [4], jeden z nejpopulárnějších a široce užívaných PHP frameworků vůbec [17].

Laravel již má implementovánu manipulaci s MySQL, uživatelskými sezeními, přihlašováním, obnovou hesla, mezipamětí, efektivní generací statických HTML (HyperText Markup Language) stránek bez zbytečných bílých znaků, CSRF (Cross-site Request Forgery) ochranou a mnoho dalších vestavěných funkcí, které jsou skvěle popsány v dokumentaci [5].

Spolu s Laravelem bude pro vývoj JavaScript modulů použit framework Vue [6].

---

[1] Seznam nejpoužívanějších webových serverů lze nalézt na webové adrese `https://w3techs.com/technologies/overview/web_server/all`

[2] CppCMS je C++ framework umožňující vývoj rychlých webových aplikací, viz `http://cppcms.com/wikipp/en/page/main`

[3] Teoreticky uvedené podmínky splňuje každý existující programovací jazyk, neboť se vždy jedná jen o abstraktní soubor pravidel [14]. Exaktním vyjádřením zde uvedených podmínek je tedy takový programovací jazyk, u kterého není nutno implementovat interpretaci na straně webových serverů či vytvářet vlastní webový server.

[4] Odkaz na oficiální webové stránky Laravel: `https://laravel.com/`

[5] Laravel dokumentace je k dispozici online: `https://laravel.com/docs`

[6] Oficiální webové stránky Vue frameworku: `https://vuejs.org/`

3 DATOVÝ MODEL

Cílem datového modelu je poskytnout kvalitní analýzu a řešení problému efektivního ukládání dat, v tomto případě do relační DB, která byla zvolena MySQL (relační SQL databáze). Formát úložiště bude transakční `InnoDB`, jelikož umožňuje provádět atomické operace za pomocí transakcí [18].

Potřebujeme ukládat data o uživatelských účtech s přístupem do IS, jejich rolemi, oprávněními, aktivními sezeními, metadaty pro obnovu hesla a vzájemnými vztahy mezi těmito daty.

Také potřebujeme ukládat data o veškerých entitách a jejich parametrech. Entitami jsou jednotlivé servery, disky, RAID, diskové řadiče, procesory, základové desky, síťové karty, transceivery, switche, napájecí zdroje, moduly, VLAN, uživatelské skupiny serverů, uživatelé serverů, WiFi účty, WiFi AP, weby, ústavy fakulty, její pracovníci, umístění na fakultě a nakonec výrobci a dodavatelé jednotlivých zařízení.

Parametry vybraných entit jsou například název, typ, umístění, dodavatel, URL adresa, login, sériové číslo, aj.

Kromě toho je také potřeba definovat všechny možné stavy těchto entit a možné přechody mezi těmito stavy, které lze uskutečnit pouze s určitým oprávněním uživatele IS. Dále je potřeba ukládat přihlašovací údaje uživatelů IS, jejich navázané role a k nim příslušné oprávnění.

3.1 Vazby entit

Některé entity mohou mít podřízené entity. Kupříkladu entita třídy `Server` může mít podřízenou entitu třídy `Základová deska`, která sama o sobě také může mít podřízené entity třídy `Transceiver`. Nebo entita třídy `Switch` může mít podřízené entity třídy `Modul`, které také mohou mít podřízené entity třídy `Transceiver`. Takže entita třídy `Transceiver` může být podřízená jak entitě třídy `Základová deska`, tak `Modul`. Někdy však může patřit rovnou entitě třídy `Switch`, či `Síťová karta`, která patří entitě třídy `Server`. Třída entit `Transceiver` tedy umožňuje patřit čtyřem různým třídám entit. Vždy ale jedna entita patří maximálně jedné nadřízené entitě. Čili jeden `Transceiver` nemůže být fyzicky zapojen do základové desky serveru a zároveň do modulu switche, či switche. Bude patřit vždy maximálně jedné nadřízené entitě.

Obecně má každá třída entit možnost mít vazbu na jinou třídu entit. Výjimkou je pouze třída entit `VLAN`, která dle systémové analýzy vždy existuje jen sama o sobě, bez jakékoliv možnosti vazby na jinou entitu.

3.1.1 Popis vazeb mezi entitami

Veškeré vazby, které mohou dle systémové analýzy nastat mezi jednotlivými třídami entit, naleznete v příloze na obrázku C.1 (Logický ER diagram IS).

Žádná z vazeb mezi entitami není vyžadována, neboť vždy může nastat situace, že byla entita zakoupena, či jinak pořízena, ale zatím není jasné, ke které konkrétní entitě se naváže. V takovém případě se nachází ve stavu *neaktivní*.

Vazby třídy[1] `Server`

Třída `Server` zajisté obsahuje nejvíce vazeb. Vlastním entitám umožňuje mít vazby na následující třídy entit: `Umístění`, `Dodavatel`, `Výrobce` a `RAID`. Na sebe pak dovoluje vázat tyto třídy entit: `Procesor`, `Diskový řadič`, `RAID`, `Disk`, `Základová deska` (touto vazbou nepřímo i `Transceiver`), `Síťová karta` (tím znova i `Transceiver`) a `Napájecí zdroj`.

Důvodem obousměrné vazby třídy `Server` s třídou `RAID` je skutečnost, že pokud má `RAID` vazbu na `Server`, tak se jedná o zprostředkování datových disků, neboť těch může `Server` mít více skupin.

[1] Pokud text pojednává o *třídě*, je tím vždy myšlena *třída entit*

Jestliže má `Server` vazbu na `RAID`, bude konkrétní `Server` podřízenou entitou třídy `RAID` z toho důvodu, že entitě třídy `Server` poskytuje přístup k systémovému disku, bez něhož by jinak nemohl fungovat.

Vazby třídy `RAID`

Obsahuje vazby na `Server` a `Výrobce`. Navázat se na ni mohou třídy `Disk` a `Server`.

Vazby třídy `Základová deska`

Tato třída má vazbu pouze na `Server`. Váže na sebe pouze třídu `Transceiver`.

Vazby třídy `Síťová karta`

Tato třída má také vazbu pouze na `Server`. Váže na sebe také pouze třídu `Transceiver`.

Vazby třídy `Switch`

Tato třída má vazby na `Výrobce` a `Umístění`. Vlastním entitám umožňuje na sebe vázat entity těctho tříd: `Větrák` , `Transceiver` , `Modul` (tím znova i `Transceiver`) a `Napájecí zdroje`.

Vazby třídy `Modul`

Existuje zde vazba pouze na `Switch`. Na sebe pak váže `Transceiver`.

Vazby třídy `VLAN`

Tato třída žádné vazby neumožňuje. U `VLAN` se v podstatě jedná o evidenci softwarového nastavení, které fyzicky není propojeno se žádnou další entitou. Je zde ale potenciál, že někdy v budoucnu vznikne na IS požadavek tuto třídu nechat propojit například s třídou `Switch`, jejíž entity pak obsahují tyto konfigurace virtuálních lokálních sítí, neboť se obě třídy nacházejí na linkové vrstvě [19].

Vazby třídy `WiFi AP`

Bez vazby na třídy `Umístění` a `WiFi platforma` by se dalo říct, že je, podobně jako `VLAN`, také izolovaná třída. Nicméně vazby zde existují, takže se o izolovanou třídu nejedná.

Vazby třídy `WiFi účet`

Třída `WiFi účet` umožňuje na vlastní entity navázat pouze entity třídy `Umístění WiFi účtu`. Tyto vazby umožňují jednotlivým entitám třídy `WiFi účet` mít více než jedno umístění. Třída `Umístění WiFi účtu` tedy umožňuje vytvořit M:N relaci mezi třídou `Umístění` a `WiFi účet`.

Vazby třídy `Webová stránka`

Tato třída dovoluje vlastním entitám navázat vazby až na tři další třídy entit. Jsou jimi `Server`, `Ústav` a `Zaměstnanci`. Na vlastní entity neumožňuje vázat žádné třídy entit.

Vazby třídy `Ústav`

`Ústav` se na žádnou třídu neváže. Pouze vlastním entitám umožňuje na sebe navázat třídy `Webová stránka` a `Zaměstnanec`.

Vazby třídy `Uživatel serveru`

Tato třída se váže na `Server` a umožňuje na sebe vázat třídu `Členství ve skupině`.

Ostatní vazby diagramu

Ostatní vazby, které pozorujete v uvedeném diagramu, se týkají teprve normalizované formy modelu, která je diskutována v následující podkapitole. Jedná se o vazby nikoliv mezi entitami, nýbrž mezi entitami a typy entit, jak bude dále vysvětleno.

3.2 Normalizace modelu

Návrh modelu na obrázku C.1 (Logický ER diagram IS) byl učiněn s ohledem na potřeby první normální formy, tedy že každý atribut každé tabulky obsahuje pouze atomické hodnoty [20].

Druhá normální forma vyžaduje, aby v jednotlivých sloupečcích tabulek nevznikaly duplicitní hodnoty, ale namísto toho byly uloženy v další tabulce a byly provázány formou cizích klíčů [20]. Pokud by tedy například tabulka `processors` obsahovala název procesoru a jeho frekvenci, tak by při pořízení dvou stejných typů procesorů vznikaly duplicitní hodnoty v obou sloupečcích. Z tohoto důvodu existuje tabulka `processor_types`, která obsahuje jednotlivé typy procesorů, tedy jejich obchodní název, frekvenci a třeba i počet jader. Při pořízení dvou stejných typů procesorů pak v tabulce `processors` vzniknou dvě řádky s totožným cizím klíčem, směřujícím na jeden řádek tabulky `processor_types`.

Druhé normální formy lze tedy dosáhnout založením nové tabulky pro každou takovou třídu entity, jež může obsahovat společné charakteristiky napříč jednotlivými entitami.

Veškeré tabulky, které vznikly aplikací druhé normální formy, jsou `server_types`, `processor_types`, `disc_types`, `wifi_platforms`, `server_groups`, `switch_types`, `motherboard_types`, `fan_types`, `power_supply_types`, `module_types`, `transceiver_types`, `disc_controller_types` a `network_card_types`.

Každá vazba mezi typem entity a entitou samotnou je vyžadována. Nemůže tedy například nastat situace, že vznikne entita třídy `Processor`, jež nemá žádnou vazbu s proprietárním typem (tedy s entitou třídy `Processor Type`), neboť by tento záznam neměl žádnou informativní hodnotu.

Třetí normální forma je automaticky splněna, pokud neexistuje jediná tabulka, která by měla více než jeden primární klíč [20]. To je také splněno. Je tedy možné navržený model označit za normalizovaný.

3.3 Zobecnění logického modelu

Navržený logický model sice naplňuje veškeré požadavky vyplývající ze systémové analýzy a normalizace, má ale jednu velmi zásadní vadu. Nelze jej snadno rozšířit či jinak pozměnit aniž by bylo potřeba zasahovat do schématu DB a do programového kódu IS.

Například již zmiňovaný požadavek na propojení třídy entit `VLAN` s třídou entit `Switch` by znamenal přeprogramovat IS, změnit schéma DB a provést migraci dat ze starého schématu do nového. To je spousta práce a existuje zde spousta problémů, které mohou nastat. U komplexnějších vazebních změn mezi třídami nemusí být migrace schématu proveditelná a jednoduše bude tím pádem potřeba veškeré entity po migraci vyplnit znova.

3.3.1 Dynamická tvorba tříd

Z těchto důvodů by bylo vhodné model přetvořit tak, aby obsahoval pouze obecné tabulky, do nichž by se z webového rozhraní daly vytvářet třídy entit a rovnou i definovat jejich možné parametry.

Také by mělo jít specifikovat, zda dané parametry mají být vyžadované či unikátní napříč všemi entitami třídy. Kromě toho by mělo být uskutečnitelné definovat, které třídy mohou být nadřízené jiné třídě. Tím by se u vytváření/editace entity dané podřízené třídy zajistilo, že vždy bude mít navázané příslušné nadřízené třídy. Kromě možnosti založit novou třídu entit by také měla samozřejmě existovat možnost naplňovat třídy jednotlivými entitami.

Třída entity tedy zatím umožňuje definovat, jaké parametry může/musí entita mít a které třídy entit mohou být s nimi ve vazbě.

Příklad

S dynamickou tvorbou tříd by tedy bylo možné (ačkoliv to není předmětem této práce) založit novou třídu entit `Elektronový mikroskop`, která bude mít vyžadované parametry *cena* a *životnost*, unikátní *jméno* a volitelné *umístění*. Pak založit novou třídu entit `Antivibrační podlaha` s vyžadovanými parametry *nosnost* a *maximální okamžité zrychlení*. Nakonec by mělo být možné definovat, že třída `Antivibrační podlahy` musí být podřízená třídě `Elektronové mikroskopy`, aby se zajistilo, že při vzniku nové entity třídy `Antivibrační podlaha` víme přesně, ke které entitě třídy `Elektronový mikroskop` náleží, což například může být požadavkem zadavatele.

3.3.2 Entity a typy entit

Nastává ale problém, zda tímto zobecněním nedojde k narušení druhé normální formy, což by se projevilo vznikáním duplicitních záznamů, které by jinak měly existovat v oddělených tabulkách. Podobně, jak bylo použito řešení u problému druhé normální formy s tabulkami `procesors` a `processor_types`, je zde zapotřebí zavést možnost vytvořit kromě entity také její typ. Měla by tedy v definici třídy entit existovat pravdivostní hodnota, zda jsou všechny entity třídy typem, či jím není žádná z nich.

Z dříve navržených tabulek `procesors` a `processor_types` se tak stávají dvě různé třídy entit, kdy jedna z nich je typem (`Typ procesoru`) a druhá typem není (`Procesor`).

Aby z datového modelu implementujícího dynamickou tvorbu tříd bylo zřejmé, které třídy typů entit patří ke kterým třídám entit, je zapotřebí nějakým způsobem uchovávat i tuto informaci. Vzhledem k tomu, že entity jedné třídy mohou mít vazbu vždy jen s entitami jedné třídy typů entit, tak se jedná o vazbu 1:1 a můžeme tuto informaci tedy uložit rovnou v třídě entit, která není typem.

Třídy entit jsou tedy v konečném důsledku předpisy pro vytváření entit a typů entit s předdefinovanými parametry, které mohou jednotlivé entity a typy entit nabývat, včetně předpisů pro vytváření předdefinovaných vazeb s entitami tříd.

Příklad

Jako příklad opět použiji třídy `Procesor` a `Typ procesoru`. Třída `Typ procesoru` bude mít parametry `název`, `frekvence` a `počet jader`. Jedná se o třídu typů[2].

Druhá třída, `Procesor`, nemá žádné parametry. Nejedná se o třídu typů. Její nadřízenou třídou může být pouze `Server`. Jejím typem je třída `Typ procesoru`.

Při vytváření nových entit třídy `Procesor` pak uživatel pouze vybere o který typ procesoru se jedná a do kterého serveru patří. Tím se vytvoří nová entita. Při zobrazení parametrů této entity se ve webovém rozhraní automaticky zobrazí i parametry jejího typu.

[2] *Třída typů* je zkrácením plného označení *třída typů entit* - jedná se o pravdivostní hodnotu určující, zda třída entit, o které je pojednáváno, bude obsahovat typy entit, či entity

3.4 Realizace datového modelu

Celý datový model, který nyní budu popisovat, včetně jeho interních vazeb tvořených cizími klíči mezi tabulkami, je vyznačen v příloze na obrázku B.1 (ER diagram informačního systému).

Uživatelé a jejich přístupy do IS jsou první skupinou, kterou jistě spatříte při pohledu na přiložený datový model. Je vyznačena okrovou barvou na pozadí. Druhou skupinou jsou *Systémové tabulky* se světle fialovou barvou pozadí. Třetí skupinou je *Evidence entit a jejich parametrů* na světle zeleném pozadí.

Všechny tyto skupiny jsou navzájem provázány formou cizích klíčů, což je mimo jiné důvod, proč je v ER diagramu uvádím všechny najednou.

3.4.1 Skupina *Systémové tabulky*

Se žádnou z těchto tabulek nelze nijak manipulovat z webového rozhraní. Jejich hodnoty jsou naplněny pouze jednou, a to při instalaci IS. Jedná se o tabulky `states`, `state_transitions`, `permissions` a `parameter_datatypes`. Jejich vazby lze vidět v příloze na obrázku B.1 (ER diagram informačního systému).

Tabulka `states` obsahuje všechny možné stavy, které IS dokáže rozlišovat. Jsou jimi `inactive`, `active`, `repairing` a `written_off`. V překladu do češtiny znamenají neaktivní, aktivní, v opravě a odepsaný.

Tabulka `permissions` má veškeré možné oprávnění, které IS dokáže rozlišovat. Více o jednotlivých oprávněních je popsáno v podkapitole 1.2.1.

Tabulka `state_transitions` uchovává informace o možných přechodech stavů s vazbou na oprávnění, které je potřeba mít pro uskutečnění přechodu. Je složena pouze z cizích klíčů, kdy dva odkazují na tabulku `states` (výchozí stav a cílový stav) a třetí na tabulku `permissions`, tedy na samotné oprávnění. Uživatel pak musí mít alespoň jednu roli, která zahrnuje odkazované oprávnění, aby mohl entitu převést z výchozího do cílového stavu.

Tabulka `parameter_datatypes` zase obsahuje názvy datových typů, které mohou být použity v předpisech parametrů jednotlivých tříd entit, tedy v tabulce s názvem `class_parameters`. Hodnoty jejich názvů korespondují s reálnými názvy sloupečků v datovém modelu tabulky `entity_parameters`. Doslova tedy říkají, ze kterého sloupečku se má hodnota požadovaného parametru vytáhnout pro zobrazení uživateli či k uložení nové hodnoty. To je mimo jiné důvod, proč je tato tabulka mezi systémovými - změna kterékoliv hodnoty by vyústila v neočekávanou chybu při každém dotazu na hodnotu parametru entity, která tento sloupeček používá. Jak již bylo zmíněno, obsahuje hodnoty totožné s názvy datových sloupečků tabulky `parameters`. Konkrétně to jsou `str`, `tiny_int`, `int`, `datetime` a `blob_data`.

3.4.2 Skupina *Evidence entit a jejich parametrů*

Z úvah uvedených v podkapitole 3.3 vyplývá, že je skutečně možné dynamicky měnit třídy evidovaných entit, a to přímo z webového rozhraní bez nutnosti měnit strukturu databáze či jakkoliv zasahovat do programového kódu informačního systému aniž by přitom byla narušena kterákoliv normální forma původního logického modelu.

Pro zjednodušení nebudu u jednotlivých tabulek uvádět společné sloupečky, jež slouží ke stejnému účelu. Jedná se o `id` (primární klíč tabulky typu `UNSIGNED INT`), `created_at` a `updated_at` (časové známky vytvoření, resp. úpravy tabulky, obě jsou typu `TIMESTAMP`).

Tabulka `entity_classes`

Tato tabulka zapouzdřuje informace o jednotlivých třídách entit a obsahuje následující parametry:

`name`	Textový řetězec určující název třídy. Například `Procesor` či `Typ Serveru`.
`is_type`	Pravdivostní hodnota, zda se jedná o třídu typů.
`require_parent`	Pravdivostní hodnota, zda entity této třídy vždy musejí mít ve vazbě alespoň jednu nadřízenou entitu ze kterékoliv třídy možných nadřízených entit.
`entity_type_class_id`	Cizí klíč směřující do vlastního `id` sloupečku. Jedná se tedy o tzv. *rekurentní vztah* [21]. Slouží k vytvoření vazby mezi třídou entity a jejím typem.
`has_manufacturer`	Pravdivostní hodnota, zda entity této třídy mohou mít výrobce.
`has_supplier`	Pravdivostní hodnota, zda entity této třídy mohou mít dodavatele.
`has_location`	Pravdivostní hodnota, zda entity této třídy mohou mít umístění.

U některých tříd potřebujeme specifikovat, že její entity musí patřit alespoň jedné nadřízené třídě, čehož dosáhneme právě parametrem `require_parent`. To je případ například třídy `Transceiver`, jejíž entity musí patřit vždy alespoň jedné entitě ze tříd `Modul`, `Switch`, `Základová deska`, či `Síťová karta`.

Tabulka `possible_parent_classes`

Uchovává informace o možných vazbách mezi nadřízenou a jí podřízenou třídou.

`child_entity_class_id`	Cizí klíč směřující do `id` sloupečku tabulky `entity_classes`. Označuje podřízenou třídu.
`parent_entity_class_id`	Cizí klíč směřující do `id` sloupečku tabulky `entity_classes`. Označuje nadřízenou třídu.

Tabulka `class_parameters`

Přesně definuje veškeré parametry, které lze u entity dané třídy evidovat.

`entity_class_id`	Cizí klíč směřující do `id` sloupečku tabulky `entity_classes`. Určuje, které třídě tento parametr náleží.
`parameter_datatype_id`	Cizí klíč směřující do `id` sloupečku tabulky `parameter_datatypes`, která je součástí systémových tabulek. Účel tohoto parametru byl vysvětlen v podkategorii 3.4.1 (Skupina *Systémové tabulky*).
`parameter_name`	Textový řetězec s názvem parametru.
`max_param_len_or_val`	Číslo, jež určuje maximální délku parametru, pokud je typ parametru řetězec; pokud je typ parametru číslo, tak jeho maximální hodnotu.
`unique`	Pravdivostní hodnota, zda je tento parametr unikátní napříč všemi entitami třídy.
`required`	Pravdivostní hodnota, zda je tento parametr vyžadován při vytváření/editaci entity.

Tabulky `locations`, `suppliers` a `manufacturers`

Mají pouze jeden parametr – název. Ačkoliv se také jedná o třídy entit, rozhodl jsem se každé z nich vytvořit zvláštní tabulku, neboť se na ně vytvářejí vazby nejčastěji a zároveň představují skupinu entit, která má se sítí samotnou pramálo společného.

V grafickém rozhraní je pak tedy nehledejte mezi ostatními entitami. Každá tato třída bude mít vlastní položku v menu.

Tabulka `entities`

Slouží k evidenci samotných entit. Dalo by se tedy říci, že je stěžejní, neboť evidence entit je náplní této práce. I přesto však neobsahuje nic jiného než cizí klíče, primární klíč a časové známky.

`parent_device_id`	Cizí klíč směřující do `id` sloupečku tabulky vlastní tabulky. Jedná se tedy opět o *rekurentní relaci*. Touto relací se realizuje vazba k nadřízené entitě.
`entity_class_id`	Cizí klíč směřující do `id` sloupečku tabulky `entity_classes`. Touto relací se jednoznačně určuje, do které třídy entit tato entita patří.
`entity_type_id`	Cizí klíč směřující do `id` vlastní tabulky. Jde tedy o další *rekurentní relaci*. Odkazuje na entitu příslušného typu, viz podkapitola 3.3.2 (Entity a typy entit).
`location_id`	Cizí klíč směřující do `id` sloupečku tabulky `locations`. Určuje tedy umístění entity, pokud jí to třída entit povoluje.
`manufacturer_id`	Cizí klíč směřující do `id` sloupečku tabulky `manufacturers`. Určuje tedy výrobce entity, pokud jí to třída entit povoluje.
`supplier_id`	Cizí klíč směřující do `id` sloupečku tabulky `suppliers`. Určuje tedy dodavatele entity, pokud jí to třída entit povoluje.
`state_id`	Cizí klíč směřující do `id` sloupečku systémové tabulky `states`. Určuje tedy v jakém stavu se entita nachází.

Tabulka `entity_parameters`

Obsahuje veškeré parametry entit.

`class_parameter_id`	Cizí klíč směřující do `id` sloupečku tabulky `class_parameters`. Odkazuje tak na svůj předpis. Touto relací lze tedy zjistit název paramteru, zda má být parametr unikátní, vyžadovaný či jeho maximální délku/hodnotu nebo i datový typ parametru.
`entity_id`	Cizí klíč směřující do `id` sloupečku tabulky `entities`. Označuje entitu, které této parametr patří.
`str`	Textový řetězec uchovávající hodnotu parametru.
`int`	Celé číslo uchovávající hodnotu parametru, ne větší než 2 147 483 647
`tiny_int`	Celé číslo uchovávající hodnotu parametru, ne větší než 127. Je vhodné pro uchovávání pravdivostní hodnoty [22]. V předpise parametru pro tento účel stačí zadat maximální hodnotu 1.
`datetime`	Časová známka uchovávající hodnotu parametru.
`blob_data`	Binární objekt uchovávající hodnotu parametru. Typicky zde může být uložen malý soubor.

Tabulka má tedy celkem 5 datových sloupečků, které jsou zároveň datovými typy. Do kterého sloupečku se data budou ukládat, určuje předpis parametru, chcete-li příslušná definice parametru třídy, tedy tabulka `class_parameters`.

3.4.3 Skupina *Uživatelé a jejich přístupy do IS*

Obsahuje následující tabulky:

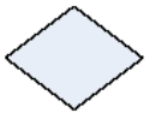
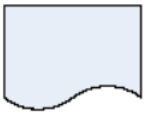
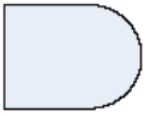
`sessions`	Stará se o sezení uživatelů IS.
`users`	Obsahuje v sobě uživatele IS, včetně jejich stavu, emailu, jména a zahashovaného hesla, viz podkapitola 1.2 (Uživatelé IS)
`password_resets`	Slouží k udržování metadat potřebných pro jednorázovou obnovu hesla.
`role_user`	Udržuje vazby mezi uživateli a jim přiřazenými rolemi.
`roles`	Eviduje seznam všech rolí IS
`permission_role`	Má vazby mezi rolemi a jim přiřazenými oprávněními.

4 PROCESNÍ MODEL

Základním stavebním kamenem tvorby procesního modelu je jeho grafická reprezentace [23]. Ta je tvořena vývojovým diagramem, který je definován obecně uznávanými symboly reprezentujícími část procesu, kterou symbol vyjadřuje.

Symbol obdélníku vyjadřuje aktivitu. Kosočtverec označuje rozhodovací bod, odkud typicky směřují alespoň dvě možná rozhodnutí do dalšího symbolu. Oválný obdélník pak vyjadřuje začátek, či konec procesního modelu. [23]

Zde, na obrázku 4.1, je možno se podívat na vysvětlení symbolů v angličtině:

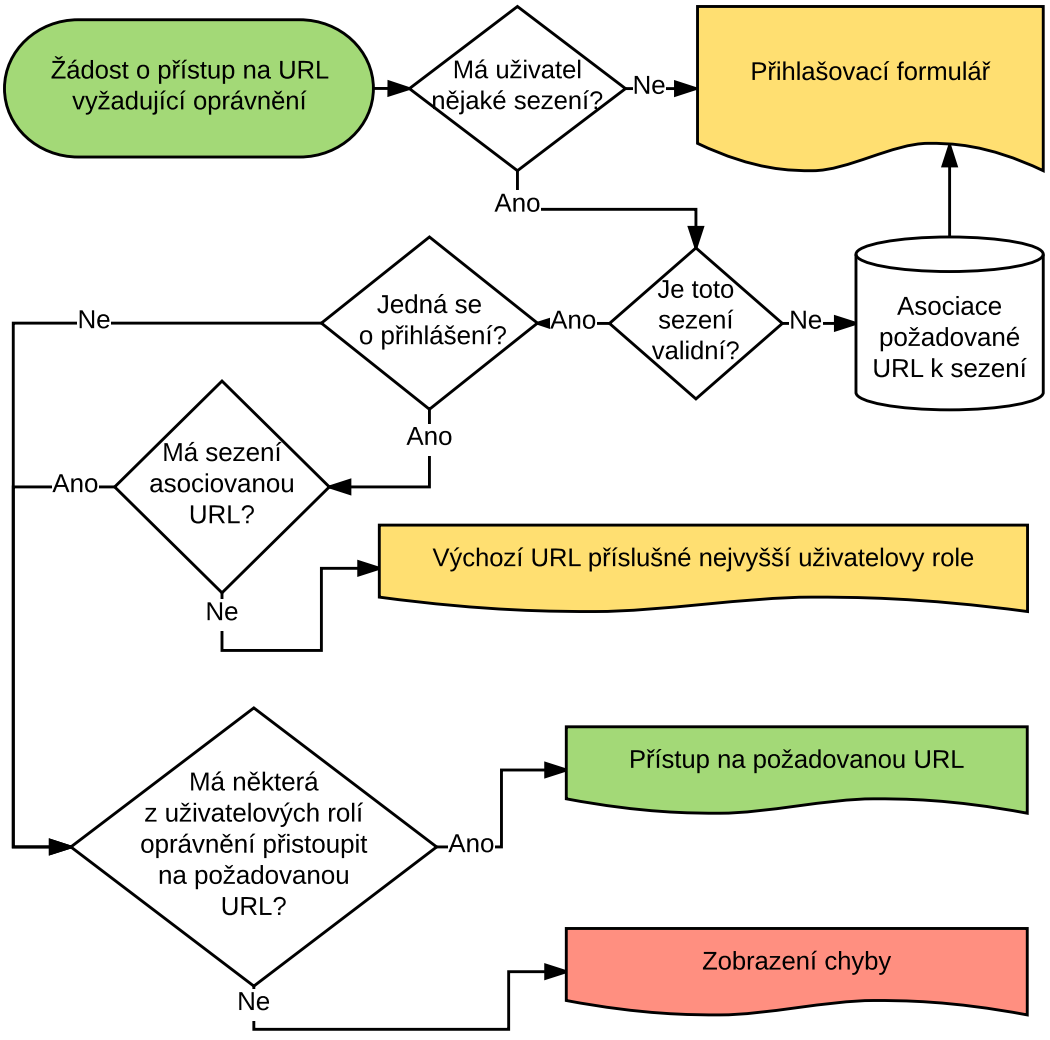
Symbol	Represents	Example
	Activity	Data is entered Purchase approved Cut PO ISO 9004.4
	Decision Point	Yes/No Agree/Disagree Pass/Fail ISO 9004.4
	Document	Purchase Requisition IV Record Expense Claim
	Start/Stop Input/Output	From (other map) To (other map) A need is identified ISO 9004.4
	On-page Connector	Go to another part of the flowchart
	Arrow	Shows the direction of flow taken by the process, through all the other symbols. ISO 9004.4
	Delay	Waiting for a service Report sitting on a desk

Obr. 4.1: Tabulka vysvětlující základní symboly procesního modelování [23]

Procesy uvnitř informačního systému jsem rozdělil do třech hlavních kategorií: Validace HTTP požadavku, Manipulace s entitami a Uživatelské procesy.

4.1 Validace HTTP požadavku

Tento proces je vykonán pro každou takovou stránku systému, která kromě přihlášení uživatele vyžaduje také, aby měl patřičné oprávnění. Proces je spouštěn ve vrstvě middleware, o které pojednává kapitola 5.1 (Architektura informačního systému).

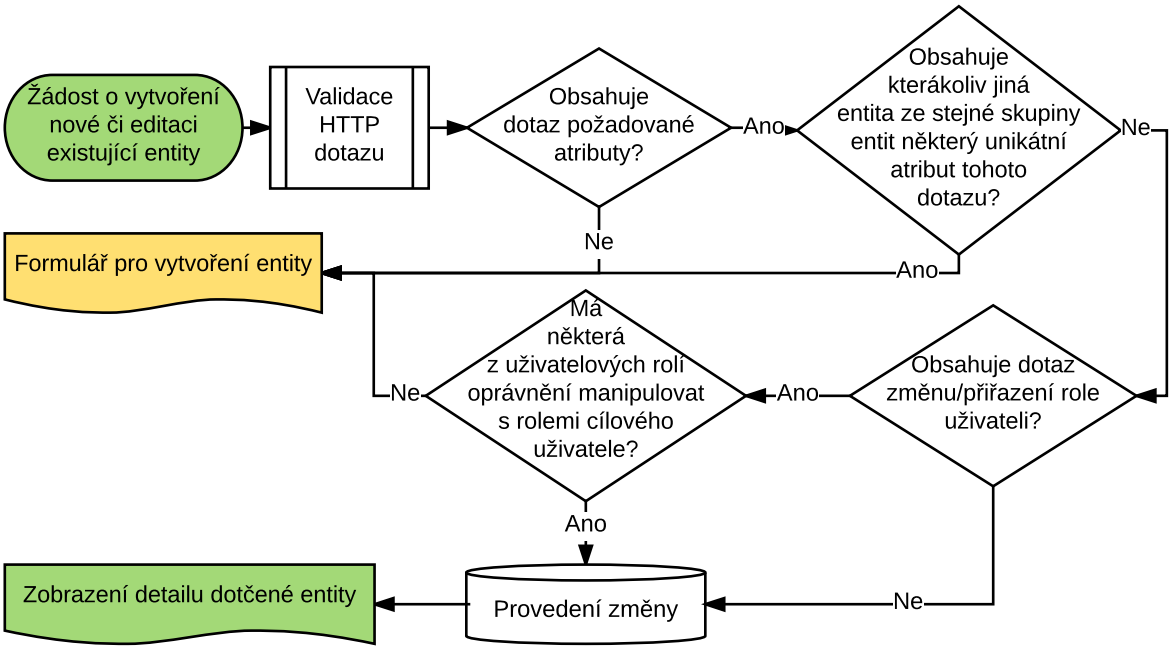


Obr. 4.2: Procesní model validace HTTP požadavku

4.2 Manipulace s entitami

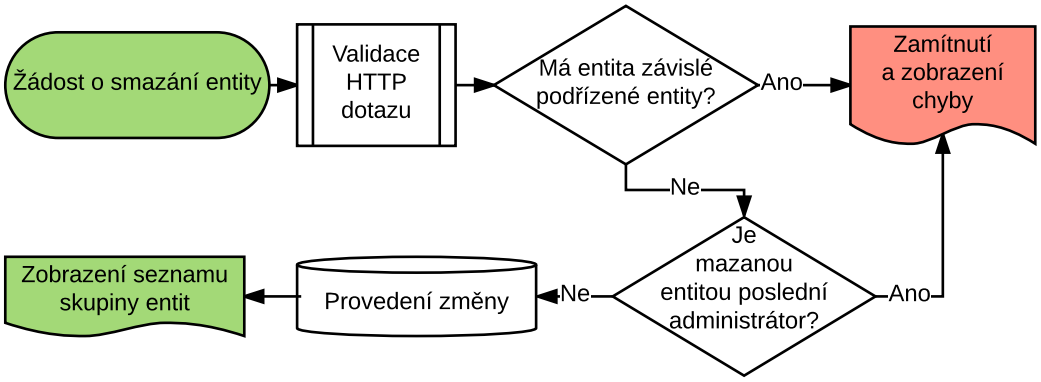
Sem jsou zařazeny procesy týkající se manipulace s entitami. Jsou jimi smazání a změna entity Viz obrázky 4.3 a 4.4.

4.2.1 Změna entity



Obr. 4.3: Procesní model změny entity

4.2.2 Smazání entity



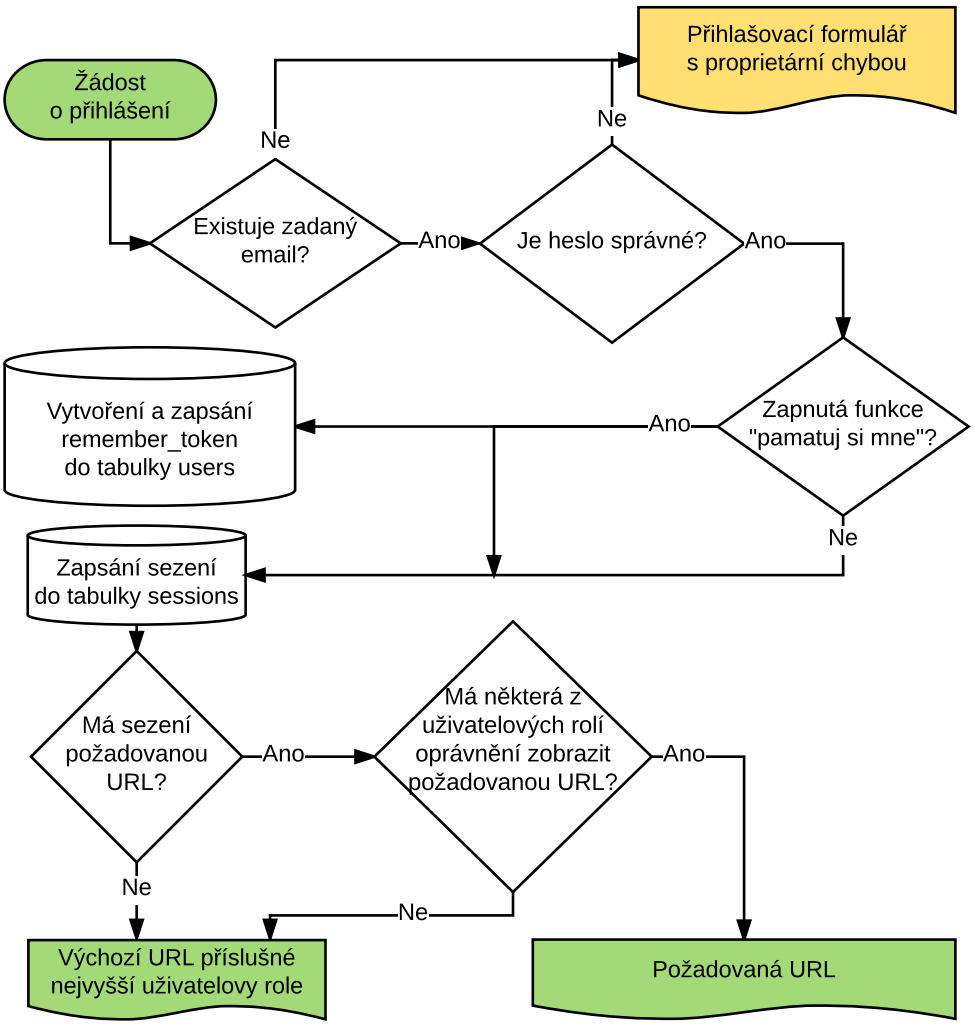
Obr. 4.4: Procesní model smazání entity

4.3 Uživatelské procesy

Tyto procesy popisují akce, které může vykonávat jakýkoliv uživatel IS. Neřadí se mezi ně editace vlastní entity, neboť tento proces je již vysvětlen v podkapitole 4.2.1 (Změna entity).

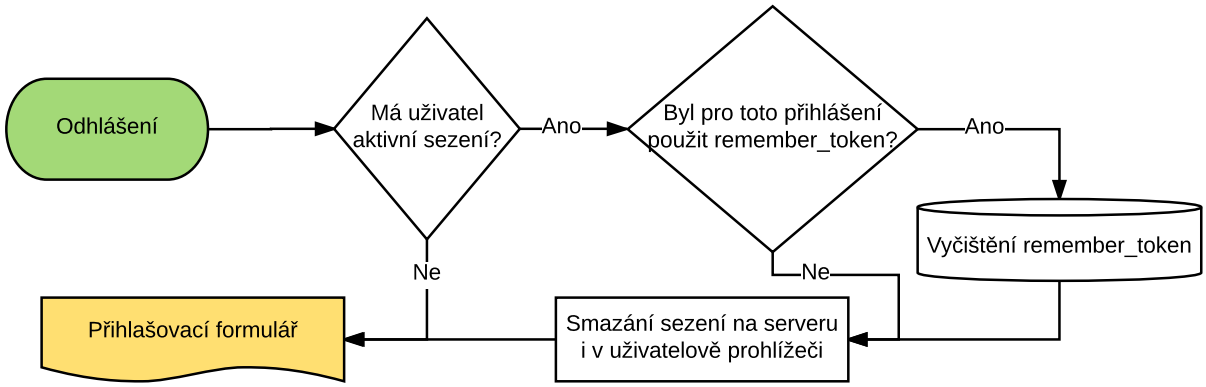
Prvním z nich je vývojový diagram vysvětlující proces přihlášení uživatele. Druhý model se zabývá jeho odhlášením. Třetí ukazuje, jak lze obnovit zapomenuté heslo.

4.3.1 Přihlášení uživatele



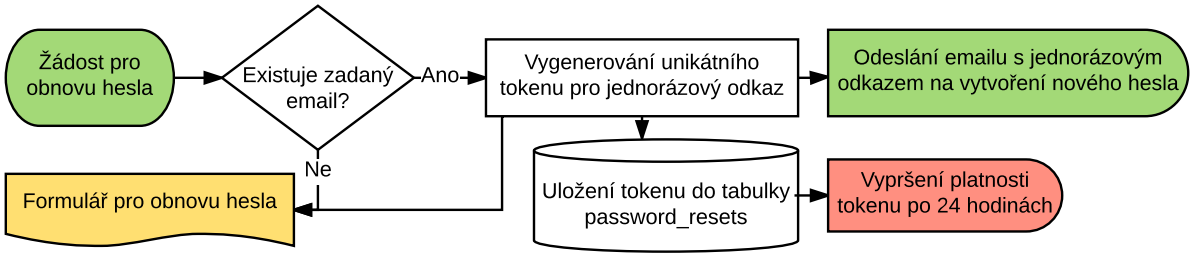
Obr. 4.5: Procesní model přihlášení uživatele

4.3.2 Odhlášení uživatele



Obr. 4.6: Procesní model odhlášení uživatele

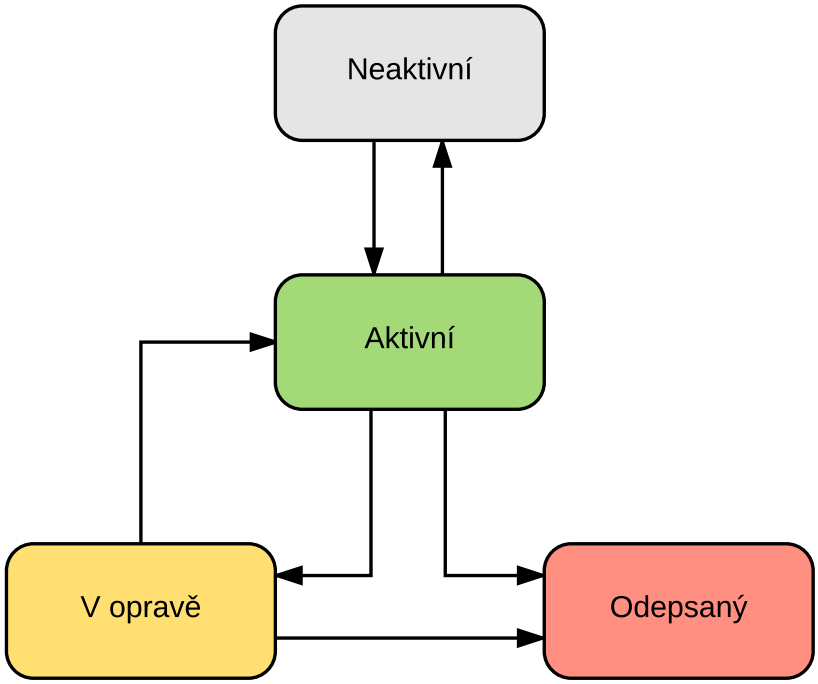
4.3.3 Obnova zapomenutého hesla



Obr. 4.7: Procesní model obnovy zapomenutého hesla

4.4 Životní cykly entit

Každá evidovaná entita může nabývat několika stavů, jak již bylo uvedeno v kapitole 3.4.1 (Skupina *Systémové tabulky*). Přechody mezi těmito stavy definuje následující diagram na obrázku 4.8:



Obr. 4.8: Procesní model životních cyklů entit

Každý přechod stavu má své odpovídající oprávnění, které musí uživatel mít, aby mohl přechod uskutečnit. Oprávnění jsou uživatelům přiřazována pomocí rolí, které obsahují určitý seznam oprávnění, viz podkapitola 1.2.1 (Uživatelské role a oprávnění).

Dle navržených rozdělení oprávnění platí, že veškeré možné přechody mezi stavy `aktivní`, `v opravě` a `odepsaný` může provádět pouze administrátor. Přechody mezi stavy `aktivní` a `neaktivní` může provádět správce zařízení a administrátor.

Auditor nemá přidělené žádné oprávnění k provedení jakéhokoliv přechodu.

Dále platí, že při vytvoření nové entity, se tato entita automaticky dostává do stavu `neaktivní`.

5 IMPLEMENTACE INFORMAČNÍHO SYSTÉMU

V této části se budu zabývat implementací systému. Jedná se o dokumentaci programového kódu IS.

5.1 Architektura informačního systému

IS implementuje klasickou architekturu MVC (Model-View-Controller) [24], [25].

Model představuje instanci objektu, která je typicky vytvořena transformací jednoho řádku jedné tabulky z relační databáze. Avšak v případě tohoto IS, jenž umožňuje evidovat entity veškerých tříd v jedné tabulce, s příslušnými parametry v jiných tabulkách, se inicializace modelu výrazně komplikuje. Jak inicializace takového modelu probíhá, vysvětlím v podkategorii 5.1.1 (Sekvenční UML diagram zobrazení entity).

View je abstrakcí, která koncovému uživateli poskytuje výsledný pohled či vzhled obrazovky. Typicky obsahuje HTML5, CSS3 a JavaScript (ECMA Script 6) definice pro uživatelův prohlížeč, který pak na základě těchto definic vytvoří cílový uživatelský zážitek.

Controller pak zpracovává vstupní uživatelovy HTTP požadavky a zároveň je zodpovědný za správné vytvoření modelu dle tohoto požadavku. Pokračuje případnou agregací vytvořeného modelu (výpočty, jejichž výsledky není možné z DB získat přímo) a jeho předáním do příslušné vrstvy View.

IS kromě MVC architektury používá i její rozšíření implementované frameworkem Laravel [17]. Rozšíření umožňuje ještě více zefektivnit vývoj aplikace. Jedná se zejména o *Routing* (možno přeložit jako směrování), *Middleware* (volně přeloženo jako prostředník) a *Facades* (velmi volně přeloženo jako průčelí).

Vrstva *Routing* jasně definuje vzory URL adres, které aplikace dokáže obsloužit a zároveň který konkrétní *Controller* je má zpracovat.

Middleware vrstva zajišťuje vykonávání předem daného souboru instrukcí nad vstupním požadavkem ještě než se tento požadavek předá konktérní instanci vrstvy *Controller*. Leží tedy mezi vrstvami *Routing* a *Controller*. Jeden soubor instrukcí typicky obsahuje nějakou podmínku, která způsobí, že se požadavek do cílové *Controller* třídy ani nedostane, ale přesměruje se na jinou.

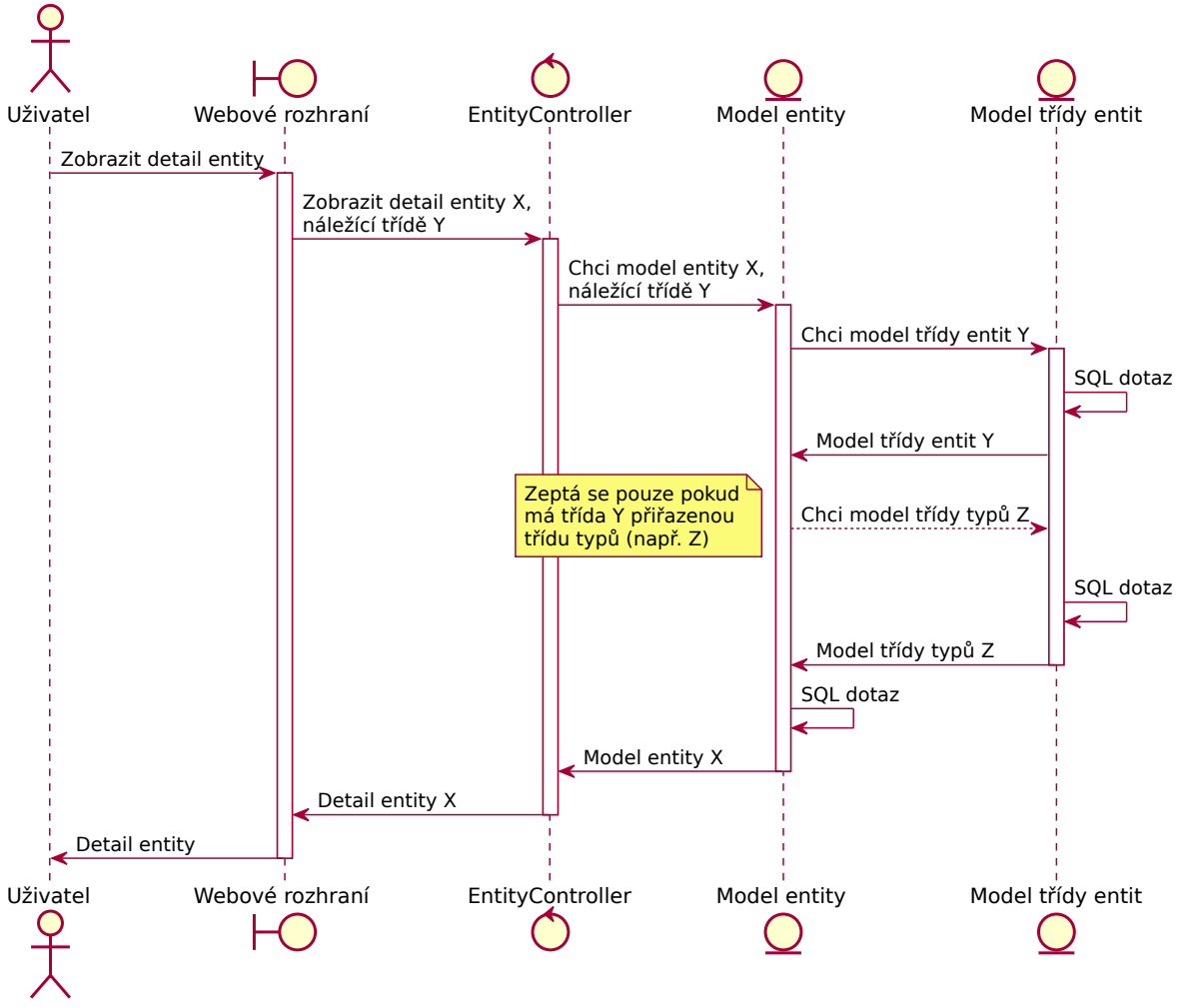
Tímto způsobem je v IS zajišťován proces Validace HTTP požadavku, jež je popsán v podkapitole 4.1. Pokud validace neuspěje, *Middleware* vyhodí výjimku,

kterou zachytí jiná vrstva Laravelu a ta uživatele přesměruje na záložní chybovou stránku.

Facades je pak koncept, který zasahuje do všech vrstev architektury. Poskytuje přístup ke statickým metodám a atributům často užívaných tříd. Její největší výhodou je, že umožňuje používat třídy, které v konfiguraci byly označeny za globální, aniž by bylo potřeba specifikovat úplný název třídy, včetně jejího jmenného prostoru. Dalo by se tedy říct, že zavádí globání prostor tříd.

5.1.1 Sekvenční UML diagram zobrazení entity

Berme prosím v úvahu, že se nejedná o zobrazení entity, která je umístěním, výrobcem, či dodavatelem, neboť tyto entity mají svou vlastní tabulku se všemi potřebnými parametry. Následující UML se tedy týká všech ostatních tříd entit.



Obr. 5.1: Sekvenční UML diagram Zobrazení detailu entity

Z diagramu je patrné, že inicializace modelu entity jen za účelem jejího zobrazení vyžaduje až 3 SQL dotazy.

První slouží nejen ke zjištění, zda třída zobrazované entity umožňuje vlastním entitám mít umístění, dodavatele či výrobce, ale i ke zjištění všech možných parametrů, které tato třída umožňuje entitám o sobě uchovávat. To je skutečně nutné, neboť bychom bez toho nevěděli, ve kterém datovém sloupečku tabulky `entity_parameters` bychom měli uložená data dané entity hledat. Kromě toho také první SQL dotaz žádá o identifikátor spjaté entity tříd, který může, ale ani nemusí být uveden. Nebude uveden například u definice třídy entit `VLAN`, neboť k této třídě vlivem aplikace druhé normální formy nevznikla žádná tabulka typů.

Ke druhému SQL dotazu ani nedojde, pokud třída zobrazované entity nemá spjatou třídu typů. Pokud ale k němu dojde, tak je jeho účelem získat všechny parametry této třídy typů a zjistit, zda svým entitám umožňuje mít výrobce. Hlavním důvodem pro sběr těchto parametrů je skutečnost, že se v detailu entity mají zobrazit hodnoty nejen svých parametrů, ale i hodnoty všech parametrů příslušného typu.

Metadata získaná předchozími SQL dotazy se použijí pro vytvoření posledního dotazu, jehož cílem je obstarat hodnoty všech parametrů entity a příslušného typu entity. Včetně umístění, dodavatele a výrobce, pokud to třída entity umožňuje. Spuštěním a zpracováním tohoto finálního dotazu je inicializace modelu hotova.

5.2 Atomicita SQL dotazů

U každého SQL dotazu hrozí, že selže. Příčin může být spousta. Chyba v kódu, nedostatek místa v úložišti databáze, nedostatek paměti RAM a mnoho dalších. V takových případech je nutné, aby data uložená v databázi zůstala v konzistentním stavu. Zde přichází na řadu transakční model zvoleného formátu úložiště – `InnoDB`.

5.2.1 Vkládání nových a editace stávajících entit

U vytváření nových entit, jejichž úplná definice je tvořena více než jednou tabulkou hrozí, že se dostanou do nekonzistentního stavu. U editace stávajících stejně tak. Proto jsou veškeré změnové operace entit zapouzdřeny do transakce v jejím výchozím nastavení. To znamená, že se automaticky vrátí všechny provedené změny do stavu před spuštěním transakce, pokud se jediná její operace nepovede.

5.2.2 Zobrazení entit

U zobrazení entit v zásadě není potřeba zajišťovat atomicitu skládání dat z více tabulek v případě, když k této operaci stačí spustit jediný SQL dotaz. Ovšem v případě, že se jedná o sekvenci různých SQL dotazů, jejichž výsledek je potřeba prezentovat uživateli IS, pak je zapotřebí atomicitu samozřejmě zajistit.

To je možné provést tak, že budou všechny SQL dotazy spuštěny v jediné transakci izolační úrovně `serializable` s vypnutou funkcí `autocommit` [26].

Příkladem zobrazení entity, která nepotřebuje více než jeden dotaz, může být zobrazení konkrétního umístění. Detail této entity sice obsahuje i seznam všech tříd entit, které na umístění odkazují, ale všechna tyto data byla z databáze vytažena jediným SQL dotazem, takže atomicitu zde zajišťuje již sama databáze při spojování informací navázaných tabulek.

Pro příklad zobrazení entity, která potřebuje více než jeden dotaz, a tím i zabezpečení pomocí transakce, se můžeme obrátit na podkapitolu 5.1.1 (Sekvenční UML diagram zobrazení entity).

5.3 Webové uživatelské rozhraní

Mělo by být navrženo pokud možno v co nejjednodušší formě, aby se v něm uživatelům dobře orientovalo. Zároveň by mělo obsahovat prvky, jejichž chování je dobře předvídatelné a co nejméně matoucí [27].

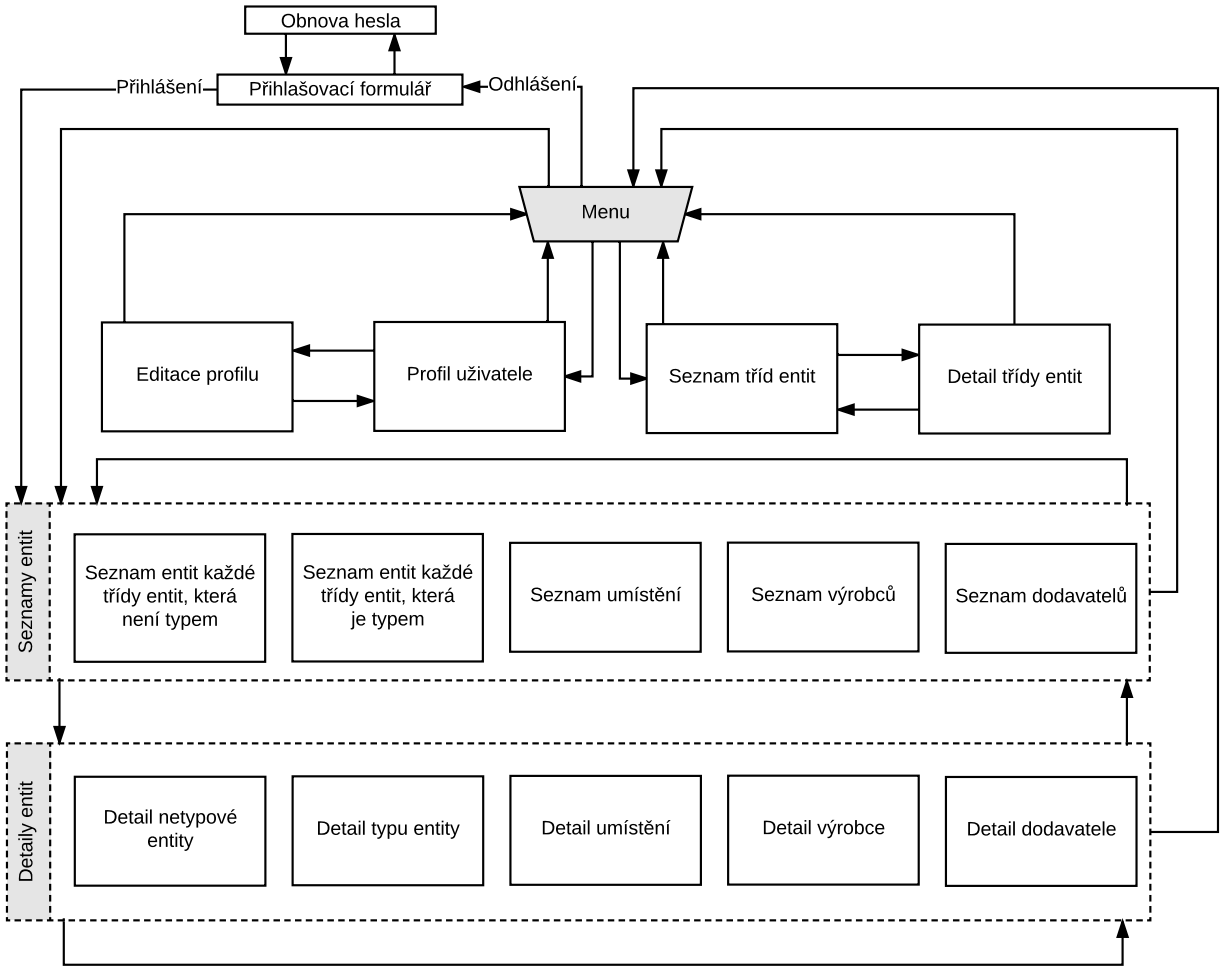
Proto jsem se rozhodl jít cestou nejmenšího odporu a do rozhraní přidával jen to, co bylo opravdu nutné. Výsledkem je průmyslový design stránek s jednoduchým horním menu, které uživateli umožňuje navigaci odkudkoliv napříč téměř celým IS.

5.3.1 Mapy přechodů obrazovek

Každá role má odlišné možnosti, co se týče navigace v systému. Začnu mapou přechodů role `Auditor`, ta jich má nejméně.

Auditor

Na následujícím schématu jsou znázorněny všechny možné obrazovky, jež může navštívit uživatel s rolí `Auditor`. Menu na schématu, ale ani na dalších, nepředstavuje samostatnou obrazovku, nýbrž prostředníka, ke kterému má uživatel přístup z každé obrazovky. Když menu pak ukazuje na některou obrazovku, znamená to, že na ni v podstatě ukazuje každá obrazovka, ze které má uživatel přístup k menu. Jinými slovy – kam ukazuje menu, mohou uživatelé po přihlášení přestoupit odkudkoliv.

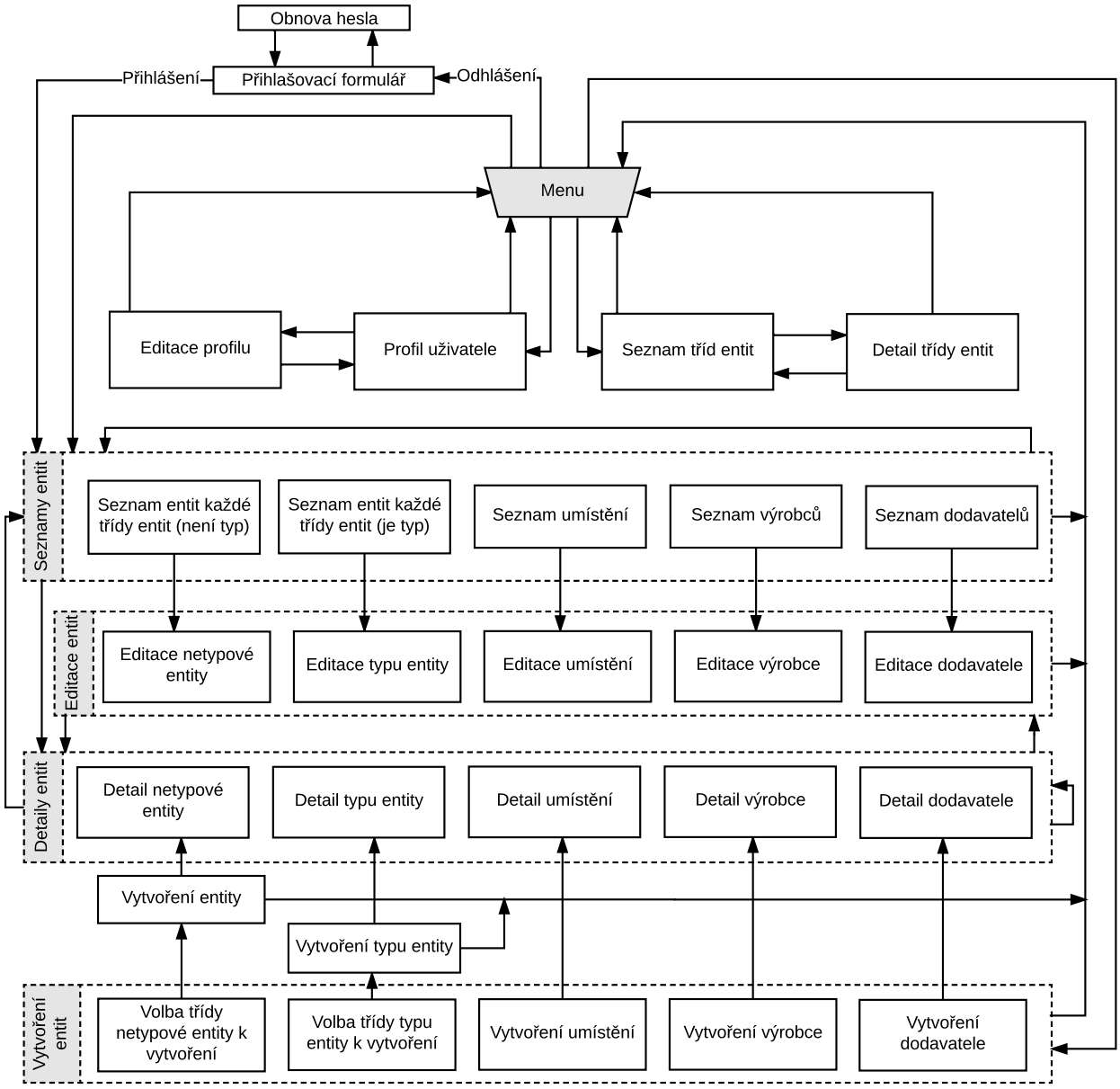


Obr. 5.2: Mapa přechodů obrazovek role `Auditor`

Všechny možné přechody mezi skupinami `Seznamy entit` a `Detaily entit` lze vyjádřit kartézským součinem jejich obrazovek. To je způsobeno skutečností, že je například v detailu entity třídy `Server` možné přejít na detail jiné entity, detail konkrétního umístění, seznam některé z třídy typů entit, atd. Tuto flexibilitu umožňují

hypertextové odkazy na každé obrazovce detailu či seznamu, jejichž hlavním cílem je umožnit uživateli procházet vazební strom závislostí napříč všemi entitami.

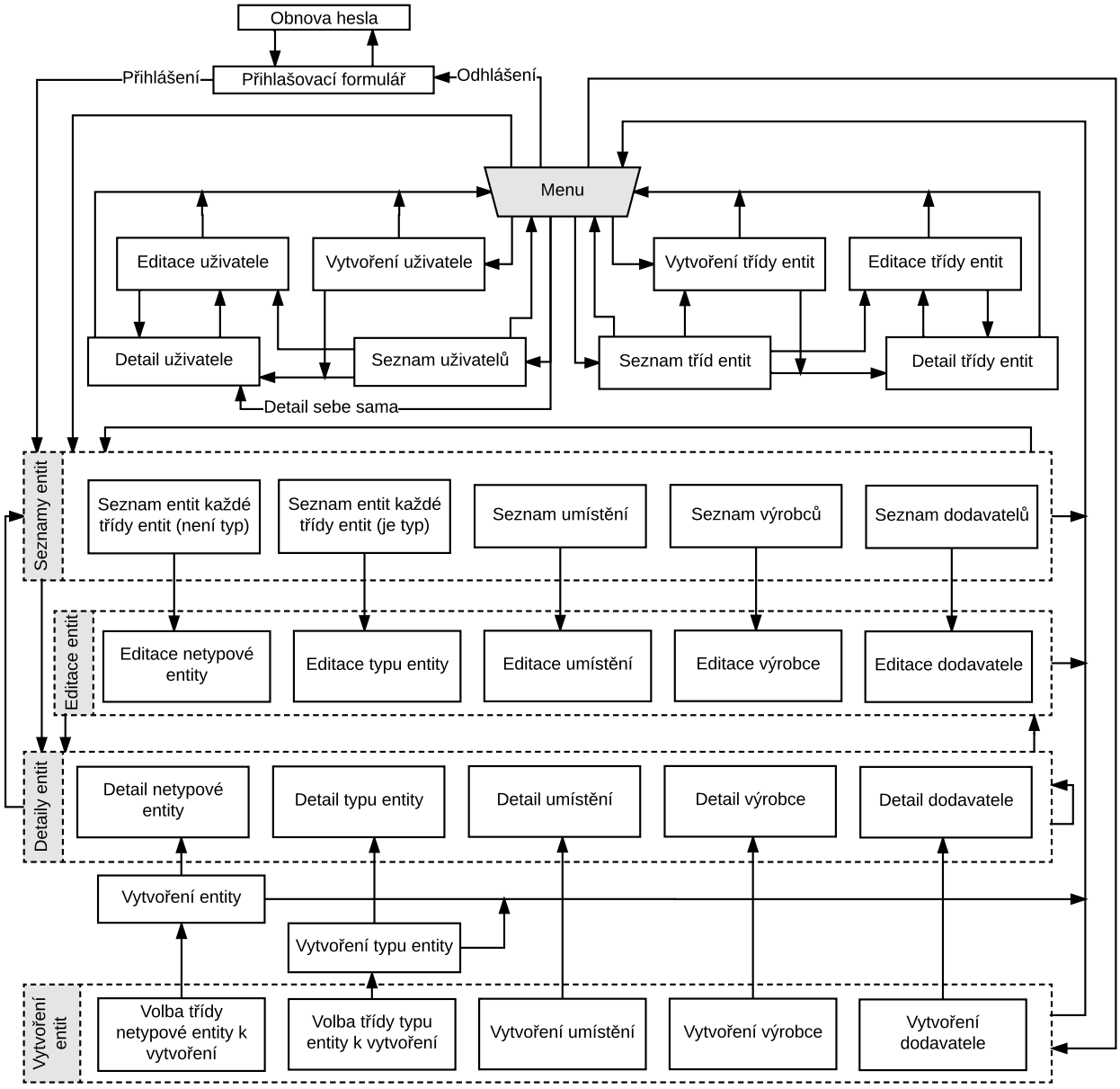
Správce zařízení



Obr. 5.3: Mapa přechodů obrazovek role `Správce zařízení`

Z diagramu lze vidět, že `Správce zařízení` má oproti roli `Auditor` možnost entity editovat a vytvářet.

Administrátor



Obr. 5.4: Mapa přechodů obrazovek role `Administrátor`

U tohoto schématu je patrné, že `Administrátor` může oproti roli `Správce zařízení` navíc ještě editovat a vytvářet třídy entit, ale také zobrazovat seznam všech uživatelů, jejich detailů, editovat je či vytvářet nové.

6 INSTALACE A KONFIGURACE APLIKACE

6.1 Instalace

Ke zprovoznění instance informačního systému je potřeba provést několik úkonů. Prvním z nich je extrakce přiloženého zip souboru do umístění, odkud bude mít webový server oprávnění číst. Může to být například `/www/entities_logger`.

6.1.1 PHP 7

Programový kód je napsán v jazyce PHP verze 7. Některé syntaxe této verze bohužel nejsou zpětně kompatibilní se staršími verzemi, a proto je nutné mít nainstalovaný interpret pro verzi 7.

Interpret navíc potřebuje mít nainstalované a povolené tyto moduly: `common`, `mbstring`, `xml` a `mysql`

6.1.2 NPM

NPM je program, který v případě tohoto IS slouží k překladu z Node.js na ECMAScript 5. Jeho instalace je potřeba také proto, že IS používá JavaScript knihovny třetích stran (včetně Vue frameworku) o jejichž stažení se NPM také postará. Je potřeba jej mít nainstalovaný alespoň ve verzi 4.2.0 [1]. Tomu odpovídá Node.js v7.

Po instalaci NPM je v adresáři IS třeba spustit příkazy `npm udpate` a `npm install`, které stáhnou všechny Node.js závislosti aplikace. Posledním krokem u NPM je překlad do ECMAScript 5, který se provede spuštěním příkazu `npm run dev`. Pro tyto operace budete potřebovat mít volných alespoň 800 MB RAM.

6.1.3 Composer

PHP Composer je správce závislostí, který se stará o stažení všech PHP knihoven třetích stran [2]. Po jeho instalaci spusťte příkaz `composer install`, opět v adresáři IS.

[1] NPM a Node.js ke stažení z URL `https://nodejs.org/en/download/package-manager/`

[2] PHP Composer ke stažní na URL `https://getcomposer.org/`

6.1.4 Webový HTTP server

K tomu, aby mohl být IS vystaven na určitém síťovém portu pro možnost komunikace s klientem za použití protokolu TCP, je potřeba nastavit libovolný webový HTTP server, jež bude uživatelské požadavky směrovat na hlavní soubor informačního systému. Tím je skript *index.php* v adresáři *public*. Stará se o inicializaci frameworku a předání vstupního požadavku vrstvě `Routing`.

V příloze se zdrojovým kódem naleznete příklad pro konfiguraci serveru `Apache httpd` pod názvem `apache.conf`. V něm je potřeba doplnit doménové jméno serveru namísto všech výskytů *ENTITIES_LOGGER_HOSTNAME*, tedy například `www.informacni-system.cz`. Pak ještě výskyty *DOCUMENT_ROOT* nahradit za absolutní cestu do adresáře *public*, tedy například `/www/entities_logger/public`.

Přiložená konfigurace je navíc přednastavena pro TLS šifrování, tekže očekává, že v adresáři `/etc/ssl/private/` nalezne šifrovací klíč a veřejnou část certifikátu pod názvy `entities-logger.key` a `entities-logger.pem`. Pro návod, jak klíče vygenerovat, se obrattně na přiložený soubor `INSTALL.md`, který popisuje instalaci systému v jazyce BASH.

V případě, že se rozhodnete pro `Apache httpd`, nezapomeňte nainstalovat a povolit jemu příslušný modul pro PHP 7. Pak ještě povolit moduly `ssl` a `rewrite`.

Povolení ukládání dat webovým serverem

Web server si do souborového systému ukládá metadata, která potřebuje ke správnému fungování. Proto je také potřeba mu tento zápis umožnit. Zde uvádím příklad, jak na to, když webový server k zápisu používá uživatele `www-data`.

Výpis 6.1: Udělení povolení k zápisu metadat IS webovým serverem

```
cd /www/entities_logger
sudo chown www-data bootstrap/cache/ -R
sudo chown www-data storage/ -R
```

6.1.5 MySQL

Po instalaci MySQL serveru je potřeba vytvořit databázového uživatele a jemu příslušnou databázi. Pak následuje vytvoření tabulek a jejich naplnění s iniciálním nastavením. Všechny tyto úkony zajišťuje přiložený BASH skript `init-db.sh`, který vytvoří schéma s názvem *entities_logger*, uživatele s loginem *entities-logger*, vygeneruje mu heslo, pokud nezadáte jinak, nastaví mu veškerá nutná oprávnění

k databázi a pak vytvořeného uživatele, včetně jeho hesla, zapíše do konfiguračního souboru `.env`. Na konec pak spustí úvodní migraci databáze, včetně jejího zasetí, které vysvětlím v podkapitole 6.2 (Konfigurace). Přiložený skript lze snadno upravit pro změnu názvu schématu, přihlašovacího jména uživatele, doménového jména cílové MySQL, jejího portu či přihlašovacího jména uživatele, pod kterým se nové údaje nastaví.

6.2 Konfigurace

Po úspěšné instalaci IS je vhodné jej ještě správně nakonfigurovat. Základní konfigurace se provádí v souboru `.env`. Rozšířená, více komplexní a podrobná konfigurace pak v adresáři `config`. Nebudu rozebírat jednotlivé konfigurační soubory tohoto adresáře dopodrobna, každá totiž má dostatek své vlastní dokumentace.

Vrátím se ale k základní konfiguraci. V té se dá nastavit například první administrátorský účet, který se má vytvořit po instalaci. Toto nastavení se ale bere v potaz pouze při úvodním naplňování databáze, takže každá změna uvodního nastavení administrátorského účtu vyžaduje přepsat celou databázi. To lze provést spuštěním obnovy databáze migrací a zasetím.

Výpis 6.2: Obnova databáze migrací a zasetím

```
cd /www/entities_logger
php artisan migrate:refresh
php artisan db:seed
```

Úvodní zasetí databáze do ní vkládá veškeré třídy entit, veškeré stavy a možné přechody mezi nimi, veškeré role a jejich příslušné oprávnění a pak i vazby těchto oprávnění na možné přechody mezi stavy, včetně úvodního administrátorského účtu. Bez tohoto zasetí by IS byl nepoužitelný.

Všechna ostatní nastavení, jež lze ve kterékoliv konfiguraci učinit, mají zpravidla okamžitou účinnost bez nutnosti restartovat server či znovu naplňovat databázi. Jedná se například o nastavení MySQL připojení, emailového serveru pro odesílání mailů s odkazem na obnovení zapomenutého hesla, dobu trvání sezení, než vyprší a automaticky odhlásí neaktivního uživatele a mnoho dalších.

7 OVĚŘENÍ FUNKČNOSTI INFORMAČNÍHO SYSTÉMU

Na závěr jsem vyzkoušel, zda systém skutečně umožňuje evidovat veškeré parametry, specifikované v logickém datovém modelu, viz obrázek C.1.

Poté jsem začal testovat, zda uživatelé s jednotlivými rolemi mají skutečně jen ta oprávnění, o kterých pojednávám v podkapitole 1.2.1.

Také jsem ověřil, že skutečně lze mezi entitami vytvářet vazby dle možných nadřízených tříd entit. To se týká editace a vytváření jednotlivých podřízených entit. Dále jsem ověřil, že lze u seznamu entit každé třídy entit vidět hodnoty všech jejich parametrů tak, jak je předepsaly jejich třídy entit a to ve spojení hodnot parametrů příslušných typů entit. Stejně tak jsem ověřil, že lze vidět hodnoty těchto parametrů, i v detailech entit.

Dále jsem otestoval, že u entit lze přecházet mezi jejich stavy dle definovaných pravidel. Také jsem otestoval, že lze vytvořit novou entitu jakékoliv třídy entit. Posléze jsem vyzkoušel, jestli jde smazat existující entitu, pokud nemá závislé podřízené, a zároveň že ji nelze smazat pokud má závislé podřízené entity. Pak jsem ověřil, že kontrola na unikátnost parametru kterékoliv třídy entit funguje správně jak u vytváření, tak editace entity. Poté jsem ověřil, že funguje i filtrování v seznamech entit.

Dalším testem bylo ověření, že se při vytváření a editaci entit, jejichž třídy jim umožňují mít výrobce, dodavatele či umístění, automaticky v příslušných polích pomocí JavaScriptu doplňují vyhovující názvy výrobců, dodavatelů či umístění, když uživatel do nich začne psát.

Také jsem ověřil, že při zobrazení detailu třídy entit lze vidět i definice jednotlivých parametrů.

Následně jsem testoval, zda správně fungují hypertextové odkazy, jejichž účelem je snadné procházení detailu navázaných entit. Otestoval jsem také, že pokud jsem uživatelem, který má právo manipulovat s rolemi ostatních uživatelů, tak že nemůže změnit svou vlastní roli.

Posléze jsem otestoval, zda si lze skutečně obnovit heslo. Zároveň se všemi výše uvedenými testy došlo k testu, zda pro každou navštívenou stránku existují dvě jazykové mutace.

Všechny provedené testy byly úspěšné.

8 ZÁVĚR

Provedením systémové analýzy jsem zjistil konkrétní požadavky na systém.

Posléze jsem za programovací jazyk zvolil PHP s napojením na databázové úložiště MySQL. Po rozhodnutí, že bude použita relační databáze, jsem pokračoval návrhem datového modelu systému, jeho normalizací a nakonec zobecněním do formy dynamického datového modelu, jenž umožňuje evidovat prakticky jakékoliv entity.

Konkrétní požadavky systému jsem pak řešil za použití procesního modelování. Navrhnul jsem tak základní chování procesů informačního systému, která jsou kritická. Tím byl hotov jak procesní, tak datový model, včetně systémové analýzy informačního systému.

Následně jsem začal IS implementovat a přitom navrhovat grafickou podobu jednotlivých obrazovek. Po implementaci IS jsem vyrobil mapu přechodů obrazovek a detailně popsal jednotlivé obrazovky.

Aby bylo možné systém dále rozšiřovat, pokračoval jsem dokumentací projektu sepsáním architektury IS, kde jsem uvedl základní architektonické koncepty pro efektivní vývoj dle obecně uznávaných postupů [24], [25]. Zabýval jsem se zde mimo jiné i atomicitou databázových dotazů a částečně i konvencemi zvoleného frameworku.

Výsledkem této práce je tedy funkční, zdokumentovaný, rozšiřitelný a z velké části odladěný informační systém, který umožňuje evidovat entity počítačové sítě, zobrazovat si je, měnit jejich evidenční údaje, spravovat přístupy a uživatelské role jednotlivých uživatelů, obnovovat zapomenuté heslo a obecně mít celkový přehled o všech prvcích na jednom místě, který je umocněn možností zobrazovat vzájemné relace jednotlivých entit s rozkliknutelnými odkazy na samotné entity, takže lze snadno procházet strom závislostí oběma směry.

Informační systém funguje a předpokládá se jeho další vylepšování a vývoj v průběhu jeho používání, mimo jiné na základě zpětných vazeb získaných od jeho uživatelů.

LITERATURA

[1] *Hubs Versus Switches – Understand the Tradeoffs* [cit. 25. května 2017] Dostupné z URL: <http://www.ccontrols.com/pdf/Extv3n3.pdf>.

[2] Rajaravivarma, V. *Virtual local area network technology and applications.* In System Theory, 1997., Proceedings of the Twenty-Ninth Southeastern Symposium on, pp. 49-52. IEEE, 1997.

[3] Perkins, Charles. *IP mobility support for IPv4, revised.* [online] (2010). [cit. 6. ledna 2017] Dostupné z URL: <http://bit.ly/is-feec-but-ipv4>.

[4] Deering, Stephen E. *Internet protocol, version 6 (IPv6) specification.* [online] (1998). [cit. 6. ledna 2017] Dostupné z URL: <http://bit.ly/is-feec-but-ipv6>.

[5] Niels Provos and David Mazières. *A Future-Adaptable Password Scheme* [online] (1999). [cit. 26. května 2017] Dostupné z URL: <https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf>.

[6] Graunke, Paul, Shriram Krishnamurthi, Steve Van Der Hoeven, and Matthias Felleisen. *Programming the Web with High-Level Programming Languages.* [online] In European symposium on programming, pp. 122-136. Springer Berlin Heidelberg, 2001. [cit. 5. ledna 2017] Dostupné z URL: <http://bit.ly/is-feec-but-euro-symposium>.

[7] Grinberg, Miguel. *Flask Web Development: Developing Web Applications with Python.* [online] O'Reilly Media, Inc., 2014. [cit. 5. ledna 2017] Dostupné z URL:<http://bit.ly/is-feec-but-python>

[8] Ford, Neal. *Art of Java Web Development: Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, WebWork.* (2003). [cit. 5. ledna 2017]

[9] Castro, Elizabeth. *Perl and CGI for the World Wide Web: Visual quickstart guide.* Peachpit Press, 2001. [cit. 5. ledna 2017]

[10] Herron, David. *Node Web Development.* [online] Packt Publishing Ltd, 2013. [cit. 5. ledna 2017] Dostupné z URL:<http://bit.ly/is-feec-but-node>

[11] Glass, Michael K., Yann Le Scouarnec, Elizabeth Naramore, Gary Mailer, Jeremy Stolz, and Jason Gerner. *Beginning PHP, Apache, MySQL Web Development.* John Wiley & Sons, 2004. [cit. 5. ledna 2017]

[12] Platt, David S. *Introducing Microsoft. Net.* Microsoft press, 2002. [cit. 5. ledna 2017]

[13] Hartl, Michael. *Ruby on rails tutorial: learn Web development with rails.* Addison-Wesley Professional, 2015. [cit. 5. ledna 2017]

[14] Motta, Quildreen. *Which programming languages are front-end and which ones are back-end?* [online] [cit. 5. ledna 2017] Dostupné z URL:<http://bit.ly/is-feec-but-quildreen-motta>

[15] Lerdorf, Rasmus, Kevin Tatroe, and Peter MacIntyre. *Programming Php.* O'Reilly Media, Inc., 2006. [cit. 5. ledna 2017]

[16] Lancor, Lisa, and Samyukta Katha. *Analyzing PHP frameworks for use in a project-based software engineering course.* In Proceeding of the 44th ACM technical symposium on Computer science education, pp. 519-524. ACM, 2013. [cit. 5. ledna 2017]

[17] Bean, Martin. *Laravel 5 Essentials.* [online] Packt Publishing Ltd, 2015. [cit. 5. ledna 2017] Dostupné z URL:<http://bit.ly/is-feec-but-laravel-essentials>

[18] MySQL, A. B. *MySQL.* [online] (2001). [cit. 5. ledna 2017] Dostupné z URL: <http://bit.ly/is-feec-but-mysql-bible>.

[19] Charlie Schluting. *Networking 101: Understanding the Data Link Layer* [online] [cit. 25. května 2017] Dostupné z URL: <http://bit.ly/is-feec-but-data-link-layer>.

[20] Michal Valenta *DBS - Normální formy, normalizace* [online] [cit. 25. května 2017] Katedra softwarového inženýrství FIT, České vysoké učení technické v Praze, (2010) Dostupné z URL: <http://bit.ly/is-feec-but-db-normalization>.

[21] Ondřej Fišar *Správa MySQL serveru* [online] [cit. 27. května 2017] Katedra počítačů, Fakulta elektrotechnická, České vysoké učení technické v Praze, (2009) Dostupné z URL: <https://dip.felk.cvut.cz/browse/pdfcache/fisaro1_2009bach.pdf>.

[22] Axmark, D., & Widenius, M. *MySQL 5.7 reference manual. - 11.1.1 Numeric Type Overview* [online] Redwood Shores, CA: Oracle, (2017). [cit. 27. května 2017] Dostupné z URL: <https://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html>.

[23] Kelly Halseth Regional Coordinator, Forms Management & Production *Process Modelling & Mapping: The Basics* [online] David Thompson Health Region [cit. 6. ledna 2017] Dostupné z URL: <http://bit.ly/is-feec-but-process-modelling-basics>.

[24] Chris Pitt. *Pro PHP MVC.* [online] Apress, 2012 [cit. 24. května 2017] Dostupné z URL: <http://bit.ly/is-feec-but-mvc>.

[25] Borek Bernard *Úvod do architektury MVC* [online] zdroják.cz, 2009 [cit. 26. května 2017] Dostupné z URL: <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc>.

[26] Axmark, D., & Widenius, M. *MySQL 5.7 reference manual. - 14.5.2.1 Transaction Isolation Levels* [online] Redwood Shores, CA: Oracle, (2017). [cit. 27. května 2017] Dostupné z URL: <https://dev.mysql.com/doc/refman/5.7/en/innodb-transaction-isolation-levels.html>.

[27] Jesse James Garrett. *Elements of User Experience,The: User-Centered Design for the Web and Beyond* [online] Pearson Education, (2010). [cit. 27. května 2017] Dostupné z URL: <https://books.google.cz/books?id=9QC6r5OzCpUC&dq=he+Elements+of+User+Experience:+User-Centered+Design+for+the+Web+and+Beyond>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

LDAP	The Lightweight Directory Access Protocol
framework	aplikační rámec
HTML	HyperText Markup Language
CSRF	Cross-site Request Forgery
ER	entitně vztahový model - entity relationhip model
DB	databáze
MySQL	relační SQL databáze
SQL	strukturovaný dotazovací jazyk - structured query language
IS	informační systém
FEKT	Fakulta elektrotechniky a komunikačních technologií
SAP	protokol pro vyhledávání a oznamování služeb v síti - service advertising protocol
VLAN	virtuální LAN
LAN	místní síť - local area network
SSD	solid-state drive
RAM	random access memory
RAID	vícenásobné diskové pole nezávislých disků - redundant array of independent disks
Wi-Fi AP	Wi-Fi přístupový bod - Wi-Fi access point
Wi-Fi	standard IEEE_802.11
WLAN	bezdrátová LAN - wireless LAN
transceiver	síťový prvek, který je vysílačem a přijímačem zároveň - z angl. spojení transmitter a receiver
switch	síťový přepínač
MVC	Model-View-Controller

view	ve spojení s MVC - architektonická vrstva pohledů
controller	ve spojení s MVC - architektonická vrstva řadičů, či jeden konkrétní řadič
middleware	volně přeloženo jako prostředník
routing	možno přeložit jako směrování
facade	velmi volně přeloženo jako průčelí
API	aplikační programovací rozhraní - Application Programming Interface
entita	je jeden konkrétní prvek, který IS eviduje
třída entit	jinými slovy kategorie entit - v logickém modelu je každá tabulka samostatnou třídou entit; například `Procesor` či `Umístění`
třída typů entit	je podmnožinou třídy entit - existuje jen kvůli druhé normální formě; například `Typ procesoru`

SEZNAM PŘÍLOH

A SOUBORY NA CD

Tato práce v souboru pod názvem `BP.pdf`.

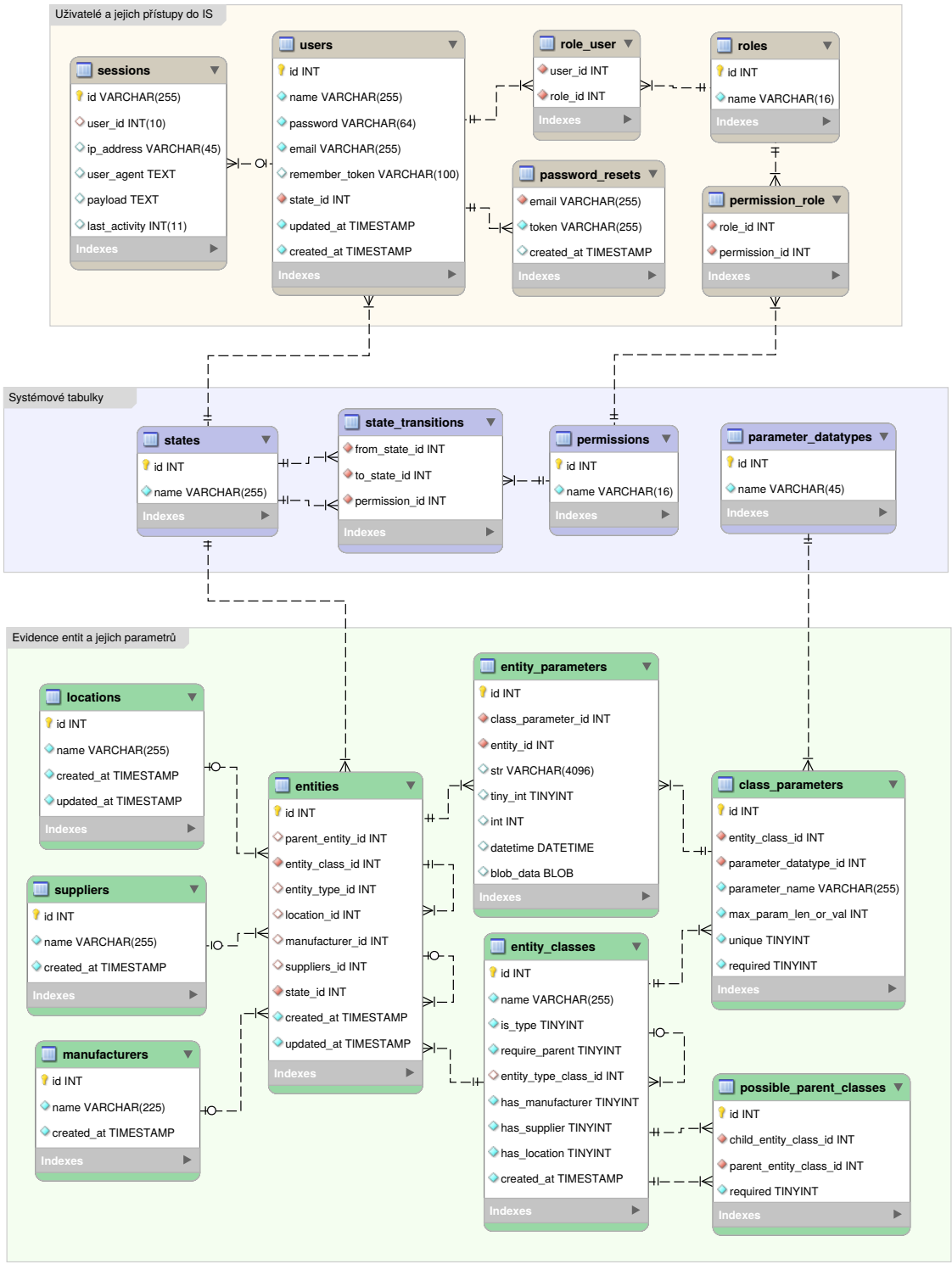
Inicializační skript pro databázi `init-db.sh`.

Instalační instrukce v jazyce BASH `INSTALL.md`.

Exemplární konfigurační soubor `apache.conf` pro webový server `Apache httpd`.

Zdrojové soubory IS v adresáři `entities_logger`.

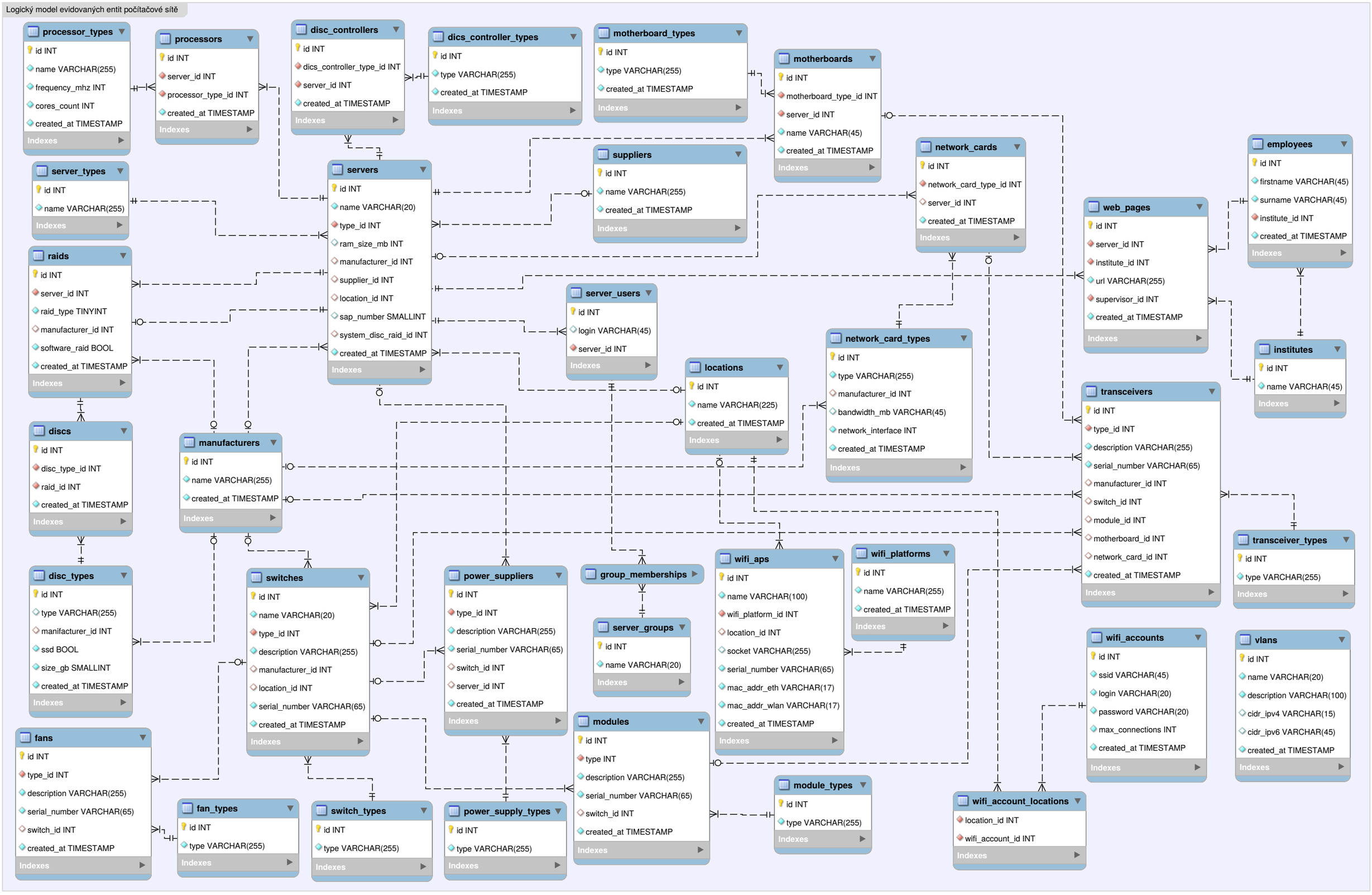
B CELÝ DATOVÝ MODEL



Obr. B.1: ER diagram informačního systému

C LOGICKÝ ER DIAGRAM IS

Viz zadní kapsa, kde logický datový model IS naleznete z důvodu nadměrných rozměrů schématu



Obr. C.1: Logický ER diagram IS