Systém pro analýzu dat operátorů a obráběcích strojů ve strojírenské výrobě

Bc. Roman Sucháček

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Roman Sucháček**
Osobní číslo:	**A15375**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Systém pro analýzu dat operátorů a obráběcích strojů ve strojírenské výrobě**
Téma anglicky:	**A System for Machine and Operator Data Analysis in Engineering Production**

Zásady pro vypracování:

1. **Nastudujte a v teoretické části popište základní technologie potřebné pro vývoj systému.**
2. **Proveďte sběr požadavků na informační systém a diskutujte možné způsoby řešení.**
3. **Stručně popište technologii třetí strany nasazenou pro sběr dat z obráběcích strojů a způsob ukládání dat do databáze.**
4. **Na základě požadavků navrhněte strukturu systému a možnosti jeho integrace s technologií pro sběr dat.**
5. **Dle návrhu implementujte funkční prototypovou aplikaci. Shrňte možnosti dalšího rozvoje a věnujte se také aspektům zabezpečení aplikace.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **Nette [online]. 2017 [cit. 2017-01-22]. Dostupné z: https://nette.org/cs/**
2. **CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. Vytváříme mobilní web a aplikace pro chytré telefony a tablety. Brno: Computer Press, 2013. ISBN 978-80-251-3763-5.**
3. **SHARKIE, Craig a Andrew FISHER. Responzivní webdesign: okamžitě. Brno: Computer Press, 2015. ISBN 9788025143841.**
4. **MariaDB. MariaDB [online]. 2016 [cit. 2017-01-30]. Dostupné z: https://mariadb.com/**
5. **Apache [online]. 2017 [cit. 2017-01-22]. Dostupné z: https://httpd.apache.org/**
6. **OWASP [online]. 2016 [cit. 2017-01-22]. Dostupné z: https://www.owasp.org/index.php/Main_Page**
7. **GASSTON, Peter. Moderní web. Přeložil Ondřej BAŠE. Brno: Computer Press, 2015. ISBN 9788025143452.**
8. **SOMMERVILLE, Ian. Softwarové inženýrství. Brno: Computer Press, 2013. ISBN 9788025138267.**

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 4.5.2014

.....
podpis diplomanta

ABSTRAKT

Tato práce se zabývá softwarovým řešením zpracování dat získaných ze strojů a dat, zadaných operátory pomocí vytvořené části systému. V rámci teoretické části práce jsou popsány použité technologie pro vývoj systému a provedená analýza systému včetně návrhu architektury. Následuje popis použité technologie pro sběr dat ze strojů. Praktická část práce je věnována návrhu architektury systému včetně návrhu databáze a výběru vhodné technologie pro provoz systému na mobilních zařízeních. Dále pak vlastní implementaci prototypové aplikace a shrnutí bezpečnostních opatření použitých v rámci implementace systému.

Klíčová slova: MariaDB, Apache, Nette, PHP, databáze, zpracování dat, webová aplikace, strojírenská výroba, sběr dat ze strojů, kontrola výroby, OEE

ABSTRACT

This thesis deals with the software solution of data processing obtained from the machines and the data entered by operator by using the created part of the system. Within the theoretical part of the thesis described the technologies used for development of the system and analysis of system, including design of architecture. Following by the description of the technology used to collect data from the machines. The practical part of the thesis is devoted to the design of the system architecture including database design and selection of an appropriate technology for using of the system on mobile devices. Furthermore, describes implementation of prototype applications and a summary of safety measures used in the implementation of the system.

Keywords: MariaDB, Apache, Nette, PHP, database, data processing, web application, engineering production, data collection of machines, production check, OEE

Chtěl bych touto cestou poděkovat panu Ing. Radkovi Valovi, Ph.D., za čas, věcné připomínky a rady, které mi během vedení této práce poskytnul, dále Ing. Luďkovi Blažkovi, ze společnosti ZNOJEMSKÉ STROJÍRNY s.r.o., za poskytnutý čas, ochotu a dobrou spolupráci při návrhu a testování systému.

Rád bych také poděkoval mé rodině za podporu, pochopení a trpělivost, bez kterých by tato práce vůbec nemohla vzniknout.

OBSAH

ÚVOD

V současné době je stále více výrobních společností, které využívají robotizovaná pracoviště, CNC obráběcí centra, automatizované výrobní linky a mnohá další zařízení. Vázanost personálu u těchto strojů není 100%, protože stroje jsou schopné po načtení instrukcí (programu), seřízení a nastavení pro danou činnost pracovat bez obsluhy i velmi dlouhou dobu. V duchu těchto okolností ovšem dochází k situacím, kdy obslužný personál zneužívá samostatné práce stroje a nevykonává v této době produktivní činnost, kterou by mohlo být například nastavení dalšího stroje na novou produkci, případně úklid pracoviště nebo příprava materiálu.

V souvislosti s tímto neproduktivním časem hledají výrobní společnosti způsob, jak dosáhnout maximálního využití strojů a personálu při současném zachování kvality. Efektivnější využití strojů a personálu zvýší produktivitu výroby a s nejvyšší pravděpodobností povede i ke snížení nákladnosti výroby. Jednou z mnoha společností, které tento problém řeší je výrobní strojírenská společnost ZNOJEMSKÉ STROJÍRNY s.r.o.[1]. Cílem vedení společnosti je dosáhnout maximálního efektivního využití strojů, a tím snížit náklady na výrobu a zároveň zkrátit dobu potřebnou na výrobu jednotlivých dílů. Podle vedení společnosti je jedním z možných řešení tohoto problému zavedení SW nástrojů. Data poskytovaná těmito SW pomáhají vedení společnosti odhalit nedostatky a skryté rezervy strojů i operátorů. Na základě informací získaných nasazenými SW nástroji může společnost přijmout pro vzniklé situace adekvátní opatření, která povedou ke zvýšení efektivity využití těchto výrobních zdrojů.

První částí tohoto řešení je nasazení SW, kterým je systém pro sběr dat ze strojů od společnosti TriDat[2]. Princip systému pro sběr dat ze strojů je založen na měření příkonu elektrické energie pomocí namontovaného měřícího transformátoru na přívodním napájecím kabelu. Snímač je umístěn v každém z vybraných strojů. Analogová data z těchto snímačů jsou následně zpracována a odesílána do databáze pomocí počítačové sítě. Druhou částí je níže popsaný systém, který je vytvořen v rámci této práce. Je realizován jako webová responzivní[3] aplikace pro sběr dat od operátorů, analýzu uložených dat ze strojů a

[1] Více na http://www.strojirny.cz/.
[2] Více na http://www.tridat.cz/.
[3] Responzivní - přizpůsobitelná různým velikostem obrazovky.

jejich vzájemnou konfrontaci. Data jsou ukládána na server do databáze. Vytvořený systém poskytuje relevantní výsledky z uložených dat, které jsou poskytovány vedení společnosti.

Náplní teoretické části práce je seznámení čtenáře s použitými technologiemi, bezpečností použitých technologií a popisem systému pro sběr dat ze strojů. Popis vlastního návrhu systému včetně implementace prototypové aplikace je popsán v praktické části.

V neposlední řadě je v závěru práce uvedena prezentace výsledků zpracovaných systémem, které jsou nejdůležitější částí pro vedení společnosti. Na základě výsledků získaných v průběhu výroby i zpětně, použitých pro vyhodnocení a porovnání, může vedení společnosti učinit patřičná nápravná opatření. Vytvořený softwarový nástroj v rukou vedení společnosti přináší zejména zvýšení efektivity využití strojního vybavení, zvýšení produktivity práce a v neposlední řadě jako důsledek dříve zmíněného i snížení nákladů na výrobu.

I. TEORETICKÁ ČÁST

1 WEBOVÉ TECHNOLOGIE

Ještě před několika lety bylo slovo „webové“ vždy spojováno zejména s pojmem „stránka“ tedy „webová stránka“. Původním účelem těchto webových stránek bylo převážně sdílet veřejný obsah, texty, obrázky a další informace pro velkou komunitu lidí, uživatelů, kteří mají k této webové stránce přístup. Přístup je zajištěn díky fenoménu jménem Internet tvořící celosvětovou síť. Postupem času však tento koncept již nebyl dostačující a začal se měnit. Statické webové stránky se měnily v dynamické, začaly s uživatelem interagovat a poskytovat konkrétní informace dle aktuálních požadavků uživatele. [1] Nastaveného trendu je i nadále využíváno. Webové stránky už nejsou určeny jen k prezentaci společností či jiného sdílení veřejných informací. Stále častěji se webové stránky více než na poskytování obsahu zaměřují na interaktivitu, a tak se z webových stránek staly spíše webové aplikace. Samozřejmě i webové stránky mají své místo a stále jsou velmi častým obsahem Internetu, případně jako směsice webových stránek a webových aplikací. Typickým příkladem může být webová stránka jako prezentace společnosti doplněná o internetový obchod v podobě webové aplikace. [1]

1.1 Výhody webových technologií

Možnost provozování aplikace v internetovém prohlížeči přináší pozitivní vlastnosti, jakými jsou dostupnost, univerzálnost a multiplatformní použití. Není nutné se omezovat na konkrétní operační systém, či na konkrétní hardware. Internetový prohlížeč je dostupný pro většinu operačních systémů a v případě webových aplikací v podstatě nahrazuje klientskou část aplikace. Pro provoz aplikace tak zpravidla stačí, aby uživatel používající webovou aplikaci měl k dispozici na svém zařízení internetový prohlížeč. [2] Absence nutnosti instalace aplikace na straně klienta přináší další výhodu, kterou je zjednodušení distribuce aktualizací webových aplikací ke klientovi. K provedení aktualizace webové aplikace postačí aktualizovat pouze serverovou část, a na straně klienta je tato aktualizace ihned dostupná. Tato vlastnost je velmi výhodná ve firemním prostředí lokální sítě, kdy zcela odpadá jakékoli fyzické provádění aktualizací na klientském zařízení. Výše zmíněný výčet výhod způsobuje nastalou situaci, kdy dochází k přesouvání dříve klientských aplikací (v podobě instalovaného klienta) na web do podoby webových aplikací. Samozřejmě tento vývoj nemá jen zmíněná pozitiva, ale tento trend sebou nese i negativa. [1]

1.2 Nevýhody webových technologií

Webový prohlížeč však není jen jeden, a díky tomu samozřejmě vznikají rozdíly v interpretaci webové aplikace, což způsobuje rozdíly nejen ve vzhledu dané aplikace, ale i v chování. Velmi důležitou podmínkou provozu je stabilní a dostatečně rychlé připojení k Internetu. Bez kvalitního připojení k Internetu nebo obecně k síti není možné webové aplikace provozovat. [2] Další nepříjemností jsou otázky bezpečnosti aplikace. Data webových aplikací jsou zpravidla přenášena v otevřené (čitelné) podobě veřejnou sítí – Internetem. Vzniká tedy riziko, že takto přenášená data může někdo zcizit a následně zneužít. Ale i mnohé další zranitelnosti, kterých je možné díky přístupu kohokoli k webové aplikaci zneužít. Je tedy nutné při vývoji webové aplikace na otázku bezpečnosti myslet a přijmout taková opatření, aby bylo riziko zneužití zejména citlivých dat aplikace minimální. [3]

2 ZVOLENÉ TECHNOLOGIE

Náplní kapitoly je stručně popsat kritéria pro výběr zvolené technologie použité k vývoji tohoto systému. Výběr zvolených technologií byl ovlivněn několika faktory. Jedním z nich je požadavek výrobní společnosti, aby náklady na licence použitých nástrojů a technologií byly co možná nejnižší, což bylo poměrně limitující. Tím se samozřejmě nabídka technologií splňujících tuto podmínku značně zmenšila. Dalším požadavkem ovlivňujícím výběr technologií je multiplatformní použití systému. Je zde předpoklad, že systém bude využíván na desktopových klientech s operačním systémem Microsoft Windows[4] a na mobilních zařízeních jako jsou tablety a mobilní telefony, na kterých převažuje operační systém Android[5]. Tyto zmíněné skutečnosti směřovaly k volbě právě webových technologií, které tuto podmínku splňují. A samozřejmě v neposlední řadě hrály ve výběru technologií roli i mé zkušenosti se zvolenými technologiemi. Pojďme tedy jednotlivá kritéria pro výběr použitých technologií popsat.

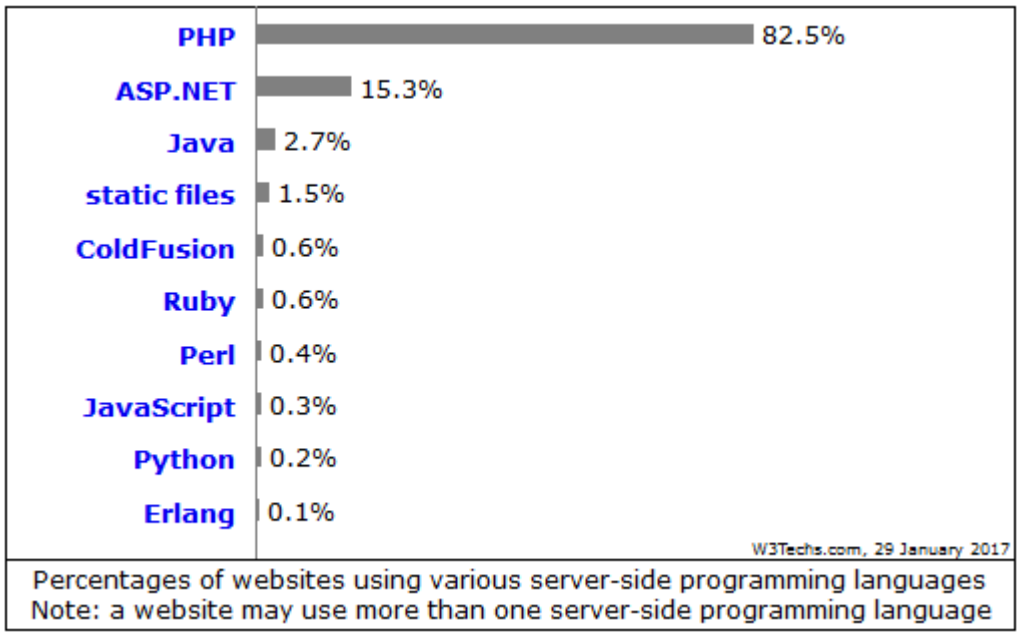
2.1 Výběr technologie

V minulé kapitole byla diskutována vhodnost volby webových technologií pro vývoj systému pro analýzu a sběr dat operátorů. Technologií pro vývoj webových aplikací však existuje více a je třeba vybrat vhodnou. Podle grafu uvedeného na (Obr. 1) je nejpoužívanějším programovacím jazykem pro vývoj webových aplikací s převahou PHP[6]. [4] PHP je velmi často používaný skriptovací jazyk pro vývoj dynamických webových aplikací. Syntaxe jazyka vychází z jazyka C, Java a Perl. [5] Skriptovací jazyk PHP je všestranně použitelný Open Source[7]. [6] Díky svobodné licenci je splněna podmínka společnosti na nízké ceny použitých technologií. V případě PHP jsou náklady na licence nulové. Je vhodným kandidátem pro vývoj systému, protože je to nejpoužívanější programovací jazyk pro webové aplikace (Obr. 1), a zároveň splňuje podmínku společnosti na nízké licenční náklady. [6]

Vývoj webové aplikace v čistém PHP je samozřejmě možný. Přináší výhody rychlejšího a pružnějšího kódu i možnost snadné přenositelnosti. Není zde vazba na knihovny třetích stran nebo licenční omezení. Naopak nevýhodou je nutnost napsat větší množství

[4] Windows – operační systém od společnosti Microsoft.
[5] Android – operační systém od společnosti Google.
[6] PHP – Hypertext Preprocesor – skriptovací programovací jazyk.
[7] Open Source – počítačový program s otevřeným zdrojovým kódem.

kódu než při použití frameworku. Nevýhodou je také složitější řešení aspektů bezpečnosti. [7]



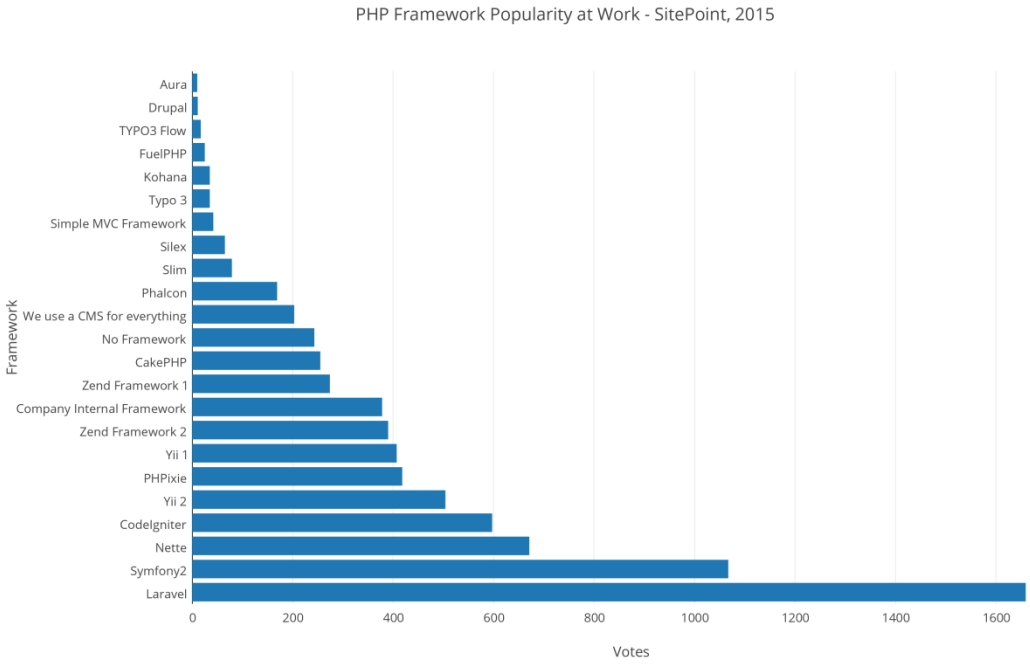
Obr. 1. Procentuální zastoupení programovacích jazyků na straně serveru. [4]

Při použití frameworku převažují spíše výhody nad nevýhodami. Jeho použití přináší menší redundanci kódu, pokročilé metody zabezpečení, řadu dostupných nástrojů a pomůcek jako součást knihoven. Frameworky jsou zpravidla založeny na nějaké architektuře (např. MVC[8]), díky které je kód rozdělen do několika vrstev. To přináší možnost pracovat na jednotlivých částech samostatně, například v týmu. Usnadňuje také testování. Nevýhodou frameworků je horší výkon resp. rychlost v porovnání s čistým PHP. Při použití frameworku je také nutné se jej naučit, ale jeho zvládnutí je obvykle možné v poměrně krátkém čase. [7] Na základě této konfrontace byl zvolen vývoj pomocí frameworku.

Při výběru vhodného frameworku bylo přihlíženo k několika kritériím. Prvním kritériem je již zmíněná cena licencí. Vybraný framework by měl být volně šiřitelný svobodný software. Framework by měl být oblíben mezi uživateli (programátory), což může částečně vypovídat o jeho použitelnosti a spolehlivosti. V ideálním případě by měl mít framework dobře zpracovanou dokumentaci, pokud možno v českém jazyce. Jak už bylo zmíněno, frameworků, založených na jazyce PHP existuje celá řada. Jejich oblíbenost je různá a

[8] MVC – Model View Controller – softwarová architektura.

v čase se mění. Na obrázku (Obr. 2) je výsledek průzkumu, který byl proveden v roce 2015. Týkal se porovnání oblíbenosti jednotlivých frameworků pro PHP. Průzkumu se zúčastnilo poměrně velké množství různých frameworků, kde se na třetím místě umístil framework Nette. V České a Slovenské republice se těší ještě větší oblíbenosti a podle průzkumu je dokonce na prvním místě. [8] Toto umístění hrálo při výběru velkou roli. Jeho tvůrci jsou z České republiky a celá komunita vývojářů je také převážně z České a Slovenské republiky. [9] To souvisí i s dostupností kvalitní dokumentace v českém jazyce a velmi silnou základnou, která se aktivně věnuje vývoji a zdokonalování frameworku Nette. Zpravidla jednou za čtvrtletí přichází nová verze frameworku. V době psaní textu byla aktuální verze 2.4. Je však již aplha verze 3.0.0. [10] Framework Nette splňuje také podmínku bezplatné licence, protože je šířen jako svobodný software pod licencí New BSD[9] a GPL[10]. [11]



Obr. 2. Porovnání popularity jednotlivých PHP frameworků. [8]

Vítěz a ve světě velmi oblíbený framework Laravel a druhý v pořadí Symphony2 [8] jsou sice také dostupné jako svobodné softwary pod licencí MIT [12] [13], nemají však doku-

[9] BSD licence – jedna z nejsvobodnějších licencí pro svobodný software.
[10] GPL licence – všeobecná veřejná licence pro svobodný software.

mentaci v českém jazyce a nemají tak silnou komunitu v České republice. Navíc s nimi nemám žádné osobní zkušenosti.

Předem stanoveným kritériím na výběr vhodného frameworku pro vývoj systému pro analýzu dat a sběr dat operátorů nejlépe vyhovuje framework Nette. Je jedním z nejpopulárnějších frameworků a je šířen jako svobodný software. Pyšní se také poměrně kvalitní dokumentací v českém jazyce a má v České republice silnou komunitu vývojářů. [8] [9] [14] Mé zkušenosti s frameworkem Nette také sehrály roli při výběru vhodného frameworku.

2.2 Framework Nette

Použitý Framework Nette je ve verzi 2.4. Pomocí něj je vývoj rychlejší, bezpečnější. Podobně jako u jiných frameworků, je také použita architektura (vzor) MVC (Model-View-Controller). Ve frameworku Nette je obecná architektura MVC (Model-View-Controller) pozměněna na MVP[11] (Model-View-Presenter), přičemž Presenter tady reprezentuje Controller. Použití této architektury přináší přehlednější a snadněji škálovatelnou aplikaci s důrazem na její rozšiřitelnost v budoucnu. Díky tomu je možná i spolupráce na projektu při práci v týmu. Samozřejmě je v rámci frameworku zabudována i podpora HTML5[12], AJAXu[13], SEO[14] a mnoho dalších moderních technologií a konceptů. [14] Ve frameworku je myšleno i na bezpečnost, což přináší další úsporu při nasazování jednotlivých bezpečnostních opatření, které jsou implementovány v rámci vývoje webového systému. Této problematice bude také později věnována kapitola 8. Framework Nette používá objektový návrh aplikace založený na PHP5. Obsahuje znovupoužitelné komponenty a událostmi řízené modelování. [14] Framework Nette má neustále rostoucí nabídku doplňků, které jsou vytvářeny velmi aktivní komunitou v České republice. Kromě vytváření doplňků a rozšíření je dostupná spousta rad a návodů. To je dáno skutečností, že základna frameworku Nette je v České republice. Framework Nette je k dispozici jako Open Source s BSD licenci nebo GPL licence ve verzi 2 nebo 3. Přičemž rozhodnutí, kterou licenci pro svůj projekt zvolíme je zcela na nás, volba je dle konkrétní potřeby. Zkrátka Nette Framework

[11] MVP – softwarová architektura frameworku Nette.
[12] HTML5 – značkovací jazyk pro tvorbu webových stránek.
[13] AJAX – změna obsahu pouze části stránky pomocí asynchronního zpracování v Javascriptu.
[14] SEO – optimalizace pro nalezení webovými prohlížeči.

je svobodný software, který je možné používat i v komerčních projektech. [11] Samozřejmostí jsou nástroje pro ladění aplikace. Při vývoji každé aplikace je dostupnost ladění velmi oceňovanou vlastností a Framework Nette má ladící nástroje velmi propracované. Podobně jako je tomu u jiných frameworků, je i v případě Nette možné se za relativně krátký čas naučit s frameworkem pracovat. V následujících kapitolách budou významné části frameworku popsány detailněji. [14]

2.2.1 MVP (Model-View-Presenter)

Vzhledem k tomu, že název *Controller* a *Presenter* reprezentují ve frameworku Nette v podstatě stejnou funkčnost [15], bude používán název *Presenter* místo *Controller*. Je to vhodné, protože adresářová struktura frameworku Nette ve své výchozí podobě používá název *Presenter*. [15] MVP je adresářová struktura rozdělující kód na tři části. *Model* obsahuje aplikační logiku. Pomocí *Modelu* jsou například modifikována data v databázi. S dalšími vrstvami komunikuje pomocí rozhraní, přičemž nemá o existenci ostatních vrstev ponětí. *Presenter* přijímá požadavky od uživatele a kontaktuje rozhraní příslušného *Modelu*, který řeší žádanou oblast aplikační logiky. Zpracované výsledky obdržené zpět od M*odelu* předá do *View* k vykreslení. Jak už bylo naznačeno, *View* se stará o zobrazení výsledku. Zpravidla k tomu používá šablonovací systém, který má informace o tom, jak zobrazit obdržené výsledky z *Modelu*, případně nějaké komponenty. Zobrazení průběhu zpracování požadavku v modelu MVP od zadání požadavku uživatelem až po vrácení vykreslené stránky zpět prohlížeči uživatele (Obr. 3). Tím se uzavírá celý kruh architektury MVP od zadání požadavku uživatele do webového prohlížeče po vrácení zpracovaného požadavku zpět uživateli v podobě zobrazené stránky ve webovém prohlížeči. [15]

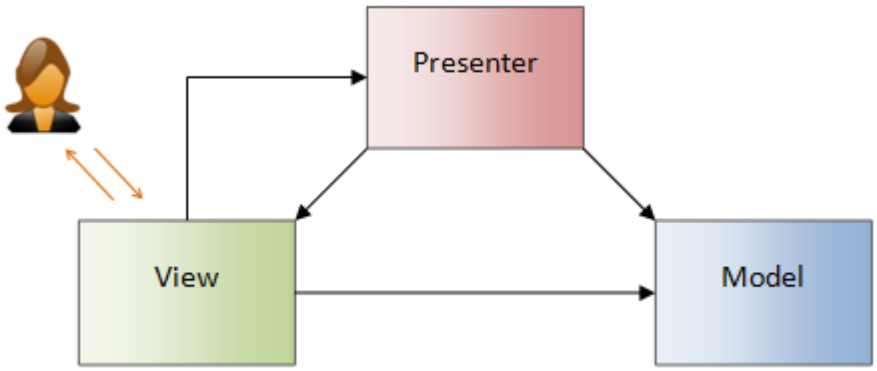
2.2.2 Model

Model je nejzákladnější částí architektury. Obsahuje zpravidla kompletní funkcionalitu pro práci s daty. Jeho součástí bývá také zpracování těchto dat, různé výpočty, apod. Pomocí *Modelu* jsou například modifikována data v databázi. S dalšími vrstvami komunikuje pomocí rozhraní, přičemž nemá o existenci dalších vrstev ponětí. Vrstva *Model* nemá žádnou přímou vazbu na ostatní vrstvy. [15] [16]

2.2.3 Presenter

Jak již bylo zmíněno v předchozím odstavci, *Presenter* tvoří prostředníka mezi *Modelem* a *View* částí aplikace. Uživatel vznáší HTTP požadavky. Všechny požadavky dostává P*re-*

senter skrze soubory *index.php* a *bootstrap.php* do objektu *$application*. Pomocí routeru zjistí, pro jaký *Presenter* je požadavek vznesen a jaká akce se má provést. Po odpovědi routeru již *$application* ví, kterému *Presenteru* je požadavek určen. Vytvoří objekt třídy *Presenter* pro konkrétní požadavek. Objekt je na základě tohoto požadavku vytvořen službou *presenterFactory*. Následně je *Presenter* požádán o provedení akce, kterou může být například akce *Show*. Odpovědí na akci *Show* je zpravidla předání výsledků do šablony, kde dojde k vykreslení HTML stránky s výsledky dotazu. [15]



Obr. 3. Schéma architektury MVP. [16]

2.2.4 View

Jak už bylo naznačeno, *View* se stará o zobrazení výsledku uživateli. Zpravidla k tomu používá šablonovací systém, který má informace o tom, jak zobrazit obdržené výsledky z *Modelu*, případně nějaké komponenty či *Presenteru*. V architektuře MVP je pomocí vrstvy *View* zpracováván i uživatelský vstup. Příkladem může být situace, kdy vrstva *View* zpracuje událost (například kliknutí myší) a zavolá příslušnou metodu vrstvy *Presenter*. To ovšem neznamená, že by měla vrstva *View* obsahovat aplikační logiku. Z výše uvedeného je patrné, že *View* má přímou vazbu na *Presenter*. V mnoha případech použití existuje také vazba na *Model*. [15] [16]

2.2.5 Životní cyklus presenteru

Akce *Presenteru* jsou volány jako jeho metody ve tvaru *render<akce>*, například pro zobrazení článků by mohl název být *renderShow*. Při vytváření *Presenterů* můžeme vytvořit více metod, mimo již zmíněnou metodu *render*. Možné klíčové metody jsou zobrazeny níže (Obr. 4). Tyto předdefinované metody je možné vytvořit nebo překrýt. Je však potom nutné zavolat i předka této metody. V další části textu si metody *Presenteru* stručně popí-

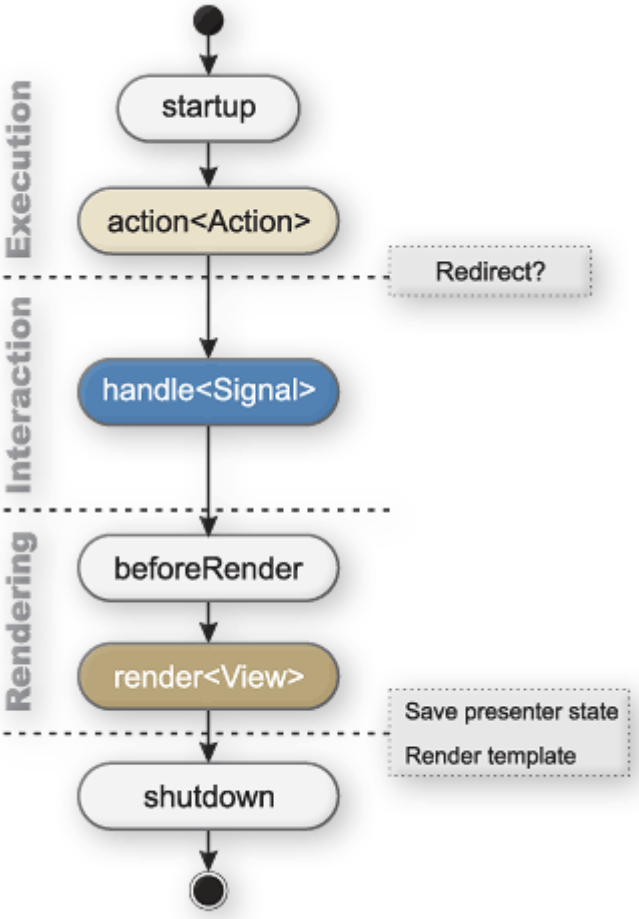
šeme. Jednotlivé metody jsou spouštěny sekvenčně v pořadí životního cyklu *Presenteru*, jak je uvedeno na obrázku (Obr. 4). [15]

2.2.5.1 Metoda startup()

Tato metoda je volána ihned po vytvoření *Presenteru*. Této metody je používáno k inicializaci proměnných. Další využití je například k ověření uživatelských oprávnění. Po jejím vykonání následuje spuštění další metody dle životního cyklu *Presenteru* (Obr. 4). [15]

2.2.5.2 Metoda action<Action>()

V rámci této metody provede *Presenter* určitý úkon (akci). Je velmi podobná metodě *render<View>*. Metoda *action* je však určena pro úkony, které nesouvisí přímo s vykreslením.



Obr. 4. Životní cyklus Presenteru. [15]

Tato metoda je vykonána před metodou *render<View>()*. Používá se například na přihlášení uživatele nebo zapsání dat do databáze. Po těchto úkonech dojde zpravidla k přesměrování jinam či zpracování dalších metod *Presenteru.* [15]

2.2.5.3 Metoda handle<Signal>()

Tato metoda je určena pro zpracování tzv. signálů neboli subrequestů. Metoda je předurčena pro zpracování AJAX požadavků a komponent. Využívá se také při zpracování formulářů pro zachycení událostí. [15]

2.2.5.4 Metoda beforeRender()

Jak je patrné z názvu, je tato metoda provedena před metodou *render<View>()* (Obr. 4). Je používána k předání proměnných pro více pohledů nebo jejím prostřednictvím je možno nastavit šablonu. [15]

2.2.5.5 Metoda render<View>()

Tato metoda je používána k naplnění dat do šablony. Část názvu metody určuje název šablony. Například pro naplnění daty a následnému vykreslení šablony s názvem *default.latte* by byla použita metoda render s názvem *renderDefault()*. [15]

2.2.5.6 Metoda shutdown()

Životní cyklus *Presenteru* je završen voláním metody *shutdown()*. Pomocí této metody je možné provést nějakou akci těsně před ukončením života *Presenteru.* [15]

2.2.5.7 Ukončení presenteru

Činnost *Presenteru* může být kdykoli během jeho životního cyklu ukončena. To je realizováno voláním speciálních metod k tomuto účelu určeným. [15]

- $presenter->terminate() – přímé ukončení presenteru. [15]
- $presenter->sendTemplate() – ukončení běhu presenteru a okamžité vykreslení šablony. [15]
- $presenter->sendPayload() – ukončení presenteru a odeslání payloadu (použití pro AJAX). [15]
- $presenter->sendResponse($response) – ukončení presenteru a následné odeslání vlastní odpovědi. [15]

- vyjímka BadRequestException – poslední možností je ukončení přesměrováním nebo vyvoláním výjimky. [15]

Předčasné ukončení životního cyklu *Presenteru* je používáno například k tomu, aby nedošlo k vykreslení šablony. Příkladem použití může být ověřování uživatelského oprávnění. Dojde-li k tomu, že uživatel není přihlášen, nezadal platné přihlašovací údaje nebo nemá dostatečná oprávnění k provedení akce, je použita některá z výše uvedených metod a životní cyklus *Presenteru* je předčasně ukončen nebo může být využito přesměrování na stránku pro přihlášení uživatele. [15]

2.2.6 Práce s databází

V rámci frameworku Nette existují nástroje pro práci s databází v podobě databázové vrstvy. V této vrstvě je implementováno několik tříd, které poskytují jednotlivé služby pro snadnější práci s databází. Ve třídě *Nette\Database* jsou k dispozici ovladače pro různé databázové servery, které jsou frameworkem podporovány. Jedná se o MySQL, PostgreSQL, Sqlite2 a 3, Oracle, MS SQL a ODBC[15]. Samozřejmě je možné implementovat vlastní ovladač a ten následně používat pro práci s databází. [17]

2.2.6.1 Třída Nette\Database\Connection

Třída *Nette\Database\Connection* poskytuje připojení k databázi a vytváří obálku nad PDO[16]. Také nabízí podporu pro provádění dotazů do databáze pomocí volání *query*. Níže je uveden příklad konfigurace připojení k databázi (Obr. 5). Skládá se z několika základních parametrů, které je nutné zadat do konfiguračního souboru frameworku Nette. Celý blok konfigurace připojení k databázi je uvozen slovem *database* s pokračováním na dalším řádku slovem *default*. Představuje výchozí připojení k databázi. Následují parametry *dsn* (Data Source Name), *user* a *password*. Parametr *dsn* se skládá z názvu databázového serveru, adresy databázového serveru a jména databáze. Přičemž název se liší podle použitého databázového serveru, resp. jemu náležejícímu ovladači databáze. Parametry *user* a *password* jsou, jak název napovídá, pro uvedení přihlašovacího jména a hesla pro připojení k databázovému serveru. Následuje sekce *options*, ze které je použit pouze parametr *lazy*. Parametr *lazy* je pro tzv. „líné“ připojení. Znamená to, že spojení s databází není vytvořeno

[15] ODBC – Open Database Connectivity – standardizované softwarové rozhraní pro připojení k databázi.
[16] PDO – PHP Data Objects – objektové rozhraní databází pro PHP [49].

ihned při vytvoření instance připojení, ale až ve chvíli, kdy je opravdu potřeba. Pro vytvoření nového připojení k databázi stačí vytvořit novou instanci této třídy. Můžeme to provést dvěma způsoby. V prvním případě to bude pomocí aplikační konfigurace. [17]

```
database:
    default:
        dsn: "mysql:host=127.0.0.1;dbname=test"
        user: "root"
        password: "password"
        options:
            lazy: true
```

Obr. 5. Příklad konfigurace pro připojení k databázi. [17]

Je to velmi snadný způsob jak nastavit připojení k databázi. Umožňuje vytvořit i více připojení. Zároveň poskytuje službu kontextu *Nette\Database\Context*. Pomocí této služby je následně možné pracovat s vrstvou *Nette\Database\Table*. Druhou možností je vytvořit připojení k databázi přímo z místa kde připojení potřebujeme, tedy zpravidla v rámci daného modelu pomocí příkazu *$connection = new Connection($dsn, $user, $password)*. S takto vytvořeným spojením do databáze už můžeme provádět dotazy do databáze. Například příkaz *$result = $connection->query('SELECT * FROM knihy');* nám vrátí všechny záznamy do proměnné *$result*. S těmito výsledky je pak možné dále pracovat. [17]

2.2.6.2 Třída Nette\Database\Table

Třída *Table* poskytuje zjednodušený a optimalizovaný přístup k datům v databázi. Idea této třídy tkví v načítání dat pouze z jedné tabulky, a tím k omezení opakovaného dotazování. Samozřejmě pokud potřebujeme načíst data z více tabulek, načte se každá tabulka zvlášť pomocí samostatných dotazů. Pro výběr dat z tabulky v databázi slouží velmi jednoduchý příkaz *table()*, například *database->table('knihy')* nám vrátí záznamy z tabulky *knihy*. [18] Data jsou načtena jako instance *ActiveRow* třídy. Tyto instance potom reprezentují jednotlivé záznamy, mezi kterými můžeme iterovat a tak s nimi pracovat. Přestože je filozofie této vrstvy určena k načítání dat vždy pouze z jedné tabulky, pomocí této třídy je také možné načítat data z více tabulek. Vztahy jsou rozděleny na dva druhy. Prvním z nich je vztah „has one" a druhý „has many". [19]

Relace „has one" reprezentuje vztah mezi dvěma tabulkami. Například může reprezentovat vztah 1:1. Tato relace se vyskytuje velice často. K vytvoření tohoto vztahu se používá metoda *ref()*, která přijímá dva argumenty. Prvním z nich je jméno tabulky, druhý potom

jméno atributu představující referenci mezi tabulkami. Metoda vrací instanci *ActiveRow* reprezentující záznam odpovídající relace nebo *NULL*, v případě, že hledaný záznam není nalezen. [19]

Relace „has many“ podobně jako v předchozím případě reprezentuje vztah mezi dvěma tabulkami, tentokrát však 1:N. Pro tuto relaci se používá metoda *related()*, která přijímá dva argumenty. Prvním z nich je jméno cílové tabulky, druhým je potom jméno atributu tabulky. Metoda *related()* následně vrací pole odpovídajících záznamů v podobě instancí *ActiveRow*. Argument může být zadán jako jeden, oddělený tečkou. V případě, že není nalezena shoda, je vrácena odpovídající výjimka. [19]

2.2.6.3 Třída Nette\Database\Table\Selection

Třída *Selection* poskytuje nástroje pro výběr a filtrování dat z databázových tabulek. Vytváří automaticky relace mezi tabulkami pomocí JOIN[17] v SQL[18]. Filtrování je prováděno zavoláním metody *where()* a zadáním podmínky. Například *$selection->where('id', $id);*. Za zmínku stojí, že vhodné operátory jsou do podmínky doplněny podle zadaných dat. Pokud chceme filtrovat záznamy hodnotou z jiné tabulky, stačí napsat jméno spojovacího klíče relace a název sloupce spojené tabulky. Ze jména sloupce, odkazující na spojovanou tabulku, je odvozen název spojovacího klíče. Způsob odvození spojovacího klíče je implementován v rámci třídy *IConventions*. Velmi vhodné je použití třídy *DiscoveredConventions*, která provádí analýzu cizích klíčů a usnadňuje práci s relacemi mezi tabulkami. [20]

2.2.7 Latte

Po zpracování požadavku *Presenterem* jsou výsledky předány k zobrazení. O zprostředkování zobrazení požadavku v podobě webové stránky se stará část architektury zvaná *View*. Ve frameworku Nette je pro vytváření vzhledu zobrazovaných stránek používán šablonovací systém Latte, který usnadňuje práci při definici vzhledu stránky. Obsahuje tzv. makra, která přináší značnou úsporu kódu a navíc je zápis s použitím maker přehlednější. Makra jsou zapisována pomocí speciálních značek a to dvěma způsoby. První způsob je za použití

[17] JOIN – klauzule SQL spojení, kdy je použita kombinace řádků ze dvou nebo více tabulek na základě společného pole mezi nimi [50].
[18] SQL – strukturovaný dotazovací jazyk.

složených závorek, např. pro zápis příkazu pro cyklus *{while expr} ... {/while}* nebo druhým způsobem pomocí *n:makra*, se kterým vypadá zápis jako * *. Šablonovací systém Latte obsahuje i makro pro automatizované překlady v šablonách. Postačí nastavit překladač a následně při použití makra ve tvaru *{_$promenna}* nebo *{_}Tex k překladu{_}* bude zadaný parametr makra přeložen, resp. doplněn zadaný překlad. Pokud bychom náhodou potřebovali makro, které není součástí frameworku, můžeme si vytvořit makro vlastní. [21]

Dalším zjednodušením při použití Latte je možnost použití filtrů tzv. *Helperů*. Pomocí těchto helperů je možné velmi snadno pracovat s řetězci, formátovat vypisované hodnoty či ovlivňovat velikost písma. A to navíc s velmi jednoduchou syntaxí. Například pro zobrazení všech velkých písmen u textového řetězce stačí použít *{$nasRetezec|upper}*. Nebo výpis čísla zformátujeme pomocí *{15238.50|number:2:',':' '}*. [22] Význam tohoto filtru je velmi užitečný a poměrně snadno čitelný. Jednotlivé parametry makra jsou od sebe odděleny dvojtečkou. První parametr *number:2* určuje, že se jedná o číselnou hodnotu s uvedením počtu desetinných míst. Druhým parametrem *','* je volba oddělovače desetinných míst a posledním parametrem *' '* je volba oddělovače tisíců. Výsledkem vyhodnocení této funkce je zobrazení *15 238,50*. A spousta dalších užitečných filtrů, například pro práci s datem nebo výpisem velikostí souborů. [22]

Velkou výhodou Latte je, že provádí automatické escapování vypisovaných proměnných. Escapování je velmi důležité, protože opomenutí escapování by znamenalo vznik bezpečnostní díry Cross Site Scripting (XSS). Navíc Latte obsahuje technologii Context-Aware Escaping. Díky této technologii Latte automaticky rozpozná, v které části dokumentu použité makro je a vybere vhodné escapování. Ale detailněji k bezpečnosti se ještě vrátíme později. Poslední zde zmíněnou výhodou je bezesporu možnost používat dědičnost šablon a vytvářet bloky. Díky této možnosti je zajištěna znovu použitelnost, čím uspoříme opět čas při vytváření šablon. [23]

2.2.8 Tracy

Nespornou výhodou frameworku Nette je nástroj na ladění při vývoji v podobě knihovny se jménem Tracy. Pro její název bývá používán i český termín „Laděnka“. Pomocí této knihovny je ladění programu mnohem příjemnější, protože jsou všechna chybová hlášení a výjimky prezentovány v přehledné formě. Zobrazená webová stránka s chybovým hlášením či zachycenou výjimkou umožňuje „proklikávat“ jednotlivé odkazy a získat tak infor-

mace z různé úrovně detailu hlášení. Samozřejmě tyto informace jsou velmi užitečné ve vývojovém prostředí, ovšem v produkčním prostředí by tomu tak již určitě nebylo. Ale na tuto situaci je v Tracy myšleno. Tracy obsahuje autodetekci produkčního a vývojového prostředí. V produkčním režimu potom Tracy zobrazí hlášení ve srozumitelné podobě pro uživatele a ostatní detailní informace zapíše do logu. K tomuto nastavení je používán parametr `Debugger::DETECT` pro automatickou detekci. [24] Pokud by automatická detekce nefungovala korektně, lze nastavit vývojový `Debugger::DEVELOPMENT`, respektive produkční `Debugger::PRODUCTION` režim ručně pomocí těchto výše uvedených příkazů. [24]

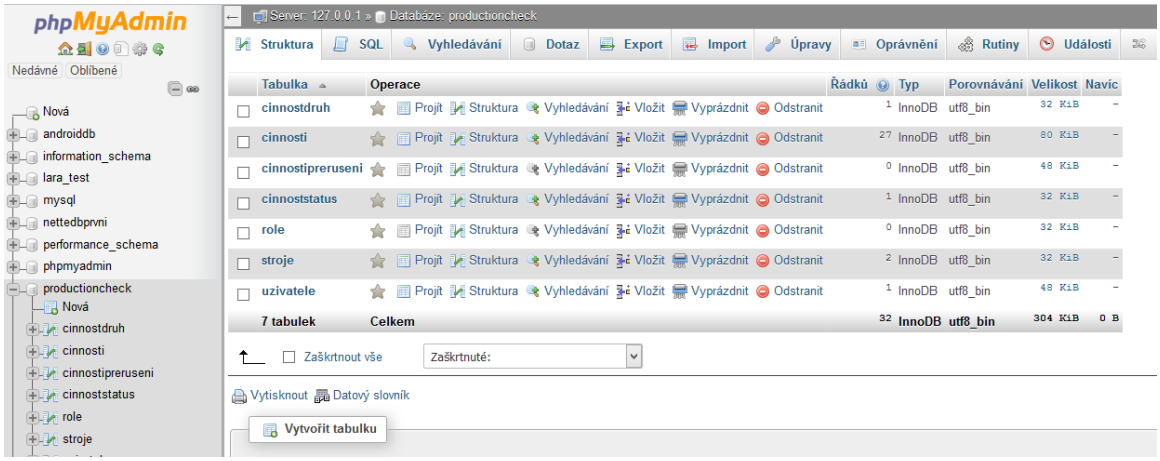
2.3 Databáze MariaDB

Databáze MariaDB je relační databázový systém. Je to rozšířená (přímá) náhrada za MySQL. MariaDB je vyvíjena komunitou vývojářů pod záštitou MariaDB Foundation. Jejími členy jsou mnozí vývojáři původního MySQL. [25] MariaDB Foundation je vlastníkem hlavního projektu a zajišťuje oficiální vývojový strom pro komunity vývojářů MariaDB a mariadb.org. Díky tomu je i nadále zaručeno, že MariaDB bude stále Open Source software v souladu s podmínkami GPL verze 2. [25] MariaDB je proto velmi vhodnou náhradou databáze MySQL, kterou dnes již vlastní společnost Oracle[19], a podléhá ochranné známce společnosti Oracle. Vývojáři MariaDB se i nadále snaží udržovat kompatibilitu mezi MariaDB a MySQL, zapracovávají nové bezpečnostní záplaty, API[20] či další příkazy a knihovny. [25] MariaDB je velmi spolehlivá a robustní databáze, její použití je jednoduché a škálovatelné. Poskytuje SQL rozhraní pro přístup k datům této relační databáze. Obsahuje velké množství zásuvných modulů a dalších nástrojů. Všechny tyto zmíněné vlastnosti činí z tohoto databázového systému velmi vhodnou databázi pro širokou škálu použití. Poslední verze MariaDB obsahuje i nástroje pro práci s geografickými daty a JSON funkce. V rámci tohoto systému je použita v době vývoje poslední stabilní verze MariaDB 10.1. [25]

[19] Oracle – softwarová společnost.
[20] API – rozhraní pro programování aplikací.

2.3.1 PhpMyAdmin

PhpMyAdmin je velmi užitečný nástroj pro správu databázového systému MySQL (MariaDB) přes grafické uživatelské rozhraní v podobě webové aplikace (Obr. 6). Jedná se o svobodný softwarový nástroj vytvořený v jazyce PHP v souladu s licencí GPL verze 2. [26] Pomocí tohoto užitečného nástroje je možné procházet a mazat databáze, tabulky, pohledy, indexy a další. Je možné provádět SQL příkazy, přidělovat uživatelská oprávnění. Samozřejmostí je také možnost vytváření a editace uložených procedur, funkcí, spouští a událostí. V neposlední řadě je možné provádět kopírování databází, export a import dat, podporu replikace či využít grafický návrhář pro vytváření tabulek a definování cizích klíčů. Pomocí PhpMyAdminu je správa databáze přehledná a snadná. PhpMyAdmin je multiplatformní a dostupný v několika jazycích, mezi které patří i český jazyk. [26]



Obr. 6. Ukázka zobrazení tabulek v databázi pomocí phpMyAdmin.

2.4 Apache

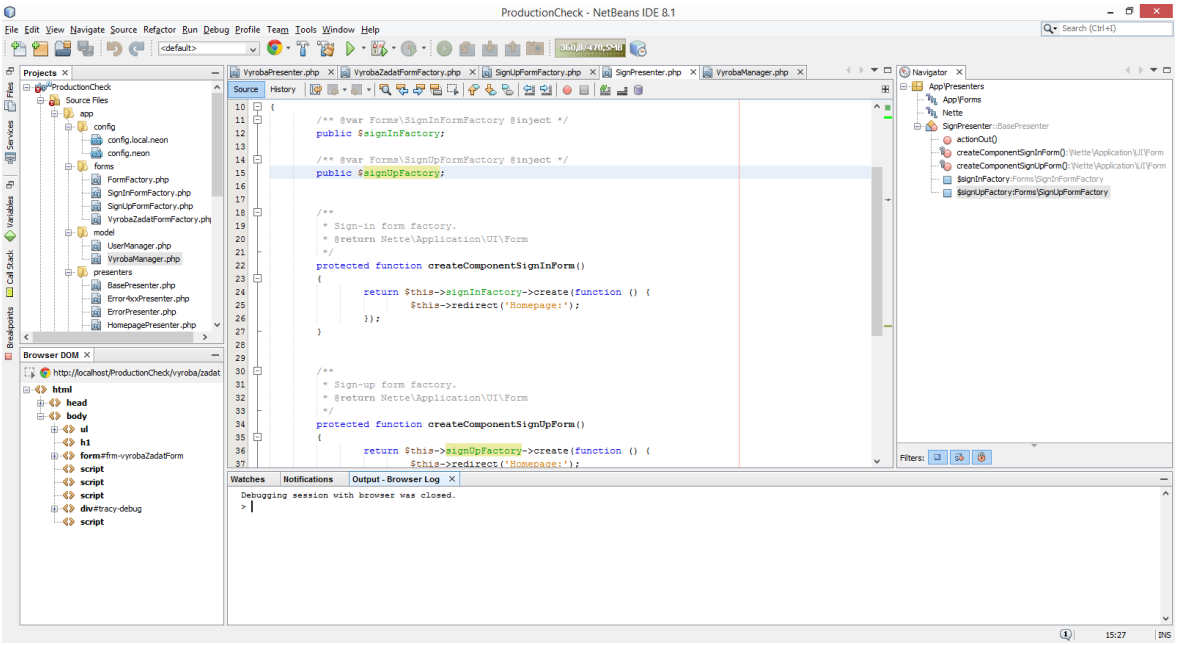
Apache je multiplatformní webový server, poskytující HTTP služby. Je k dispozici jako svobodný software v souladu s Apache licencí verze 2, která je kompatibilní s GPL licencí verze 2. Je vyvíjen celosvětovou komunitou vývojářů z celého světa. [27]

2.5 Vývojové prostředí NetBeans

Integrované vývojové prostředí NetBeans je také multiplatformní. [28] Podobně jako všechny dříve zmíněné použité technologie je i NetBeans svobodný software vyvíjený rostoucí komunitou vývojářů a partnerů po celém světě. NetBeans je možné bezplatně používat pro soukromé, ale i komerční projekty. Zakladatelem tohoto Open Source projektu je

společnost Sun Microsystems a učinila tak již v roce 2000. V rámci tohoto projektu existují dva produkty. [29]

V prvním případě se jedná o již zmíněné vývojové prostředí NetBeans (Obr. 7). V rámci tohoto prostředí je možné psát kód, ladit a překládat. Možné je provádět distribuci vytvořené aplikace. Vývojové prostředí NetBeans je vytvořeno v programovacím jazyce Java.



Obr. 7. Ukázka vývojového prostředí NetBeans.

V rámci něj je však možné vyvíjet v široké škále programovacích jazyků jako je například PHP, Java, C++, HTML a mnohé další. [28] NetBeans lze ještě dále rozšiřovat o další dostupné moduly. Jedním z modulů, který je využíván v rámci tohoto systému je framework Nette. Je tedy možné vytvořit nový PHP projekt přímo s podporou frameworku Nette. Další velmi příjemnou funkcionalitou je automatická kompletace kódu. [28]

V druhém případě se jedná o vývojovou platformu Netbeans. Jedná se o základ pro vytváření rozsáhlých desktopových aplikací, který je modulární a dále rozšiřitelný. Spektrum modulů je velmi bohaté a snadno integrovatelné. Ty mohou být následně použity k vývoji dalších nástrojů a řešení. [30]

3 BEZPEČNOST WEBOVÝCH APLIKACÍ

Bezpečnost softwarových produktů je v dnešní době velmi diskutovaná a důležitá oblast vývoje webových aplikací. I přes rostoucí informovanost o bezpečnostních rizicích a zneužitelných nedostatcích systémů je napadení webů významných společností, ale i státních institucí poměrně běžné. Stále častěji jsou také napadány i relativně nenápadné nebo neznámé systémy, kterých je následně zneužito pro napadení dalších subjektů. [3]

V současné době již existují způsoby, jak zvýšit ochranu před známými praktikami, které jsou nejčastěji používány při napadení webových aplikací. Také u většiny z používaných praktik při napadení je znám způsob ochrany. [3] V souvislosti se zvolenou technologií pro vývoj webové části systému popsanou v předchozích odstavcích, jsou v kapitole 8 uvedeny způsoby zabezpečení, které jsou k dispozici ve zvoleném frameworku Nette ve verzi 2.4. [31]

3.1 CSRF útok

CSRF (Cross-Site Request Forgery) je útok vedený na uživatele (návštěvníka nebo klienta) webové stránky nebo aplikace za účelem zneužití jeho identity jako přihlášeného uživatele. Server vyřizující požadavek není schopen rozlišit, jestli se jedná o požadavek korektně přihlášeného uživatele nebo je jeho identita zneužita útočníkem. [32] Útočník mnohdy vychází ze skutečnosti, že danou webovou stránku či aplikaci zná, protože se jedná např. o nějaký CSM[21]. Nic netušícímu uživateli je zpravidla podstrčen webový odkaz (např. zaslaný emailem, lákavý obrázek nebo formou sociálního inženýrství[22]), který směřuje na předem připravenou stránku útočníka. Tímto kliknutím na odkaz je následně provedena akce, kterou útočník požadoval. Uživatel, který na zaslaný podvržený odkaz kliknul, mohl útočníkovi zaslat citlivá data, jako jsou například přihlašovací údaje. [33] [34]

Způsobů ochran před tímto typem útoku může být několik, ne však každá je zcela úspěšná. Na ochranu lze nahlížet ze dvou úhlů. V prvním případě se jedná o ochranu na straně uživatele, v druhém potom o ochranu na straně serveru. [34] V případě ochrany na straně uživatele je možností poměrně málo, navíc některé z nich mohou významně ovlivnit komfort při prohlížení webových stránek. Jednou z nich by mohlo být přísnější nastavení

[21] CSM – (*Content Management Systém) Systém pro správu obsahu.*
[22] Sociální inženýrství – způsob manipulace s lidmi za účelem získání informací nebo provedení akce. [51]

prohlížeče v podobě zakázání spouštění scriptů, případně dokonce stahování obrázků. Další možností by bylo nastavení firewallu[23] počítače na interaktivní režim, při kterém bude uživatel vždy vyzván k potvrzení komunikace. [34] I v tomto případě bude práce velmi nepohodlná, protože požadavků o síťové spojení může být mnoho. [34] [35]

Ochrana na straně serveru je na vývojáři webové stránky nebo aplikace, aby stránku dostatečně zabezpečil před tímto útokem. Jednou z možností jak provést ochranu proti CSRF útoku je za pomoci využití hlavičky „referer“. [34] Hlavička referer je součástí HTTP dotazu odeslaného uživatelem skrze webový prohlížeč. Právě webový prohlížeč tuto informaci do HTTP dotazu vloží. Je však možné nastavením webového prohlížeče omezit odesílání této informace. Použitím této ochrany by mohlo dojít k zamezení přístupu k webové stránce nebo aplikaci i legitimním uživatelům. [34] Další možností ochrany je pomocí proměnné hodnoty v URL. Do URL je přidána proměnná v podobě jednoznačného identifikátoru v souvislosti se session[24]. URL potom bude vždy jiná pro různé identifikátory vznikající na základě *sessions*. Důležité při použití této ochrany je, aby byla precizní v celém webu. Přesná URL bez doplněného identifikátoru na základě session nesmí být zobrazena. [34] Poslední zde zmíněnou ochranou je použití autorizačního tokenu. Jedná se o princip, při kterém je pro každou operaci v dané relaci vygenerován náhodný řetězec, který je následně uložen na straně serveru. V případě, že uživatel operaci provede, je nejdříve kontrolována shoda identifikátoru, neboli autorizačního tokenu s hodnotou uloženou na serveru, a teprve potom je požadovaná operace provedena. [35] Provedení operace je pouze na základě shody identifikátorů. Potom je tento identifikátor ze serveru odstraněn. [34] [35]

3.2 XSS útok (cross-site scripting)

XSS (Cross-Site Scripting) je označení dalšího známého a často používaného způsobu útoku na webové aplikace. Jeho podstata je založena na nedostatečném ošetření vstupů, které jsou posléze zpracovány. Díky tomuto nedostatku může útočník vložit svůj kód do webové stránky. K těmto útokům jsou zpravidla využívány skripty na straně klienta a vzhledem k zastoupení ve většině prohlížečů webových stránek se jedná nejčastěji o Javascript. [36]

---

[23] Firewall – síťové rozhraní sloužící pro oddělení vnitřní a vnější sítě a následně kontrole síťového provozu.
[24] Session – „sezení“ vytvořené při spojení mezi klientem a serverem.

Existuje několik typů XSS útoků. Prvním z nich je tzv. „Persistentní“ neboli trvalý. S tímto typem XSS útoku se setkáme všude tam, kde je možné svůj příspěvek na webové stránce uložit natrvalo. Jedná se například o diskuzní fóra, příspěvky, apod. [36] Pokud tedy vstup uživatele není dostatečně ošetřen, může uživatel vložit nebezpečný skript. Ten potom může být následně spuštěn prohlížečem všech uživatelů, kteří k této upravené stránce přistoupí. Vložený skript může provádět například jen jednoduché přesměrování na stránky útočníka (Obr. 8), změnu obsahu stránky či úplnou kontrolu nad prohlížečem oběti. [36]

```
<script>document.location='http://www.atacker.com';</script>
```

Obr. 8. Skript pro nežádoucí přesměrování uživatele. [36]

Dalším typem XSS útoku je tzv. „Non-persistentní“. [36] Princip tohoto útoku je založen na zakomponování škodlivé části na straně serveru skrze požadavek na stránku. Stránka je s touto změnou následně zobrazena uživateli. Častá je tato zranitelnost u vyhledávačů, nebo na stránkách, které předávají zprávy přes parametry URI. [36]

Posledním zmíněným typem XSS útoku je „DOM-based“[25]. [36] Ovšem zde probíhá integrace útočníkova skriptu do webové stránky na straně klienta. Útočník může zneužít údajů obsažených v URL a použije získaná data pro zobrazovanou stránku. Ukázka kódu stránky, který je možné zneužít pro tento typ útoku (Obr. 9). [36]

```
<script>
    var pos=document.URL.indexOf("jmeno=")+6;
    document.write("Ahoj "+document.URL.substring(pos,document.URL.length));
</script>
```

Obr. 9. Příklad kódu stránky citlivé na útok „DOM-based“. [36]

Ochranou před XSS útokem na straně klienta by mohlo zakázání spouštění skriptů ve webovém prohlížeči. To by ovšem v dnešním světě webových stránek, které jsou doslova protkány Javascriptem vedlo minimálně k nekorektní funkčnosti, zobrazení chyby nebo úplné nefunkčnosti. [36]

[25] DOM (Document Object Model) – objektový model dokumentu.

Ochranou proti tomuto typu útoku je zejména správné ošetření vstupů resp. výstupů z aplikace. Na straně vstupu je vhodné zamezit vložení neplatných znaků či dokonce skriptu útočníka. Nebezpečné znaky je vhodné na vstupu kódovat. K tomu účelu lze použít funkci *htmlentities()*. [36] Naopak ošetření výstupů lze zajistiti vestavěnou funkcí programovacích jazyků (pro PHP je to funkce *htmlspecialchars()*) nebo použít např. sadu povolených znaků omezující zadání znaků nebezpečných. [37]

Dalším způsobem jak zvýšit ochranu proti tomuto útoku je použití příznaku „HttpOnly“, který je zahrnut do hlavičky HTTP odpovědi v Set-Cookie. Tento příznak omezuje přístup ke cookies[26] na straně klienta (např. použitím Javascriptu) pouze na požadavek HTTP či HTTPS. Podmínkou této popsané funkcionality příznaku je podpora v rámci použitého webového prohlížeče. Dobrou zprávou ovšem je, že většina dnes používaných moderních prohlížečů toto nastavení podporuje. [38]

3.3 Session hijacking, session stealing

Základním principem těchto útoků je zneužití session ID platného uživatele útočníkem. Jedná se o jednoznačný identifikátor, který server potřebuje pro rozpoznání uživatele. Pokud se útočníkovi podaří tento identifikátor zneužít, tváří se ve vztahu k serveru jako legitimní uživatel. Díky tomu má útočník přístup k webovému serveru. [39] Session ID může být ve webové aplikaci použit na různých místech. Prvním z nich může být URL adresa. Jednoznačný identifikátor je uložen do URL adresy. [39] Druhým způsobem může být uložení session ID v HTTP hlavičce „referer“ požadavku. Stačí kliknout na podvržený odkaz útočníka a session ID je předáno útočníkovi. [39]

Ochranou proti tomuto typu útoku je nepoužívání URL pro přenášení session ID. Vhodnější je pro přenášení session ID použít cookies. V případě PHP lze přímo v konfiguračním souboru (php.ini) parametrem *session.use_only_cookies true* možné nastavit zákaz používání session ID v URL. [39]

3.4 Session fixation

Podobně jako v předchozí kapitole i tato zranitelnost se týká zneužití uživatelova session ID. V tomto případě se však nejedná úplně o ukradení session ID, ale spíše o přesvědčení

[26] Cookies – malá textová informace zaslaná serverem, uložená u klienta.

uživatele na použití podstrčeného konkrétního identifikátoru. Dochází tedy k „fixaci“ již zavedené relace. [40] Způsoby pro provedení útoků jsou velmi podobné jako v předchozím případě. Jednou z možností jak podstrčit přihlášenému uživateli session ID je, předáním session ID pomocí URL. Postačí oběti zaslat falešné session ID a od té chvíle používá oběť podstrčený identifikátor. [40] Druhou zde zmíněnou možností je situace, kdy je session ID předáváno pomocí cookie. V tomto případě je na straně klienta zpravidla využito dalších technik jako XSS útok popsaný v kapitole 3.2, pomocí kterého je možné modifikovat cookie na fixní identifikátor pro komunikaci se serverem. Způsobů provedení útoku může být více, např. úpravou parametrů set-cookie v odpovědi HTTP hlavičky. [39] [40]

Ochranou před tímto druhem útoku je změna session ID po každém úspěšném přihlášení uživatele. [39] Ještě lepším způsobem ochrany je generování nového session ID při každém požadavku. Díky tomu bude session ID pro každou operaci jednoznačné a zamezí přihlášení pomocí něj i v situaci, že by útočník zachytil komunikaci a získal tak session ID oběti. Případně by bylo možné nastavit odmítání všech identifikátorů nepocházejících ze serveru. Pro nastavení tohoto striktního módu je v PHP k dispozici *session.use_strict_mode*. [39]

3.5 Autentizace uživatelů

Autentizace uživatelů je velmi častou a velmi důležitou částí v oblasti zabezpečení webových aplikací. Autentizací uživatele rozumíme situaci, při které je ověřen „identifikován“ uživatel, který přistupuje k dané webové aplikaci. Autentizací získáme identitu uživatele, jinými slovy je možné říci, že ověříme jeho totožnost. [41] Způsobů provedení autentizace je několik, ale nejčastěji se jedná o vnitřní část systému, ve které jsou uživatelé evidováni a ověřováni na úrovni tabulky uživatelů v databázi. Evidence uživatelů, která obsahuje identifikační údaje, jako jsou jména či hesla, bývá zpravidla ještě dále zabezpečena pomocí šifrování alespoň hesel uživatelů. Nejčastěji používaný způsob autentizace je pomocí jména a hesla. [41] Uživatel je vyzván k zadání svého uživatelského jména a hesla a tyto údaje jsou ověřeny, jestli jsou platné. Ale k přihlašování je možné využít více způsobů než jen obvyklé jméno a heslo. Dnes jsou již zcela běžné způsoby autentizace za použití biometrických dat. Při tomto typu autentizace jsou použita ověření pomocí např. otisků prstů, otisků dlaně, snímání obličeje, zkrátka nějaké části těla, která jednoznačně identifikuje přihlašovaného uživatele. [42] Hojně je také využíváno generátorů hesel v podobě specializovaného generátoru. Pomocí tohoto způsobu je heslo pro přihlášení uživatele platné na-

příklad na omezený čas, případně na jedno přihlášení. Tato metoda může být doplněna ještě požadavkem na zadání PINu. [42] Dalším způsobem autentizace je pomocí čipové karty nebo USB tokenu. Na kartě či USB tokenu je uložena identifikace uživatele v podobě PKI. Pomocí tohoto klíče je možné se ne jenom autentizovat do aplikace, ale využít jej také např. k šifrování dat. [42]

V souvislosti s přihlašovacími údaji je vhodné nastavit například jejich dobu platnosti, po které je uživatel vyzván ke změně zadaného hesla. V rámci systému by také mělo být možné omezit možnost autentizace uživatele například jen v určitou dobu. [41] Dále by mělo být možné údaje uživatele zneplatnit, pokud již uživatel není oprávněn k autentizaci do systému. Velmi vhodné je také logování jednotlivých autentizací uživatelů. Tyto informace jsou následně použitelné pro analýzu problému, případně zjištění o neoprávněné autentizaci některého uživatele. [41]

3.6 Autorizace uživatelů

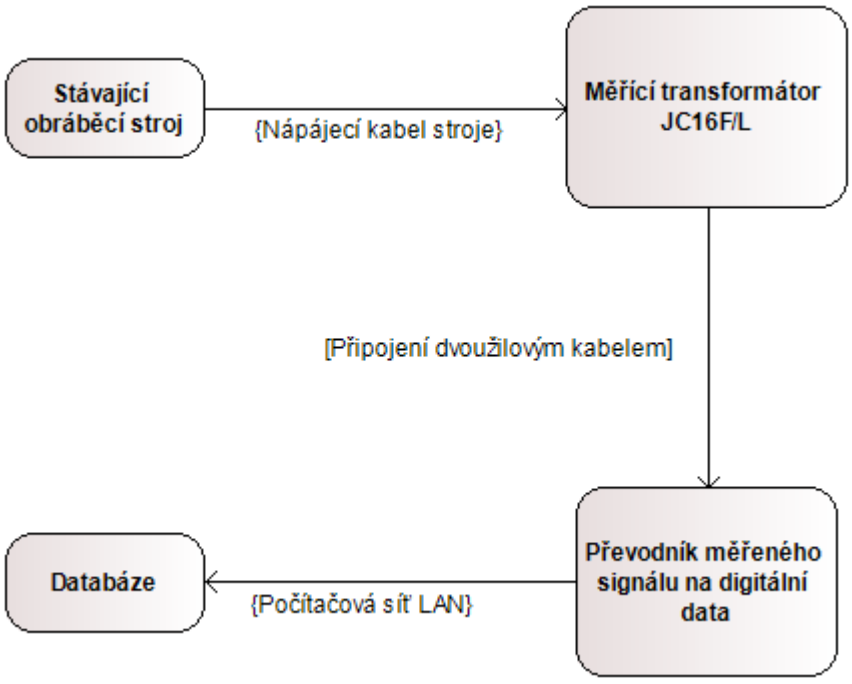
Autorizace je zpravidla proces, který přímo navazuje na autentizaci, popsané v kapitole 3.5. Autorizace spočívá v zjištění, jestli má autentizovaný uživatel dostatečnou úroveň oprávnění (někdy také roli) k provedení požadované operace či akce. [41] Úrovně oprávnění jsou zpravidla předem dány a jsou definovány administrátorem aplikace. Mohou být velmi specifické a mít složitou strukturu, ale v některých situacích si vystačíme jen s modelem administrátor či správce a uživatel. V každém případě je velmi vhodné vždy ověřovat, jestli má aktuální uživatel dostatečná oprávnění pro přístup k dané části webové aplikace nebo provedení požadované akce. [41]

4 SYSTÉM SBĚRU DAT ZE STROJŮ

Systém pro sběr dat ze strojů je ve výrobní společnosti již implementován. Dodávku realizovala společnost TriDat. Součástí dodaného řešení byl i HW pro měření příkonu strojů, převodník pro uložení naměřených dat jako elektrického příkonu, databáze pro uložení naměřených dat a uživatelský software pro zobrazení naměřených dat. Tento systém sběru dat ze strojů byl prozatím implementován na 4 obráběcí stroje ve výrobní společnosti.

4.1 Popis HW

Měření elektrického příkonu strojů je realizováno pomocí měřících proudových transformátorů, které jsou použity pro měření střídavého proudu až do několika set ampér. [43] Měřící transformátor je dělený, lze jej snadno otevřít. Díky tomu je montáž měřícího transformátoru snadná a poměrně rychlá. [43] Měřící transformátor se montuje na kabel z vnější strany, přesněji kabel je obalen tímto transformátorem. Na transformátoru je tedy otvor, do kterého se umístí kabel, na kterém chceme měřit protékající proud. V tomto případě se jedná o přívodní kabel na měřeném stroji. Blokové schéma zařízení je uvedeno na obrázku (Obr. 10).



Obr. 10. Blokové schéma zařízení pro sběr dat ze strojů.

Výstup z transformátoru je zapojen do blíže nespecifikovaného modulu (dodavatel zařízení neposkytnul bližší informace o funkčnosti) a v tomto modulu je naměřený elektrický signál

zpracován. Modul je připojen přímo do počítačové sítě společnosti pomocí sítě Ethernet a data jsou přenášena protokolem TCP/IP. Pomocí tohoto připojení jsou naměřená data ukládána přímo do databáze, která je umístěna na interním serveru společnosti.

4.2 Popis SW

Softwarová část tohoto implementovaného řešení se skládá ze dvou základních částí. První část se týká SQL databáze jako uložiště naměřených dat strojů, druhou část potom tvoří webová aplikace, která slouží pro statistické vyhodnocení naměřených dat.

4.2.1 Databáze pro uložení naměřených dat

Pro ukládání dat je využito databázového serveru MariaDB. Jméno databáze je „*tdmaxt*“. Databáze obsahuje několik tabulek. Mimo tabulky vztahující se k vyhodnocení dat pomocí webové aplikace a několika dalších tabulek, je zde i tabulka pro evidenci pracovišť. V tabulce pracovišť jsou evidovaná data o strojích (název, typové označení, atd.), jejich hranice příkonu a identifikace strojů pro ukládání dat. Další tabulky obsahují data pro nastavení parametrů pro sběr dat. Nejdůležitější jsou tabulky s názvy „max_input_analog_x“, kde „x“ v názvu tabulky označuje číslo stroje. Naměřená data z každého připojeného stroje jsou tak uložena do samostatné tabulky. Struktura této tabulky je poměrně jednoduchá a skládá se pouze ze 4 atributů. Prvním atributem je „OID“ jako ID tabulky s primárním klíčem. Atribut „MaxPortID“ obsahující identifikaci stroje a zároveň je tento atribut stejný s číselným doplňkem názvu tabulky. Třetí atribut „DT“ slouží pro uložení data a času vloženého záznamu a v posledním jak název napovídá „Data“ jsou uloženy naměřené hodnoty příkonu stroje.

OID	MaxPortID	DT	Data
23673880	5	2017-01-05 10:00:11.00	0.11
23673881	5	2017-01-05 10:00:22.00	0.11
23673882	5	2017-01-05 10:00:33.00	0.13
23673883	5	2017-01-05 10:00:44.00	0.12
23673884	5	2017-01-05 10:00:55.00	0.13

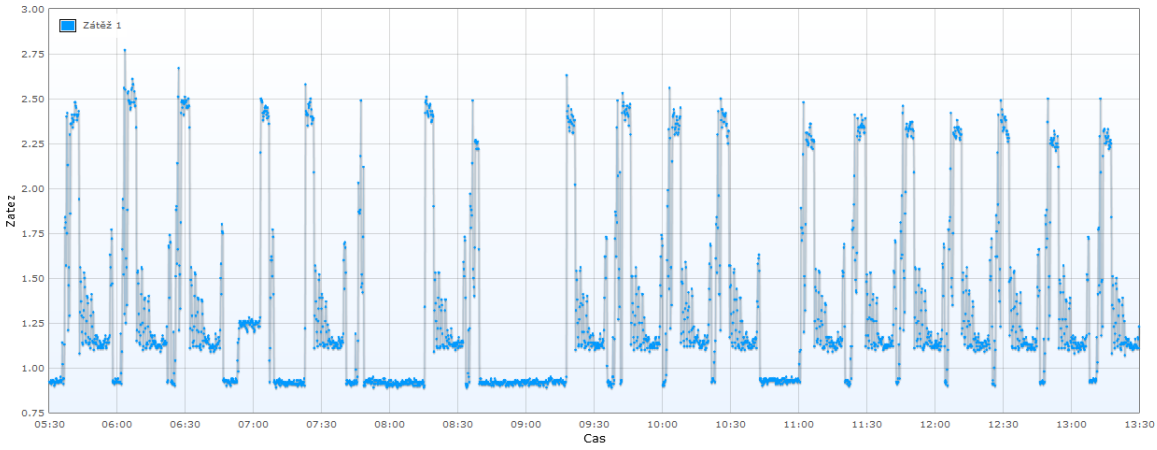
Obr. 11. Ukázka tabulky s uloženými daty ze strojů.

Záznamy jsou do tabulek strojů vkládány každých 11 sekund (Obr. 11). Tato časová hodnota je uživatelsky nastavitelná. Záznamy s naměřenými hodnotami jsou do tabulky uklá-

dány dle zvoleného časového intervalu bez ohledu na naměřenou hodnotu příkonu. Může tedy nastat situace, kdy je stroj vypnutý například z důvodu poruchy, ale do tabulky jsou stále vkládány nové záznamy s nulovou naměřenou hodnotou příkonu.

4.2.2 Webová aplikace

Součástí dodaného řešení firmou TriDat je i webová aplikace. V rámci této webové aplikace je k dispozici nastavení základních parametrů systému sběru dat a vyhodnocení nasbíraných dat ze strojů. Nastavení umožňuje definovat parametry pro identifikaci síťového komunikačního rozhraní zařízení, záznamy o jednotlivých strojích a nastavení uživatelů pro přístup k aplikaci. Kromě nastavení celého systému pro sběr dat ze strojů je v rámci této webové aplikace k dispozici vyhodnocení dat v podobě grafů. Vyhodnocení se týká hodnot naměřených dat v jednotlivých časech. Tyto výstupy bohužel nemají příliš velkou vypovídající hodnotu pro společnost, protože sice zobrazují průběh stavu stroje v čase, ale stavy jsou rozděleny do tří skupin. Jednou ze skupin je aktivní čas, při kterém se naměřený příkon stroje nachází nad zadanou hranicí. Naopak druhým stavem je neproduktivní čas stroje, při kterém jsou hodnoty naměřeného příkonu pod stanovenou hranicí. Posledním třetím stavem je situace, kdy v daném čase nejsou naměřena žádná data. Níže je ukázka grafu znázorňující hodnoty naměřeného příkonu stroje v průběhu určitého časového úseku (Obr. 12).



Obr. 12. Graf reprezentující naměřená data příkonu stroje.

II. PRAKTICKÁ ČÁST

5 ANALÝZA SYSTÉMU

Cílem tohoto projektu je vytvoření systému pro analýzu dat operátorů a obráběcích strojů ve strojírenské výrobě. Systém bude evidovat práci operátorů na jednotlivých strojích, evidenci činnosti související s přerušením práce na zakázce. V rámci systému budou analyzována data ze strojů. Na základě této analýzy bude operátor stroje upozorňován na neshodu mezi zadanými daty operátory a daty sesbíranými ze strojů. Systém bude doplněn o správu základních číselníků strojů, druhů činností (přerušení) a uživatelů.

Obsluhu systému budou provádět tyto uživatelské role:

- operátor/obsluha stroje
- mistr/vedoucí pracovník
- manažer
- správce systému (Administrátor)

Systém bude evidovat data operátorů. Každý operátor bude systémem požádán o zadání aktuální prováděné činnosti na stroji. Zadání činnosti bude spočívat ve výběru stroje, na kterém je činnost prováděna a zadání označení zakázky, která je vystavena v rámci podnikového informačního systému. Tato zadaná data budou po potvrzení operátorem uložena do databáze systému. Zadání činnosti operátora může být provedeno pro více strojů, protože se předpokládá, že operátor bude obsluhovat více než jen jeden stroj. V případě, že dojde k přerušení produktivní činnosti na zadaném stroji a zakázce, zadá operátor data přerušení (neproduktivní činnost). Pro zadání přerušení produktivní činnosti na stroji bude třeba, aby operátor vybral již dříve zadanou činnost. K této zadané činnosti následně operátor vybere typ přerušení, které nastalo při provádění produktivní činnosti. Přerušení bude po zadání operátorem uloženo do databáze systému.

Data zadaná operátorem budou průběžně analyzována systémem a porovnávána s daty, která jsou průběžně sbírána ze strojů a ukládána do databáze. Analýza bude spočívat v porovnání dat ze strojů a dat, která zadají operátoři. Bude porovnáno, jestli činnost zadaná operátorem je opravdu prováděna, resp. jestli této činnosti odpovídají data ze strojů. Snímání dat ze strojů a jejich uložení v databázi bylo popsáno v kapitole 4. Nastane-li situace, kdy operátor zadá produktivní činnost na zakázce, ale data ze stroje nebudou odpovídat produktivní činnosti, bude po uběhnutí stanovené doby prostoje, operátor na tuto situaci upozorněn. Pro upřesnění nastalé situace provede operátor zadání činnosti, která zastupuje aktuálně prováděnou neproduktivní činnost (např. seřizování stroje). Samozřejmě může

nastat i opačná situace, kdy bude mít operátor zadanou informaci o neproduktivní činnosti, ale stroj bude vykazovat parametry produkce. Nebude-li operátor reagovat na výzvu systému k upřesnění nastalé situace, bude tato nesrovnalost eskalována nadřízenému k prověření tohoto nesouladu. Předání informace bude provedeno odesláním emailu s upozorněním kompetentní osobě.

V rámci systému bude k dispozici také několik základních číselníků pro nastavení parametrů systému, číselník strojů, uživatelů, nastavení oprávnění uživatelů a druhů činností. Číselník strojů bude obsahovat informace o jednotlivých strojích. V rámci tohoto číselníku bude mimo číselného označení a názvu stroje u každého stroje evidována informace o nastavené hranici produktivního příkonu, doba, po kterou bude na daném stroji tolerován neproduktivní čas. Součástí systému bude také číselník uživatelů a jejich rolí, pro nastavení uživatelských oprávnění a úrovní pro přístup do systému. Číselník činností bude obsahovat informace o typech prováděných činností na stroji.

Poslední částí systému bude prezentace výsledků porovnání dat. Hlavním výstupem bude zobrazení statistiky efektivního využití strojů. Dále bude v systému dostupný panel s přehledem aktuálního stavu strojů.

5.1 Specifikace požadavků

Specifikace požadavků na systém byla vytvořena na základě několika provedených interviewu, která byla vedena mezi mnou a zástupcem výrobní společnosti. Při prvním setkání vznikla kostra požadavků, v rámci které byly zmíněny hlavní části systému. Při hlubší analýze byly následně upřesňovány a doplňovány o nově zjištěné požadavky. Požadavky jsou rozděleny do následujících skupin.

5.1.1 Funkční požadavky

- Systém bude obsahovat evidenci prováděné produktivní činnosti operátory na strojích. Bude obsahovat čas zahájení a ukončení činnosti, označení zakázky, označení stroje, na kterém je činnost prováděna a identifikaci obsluhy stroje.
- Systém bude obsahovat evidenci přerušení produktivní činnosti na strojích, tedy evidenci neproduktivní činnosti jako je seřizování, kontrola, údržba a podobně. Evidence bude obsahovat druh prováděné neproduktivní činnosti, čas zahájení a ukončení přerušení, stroj, operátora a výrobní zakázku, ke které se neproduktivní činnost váže.

- Systém bude průběžně zpracovávat data sesbíraná ze strojů uložená v databázi třetí strany.
- Systém bude průběžně analyzovat data ze strojů a porovnávat s daty zadanými operátory. Analýzou dat z obou systémů bude kontrolována shoda informací mezi daty ze strojů a daty zadanými operátory.
- Systém bude při vzniku nesouladu mezi daty operátorů a strojů na tuto skutečnost operátora upozorňovat. Pokud nebude po opakovaném upozornění operátora sjednána náprava, upozornění bude zasláno kompetentnímu nadřízenému pracovníkovi.
- Systém bude řídícím pracovníkům poskytovat prezentaci zpracovaných dat v přehledné formě.
- Systém bude obsahovat číselník pro správu jednotlivých strojů.
- Systém musí obsahovat evidenci nastavení a parametrů pro stanovení produktivní činnosti stroje. Dále musí být možnost uložit maximální povolený čas pro přerušení produktivní činnosti stroje.
- Systém bude obsahovat evidenci operátorů strojů pro jejich jednoznačnou identifikaci. Mimo jména a příjmení bude systém u operátorů evidovat také jejich osobní číslo.

5.1.2 Nefunkční požadavky

- Systém bude podporovat uživatelská oprávnění. Bez platného uživatelského jména a hesla a dostatečných oprávnění k provedení dané operace nebude možné operaci provést.
- Systém musí být možné provozovat na mobilních zařízeních (tablety, mobilní telefony) s různou velikostí displeje.
- Obsluha systému bude velmi snadná. Cílem je, aby operátoři pro práci se systémem potřebovali co nejméně času.
- Náklady na licence použitých technologií musí být minimální.

5.1.3 Doménové požadavky

- Systém bude provozován na existujícím technickém vybavení společnosti.
- Systém bude zpracovávat data ze strojů, jež jsou sbírána již provozovaným systémem třetí strany.

5.2 Případy užití

V rámci přehlednějšího zobrazení všech případů užití celého systému, jsou tyto rozděleny do menších skupin, které funkčně vytváří samostatné celky. U každého diagramu jsou dále doplněny scénáře jednotlivých případů užití.

5.2.1 Kontrola produkce strojů a operátorů

V tomto digramu jsou obsaženy všechny funkčně důležité části systému, které poskytují funkcionalitu pro sběr dat od operátorů, zpracování dat ze strojů a systém upozornění na zjištěné nedostatky mezi daty ze strojů a operátorů. V diagramu je uvedeno hned několik aktérů (Obr. 13). Tři aktéři zastupující uživatele, další dva potom připojený systém pro sběr dat ze strojů a systém pro zprávu dat od operátorů. Mezi aktéry operátor, mistr a manažer je zavedeno zobecnění, znázorňující pravomoci, resp. úroveň oprávnění. Zjednodušeně by se dalo říci, že vše co smí operátor, smí i mistr, přičemž mistr má navíc další dostupné funkce. Podobná je i situace u manažera. Manažer má všechny funkce jako mistr (tedy i ty, které má operátor) a navíc funkcionalitu, kterou mistr ani operátor nemají.

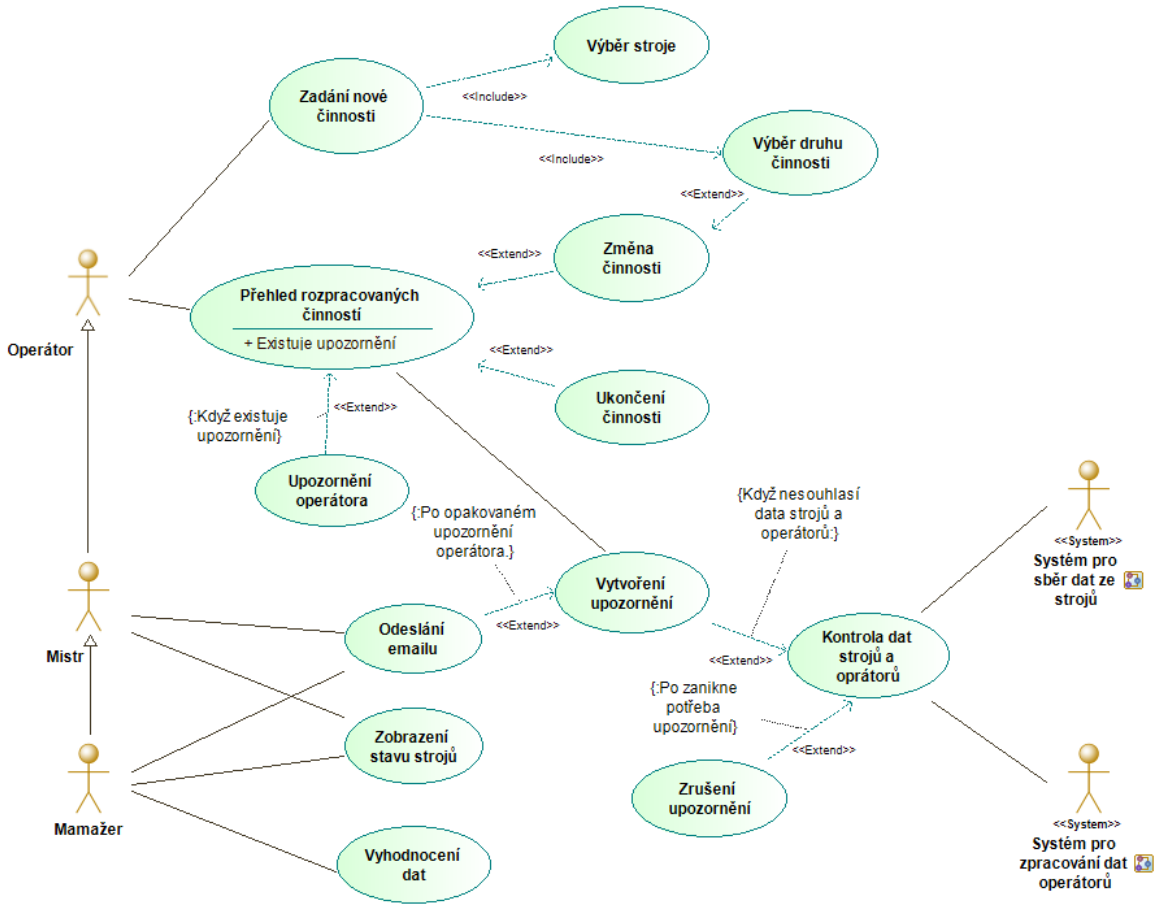
5.2.1.1 Založení nové činnosti

1. Případ užití začíná, když oprávněný operátor zvolí z menu položku pro zadání nové činnosti.
2. Systém zobrazí formulář pro zadání nové činnosti.
3. Operátor vyplní požadované informace do formuláře.
    a. INCLUDE Výběr stroje.
    b. INCLUDE Výběr druhu činnosti.
4. Operátor po vyplnění všech údajů zvolí volbu pro vložení nové činnosti.
5. Systém provede vložení nové činnosti.
6. Případ užití končí.

5.2.1.2 Přehled rozpracovaných činností

1. Případ užití začíná, pokud operátor zvolí z menu položku pro zobrazení rozpracované činnosti.
    a. ALT Případ užití začíná, když se operátor přihlásí do systému.
    b. ALT Případ užití začíná, když operátor zadá novou činnost.
    c. ALT Případ užití začíná, když operátor změní činnost.

2. Systém zobrazí přehled rozpracovaných činností přihlášeného operátora.
3. IF operátor zvolí změnu činnosti.
    a. EXTEND Změna činnosti.
4. IF operátor zvolí ukončení činnosti.
    a. EXTEND Ukončení činnosti.
5. Systém provádí v pravidelných intervalech kontrolu existence upozornění pro přihlášeného operátora.
6. IF existuje-li upozornění pro přihlášeného operátora.
    a. EXTEND Upozornění operátora.
7. Případ užití končí.



Obr. 13. Diagram případů užití pro kontrolu produkce strojů a operátorů.

5.2.1.3 Změna činnosti

1. Případ užití začíná, když operátor zvolí volbu pro změnu činnosti.
2. Systém zobrazí formulář pro zadání změny činnosti u vybraného záznamu.
3. IF operátor vybere volbu pro výběr druhů činností.

a. EXTEND Výběr druhu činnosti.

4. Operátor po vyplnění všech údajů zvolí volbu pro uložení změny činnosti.
5. Systém provede uložení změny činnosti. Při uložení zaeviduje datum a čas provedené změny.
6. Případ užití končí.

5.2.1.4 Ukončení činnosti

1. Případ užití začíná, když operátor zvolí volbu pro ukončení činnosti.
2. Systém provede ukončení vybrané činnosti. Při ukončení zaeviduje datum a čas provedeného ukončení.
3. Případ užití končí.

5.2.1.5 Výběr stroje

1. Případ užití začíná, když operátor zvolí volbu pro výběr stroje.
2. Systém zobrazí seznam s dostupnými stroji v systému.
3. Operátor vybere jednu položku ze zobrazeného seznamu.
4. Systém po výběru stroje převezme hodnotu do formuláře a zavře otevřený seznam.
4. Případ užití končí.

5.2.1.6 Výběr druhu činnosti

1. Případ užití začíná, když operátor zvolí volbu pro výběr druhu činnosti.
2. Systém zobrazí seznam s dostupnými druhy činností v systému.
3. Operátor vybere jednu položku ze zobrazeného seznamu.
4. Systém po výběru druhu činnosti převezme hodnotu do formuláře a zavře otevřený seznam.
5. Případ užití končí.

5.2.1.7 Upozornění operátora

1. Případ užití začíná, když operátor zvolí volbu pro zobrazení rozpracované činnosti.
2. Systém vybere upozornění pro přihlášeného operátora a stroj.
3. Systém zobrazí operátorovi upozornění s informací o nesouladu dat ze strojů a operátorů.
4. Případ užití končí.

5.2.1.8 Kontrola dat strojů a operátorů

1. Případ užití je automaticky spouštěn v pravidelných časových intervalech pomocí plánovače úloh.
2. Systém provede postupně kontrolu u každého stroje.
3. Systém u jednotlivých strojů porovná data ze strojů s daty od operátorů.
4. IF nesouhlasí data ze strojů s daty od operátorů.
    a. EXTEND Vytvoření upozornění.
5. IF zanikne potřeba upozornění operátora, tedy data ze strojů souhlasí s daty od operátorů.
    a. EXTEND Zrušení upozornění.
6. Případ užití končí.

5.2.1.9 Vytvoření upozornění

1. Případ užití začíná, když nesouhlasí data strojů a operátorů.
2. Systém vloží nový záznam pro upozornění operátora.
3. IF pokud se jedná o opakované upozornění k jednomu stroji. Počet opakování je 3.
    a. EXTEND Odeslání emailu
4. Případ užití končí.

5.2.1.10 Zrušení upozornění

1. Případ užití začíná, když zanikne potřeba pro upozornění operátora.
2. Systém provede zrušení upozornění pro daný stroj.
3. Případ užití končí.

5.2.1.11 Odeslání emailu

1. Případ užití začíná po opakovaném upozornění operátora stroje.
2. Systém odešle email s upozorněním na nesoulad mezi daty ze strojů a daty operátorů. Email odešle na uložené adresy aktérů mistr a manažer.
3. Systém uloží informaci o odeslání emailu kompetentním osobám.
4. Případ užití končí.

5.2.1.12 Zobrazení stavu strojů

1. Případ užití začíná, když mistr nebo manažer zvolí z menu volbu pro zobrazení stavu strojů.

a. ALT Případ užití začíná, když se mistr nebo manažer přihlásí do systému.

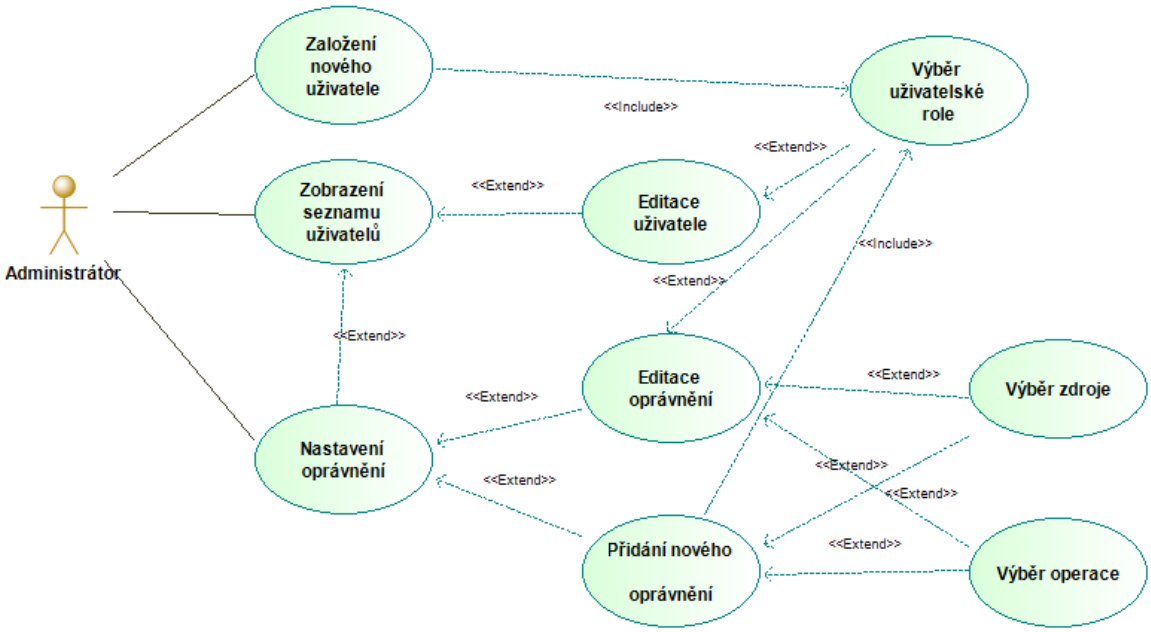
2. Systém zobrazí aktuální stav, v jakém se nacházejí jednotlivé stroje i s doplňujícími údaji.
3. Případ užití končí.

5.2.1.13 Vyhodnocení dat

1. Případ užití začíná, když uživatel zvolí z menu vyhodnocení dat.
2. Systém zobrazí data efektivity využití všech strojů po měsících za poslední rok.
   a. ALT Systém zobrazí data efektivity využití podle strojů a měsíců za poslední rok.
3. Případ užití končí.

5.2.2 Správa uživatelů

V rámci tohoto funkčního celku je řešena správa uživatelů systému a jejich uživatelských oprávnění (Obr. 14). Dle požadavků smí správu uživatelů provádět pouze administrátor systému. Podklady pro nastavení uživatelských oprávnění poskytuje administrátorovi vedení společnosti.



Obr. 14. Diagram případů užití pro správu uživatelů.

5.2.2.1 Založení nového uživatele

1. Případ užití začíná, když přihlášený uživatel *Administrátor* zvolí volbu z menu pro založení nového uživatele.
2. Systém zobrazí formulář pro přidání nového uživatele.
3. Uživatel vyplní požadované údaje o uživateli.
    a. INCLUDE Výběr uživatelské role.
4. Uživatel po vyplnění všech údajů zvolí volbu pro vytvoření účtu.
5. Systém provede vložení nového uživatele.
6. Případ užití končí.

5.2.2.2 Zobrazení seznamu uživatelů

1. Uživatel Administrátor vybere z menu volbu pro správu uživatelů.
2. Systém zobrazí seznam všech uživatelů.
3. IF uživatel zvolí volbu pro editaci uživatele.
    a. EXTEND Editace uživatele.
4. Případ užití končí.

5.2.2.3 Editace uživatele

1. Případ užití začíná zvolením editace uživatele.
2. Systém zobrazí formulář pro editaci uživatele.
3. Uživatel provede editaci údajů o uživateli.
4. IF uživatel potřebuje změnit roli uživatele.
    a. EXTEND Výběr uživatelské role.
5. Uživatel po vyplnění všech údajů zvolí volbu pro vytvoření účtu.
6. Systém provede uložení provedených změn uživatele.
7. Případ užití končí.

5.2.2.4 Výběr uživatelské role

1. Případ užití začíná, když uživatel zvolí volbu pro výběr uživatelské role.
2. Systém zobrazí seznam s dostupnými uživatelskými rolemi v systému.
3. Uživatel vybere jednu položku ze zobrazeného seznamu.
4. Systém po výběru role převezme hodnotu do formuláře a zavře otevřený seznam.
5. Případ užití končí.

5.2.2.5 Nastavení oprávnění

1. Případ užití začíná, když uživatel vybere volbu pro nastavení oprávnění.
2. Systém zobrazí přehled zadaných oprávnění v systému.
3. IF uživatel zvolí volbu pro editaci oprávnění.
    a. EXTEND Editace oprávnění.
4. IF uživatel zvolí volbu pro přidání nového oprávnění.
    a. EXTEND Přidat nové oprávnění.
5. Případ užití končí.

5.2.2.6 Přidání nového oprávnění

1. Případ užití začíná, když uživatel vybere volbu pro přidání nového oprávnění.
2. Systém zobrazí formulář pro zadání nového oprávnění.
3. IF uživatel zvolí volbu pro výběr role.
    a. INCLUDE Výběr uživatelské role.
4. IF uživatel zvolí volbu pro výběr zdroje.
    a. EXTEND Výběr zdroje.
5. IF uživatel vybere volbu výběr operace.
    a. EXTEND Výběr operace.
6. Uživatel po vyplnění všech údajů zvolí volbu pro vložení oprávnění.
7. Systém provede vložení nového oprávnění.
8. Případ užití končí.

5.2.2.7 Editace oprávnění

1. Případ užití začíná, když uživatel vybere volbu pro editaci oprávnění.
2. Systém zobrazí formulář pro editaci vybraného oprávnění.
3. IF uživatel zvolí volbu pro výběr role.
    a. INCLUDE Výběr uživatelské role.
4. IF uživatel zvolí volbu pro výběr zdroje.
    a. EXTEND Výběr zdroje.
5. IF uživatel vybere volbu výběr operace.
    a. EXTEND Výběr operace.
6. Uživatel po editaci údajů zvolí volbu pro uložení oprávnění.
7. Systém provede uložení oprávnění.

8. Případ užití končí.

5.2.2.8 Výběr zdroje

1. Případ užití začíná, když uživatel zvolí volbu pro výběr zdroje.
2. Systém zobrazí seznam s dostupnými zdroji v systému.
3. Uživatel vybere jednu položku ze zobrazeného seznamu se zdroji.
4. Systém po výběru zdroje převezme hodnotu do formuláře a zavře otevřený seznam.
5. Případ užití končí.

5.2.2.9 Výběr operace

1. Případ užití začíná, když uživatel zvolí volbu pro výběr operace.
2. Systém zobrazí seznam s dostupnými operacemi v systému.
3. Uživatel vybere jednu položku ze zobrazeného seznamu s operacemi.
4. Systém po výběru operace převezme hodnotu do formuláře a zavře otevřený seznam.
5. Případ užití končí.

5.2.3 Nastavení

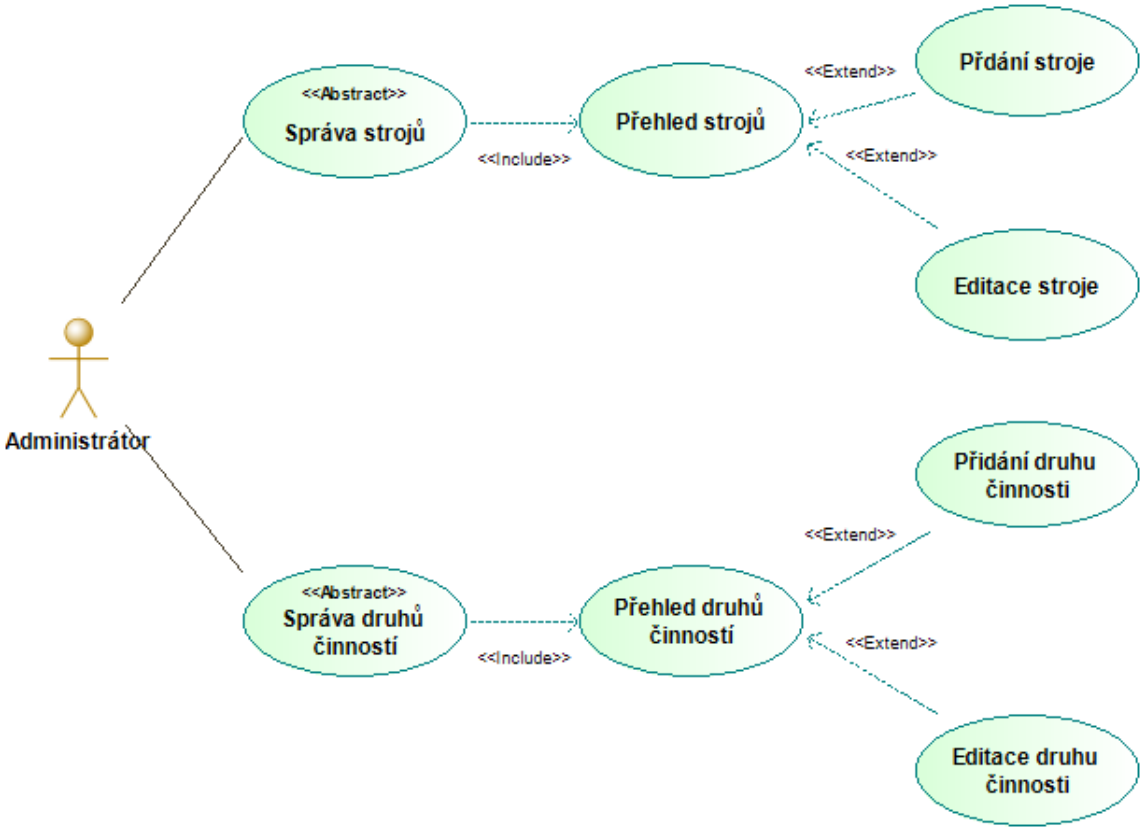
Následující případy užití pokrývají funkční oblast pro nastavení číselníků systému. Jedná se o správu číselníku strojů a správu číselníku druhů činností. Diagram je na obrázku (Obr. 15).

5.2.3.1 Správa strojů

Tento abstraktní případ užití je vytvořen pouze pro přehlednost. Správa strojů reprezentuje případy užití, týkající se správy dat strojů.

5.2.3.2 Přehled strojů

1. Případ užití začíná, když uživatel klikne na volbu správa strojů
2. Systém zobrazí přehled stojů evidovaných v systému.
3. IF uživatel zvolí volbu editovat.
    a. EXTEND Editovat stroj.
4. IF uživatel zvolí volbu přidat stroj.
    a. EXTEND Přidat stroj.
5. Případ užití končí.



Obr. 15. Diagram případů užití pro obsluhu základních číselníků systému.

5.2.3.3 Přidání stroje

1. Případ užití začíná po kliknutí na volbu pro přidání stroje.
2. Systém zobrazí formulář pro přidání nového stroje.
3. Uživatel po zadání údajů zvolí volbu pro vložení stroje.
4. Systém provede vložení zadaného stroje.
5. Případ užití končí.

5.2.3.4 Editace stroje

1. Případ užití začíná po kliknutí na volbu pro editaci stroje.
2. Systém zobrazí formulář pro editaci vybraného stroje.
3. Uživatel po úpravě údajů zvolí volbu pro uložení stroje.
4. Systém provede uložení změn vybraného stroje.
5. Případ užití končí.

5.2.3.5 Správa druhů činností

Abstraktní případ užití správa druhů činností reprezentuje případy užití, týkající se správy dat druhů činností.

5.2.3.6 Přehled druhů činností

1. Případ užití začíná, když uživatel klikne na volbu druhy činností.
2. Systém zobrazí přehled druhů činností evidovaných v systému.
3. IF uživatel zvolí volbu editovat.
    a. EXTEND Editovat druh činnosti.
4. IF uživatel zvolí volbu přidat druh činnosti.
    a. EXTEND Přidat druh činnosti.
5. Případ užití končí.

5.2.3.7 Přidání druhu činnosti

1. Případ užití začíná po kliknutí na volbu pro přidání druhu činnosti.
2. Systém zobrazí formulář pro přidání nového druhu činnosti.
3. Uživatel po zadání údajů zvolí volbu pro vložení druhu činnosti.
4. Systém provede vložení zadaného druhu činnosti.
5. Případ užití končí.

5.2.3.8 Editace druhu činnosti

1. Případ užití začíná po kliknutí na volbu pro editace druhu činnosti.
2. Systém zobrazí formulář pro editaci vybraného druhu činnosti.
3. Uživatel po úpravě údajů zvolí volbu pro uložení druhu činnosti.
4. Systém provede uložení změn vybraného druhu činnosti.
5. Případ užití končí.

5.3 Webová vs. mobilní aplikace

Jedním ze základních požadavků na systém byla skutečnost, že systém bude provozován na běžných osobních počítačích, ale také mobilních zařízeních jako jsou mobilní telefony a tablety. Tento poměrně vágně vypadající požadavek sebou ovšem nese problematiku jistých rozdílů při provozování aplikací na osobním počítači a mobilním zařízení. Zjednodušeně by se dalo říci, že v případě mobilních zařízení máme zpravidla menší velikost zobra-

zovacího displeje a nižší výkon mobilního zařízení, než u osobních počítačů. To ovšem nejsou jediné rozdíly. Liší se také operačním systémem. V souvislosti s těmito fakty byla při návrhu aplikace diskutována otázka, jestli bude systém vyvinut jako webová aplikace s přizpůsobením vzhledu pro použití i na mobilních zařízeních nebo webová aplikace pro osobní počítače a mobilní aplikace pro mobilní zařízení. Výhody a nevýhody každé z těchto variant jsou diskutovány v následujících odstavcích.

5.3.1 Webová aplikace

O výhodách webových aplikací již bylo hovořeno v kapitole 1. Pro běh vlastní webové aplikace je potřeba „jen“ webový prohlížeč. Webový prohlížeč je pro webovou aplikaci jakýmsi klientem. Tohoto klienta má drtivá většina operačních systémů k dispozici již v rámci instalace operačního systému nebo je možné webový prohlížeč do operačního systému doinstalovat. Díky tomuto faktu je tak možné provozovat webovou aplikaci téměř na jakémkoli zařízení a operačním systému. To přináší výhodu v distribuci aktualizací na klienta, která v podstatě odpadá. Částečným omezením použití webových aplikací na mobilních zařízeních je velikost displeje a jeho rozlišení. Na těchto mobilních zařízeních je zpravidla menší velikost displeje než na osobních počítačích a zároveň bývá nižší i rozlišení displeje. To sebou samozřejmě nese komplikace v podobě nekorektního zobrazení webové aplikace na displeji mobilního zařízení. Tuto nepříjemnost je ovšem možné řešit vytvořením webové aplikace s responzivním designem. Touto vlastností je dosaženo korektního zobrazení webové aplikace i na omezeném displeji mobilního zařízení. Důsledkem využití implementace systému jako webové aplikace s responzivním designem je vytvoření pouze jedné verze systému, a to jak pro osobní počítače, tak pro mobilní zařízení.

Hlavními argumenty proti vytvoření webové aplikace je bezpečnost webových aplikací, která byla detailněji popsána v kapitole 3. Jedním z dalších argumentů proti vytvoření webové aplikace může být nekompatibilita klientů webových aplikací, tedy webových prohlížečů. Touto existující vzájemnou nekompatibilitou mezi jednotlivými webovými prohlížeči může dojít k nesprávnému zobrazení aplikace nebo dokonce k jejímu nekorektnímu fungování. Poslední zmíněnou nevýhodou je absence rozšířených možností ovládání zejména mobilních zařízení ve srovnání s nativní mobilní aplikací.

5.3.2 Mobilní aplikace

Na druhou stranu, výhod mobilních aplikací by mohlo být hned několik. Dnešní moderní vývojové technologie pro mobilní zařízení také umožňují vytváření multiplatformní aplikace, podobně jako je tomu u webových technologií. Ale ne jenom to, umožňují vytvoření téměř jakékoli aplikace. Ovšem v souvislosti s tímto vytvářeným systémem jako mobilní aplikace byla řešena zcela jiná otázka. V souvislosti s požadavky na tento vytvářený systém bylo jedním z nefunkčních požadavků provozování systému na mobilních zařízeních. Zprvu vypadalo rozhodnutí poměrně jednoznačně a při úplně prvních setkáních v rámci analýzy to vypadalo, že část systému určená pro operátory bude vytvořena jako mobilní a část pro management podniku a administrátora systému jako webová aplikace. Následně však při opakovaných setkání se zástupcem společnosti byly řešeny další otázky, které se mimo jiné týkaly instalace systému, distribuce aktualizací a jeho správy.

Při diskuzi se zástupcem firmy a vyjmenování všech požadavků patřila mezi výhody mobilní aplikace možnost zasílat operátorům „push notifikace[27]“. Další diskutovanou výhodou byla skutečnost, že pomocí mobilní aplikace je možné lépe ovládat dané zařízení. Zejména možnost obsluhy zobrazovacího displeje a reproduktorů přístroje pro přehrávání výstražných zvuků byla rozhodně vlastnost, která přidávala plusové body pro volbu mobilní aplikace. Možnost pracovat s fotoaparátem mobilního zařízení, zamezit samovolnému vypnutí displeje mobilního zařízení nebo obsluha vibrací byly argumenty pro vytvoření mobilní aplikace.

V souvislosti s rozhodnutím, byly nastíněny také negativní vlastnosti mobilní aplikace. Jednou z prvních diskutovaných myšlenek byla situace, kdy bude třeba aplikaci naistalovat do každého použitého mobilního zařízení. Další navazující negativní vlastností byla otázka aktualizací aplikace, tedy nahrávání nové verze aplikace. Instalace nebo aktualizace by zpravidla vyžadovala asistenci IT personálu pro buďto provedení těchto úkonů nebo alespoň pomoc s jejich provedením. Ovšem společnost nemá IT pracovníka dostupného trvale a tak tato otázka hrála významnou roli. Dalším argumentem byla skutečnost, že bude třeba vytvořit dvě aplikace, protože systém musí být samozřejmě možné provozovat také na

[27] Push notifikace – oznámení, zaslané serverem klientovi.

desktopových počítačích. Znamenalo by to tedy vytvoření dvou aplikací, přičemž část aplikace pro desktopové systémy by byla navíc částečně duplicitní.

5.3.3 Rozhodnutí

Na základě všech argumentů uvedených v několika předcházejících odstavcích bylo rozhodnuto o vytvoření pouze webové aplikace, která bude přizpůsobena pro provoz na mobilních zařízeních s menší velikostí displeje. Vytvořenou webovou aplikaci nebude třeba na osobní počítače ani na mobilní zařízení instalovat, distribuce aktualizací bude automatická díky provedení aktualizací správcem systému přímo na serveru společnosti. Díky volbě webové aplikace je vytvořen pouze jeden systém, který je možné provozovat jak na osobních počítačích, tak na mobilních zařízeních. Argumenty pro webovou aplikaci převýšily výhody mobilní aplikace, jakými jsou dostupnost „push notifikací“ nebo větších možností při ovládání zařízení.

5.4 Diagram modelu tříd

Diagram modelu tříd je vzhledem ke své velikosti graficky znázorněn v příloze PII. Při vytváření modelu jsem se dopustil drobného zjednodušení zejména z důvodu přehlednosti. Cílem analytického modelu je zachytit vztahy mezi jednotlivými třídami. Dále bylo přihlédnuto ke skutečnosti, že je již předem známa technologie, ve které bude tento systém implementován. V diagramu modelu tříd jsou znázorněny pouze třídy z vrstvy *presenter* a *model*. Tyto vrstvy tvořící architekturu aplikace jsou podrobněji popsány v kapitole 6.

Z diagramu v příloze je patrné, že vztahy mezi třídami jsou mezi vrstvami, nikoli mezi třídami v rámci jednotlivých vrstev. Výjimku tvoří pouze *NoticeManager*, který je ve vztahu s *ProductionManager*, za účelem zjištění rozpracované výroby pro stroj a uživatele. Druhou výjimkou je vztah mezi třídou *SecurityManager* a *SecurityDataManager*. Pomocí *SecurityDataManager* jsou načítána data oprávnění uživatelů z databáze a ukládána do instance třídy *SecurityManager*. Všechny ostatní vytvořené vztahy mezi třídami jsou napříč vrstvami.

SignPresenter je třída obsahující funkcionalitu pro řízení správy uživatelů. Jedná se zejména o zajištění přihlášení a odhlášení uživatelů. Dále je to také přidávání nových uživatelů a provádění jejich změn. Obsahuje závislost na třídu *UserManager*, která poskytuje prostředky pro manipulaci s daty uživatelů.

Třída *HomepagePresenter* vyřizuje požadavky na zjištění aktuálního stavu všech strojů. Tyto požadavky jsou předávány pomocí vytvořené závislosti třídě *ProductionManager* ke zpracování. Třída *ProductionManager* poskytuje všechny prostředky pro správu dat o probíhajících činnostech uživatelů na strojích. Díky již zmíněné spolupráci s třídou *NoticeManager* poskytuje také relevantní data pro vznik upozornění.

Na tuto třídu má vytvořenu závislost další třída se jménem *ProductionPresenter*. Tato třída je základní pro řízení všech požadovaných operací v oblasti správy výroby resp. prováděných činností. Vzhledem ke komplexnosti poskytovaných informací je v této třídě vytvořena ještě závislost na třídu *NoticeManager*. Pomocí této závislosti řídí *ProductionPresenter* zpracování požadavků na upozornění uživatelů na nesoulad dat strojů a operátorů.

Třída *MachinePresenter* je řídícím prvkem pro akce spojené s agendou správy strojů. Je navázána na třídu *MachineManager*, která obsahuje funkcionalitu pro správu strojů. Velmi podobnou strukturu má i další v modelu uvedená třída se jménem *KindActivityPresenter*. Jejím úkolem je správa agendy druhů činností. I zde je vytvořena závislost na třídu *KindActivityManager*, která umožňuje manipulovat s daty druhů činností v databázi a provádět další požadované činnosti.

Řídící třídou pro správu uživatelských oprávnění je *SecurityDataPresenter*. Obsahuje funkcionalitu pro předání požadavků na správu oprávnění a vytváření formulářů pro tyto účely. Obsahuje závislost na třídu *SecurityDataManager*, která má dostupné metody pro práci s daty oprávnění uživatelů v databázi.

Poslední částí popisovaného modelu je třída *CheckMachinePresenter*. Tato třída má jednoduchou strukturu, která obsahuje pouze jednu metodu. Touto metodou je volání kontroly stavu strojů v jiné databázi a stavu dat zadaných operátory. Celá tato aplikační logika je obsažena v závislé třídě, kterou je *CheckMachineDataManager*. V této třídě je dostupná funkcionalita pro připojení obou databází a pro provedení patřičných kontrol.

6 NÁVRH ARCHITEKTURY APLIKACE

Jak již bylo zmíněno v úvodu tohoto textu, pro vývoj systému bude použit framework Nette. V rámci něj je systém rozdělen minimálně do třech vrstev za použití MVP architektury. Vedle tohoto základního členění je ještě doplněna vrstva pro vytváření formulářů.

6.1 Vrstva model

Vrstva *model* je umístěna v adresářové struktuře frameworku Nette se stejnojmenným názvem adresáře. Tato vrstva jako jediná v celé architektuře pracuje s daty v databázi. Vrstva obsahuje několik tříd, které jsou pojmenovány podle vzoru *<NazevModelu>Manager*. V těchto třídách jsou definovány metody pracující s daty. Mimo to jsou zde implementovány i metody, které provádí zpracování dat pro nadřazené vrstvy nebo naopak od nadřazených vrstev. Avšak v žádném případě model neví o existujících vyšších vrstvách architektury. Tato vrstva vytváří pro vyšší vrstvy jen rozhraní pro práci s databází a aplikační logiku.

6.2 Vrstva presenters

Vrstva *presenters* tvoří prostřední vrstvu mezi vrstvou *model* a vrstvou *view*. Přijímá požadavky od uživatele, předává je ke zpracování vrstvě *model* a výsledky vrácené modelem předá do *view*, tedy do šablony k vykreslení. Přes tuto vrstvu prochází všechno dění v systému. Vrstva *presenters* je v pozici řídícího článku celé architektury systému. Všechny *presentery* v systému jsou potomky třídy *Nette\Application\UI\Presenter*. Nachází se v adresáři *presenters* v adresářové struktuře frameworku Nette. Jméno *presenteru* je tvořeno z názvu modelu. Obecně tedy *<NazevModelu>Presenter*. Neplatí ovšem, že počet *modelů* musí být roven počtu *presenterů*.

6.3 Vrstva view

View tvoří poslední vrstvu architektury. Jejím posláním je zobrazování výsledků uživateli, které zpravidla obdrží od vrstvy *presenters*. Provádí také zobrazení vytvořených komponent. Jak správně obdržená data zobrazit je informace, kterou zná šablonovací systém Latte, blíže popsaný v kapitole 2.2.7. Pomocí šablonovacího systému je řízeno rozvržení zobrazení dat na stránce. Šablony ve frameworku Nette jsou uloženy v adresáři *templates* v adresářové struktuře frameworku Nette. Adresáře jednotlivých šablon, které se váží k *presenterům* jsou pojmenovány podle jmen *presenterů*, ale neobsahují část slova „*presen-*

ter“. Například adresář pro šablony výroby vychází z *presenteru ProductionPresenter* a adresář pro šablony je pouze *Production*. Šablona *@layout* tvoří základ pro rozvržení stránky (layout) pro všechny ostatní šablony systému. Všechny ostatní šablony následně používají pro rozvržení stránky právě toto výchozí nastavení.

7 IMPLEMENTACE PROTOTYPOVÉ APLIKACE

Tato kapitola je věnována popisu implementace prototypové aplikace. Obsahem kapitoly není detailní popis kódu každé třídy, ale jen základní struktura a některé specifické detaily. V rámci této kapitoly je také popsána struktura databáze. Diagram struktury databáze je uveden v příloze P I.

7.1 Struktura databáze

Databázová struktura byla definována s ohledem na provedenou analýzu systému a v rámci ní vytvořené požadavky na systém. Podle těchto podkladů již bylo možné definovat jednotlivé tabulky systému a jejich atributy. Také bylo možné definovat vztahy mezi tabulkami. V rámci tabulek jsou doplněny integritní omezení atributů, poskytující pravidla pro správnost zadávaných dat. Jméno databáze je *productioncheck*. Kompletní diagram databáze je zobrazen z důvodu velikosti v příloze P I tohoto textu.

7.1.1 Tabulka uzivatele

Tato tabulka (Tab. 1) uchovává všechna data o uživatelích. Data uživatelů z této tabulky jsou používána pro přihlašování uživatelů do systému. Primárním klíčem[28] tabulky je atribut *idUzivatel*. Je u něj nastavena automatická inkrementace. Další atributem je *prihlasovaciJmeno* a slouží pro uložení přihlašovacího jména uživatele do systému. Atribut pro zadání přihlašovacího jména má dále nastaven unikátní klíč[29], aby nenastala situace, že budou mít dva uživatelé stejné přihlašovací jméno do systému. U tohoto atributu není povoleno zadání nedefinované hodnoty „null“[30].

Tab. 1. Struktura tabulky uzivatele.

Atribut	**Doména**	**Null hodnota**
idUzivatel	int(11)	Ne
prihlasovaciJmeno	varchar(100)	Ne
heslo	varchar(200)	Ne

[28] Primární klíč (Primary key) – jednoznačně identifikuje každý záznam v tabulce.
[29] Unikátní klíč (Unique key) – omezení vložení duplicitního záznamu do tabulky.
[30] Null – nedefinovaná hodnota.

Atribut	**Doména**	**Null hodnota**
idRole	int(11)	Ne
osobniCislo	varchar(5)	Ano
jmeno	varchar(200)	Ne
prijmeni	varchar(250)	Ne
token	varchar(100)	Ano
zalozeno	timestamp	Ne
email	varchar(100)	Ano
upozorneni	tinyint(1)	Ano
upozorneniEmail	varchar(100)	Ano
blokovan	tinyint(1)	Ne
blokovanOd	timestamp	Ano

Atribut *heslo* je podle názvu připraven pro uložení hesla uživatele. Zadané heslo je v tabulce uloženo v hash podobě viz kapitola 8.6. Zadání hesla je pro uživatele povinné, není povoleno zadání hodnoty „null“. Atribut *idRole* je cizím klíčem do tabulky *uziv_role*. Jeho zadání je povinné. Z důvodu zastupující vlastnosti cizího klíče, je na tomto atributu umístěn index *roleId*. Dalším atributem této tabulky je *osobniCislo*. Slouží pro uložení osobního čísla zaměstnance. *OsobniCislo* není údaj povinný a není nutné jej vyplnit. Pokud ovšem vyplněn je, provádí se kontrola na duplicitu pomocí unikátního klíče *osobniCislo*. Následuje *jmeno* a *prijmeni*. Je z názvu zřejmé, že se jedná o atributy pro zadání jména a příjmení uživatele. Jejich vyplnění je povinné. Atribut *token* je připraven pro ukládání identifikačního klíče uživatele. Prozatím tato funkcionalita není využívána, a tudíž není nijak využit ani tento atribut. *Zalozeno* obsahuje údaje o vzniku záznamu v podobě data a času. Je automaticky doplňován při vzniku nového záznamu. Atribut *email* slouží pro uložení emailové adresy uživatele. Zadání emailové adresy není povinné, ale pokud zadání proběhne, musí být zajištěno, že zadaná emailová adresa v databázi neexistuje. K tomuto účelu slouží definovaný unikátní klíč. Pravdivostní atribut *upozorneni* slouží k uložení informace, zda má daný uživatel dostávat emaily s upozorněním na nesoulad stavu strojů.

Pokud obsahuje hodnotu „1“, bude dostávat upozornění, v opačném případě nikoli. Emailová adresa pro zasílání upozornění je uložena v *upozorneniEmail*. Její zadání není povinné. Dalším atributem obsahující pravdivostní hodnotu je *blokovan*. Zadání je povinné a výchozí hodnota je nastavena na 0, tedy na nepravdu. V případě nastavení blokování uživatele je do atributu *blokovanOd* vhodné uložit aktuální datum a čas této události. Zadání není povinné.

7.1.2 Tabulka uziv_role

Tabulka *uziv_role* je v relaci s tabulkou uživatelů. Obsahuje několik atributů, které jsou názorně uvedeny v tabulce (Tab. 2).

Tab. 2. Struktura tabulky uziv_role.

Atribut	**Doména**	**Null hodnota**
idRole	int(11)	Ne
nazev	varchar(50)	Ne
rodic	varchar(50)	Ano
popis	varchar(250)	Ano

Atribut *idRole* je primárním klíčem tabulky s nastavením automatické inkrementace. Dalším atributem je *nazev*. Slouží pro zapsání názvu role. Zadání je povinné a díky unikátnímu indexu nesmí být duplicitní. Atribut *rodic* slouží pro uvedení názvu role, ze které zadaná role zdědí oprávnění. Zadání není povinné, protože každá role nemusí mít svého předka. Je zde povolena duplicita zadaných hodnot, protože nastávají situace, kdy několik zadaných rolí dědí z role jedné. Posledním atributem této tabulky je *popis*. Tento atribut je určen pro zapsání krátkého popisu, vysvětlující zadanou roli. Zadání pole není povinné, je jen na administrátorovi systému, jestli popis použije.

7.1.3 Tabulka uziv_prava

Tato tabulka je určena pro evidenci uživatelských oprávnění uživatelů. Pomocí těchto dat jsou vytvářena uživatelská oprávnění administrátorem systému volbou povolení nebo zakázání přístupu ke konkrétní kombinaci uživatelské role, zdroje a prováděné akce. Struktura tabulky je znázorněna v tabulce (Tab. 3). Prvním atributem je *idPrava*, který je primár-

ním klíčem tabulky s nastavením automatického inkrementování id. Atribut *povoleno* slouží k zadání informace o tom, jestli kombinace hodnot role, zdroj a operace je povolena (allow) nebo zamítnuta (deny). Hodnota „1“ znázorňuje hodnotu povoleno (allow), hodnota nula potom opak, tedy zamítnuto (deny). Zadání tohoto atributu je povinné.

Tab. 3. Struktura tabulky uziv_prava.

Atribut	**Doména**	**Null hodnota**
idPrava	int(11)	Ne
povoleno	tinyint(1)	Ne
idRole	int(11)	Ne
idZdroj	int(11)	Ano
idOperace	int(11)	Ano

Atribut *idRole* slouží pro zadání vazby na roli uživatele. Zadání je povinné. Je cizím klíčem do tabulky uživatelských rolí s názvem *role*. Další atribut s názvem *idZdroj* slouží pro zadání reference do tabulky zdrojů s názvem *uziv_zdroje*. Přesněji řečeno je cizím klíčem do této tabulky. Ovšem její zadání není povinné. V případě, že není zadána žádná hodnota zdroje (výchozí hodnota je null), je toto považováno za oprávnění ke všem zdrojům. Této hodnoty je použito v případě oprávnění administrátora, který má oprávnění na vše. Posledním atributem této tabulky je *idOperace*. Je cizím klíčem do tabulky prováděných operací se jménem *uziv_operace*. Její zadání není povinné ze stejných důvodů, jako bylo zmíněno u atributu *idZdroj*. V této tabulce je vytvořen unikátní klíč přes atributy *idRole*, *idZdroj* a *idOperace*. Tímto unikátním klíčem je zajištěno, že není možné zadat dva stejné záznamy, které se shodují ve výše vyjmenovaných atributech. Pokud by tento klíč neexistoval, mohlo by snadno dojít k nefunkčnosti oprávnění. Mohla by nastat situace, kdy bude pro stejnou kombinaci uživatelské role, zdroje a požadované operace jednou zadáno jako povolená a jednou jako zakázaná kombinace. Možným důsledkem by následně bylo, že se uživatel v rámci systému dostane k datům, na která oficiálně nemá oprávnění.

7.1.4 Tabulka uziv_zdroje

Tato tabulka obsahuje jména zdrojů odrážející jména presenterů, které existují v systému. Jméno zdroje je tvořeno jako název presenteru bez společného *presenter*. Například

v případě jména presenteru *SecurityDataPresenter* je hodnotou pro zdroj oprávnění pouze *SecurityData*. Struktura této tabulky je podrobněji znázorněna v tabulce (Tab. 4).

Tab. 4. Struktura tabulky uziv_zdroje.

Atribut	**Doména**	**Null hodnota**
idZdroj	int(11)	Ne
nazev	varchar(100)	Ne
rodic	varchar(100)	Ano
popis	varchar(200)	Ne

Prvním atributem je *idZdroj*. Je to primární klíč tabulky a je u něj nastavena automatická inkrementace hodnoty id. Atribut *nazev* je určen pro uložení názvu zdroje. Jeho zadání je povinné a obsahuje unikátní index pro zamezení zadání duplicitní hodnoty. *Rodic* je atribut pro zadání jména předka zdroje. Jeho uplatnění v tomto systému není adekvátní, protože zdrojů je jen několik. Ovšem, pokud se jedná o velmi rozsáhlý systém, může tento atribut pomoci při sestavování úrovní oprávnění. Atribut *popis* jak název vypovídá, slouží pro uložení popisu zdroje. Zadání je povinné, protože krátký název zdroje vznikající z názvu *presenteru* nemusí být zcela vypovídající. Proto je zde vyžadováno zadání stručného popisu jednotlivých zdrojů.

7.1.5 Tabulka uziv_operace

Tabulka *uziv_operace* slouží pro uložení dat o operacích. Uložená data odrážejí názvy akcí, které mají být provedeny. V tomto případě je názvem akce myšlen název metody v *presenteru* bez přidruženého názvu životního cyklu *presenteru*. Například vezmeme-li *SecurityDataPresenter* a v něm akci *actionEdit*, je jako název operace brána pouze část *Edit*.

Tab. 5. Struktura tabulky uziv_operace.

Atribut	**Doména**	**Null hodnota**
idZdroj	int(11)	Ne
nazev	varchar(100)	Ne
rodic	varchar(200)	Ne

Primárním klíčem této tabulky (Tab. 5) je atribut *idZdroj*. Je u něj nastaveno automatické inkrementování hodnoty id. Atribut *nazev* slouží pro uložení názvu operace (akce). Zadání hodnoty je povinné. Zadaná hodnota musí být unikátní. To je zajištěno díky existujícímu unikátnímu klíči. Je předpokladem, že pokud bude například akce *Edit* existovat v rámci více zdrojů, není nutné zadávat hodnotu akce pro každý zdroj opakovaně, ale pouze jednou pro všechny. Pro uložení popisu operace slouží atribut *popis*. Jeho zadání je povinné, protože bez zadání popisu, resp. vysvětlení zadané hodnoty operace, nemusí být zřejmé o jakou operaci (akci) se jedná.

7.1.6 Tabulka uziv_prihlaseni

Informace o přihlášení a odhlášení uživatelů systému jsou ukládány do této tabulky. Struktura této tabulky je poměrně jednoduchá viz tabulka (Tab. 6) a je také poslední ve vztahu k uživatelům nebo k jejich oprávnění.

Tab. 6. Struktura tabulky uziv_prihlaseni.

Atribut	**Doména**	**Null hodnota**
idPrihlaseni	int(11)	Ne
idUzivatel	int(11)	Ne
prihlasen	timestamp	Ne
odhlasen	timestamp	Ano

Atribut i*dPrihlaseni* je primárním klíčem tabulky s automatickou inkrementací id. Další atribut *idUzivatel* je cizím klíčem do tabulky u*zivatele*. Atribut *prihlasen* nese informaci o datu a času přihlášení uživatele. Zadání je povinné a je vytvářeno automaticky při vzniku nového záznamu v tabulce. Datum a čas odhlášení uživatele je zaznamenáván do atributu *odhlasen*. Zadání samozřejmě není povinné, protože uživatel je vždy nějakou dobu jen přihlášen a teprve později odhlášen.

7.1.7 Tabulka cinnostdruh

Tabulka *cinnostdruh* (Tab. 7) slouží k uložení dat pro číselník druhů prováděných činností. Díky tomuto číselníku je možné nastavit pojmenování druhů činností podle svého uvážení. Prvním atributem tabulky je *idDruh*. Je primárním klíčem tabulky s automatickým inkre-

mentováním id. Druhým atributem je *kod*. Zadání je povinné a je určen pro uložení označení činnosti číselnou hodnotou podle uvážení a potřeby. Jedinou podmínkou je, aby hodnota kódu „1“ byla spjata s produktivní činností stroje. Ostatní hodnoty je možné nastavit libovolně. Je však pomocí unikátního indexu zamezeno zadání duplicitního kódu činnosti.

Tab. 7. Struktura tabulky cinnostdruh.

Atribut	**Doména**	**Null hodnota**
idDruh	int(11)	Ne
kod	int(11)	Ne
popis	text	Ne

Atribut *popis* slouží k textovému označení činnosti. Je vysvětlujícím popisem daného kódu činnosti. Zadání je také povinné, protože pouze číselná hodnota v podobě označení kódem by nebyla pro uživatele systému srozumitelná.

7.1.8 Tabulka cinnosti

Jednou z nejdůležitějších tabulek v systému je tabulka *cinnosti* (Tab. 8). Slouží pro zadání stěžejních informací o prováděné činnosti na stroji. Její struktura není složitá. Tvoří v podstatě „hlavičku“ pro zadání prováděných činností se stroji. Je tvořena zejména cizími klíči.

Tab. 8. Struktura tabulky cinnosti.

Atribut	**Doména**	**Null hodnota**
idCinnost	int(11)	Ne
zakazka	varchar(100)	Ne
idStroj	int(11)	Ne
idUzivatele	int(11)	Ne
idStatusCinnosti	int(11)	Ne

Jako u všech tabulek je prvním atributem *idCinnosti*, který je ve funkci primárního klíče tabulky s nataveným automatickým inkrementováním id. Atribut *zakazka* slouží pro ulože-

ní označení výrobní zakázky. Označení čísla zakázky je přebráno z informačního systému společnosti. Pomocí něj je vytvářena vazba mezi daty z informačního systému a daty strojů, aby bylo možné získané informace správně přiřadit. Zadání označení zakázky je povinné. Ostatními atributy tabulky jsou už jen cizí klíče do závislých tabulek. První z nich je *idStroj*, který je cizím klíčem do tabulky *stroje*. Druhým závislým atributem je *idUzivatele*, který je cizím klíčem do tabulky *uzivatele*. Třetím a posledním atributem je *idStatusCinnosti*. Je cizím klíčem do tabulky *cinnoststatus*. Všechny tyto zmíněné cizí klíče jsou pro zadání povinné a je tedy nutné je vyplnit.

7.1.9 Tabulka cinnostipreruseni

Další velmi důležitou tabulkou v systému je tabulka *cinnostipreruseni* (Tab. 9). Je ve vztahu k tabulce *cinnosti*, ke kterým je v roli „položek“. Slouží pro uložení dat o jednotlivých činnostech (přerušeních) prováděných na stroji ve vztahu k nadřízené tabulce *cinnosti*.

Tab. 9. Struktura tabulky cinnostipreruseni.

Atribut	**Doména**	**Null hodnota**
idCinnostPreruseni	int(11)	Ne
idCinnosti	int(11)	Ne
zahajeni	timestamp	Ne
idDruhCinnosti	int(11)	Ne
ukonceni	timestamp	Ano
důvod	varchar(250)	Ano

Atribut *idCinnostPreruseni* je primárním klíčem tabulky s automatickou inkrementací. Následuje atribut *idCinnosti*, který je cizím klíčem do tabulky *cinnosti*. Vytváří tak vazbu mezi „hlavičkou“ činnosti a jejími „položkami“. Hodnota musí být vyplněna, aby bylo možné vytvořit vazbu mezi tabulkami. Atribut *zahajeni* je určen pro zadání data a času. Jeho hodnota je vyplněna automaticky při vložení nového záznamu činnosti do tabulky. Přestože vzniká automaticky, je u něj nastavena povinnost pro zadání. Dalším atributem této tabulky je *idDruhCinnosti*. Je v roli cizího klíče do tabulky *cinnostdruh* a jeho zadání je povinné. Pátým atributem je *ukonceni*, který slouží pro uložení data a času ukončení činnosti. Posledním atributem této tabulky je *duvod*. Je vytvořen pro možnost doplnit dů-

vod k zadané prováděné činnosti. Jeho zadání není povinné, protože například při zadání produktivní činnosti by byl jakýkoli komentář zbytečný.

7.1.10 Tabulka cinnoststatus

Tabulka *cinnoststatus* je systémovou tabulkou (Tab. 10). V rámci systému pro ni není vytvořeno žádné prostředí pro její správu. Tento fakt je záměrný, protože se jedná o tabulku, kde jsou její hodnoty používány pro rozlišení, v jakém stavu se dané činnosti nacházejí. Nebylo by vhodné, kdyby například administrátor systému průběžně modifikoval její hodnoty. Zadané hodnoty ovšem mají jasně nastaven svůj význam, který bude vysvětlen při popisu jednotlivých atributů tabulky.

Tab. 10. Struktura tabulky cinnoststatus.

Atribut	**Doména**	**Null hodnota**
idStatus	int(11)	Ne
status	int(11)	Ne
popis	text	Ano

Atribut *idStatus* je primárním klíčem tabulky a i u něj je nastavena automatická inkrementace. Druhým atributem je *status*. Jeho zadání je povinné a musí být jednoznačné, což zajišťuje vytvořený unikátní klíč. Jedná se o numerickou hodnotu, která udává, v jaké situaci se tato hodnota nachází. Pro zadání hodnot platí pravidlo, při kterém nejnižší hodnota zastupuje status zahájení činnosti a naopak nejvyšší potom status ukončení činnosti. Hodnot statusů může být libovolný počet v závislosti na tom, jak je požadováno podrobně sledovat, v jaké fázi (statusu) se daná probíhající činnost nachází. V rámci tohoto systému jsou použity pouze dva statusy podle zmíněného pravidla. Prvním je hodnota „1“ znázorňující „zahájení“ činnosti na stroji, druhou hodnotou je „2“, která znázorňuje „ukončení“ činnosti na stroji. Poslední atribut této tabulky je *popis*. Jeho zadání není povinné, ale je velmi vhodné z důvodu snadnější orientace nebo pro zobrazení v přehledech a vyhodnoceních. Je textovým vyjádřením významu hodnoty, zadané v atributu *status*.

7.1.11 Tabulka stroje

Tabulka *stroje* je určena pro údaje, zastupující zadaná relevantní data o strojích a jejich vlastnostech či parametrech (Tab. 11). Pro systém je tedy v pozici „číselníku strojů“.

Tab. 11. Struktura tabulky stroje.

Atribut	**Doména**	**Null hodnota**
idStroje	int(11)	Ne
cislo	int(11)	Ne
nazev	varchar(200)	Ne
popis	text	Ano
hranicePrikonu	float	Ne
dobaProstoje	time	Ne
externiCisloStroje	int(11)	Ne

IdStroje je jako vždy primárním klíčem tabulky a je u něj nastavena automatická inkrementace hodnoty id. Atribut *cislo* slouží pro zadání číselného označení stroje. Jeho zadání je povinné a nesmí být duplicitní, což je zajištěno pomocí unikátního klíče. Další atribut *nazev* je určen pro zadání názvu stroje. Zadání názvu je povinné také proto, že je názvu používáno v různých přehledech. Ale *popis* již povinný není. Slouží v případě potřeby pro doplnění dalšího popisu zadaného stroje. Následující atribut *hranicePrikonu* slouží k zadání číselné desetinné hodnoty, která udává práh pro produktivní příkon stroje. Jeho zadání je povinné. Pokud je tato hodnota (hranice) překročena, je považován stav stroje za produktivní. *DobaProstoje* je atributem pro zadání doby (času), po kterou se může stroj nacházet v neproduktivním čase a přesto to je považováno za produkci. Nastávají totiž situace, kdy je ve stroji vyměňován obráběný díl. Stroj sice nemá příkon jako při produkci, ale nepovažuje se za neproduktivní čas, protože jej není možné během této doby lépe využít. Z toho také plyne, že doba výměny obráběného dílu je individuální pro stroj. Atribut *externiCisloStroje* umožňuje uložit číselné označení stroje. Toto označení je určeno pro provázání se systémem pro sběr dat ze strojů. Slouží jako identifikace tabulky, do které jsou ukládána data ze strojů v databázi strojů viz kapitola 4.2. Zadání je povinné a nesmí být duplicitní. I předchozí dva atributy (*hranicePrikonu* a *dobaProstoje*) slouží pro zpracování dat ze strojů. Jejich hodnoty jsou používány pro rozlišení, jestli se stroj nachází v produktivním nebo neproduktivním stavu.

7.1.12 Tabulka upozorneni

Další významnou tabulkou tohoto systému je tabulka *upozorneni* (Tab. 12). Její význam, jak již název napovídá, je pro ukládání dat o upozorněních, která v rámci systému vznikají. Je další tabulkou, ke které v systému neexistuje uživatelské rozhraní. Data v této tabulce jsou modifikována na pozadí během programu.

Tab. 12. Struktura tabulky upozorneni.

Atribut	**Doména**	**Null hodnota**
idUpozorneni	int(11)	Ne
vytvoreno	timestamp	Ne
zmeneno	timestamp	Ne
idStroje	int(11)	Ne
pocetUpozorneni	int(11)	Ano
emailOdeslan	tinyint(1)	Ano
textZpravy	text	Ano
vyrizeno	tinyint(1)	Ne

Primárním klíčem této tabulky je atribut *idUpozorneni*, u kterého je nastaveno automatické inkrementování hodnoty id. Atribut *vytvoreno* slouží pro uložení data a času vzniku záznamu o upozornění. Jeho zadání je povinné. Dalším atributem je *zmeneno*, který také slouží pro uložení data a času provedené změny záznamu. Tato hodnota je tedy přepisována při každé změně záznamu. Zadání není povinné, protože v době vytvoření ještě není dostupná informace o jeho změně. Atribut *idStroje* je cizím klíčem do tabulky *stroje*. *PocetUpozorneni* slouží pro uložení hodnoty o počtu upozornění. Pomocí něj je následně možné zjistit, kolikrát byl operátor stroje upozorněn, než bylo upozornění vyřízeno. Jeho zadání není povinné, ale i přesto je prováděno automaticky v situaci, kdy stav upozornění pokračuje. Následující atribut *emailOdeslan* slouží k zaznamenání hodnoty o tom, zda-li byl odeslán email s upozorněním nadřízenému, či nikoliv. Podobně je tomu u atributu *textZpravy*, který slouží k uložení textu zprávy, která byla zaslána na adresy určené pro zasílání upozornění. Jeho zadání není povinné. Jeho hodnota je zadána jen v případě, že byl odeslán email s upozorněním. Posledním atributem této tabulky je *vyrizeno*. Slouží k zadá-

ní pravdivostní hodnoty o stavu upozornění. Pokud je jeho stav nastaven na „1“, znamená to, že dané upozornění je již ve stavu „vyřízeno“. Pokud hodnota není rovna „1“, je stav upozornění stále aktivní a může se v průběhu času měnit.

7.1.13 Uložené procedury

V databázi jsou mimo tabulek uloženy ještě procedury. Jsou určeny k provádění složitějších výpočtů s daty. V systému poskytují výpočty efektivity strojů. Výsledkem zpracování procedur je pole dat, obsahující všechny vypočtené záznamy. Výsledná data jsou zpracována a formou grafu a tabulky prezentována uživateli systému.

7.2 Třídy vrstvy model

Každá třída ve vrstvě *model* obsahuje konstruktor. Pomocí konstruktoru je předávána závislost na třídu *Nette\Database\Context*. Díky této závislosti je podle konfiguračního souboru *config.local.neon* a třídě *Nette\Database\Connection* vytvořeno připojení k databázi. Níže je uvedena ukázka kódu používaného konstruktoru pro vytvoření přístupu k databázi:

```
/** @var Nette\Database\Context */
private $database;
public function __construct(Nette\Database\Context $database)
{
      $this->database = $database;
}
```

Nyní je již možné pracovat s databází a to hned dvěma způsoby. V prvním případě je možné zapisovat SQL dotazy přímo, pomocí metody *query*. Příklad použití je uveden níže.

```
public function getRulesToList()
{
    try{
            $rules    =    $this->database->query('SELECT    p.idPrava,
p.povoleno, r.nazev as role, z.nazev as zdroje, o.nazev as operace
 FROM ' . self::TABLE_PRAVA . ' p
 INNER JOIN ' . self::TABLE_ROLE . ' r ON (p.idRole = r.idRole)
 LEFT JOIN ' . self::TABLE_ZDROJE . ' z ON (p.idZdroj = z.idZdroj)
 LEFT JOIN ' . self::TABLE_OPERACE . ' o ON (p.idOperace = o.idOperace)
ORDER BY r.nazev ASC, z.nazev ASC, o.nazev');

        return $rules;
    }
        catch (Nette\Database\DriverException $e){
            throw new DriverSecurityDataException;
        }
    }
```

Tato možnost je vhodná pro vytváření složitějších dotazů, při dotazech přes více tabulek apod. V druhém případě je možné využít vrstvy *Nette\Database\Table*. Data získaná pomocí této vrstvy z databáze jsou vrácena jako instance *ActiweRow*. Použití *Nette\Database\Table* je vhodnější zejména k provádění jednodušších dotazů. Je také možné vkládat data nová nebo data modifikovat. Příklad přidání nového záznamu do databáze je uveden v níže uvedeném kódu.

```
public function addKind($kod, $popis)
{
      try {
            $this->database->table(self::TABLE_NAME)->insert([
                        self::COLUMN_KOD => $kod,
                        self::COLUMN_POPIS => $popis
                  ]);
      } catch (Nette\Database\DriverException $e) {
                  throw new DriverKindException;
      }
}
```

V některých třídách vrstvy *model* bylo také potřeba zajistit spolupráci mezi třídami v rámci vrstvy. To je zajištěno předáním závislosti na požadovanou třídu pomocí konstruktoru podobně, jako bylo výše ukázáno předání závislosti na context databáze. Jednotlivé třídy modelů a jejich vzájemné vztahy jsou blíže popsány v kapitole 5.4.

7.2.1 Třída CheckMachineDataManager

V třídě je implementována funkcionalita pro kontrolu stavu strojů a dat od operátorů. Jako jediná třída v celém systém systému používá připojení ke dvěma databázím současně. To je vyřešeno přidáním dalšího připojení k databázi do souboru *config.local.neon*, který je součástí základní struktury frameworku Nette. Obsah konfiguračního souboru je na obrázku (Obr. 16). Dalším nastavením pro funkčnost dvou databází současně je nastavení předání obou kontextů databází v části služeb souboru *config.neon*. To je provedeno pomocí příkazu na následujícím řádku:

```
App\Model\CheckMachineDataManager(@database.default.context,
@database.dbstroje.context)
```

Díky těmto výše uvedeným krokům v konfiguraci je možné pomocí konstruktoru této třídy vytvořit připojení k oběma databázím současně. S takto vytvořeným spojením je pracováno v metodě *dataMachine()*, která provádí celou kontrolu. Nejdříve zjistí, v jakém stavu se nachází stroj pomocí metody *isMachineWork()* a následně kontroluje stav činnosti operátora stroje metodou isUserWork(). Vrácené výsledky jsou porovnány a je rozhodnuto o vzni-

ku upozornění. V případě, že jsou shodná, je provedena kontrola, jestli pro stejná kritéria neexistuje upozornění. Pokud ano, je upozornění označeno jako vyřízené.

```
database:
    default:
        dsn:    'mysql:host=127.0.0.1;dbname=productioncheck'
        user:   root
        password:
        options:
            lazy:   yes


    dbstroje:
        dsn:    'mysql:host=127.0.0.1;dbname=tdmaxt'
        user:   root
        password:
        options:
            lazy: yes
```

Obr. 16. Ukázka konfiguračního souboru config.local.neon pro dvě databáze.

7.2.2 Třída KindActivityManager

Tato třída obsahuje kompletní funkcionalitu pro správu číselníku druhů činností. Nejdříve je pomocí konstruktoru předán kontext databáze. Následují metody pro přidání nového druhu a metoda pro editaci již existujících. Součástí třídy jsou ještě metody pro kompletní výpis všech druhů, výpis druhu podle zadaného identifikátoru záznamu a vrácení dat druhů připravená jako zdroj pro výběrový seznam druhů činností.

7.2.3 Třída NoticeManager

Třída *NoticeManager* je určena pro práci s upozorněními. Ve třídě je metoda *getNotice()*, která nejdříve zjistí stav rozpracovaných činností pro přihlášeného operátora. K těmto rozpracovaným činnostem je následně pomocí další metody *existsNoticeForMachine()* podle zadaných strojů zjištěno, jestli pro něj existuje nějaké upozornění. Po kontrole všech rozpracovaných činností jsou vráceny výsledky a metoda je ukončena.

7.2.4 Třída SecurityDataManager

Správa dat uživatelských oprávnění je implementována v této třídě. Umožňuje přidávat a editovat oprávnění uložená v databázi. Dalšími metodami jsou zejména obslužné rutiny, které vrací kompletní výpis všech zadaných práv v databázi, výpis všech rolí a oprávnění

uživatelů. Také jsou v rámci třídy implementovány metody pro přípravu dat pro výběrové seznamy a to pro výběrový seznam rolí, zdrojů a oprávnění.

7.2.5 Třída SecurityManager

Tato velmi jednoduchá třída je potomkem třídy *Nette\Security\Permission.* Obsahuje pouze jednu metodu a tou je konstruktor. Kromě předání kontextu databáze konstruktor obsahuje volání pro vytvoření nové instance. Pomocí vytvořené instance třídy *SecurityDataManager* jsou volány jednotlivé metody pro načtení dat uživatelských rolí, zdrojů a oprávnění. V posledním kroku jsou vytvořena uživatelská oprávnění, vzniklá kombinací pravdivostní hodnoty povoleno/zakázáno, uživatelské role, zdroje a oprávnění.

7.2.6 Třída MachineManager

Třída *MachineManager* je určena pro správu dat o strojích. Kromě standartního konstruktoru pro předání kontextu databáze obsahuje jen několik málo metod. Samozřejmostí je metoda pro funkcionalitu přidání nového a editaci existujícího stroje v databázi. Další metodou je metoda pro výpis všech strojů a výpis dat stroje podle zadaní id stroje. Poslední metodou v této třídě je příprava dat pro vytvoření výběrového seznamu strojů.

7.2.7 Třída UserManager

Třída, ve které je implementováno rozhraní *Nette\Security\IAuthenticator*. Obsahuje správu dat uživatelů systému. Umožňuje tedy přidat nového a editovat stávajícího uživatele. Její nejdůležitější metodou je však metoda *authenticate()*, která provádí přihlášení uživatele. Jako parametr přebírá přihlašovací údaje (přihlašovací jméno a heslo) a ověří jejich platnost. Pokud jsou přihlašovací údaje platné, vrátí metoda instanci třídy *Nette\Security\Identity*. V opačném případě je vyhozena výjimka o neplatných údajích. Dalšími metodami je zajištěno zaznamenání přihlášení a odhlášení do databáze. Dalšími metodami jsou výpisy dat uživatelů z databáze podle různých kritérií. V rámci třídy je také implementována metoda, která vrací seznam emailových adres pro zasílání upozornění.

7.2.8 Třída ProductionManager

Správa dat všech činností (výroby) je spravována touto třídou. Umožňuje přidat novou činnost (výrobu) nebo přidání přerušení (neproduktivní činnost). Má implementovanou metodu pro vrácení dat rozpracované výroby pro operátora nebo bez omezení, tedy všech rozpracovaných činností. Metoda pro ukončení činnosti samozřejmě nesmí chybět.

7.3 Třídy vrstvy presenters

Základní třídou vrstvy *presenters* je abstraktní třída *BasePresenter*. Tato abstraktní třída je potomkem *Nette\Application\UI\Presenter.* Ostatní presentery jsou potomky této abstraktní třídy. Třída *BasePresenter* má dva úkoly. Prvním z nich je pomocí metody *beforeRender()* předat všechny položky menu systému do výchozí šablony se jménem *@default.* Jednotlivé položky menu jsou v metodě zapsány v podobě pole[31]. Druhým velmi důležitým úkolem je kontrola přihlášení uživatelů a úrovně uživatelského oprávnění, která se provádí pomocí metody *checkRequirements($element)*. Před kontrolou oprávnění je testováno, jestli je kontrola oprávnění volána z presenteru, který oprávnění vyžaduje. Tato kontrola je navázána na existenci anotace *@secured*, kterou třída *Basepresenter* detekuje z předaného parametru *$element*. Nicméně v systému existuje pouze jedna třída, která nemá anotaci *@secured* a je jím *CheckMachinePresenter*. K této výjimce se vrátíme v dalším odstavci.

7.3.1 Třída CheckMachinePresenter

V minulém odstavci zmíněná třída *CheckMachinePresenter* není přístupná z webové části systému, ale je určena pro spouštění metod v tzv. konzolovém módu. Je to speciální třída, volaná pomocí naplánované úlohy, která je v pravidelných časových intervalech spouštěna na webovém serveru. V této třídě je kromě konstruktoru předávající vazbu na třídu *CheckMachineDataManager*, metoda *actionCheckMachine()*, která hned v úvodu detekuje, jestli volání metody opravdu pochází z příkazové řádky. V případě kladné odpovědi je spuštěno volání metody, která spouští kontrolu stavu strojů pomocí metody ze třídy *CheckMachineDataManager* a po jeho dokončení je činnost ukončena voláním *$this->terminate()*.

7.3.2 Třída HomePagePresenter

Zobrazení aktuálního stavu strojů zpracovává třída *HomepagePresenter*. Její jedinou úlohou je předat třídě *ProductionManager* dotaz, který vzniká v šabloně *default* jako AJAXové volání. Po zpracování dat a vrácení výsledků třídou *ProductionManager*, je překreslena část šablony *default* vymezená blokem s názvem makra *{snippet}*. V těle makra *{snippet}* jsou předaná data zpracována do požadované podoby.

[31] Pole – datová struktura.

7.3.3 Třída KindActivityPresenter a MachinePresenter

Třídy *KindActivityPresenter* a *MachinePresenter* zprostředkovávají akce pro zpracování požadavků na práci s daty druhů činností a strojů. Samozřejmostí je předání závislostí na potřebné spolupracující třídy pomocí konstruktoru. Obě třídy obsahují metody pro volání vytváření formulářů pomocí komponent. Vytvořené formuláře jsou dále předány do šablony, která se o jejich vykreslení postará. V těchto dvou třídách jsou formuláře používány pro přidávání nových záznamů a editaci stávajících.

7.3.4 Třída SignPresenter a SecurityDataPresenter

SignPresenter a *SecurityDataPresenter* slouží pro řízení činnosti v oblasti zabezpečení, konkrétně pro autentizaci a autorizaci uživatelů. *SignPresenter* obsahuje obsluhu vytváření komponent formulářů pro editaci, přidávání nových uživatelů a pro přihlašování uživatelů. Dále řídí akci pro odhlášení uživatelů a předává data z třídy *UserManager* do šablony pro výpis všech uživatelů. Druhý *SecurityDataPresenter* provádí řídící činnost pro správu uživatelského oprávnění uživatelů. Vytváří komponenty formulářů pro přidávání a editaci záznamů o oprávněních. Další činností je požádání třídy *SecurityDataManager* o vrácení výpisu všech uživatelských oprávnění.

7.3.5 Třída MachinePresenter

Další třída *MachinePresenter* spolupracuje s třídou *MachineManager*. Jak z názvu vypovídá, jedná se o třídu, která řídí činnosti v oblasti správy strojů. Tyto činnosti zahrnují vytváření dvou formulářů, pro přidávání a editaci. Provádí také zpracování požadavku na výpis strojů skrze šablonu.

7.3.6 Třída ProductionPresenter

ProductionPresenter obsahuje mimo vytváření komponent formulářů pro přidávání a změnu činností také další akce jako je ukončení rozpracovaných činností nebo zprostředkování různých druhů výpisů dat o výrobě. Třída má závislost na třídě *ProductionManager*, která zpracovává všechna data o výrobě, resp. prováděných činnostech. Má však vytvořenou závislost ještě na třídě *NoticeManager*, která poskytuje informace o existujících upozorněních. Existující upozornění jsou ve třídě volány pomocí AJAXu. Metoda *handleUpozorneni()* zjistí, jestli existují nějaká platná upozornění a pokud ano, připraví data, která jsou doručena zpět šabloně a v ní vykreslena. Tato metoda je volána opakovaně každou minutu.

8 APLIKOVANÁ BEZPEČNOSTNÍ OPATŘENÍ

Obecné principy jednotlivých známých útoků a ochrana před nimi byly podrobně popsány v kapitole 3. Obsahem této kapitoly tedy nebude popis jednotlivých postupů útoků, ale jen shrnutí implementovaných opatření pro zvýšení zabezpečení webové aplikace. Velkou měrou pomohl ke zvýšení zabezpečení použitý framework Nette. Pro zvýšení zabezpečení aplikace byly využity možnosti, které framework nabízí. Navíc riziko zneužití je do značné míry omezeno faktem, že aplikace bude provozována v interní síti výrobní společnosti. Bude tedy oddělena od sítě Internet firewallem a hraničním směrovačem.

8.1 Cross-Site Scripting (XSS)

Princip tohoto útoku je detailněji popsán v kapitole 3.1, ale základní myšlenkou tohoto útoku je zneužití nesprávně ošetřených vstupů webové aplikace. Framework Nette má na tento útok ochranu. Ochrana spočívá v implementované technologii Context-Aware Escaping. [44] Tato technologie je součástí šablonovacího systému Latte (viz kapitola 2.2.7), která provádí automatické escapování. Toto automatické ošetření výstupů se přizpůsobuje místu, kde se makro nachází a podle toho framework vybere vhodné escapování. Automatické ošetření má výhodu, protože se nestane, že na něj programátor zapomene a vytvoří bezpečnostní díru v aplikaci. [44] Na obrázku (Obr. 17) je uveden příklad fungování šablony vytvořené pomocí systému Latte.

```
<p onclick="alert({$movie})">{$movie}</p>

<script>var movie = {$movie};</script>
```

Obr. 17. Ukázka vytvořené šablony v Latte. [44]

Automatické escapování upraví výstup, jak již bylo zmíněno výše, podle umístění v dokumentu. Na uvedeném příkladu je obsahem proměnné textový řetězec *'Amarcord & 8 1/2'*. [44]

```
<p onclick="alert(&amp;quot;Amarcord &amp;amp; 8 1\/2&amp;quot;)">Amarcord &amp;amp; 8 1/2</p>

<script>var movie = "Amarcord & 8 1\/2";</script>
```

Obr. 18. Ukázka výstupu po automatickém escapování šablonovacím systémem Latte. [44]

Jak je vidět na obrázku (Obr. 18) je jiné escapování použito na proměnnou *$movie* v atributu *onclick*, jiné u HTML a v neposlední řadě je také odlišné při použití v Javascrip-

tu. Šablonovací systém Latte se takto stará o automatickou ochranu před útokem XSS za nás. [31] [44]

8.2 Cross-Site Request Forgery (CSRF)

Další z implementovaných zabezpečení tohoto systému je ve frameworku Nette dostupná funkce, která provádí ochranu před touto zranitelností. Její použití je pomocí následující funkce *$form->addProtection();*. [31] Tato funkce přijímá dva parametry. Prvním z nich je textová zpráva, která se zobrazí při neúspěšném ověření identifikátoru nebo po vypršení doby platnosti session. Platnost session je nastavována při přihlášení nebo je ji možné zkrátit zadáním druhého parametru. Druhý parametr tedy přijímá číslo udávající počet sekund platnosti vytvořeného identifikátoru. Podstatou této funkce je vytvoření jednoznačného klíče, který je následně ověřován, jestli nebyl změněn. Doba platnosti klíče je shodná se session nebo s hodnotou, která je zadána do funkce jako druhý parametr. V případě negativního ověření je vypsána zpráva, která je parametrem této funkce (viz příklad výše). [45] Navzdory doporučení obsažené v dokumentaci na vložení této funkce na důležité (administrační) části aplikace [45], je v rámci implementace tohoto vytvořeného systému tato funkce doplněna do všech formulářů s platností na 120 sekund. [31] [45]

8.3 Callbacky formulářů

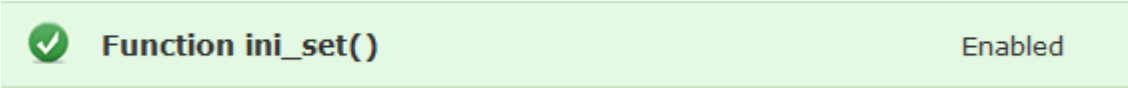
Zabezpečení formulářů je ještě zvýšeno zabezpečením tzv. callbacků, protože by se mohlo stát, že útočník pozmění adresu a mohl by se dostat k odeslání formuláře i bez přihlášení. [46] Proto je třeba implementovat ověření přihlášeného uživatele ještě před zpracováním dat formuláře a to velmi jednoduchým způsobem. Pomocí funkce *isLoggedIn()* ověříme, jestli je uživatel přihlášen. Takto je zajištěno, že pro provedení akce je uživatel přihlášen. [46]

8.4 URL attack, control codes, invalid UTF-8

Existuje několik různých způsobů, jak může útočník podstrčit webové aplikaci neplatný nebo škodlivý vstup. [31] Následky nesprávného vstupu mohou vést k poškození dat nebo ke zneužití citlivých dat. Ošetřování vstupů na úrovni bajtů je možný způsob ochrany před těmito útoky. [31] Součástí frameworku Nette je automatické ošetřování všech vstupů na úrovni jednotlivých bajtů. V rámci implementace není třeba dodatečně nic nastavovat. [31]

8.5 Session hijacking, session stealing, session fixation

Všechny tyto útoky, jak jejich názvy napovídají, se týkají správy session ID. Blíže jsou tyto zranitelnosti popsány v kapitole 3.3 a 3.4. Základním principem ochrany před těmito útoky je správná konfigurace serveru a PHP. [31] V rámci frameworku Nette je toto opět zajištěno automaticky, podmínkou však je povolení funkce *ini_set()*. [31]



Obr. 19. Znázornění povolené funkce ini_set() v Nette Checker.

Na obrázku (Obr. 19) je zobrazen pozitivní výsledek nastavení funkce *ini_set()*. Tento výpis byl pořízen pomocí nástroje *Nette Framewotk Requirements Checker*[32], který je součástí frameworku Nette. [31]

8.6 Autentizace a autorizace

Do systému je také implementován systém pro autentizaci a autorizaci uživatelů, který je obecně popsán v kapitole 3.5 a 3.6. Systém pro autentizaci a autorizaci uživatelů je vytvořen jako dynamický, aby jej bylo možné snadno spravovat administrátorem systému. Všechny informace potřebné pro ověření uživatelů jsou uvedeny v databázi systému. Samozřejmě je k vytvoření autentizace a autorizace využito nástrojů frameworku Nette. Pro autentizaci je to vlastní implementace rozhraní *Nette\Security\IAuthenticatoru* [47] ve třídě *UserManager*. V databázi v tabulce *uzivatele* jsou uloženy všechny informace o uživatelích systému. Při přihlášení je tedy uživatel ověřován podle uživatelského jména a hesla proti datům zadaným v databázi. Hesla jsou v databázi uložena bezpečně. K tomu je využita funkce *Nette\Security\Passwords::hash($heslo)*. [48] Tato integrovaná funkce ve Frameworku Nette se používá k hashování hesla pomocí algoritmu bcrypt. Funkce *hash* přebírá dva parametry. První z nich je již zmíněný text k zahešování, druhý nepovinný parametr udává počet iterací algoritmu. Pokud není parametr zadán, je použita výchozí hodnota s počtem iterací 10. [48] V případě platného jména a hesla uživatele je vytvořena nová instance třídy *Nette\Security\Identity*, která je také součástí frameworku Nette. [47] V instanci této třídy jsou uchována všechna data o přihlášeném uživateli. Výjimku tvoří

[32] Nette Framewotk Requirements Checker – skript, který kontroluje splnění požadavků Frameworku na nastavení serveru a PHP.

pouze uživatelské heslo, které se do třídy nepřidává. Takto vytvořená identita uživatele je v systému dostupná v proměnné *Nette\Security\User*. Pomocí funkce *isLoggedIn()*, je možné ověřit, jestli je uživatel do systému přihlášen. [47]

Pro autorizaci je využito vlastní třídy *SecurityManager*, která je potomkem třídy *Nette\Security\Permission*. Třída *Nette\Security\Permission* uchovává údaje o rolích uživatelů, zdrojích a operacích. Dále obsahuje také informace o právech uživatelů. [47] Všechna data o rolích, zdrojích a operacích jsou, podobně jako data o uživatelích, uložena v databázi pro snadnou správu administrátorem systému. Nastavení uživatelských oprávnění v systému je díky této struktuře dynamické a možné jej měnit bez potřeby jakékoli změny zdrojového kódu. Pomocí systému může administrátor ovlivňovat nastavení oprávnění uživatelů systému. Práva uživatelů jsou vytvářena na základě těchto dat z databáze a za běhu systému jsou uložena v instanci třídy *Nette\Security\Permission*. [47] V Nette frameworku je dostupná funkce pro ověření uživatelského oprávnění podobně jako v předchozím případě přes funkci třídy *Nette\Security\User*. [47] Funkce pro ověření oprávnění se jmenuje *isAllowed()*. Jejími parametry jsou informace o zdroji a požadované akci, ke které uživatel přistupuje a samozřejmě doplněné o role uživatele, která je získána z vytvořené identity přihlášeného uživatele. [47] Je tedy zřejmé, že autorizaci je možné provést až po úspěšné autentizaci uživatele, díky které je vytvořena identita přihlášeného uživatele.

Pokud administrátor provede nějakou změnu v databázi s oprávněním uživatelů, okamžitě se tato data promítnou do systému, resp. ihned po obnovení zobrazené stránky nebo při přechodu na jinou část aplikace.

9 PREZENTACE VÝSLEDKŮ ZE SYSTÉMU

V této kapitole jsou představeny výsledky, kterých je dosaženo při zpracování dat strojů a operátorů. Výsledky můžeme rozdělit do dvou základních skupin. První skupinou jsou data, která jsou získávána ze systému průběžně. Jejich načítání je opakované pro zajištění aktuálních informací. Druhou skupinou potom tvoří spíše statistická vyhodnocení, která analyzují nasbíraná data zpětně.

Aktuální situaci o stavu strojů zjistí uživatel s dostatečným oprávněním (mistr, vedoucí/manažer a samozřejmě správce) pod položkou v menu s názvem „Stav“. Po výběru volby z menu je zobrazena tabulka viz obrázek (Obr. 20). V tabulce jsou zobrazeny detailní

Číslo stroje	Stroj	Osobní číslo	Operátor	Zakázka	Zahájeno	Činnost
1	ZPS VMC 560	11	Příjmení Test3 Jménotest	A268965B	2017-04-15 08:02:50	Kontrola prvního kusu
2	Finetech SMV-1370-H3L	11	Příjmení Test3 Jménotest	A256897B	2017-04-15 15:19:21	Porucha
3	Doosan Puma 300	001	Operátor Testovací	A236987B	2017-04-09 16:53:55	Produktivní činnost
4	Doosan Puma TT1800SY	001	Operátor Testovací	A123456B	2017-04-13 21:39:54	Produktivní činnost

Obr. 20. Ukázka vzhledu tabulky znázorňující aktuální stav strojů.

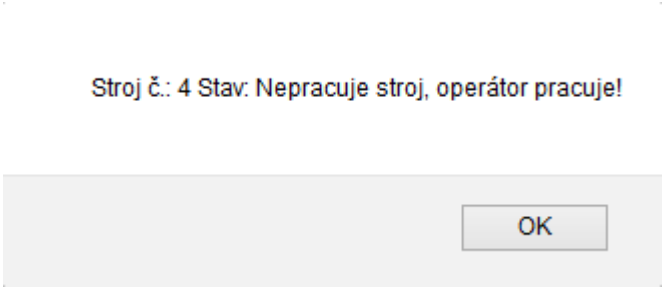
informace o stavu jednotlivých strojů s uvedením aktuálního operátora stroje, zakázky, data a času zahájení činnosti. Pro rychlejší orientaci je graficky oddělen produktivní a neproduktivní čas stroje. Zelená barva znázorňuje produktivní čas stroje, červená stav neproduktivní. Data v tabulce jsou každých 30 sekund aktualizována.

Aktuální stav strojů je k dispozici i pro operátora, ale v trochu jiné podobě. Přihlášený operátor nevidí přímo zobrazení stavu stroje a velmi podobnou tabulku, která zobrazuje aktuální činnosti operátora. Je ukryta pod položkou menu s názvem „Rozpracovaná výroba“. Obsahuje velmi podobné údaje jako v předchozím případě. Postrádá informaci o operátorovi, protože zobrazená data se váží jen k přihlášenému operátorovi. Je však ještě doplněna o možnost změnit druh vybrané činnosti, případě činnost ukončit (Obr. 21). Jedná se o výchozí stránku, která je automaticky po přihlášení operátora zobrazena. Na této stránce je operátor upozorňován na situace, kdy stav podle dat naměřených ze stroje není shodný s daty, která do systému zadal.

Číslo stroje	Stroj	Zakázka	Zahájena od	Akce ukončit	Změna činnosti	Kod činnosti	Činnost
3	Doosan Puma 300	A236987B	09.04.2017 16:53:55	Ukončit	Změnit	1	Produktivní činnost
4	Doosan Puma TT1800SY	A123456B	13.04.2017 21:39:54	Ukončit	Změnit	1	Produktivní činnost

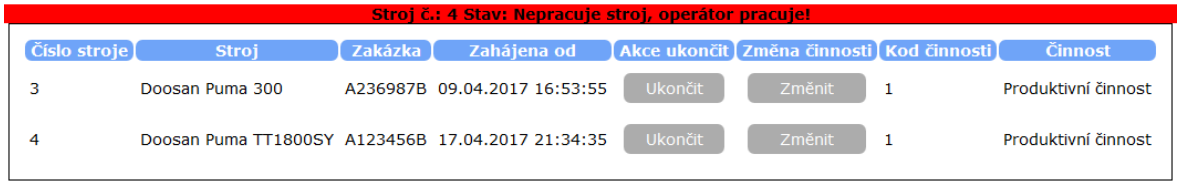
Obr. 21. Ukázka vzhledu tabulky rozpracovaných činností pro operátora.

V takovém případě je operátor informován zobrazením dialogového okna (Obr. 22), které jej informuje o nastalé situaci.



Obr. 22. Dialogové okno upozornění.

Pro zdůraznění upozornění operátorovi o nekonzistenci činností je po zavření dialogového okna (Obr. 22) ještě doplněno varovným pruhem nad výpisem rozpracované výroby (Obr. 23). Tento pruh po potvrzení dialogového okna s upozorněním ještě několikrát zabliká.



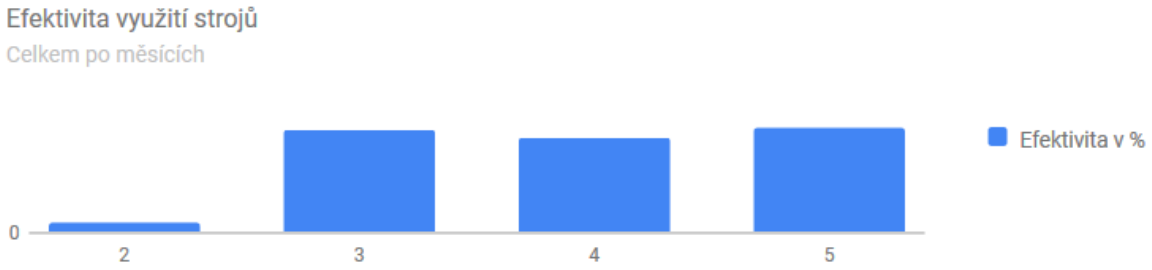
Obr. 23. Ukázka zobrazení pruhu s upozorněním pro operátora.

Je velmi důležité, aby byl operátor na vzniklou situaci důrazně upozorněn. Je žádoucí, aby nastalou nekonzistenci mezi daty ze strojů a daty zadanými operátorem, řešil co nejdříve. Pokud i po několika opakováních upozornění operátor nastalou nekonzistenci mezi daty strojů a operátorů neřeší, je o této situaci informován nadřízený pracovník pomocí zaslaného emailu. Nadřízený je tak o vzniklé situaci informován a může podniknout patřičné kroky k napravení vzniklé situace.

Druhou skupinou vyhodnocení jsou nikoliv data získávaná aktuálně v průběhu činností, ale zpětně za již ukončené činnosti. Tato data souvisejí s vyhodnocením efektivity vyu-

žití strojů. Prozatím jsou součástí systému dvě vyhodnocení. Vyhodnocení jsou dostupná pod položkou menu s názvem „Vyhodnocení“.

Prvním z nich je vyhodnocení celkové efektivity všech strojů po měsících. Efektivita strojů je vypočtena jako poměr produktivního času k celkovému času stroje. Výsledek je vypočten v procentech. Vypočtená efektivita vytváří obraz o využití strojů, u kterých je ve společnosti v současnosti nasazen tento systém. Ukázka je zobrazena na obrázku (Obr. 24).



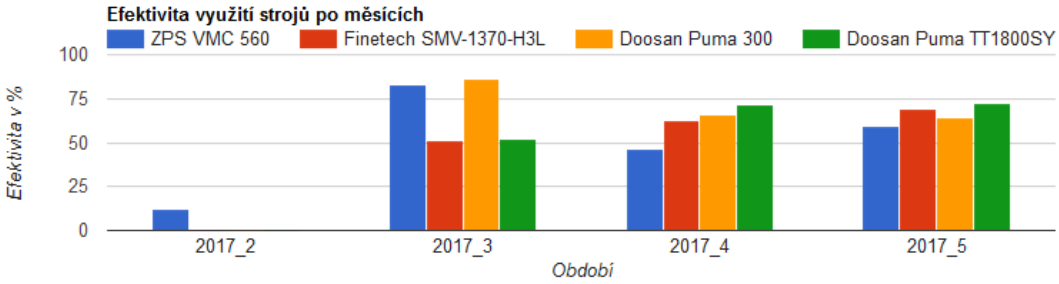
Rok	Měsíc	Produkce(min)	Celkem(min)	Efektivita(%)
2017	2	1569	21722	7.22309
2017	3	235809	351897	67.0108
2017	4	122101	196923	62.0044
2017	5	14983	21974	68.1851

Obr. 24. Ukázka vyhodnocení celkové efektivity po měsících.

Bohužel jsou data k dispozici pouze za několik málo týdnů a tak ještě není možné vzájemně porovnávat hodnoty z uplynulých měsíců.

Částečnou obměnou předchozího vzniklo vyhodnocení, které zobrazuje efektivitu v % pro jednotlivé stroje po měsících. Jedná se o vyhodnocení dat zpětně, podobně jako v předchozím případě. Vyhodnocení ukazuje, s jakou efektivitou pracovaly stroje za uplynulé měsíce. Ukázka tohoto vyhodnocení je na obrázku (Obr. 25).

Zmíněná, společností požadovaná, vyhodnocení jsou již v systému implementována. Vyhodnocení poskytují data prozatím za poměrně krátkou dobu, protože nasazení systému proběhlo před několika týdny. Přesto i v tuto chvíli poskytují společnosti relevantní informace pro vytvoření obrazu o efektivnosti využití jednotlivých strojů. Je velmi pravděpodobné, že bude společnost chtít připravit další vyhodnocení, která pomohou dále analyzovat stav o využití strojů.



Rok	Měsíc	Stroj	Název	Produkce(min)	Celkem(min)	Efektivita(%)
2017	2	1	ZPS VMC 560	1567	12939	12.1107
2017	2	2	Finetech SMV-1370-H3L	2	8783	0.0227712
2017	3	1	ZPS VMC 560	120960	145833	82.9442
2017	3	2	Finetech SMV-1370-H3L	74625	146242	51.0284
2017	3	3	Doosan Puma 300	23054	26727	86.2573
2017	3	4	Doosan Puma TT1800SY	17170	33095	51.8809
2017	4	1	ZPS VMC 560	20282	43889	46.212
2017	4	2	Finetech SMV-1370-H3L	35628	56847	62.6735
2017	4	3	Doosan Puma 300	29937	45533	65.7479
2017	4	4	Doosan Puma TT1800SY	36254	50654	71.5718
2017	5	1	ZPS VMC 560	1794	3012	59.5618
2017	5	2	Finetech SMV-1370-H3L	2753	3971	69.3276
2017	5	3	Doosan Puma 300	2913	4548	64.0501
2017	5	4	Doosan Puma TT1800SY	7523	10443	72.0387

Obr. 25. Ukázka vyhodnocení efektivity podle strojů za měsíce.

10 DALŠÍ VÝVOJ SYSTÉMU

Navržená struktura systému vychází z požadavků konkrétní výrobní společnosti, ale při jejím návrhu bylo přihlíženo na obecnější pohled na požadavky. S tímto ohledem vznikl tento systém, který pokrývá požadavky na systém zadavatele. Jedná se však o prototypovou aplikaci, která bude dále vyvíjena k širšímu využití. Některé její části budou přepracovány. Jedná se zejména o vzhled uživatelského rozhraní systému. Na ten nebyl kladen důraz, protože se jedná o funkční prototyp systému.

Jednou z prvních možností pro zlepšení implementovaného systému je napojení na stávající informační systém výrobní společnosti. V první fázi by bylo napojení zaměřeno na kontrolu vkládaného čísla zakázky operátorem v rámci zadání produktivní činnosti. Smyslem tohoto propojení bude zejména validovat platnost zadání výrobní zakázky. Po zadání výrobní zakázky operátorem stroje bude číslo zakázky ověřeno v informačním systému. Implementace tohoto napojení zabezpečí udržení spolehlivé vazby mezi informačním systémem výrobní společnosti a tímto systémem, vytvořeným v rámci této práce.

Dalším rozšířením systému bude vytvoření informačního panelu. Informační panel bude sloužit k rychlému přehledu o stavu všech strojů a jejich operátorů. Tento informační panel bude zobrazen na velkoplošné obrazovce, která bude umístěna na dobře viditelném místě na dílně. Operátoři i mistr výroby bude díky tomu mít na přehledném místě k dispozici informace o stavu stroje, na kterém sami právě pracují. Ovšem hlavním důvodem bude dostupnost informací i o ostatních pracovištích se stroji na dílně. Předpokládaným přínosem tohoto rozšíření bude vzájemná kontrola operátorů o jejich stavu, která by měla vést ke zvýšení produktivity operátorů.

Poslední zde uvedenou možností vývoje tohoto systému je vytvoření vlastního systému pro sběr dat ze strojů. Jedním z důvodů pro tento krok je finanční náročnost řešení třetí strany a nedostatečná dokumentace a podpora k tomuto systému. Díky vlastnímu řešení dojde k úspoře nákladů na připojení stroje a navíc bude získána plná kontrola nad zpracováním naměřených dat. Druhým důvodem je vytvoření systému, který bude zcela jiného konceptu, zejména v oblasti ukládání naměřených dat do databáze. V rámci tohoto rozšíření by byly ukládány pouze změny stavů stroje, nikoli kontinuální ukládání stále stejného stavu, jako je tomu u stávajícího systému pro sběr dat ze strojů. V případě setrvalého stavu stroje by žádná data nebyla ukládána. Přínosem tohoto rozšíření bude výrazné snížení zátěže databázového serveru a úspora místa.

ZÁVĚR

Tato práce je věnována návrhu a implementaci systému pro kontrolu výroby. Jedná se o porovnání dat naměřených ze strojů a dat zadaných operátory. Pro dosažení tohoto záměru bylo nutné nejdříve důkladně analyzovat již implementovaný systém pro sběr dat ze strojů, který je dodán třetí stranou a provozován několik měsíců. Pomocí této analýzy byla zjištěna funkcionalita dodaného řešení. Dále bylo nutné zjistit požadavky na vytvářený systém. Konzultacemi se zástupcem výrobní společnosti byl postupně proveden sběr požadavků. Z těchto požadavků bylo možné následně vytvořit jednotlivé diagramy případů užití. Při jejich tvorbě bylo potřeba opakovaně konzultovat požadavky, protože některé z nich nebyly zcela jednoznačné nebo byly protichůdně definované. Po vytvoření všech případů užití bylo již možné přistoupit k vlastnímu návrhu aplikace. Bylo již také možné zvolit technologie pro vývoj systému.

Během analýzy požadavků na systém byla vyslovena podmínka na provozování systému i na mobilních zařízeních s omezenou velikostí displeje. K tomuto požadavku bylo při návrhu systému přihlédnuto. V rámci návrhu aplikace bylo tedy nutné vyřešit tento požadavek. Vzhledem ke dvěma možným způsobům řešení, jako mobilní aplikace nebo jako webová aplikace v responzivním designu, bylo nutné rozhodnout o vhodné variantě. Po zvážení všech požadavků bylo rozhodnuto o webové variantě řešení implementace tohoto systému. Bylo tedy nutné vybrat vhodné webové technologie pro vývoj. V tomto případě sehrál svůj význam požadavek na vytvoření systému pomocí bezplatných licencí. S ohledem na tyto požadavky byl pro vývoj systému zvolen programovací skriptovací jazyk PHP v rámci frameworku Nette. Jako databáze byla zvolena databáze MariaDB. Během vývoje byl kladen důraz na co možná nejsnadnější ovládání. Cílem je vytvoření systému, který bude optimálně získávat data od operátorů strojů, ale přitom je nebude zaneprázdňovat natolik, aby byla práce se systémem kontraproduktivní. Dalším aspektem pro vývoj bylo vytvoření systému, který vzhledem k plánovanému použití na mobilních zařízeních s omezeným výkonem nebude příliš náročný na zařízení. Při implementaci systému byla také zohledněna možná bezpečnostní rizika, která tento typ systémů mohou ohrožovat. Na tyto známé zranitelnosti byly implementovány dostupné opatření pro jejich omezení.

Všechny tyto požadavky byly dodrženy a zohledněny při návrhu a implementaci systému. Implementovaný systém byl otestován na vytvořených testovacích datech. Testová-

ním byly zjištěny nedostatky, které byly odstraněny. V průběhu měsíce dubna roku 2017 byl systém nasazen do reálného testování uživateli s online napojením na data ze strojů. V současné době je systém reálně testován. Jedná se stále o prototyp aplikace a je tedy kladen stále důraz na zajištění funkčnosti, než vzhledu či komfortu obsluhy systému. Během testování v reálném provozu jsou zjištěné nedostatky shromažďovány a jsou průběžně opravovány.

Velmi důležitou částí systému je také vyhodnocení dat ze strojů. Sběrem dat je tedy možné získat jistou množinu tvořící informace, které již mohou poskytovat relevantní obrázek o skutečném využití strojů. Získaná data jsou cílem vytvořeného systému, protože na základě nich je možné zjistit přesnější využití strojů. Je tak možné, aby vedení společnosti provedlo patřičné kroky pro zajištění vyšší produktivity strojů, která společnosti přinese nižší náklady na výrobu a snad také zkrácení času potřebného pro vyrobení jednotlivých zakázek.

Vytvořený systém bude i nadále vyvíjen. Jedním z prvních plánovaných rozšíření je napojení na stávající informační systém výrobní společnosti. Prozatím by se jednalo o zajištění propojení na informace o výrobních zakázkách. Vytvoření informačního panelu na dílnu společnosti je jedním z dalších plánovaných rozšíření, které přinese větší informovanost operátorů. Stávající systém sběru dat není úplně optimální a tak dalším plánovaným rozšířením systému je vytvoření vlastního řešení pro sběr dat. Toto řešení bude realizováno kompletně, tedy jak po HW stránce, tak i po stránce SW.

SEZNAM POUŽITÉ LITERATURY

[1] *Informatika a výpočetní technika pro střední školy: Teoretická učebnice* [online]. 1. Brno: Computer Press, 2012 [cit. 2017-04-22]. ISBN 978-80-251-3228-9. Dostupné z: https://books.google.cz/books?id=qw0nCwAAQBAJ&printsec=frontcover&hl=cs#v=onepage&q&f=false

[2] Webové aplikace. *Jak na Internet* [online]. 2014 [cit. 2017-04-22]. Dostupné z: https://www.jaknainternet.cz/page/1262/webove-aplikace/

[3] Bezpečnostní rizika pro webové aplikace. *SystemOnLine* [online]. 2016 [cit. 2017-04-14]. Dostupné z: http://www.systemonline.cz/clanky/bezpecnostni-rizika-pro-webove-aplikace.htm

[4] Usage of server-side programming languages for websites. *W3Techs* [online]. 2017 [cit. 2017-01-29]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all

[5] Preface. *PHP* [online]. 2017 [cit. 2017-04-14]. Dostupné z: http://cz1.php.net/manual/en/preface.php

[6] PHP Licensing. *PHP* [online]. 2017 [cit. 2017-04-14]. Dostupné z: https://secure.php.net/license/

[7] Raw PHP vs PHP Frameworks. *Carmatec* [online]. b.r. [cit. 2017-04-14]. Dostupné z: https://www.carmatec.com/blog/raw-php-vs-php-frameworks/

[8] The Most Popular Framework of 2015. *Sitepoint* [online]. 2015 [cit. 2017-01-29]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[9] Hall of Fame. *Nette* [online]. 2017 [cit. 2017-04-14]. Dostupné z: https://nette.org/contributors

[10] Release Announcements & News. *Nette* [online]. 2017 [cit. 2017-04-14]. Dostupné z: https://forum.nette.org/en/f78-release-announcements-news

[11] Licenční politika. *Nette* [online]. 2017 [cit. 2017-04-09]. Dostupné z: https://nette.org/cs/license

[12] Symfony License. *Symfony* [online]. 2017 [cit. 2017-04-14]. Dostupné z: http://symfony.com/doc/current/contributing/code/license.html

[13] License & Credits. *LaravelBoilerplate* [online]. 2016 [cit. 2017-04-14]. Dostupné z: http://laravel-boilerplate.com/license.html

[14] Hlavní přednosti. *Nette* [online]. 2017 [cit. 2017-04-15]. Dostupné z: https://nette.org/cs/#toc-features

[15] MVC aplikace & presentery. *Nette* [online]. 2017 [cit. 2017-02-05]. Dostupné z: https://doc.nette.org/cs/2.4/presenters

[16] Prezentační vzory z rodiny MVC. *Zdroják.cz* [online]. 2009 [cit. 2017-04-16]. Dostupné z: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[17] Databáze. *Nette* [online]. 2017 [cit. 2017-02-05]. Dostupné z: https://doc.nette.org/cs/2.4/database

[18] Database\Table. *Nette* [online]. 2017 [cit. 2017-02-05]. Dostupné z: https://doc.nette.org/cs/2.4/database-table

[19] Database: ActiveRow. *Nette* [online]. 2017 [cit. 2017-02-05]. Dostupné z: https://doc.nette.org/cs/2.4/database-activerow

[20] Database: Selection. *Nette* [online]. 2017 [cit. 2017-02-05]. Dostupné z: https://doc.nette.org/cs/2.4/database-selection

[21] Výchozí Latte makra. *Latte* [online]. 2017 [cit. 2017-03-20]. Dostupné z: https://latte.nette.org/cs/macros

[22] *Výchozí helpery šablon: Latte* [online]. 2017 [cit. 2017-03-20]. Dostupné z: https://doc.nette.org/cs/2.1/default-helpers

[23] Latte. *Nette* [online]. 2017 [cit. 2017-01-30]. Dostupné z: https://latte.nette.org/cs/

[24] Tracy. *Nette* [online]. 2017 [cit. 2017-01-29]. Dostupné z: https://tracy.nette.org/

[25] MariaDB. *MariaDB* [online]. 2016 [cit. 2017-01-30]. Dostupné z:

https://mariadb.com/

[26] PhpMyAdmin. *PhpMyAdmin* [online]. 2017 [cit. 2017-02-01]. Dostupné z: https://www.phpmyadmin.net/

[27] *Apache* [online]. 2017 [cit. 2017-01-22]. Dostupné z: https://httpd.apache.org/

[28] NetBeans IDE Features. *NetBeans* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://netbeans.org/features/index.html

[29] Welcome to the NetBeans Community. *NetBeans* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://netbeans.org/about/index.html

[30] The NetBeans Platform. *NetBeans* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://netbeans.org/features/platform/index.html

[31] Zabezpečení před zranitelnostmi. *Nette* [online]. 2017 [cit. 2017-04-01]. Dostupné z: https://doc.nette.org/cs/2.4/vulnerability-protection

[32] Cross-Site Request Forgery (CSRF). *Open Web Application Security Project* [online]. 2016 [cit. 2016-11-01]. Dostupné z: https://www.owasp.org/index.php/Cross-Site_Request_Forgery

[33] Rady pro administrátory. *CSIRT.CZ* [online]. 2016 [cit. 2016-11-02]. Dostupné z: https://www.csirt.cz/page/2797/zakladne-minimum-zabezpecenia-webovej-stranky/

[34] Cross Site Request Forgery. *SOOM.CZ* [online]. 2008 [cit. 2016-11-01]. Dostupné z: http://www.soom.cz/clanky/484--Cross-Site-Request-Forgery

[35] Cross-Site Request Forgery. *PHP triky* [online]. 2006 [cit. 2016-11-02]. Dostupné z: https://php.vrana.cz/cross-site-request-forgery.php

[36] Pokročilé techniky XSS. *SOOM.CZ* [online]. 2008 [cit. 2016-11-08]. Dostupné z: http://www.soom.cz/clanky/485--Pokrocile-techniky-XSS

[37] Cross-site scripting (XSS). *OWASP* [online]. 2016 [cit. 2016-11-08]. Dostupné z: https://www.owasp.org/index.php/XSS

[38] HttpOnly. *OWASP* [online]. 2017 [cit. 2017-04-02]. Dostupné z: https://www.owasp.org/index.php/HttpOnly

[39] Zranitelnosti a útoky spojené se session managementem. *SOOM* [online]. 2016 [cit. 2017-04-02]. Dostupné z: https://www.soom.cz/clanky/1184--Zranitelnosti-a-utoky-spojene-se-session-managementem#c3

[40] Session fixation. *OWASP* [online]. 2014 [cit. 2017-04-02]. Dostupné z: https://www.owasp.org/index.php/Session_fixation

[41] Autentizace a autorizace. *TRISUL* [online]. 2016 [cit. 2016-11-17]. Dostupné z: http://www.trisul.cz/bezpecnost-autentizace-autorizace/

[42] Základní autentizační metody. *ICT SECURITY* [online]. 2009 [cit. 2016-11-17]. Dostupné z: http://www.ictsecurity.cz/component/content/article?id=2721

[43] Dělená měřicí proudová trafa s vysokým poměrem (mA). *KMB Systems* [online]. 2011 [cit. 2017-04-05]. Dostupné z: http://www.kmb.cz/index.php/cs/prislusenstvi/delena-merici-proudova-trafa-ma

[44] Context-Aware Escaping. *Latte* [online]. 2017 [cit. 2017-04-01]. Dostupné z: https://latte.nette.org/cs/#toc-context-aware-escaping

[45] Obrana před Cross-Site Request Forgery (CSRF). *Nette* [online]. 2017 [cit. 2017-04-02]. Dostupné z: https://doc.nette.org/cs/2.4/forms

[46] Autentifikace. *Nette* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://doc.nette.org/cs/2.4/quickstart/authentication#toc-callbacky-formularu

[47] Přihlašování & oprávnění uživatelů. *Nette* [online]. 2017 [cit. 2017-04-19]. Dostupné z: https://doc.nette.org/cs/2.4/access-control

[48] Práce s hesly – Nette\Security\Passwords. *Nette* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://doc.nette.org/cs/2.4/passwords

[49] PDO. *PHP* [online]. 2017 [cit. 2017-02-05]. Dostupné z: http://php.net/manual/en/intro.pdo.php

[50] SQL Joins. *W3Schools.com* [online]. 2017 [cit. 2017-02-05]. Dostupné z: http://www.w3schools.com/sql/sql_join.asp

[51] Co je sociální inženýrství? - 1. díl. *PCWorld* [online]. 2012 [cit. 2016-11-01].

Dostupné z: http://pcworld.cz/internet/co-je-socialni-inzenyrstvi-1-dil-44361

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SW	Software.
HW	Hardware.
PHP	Hypertext Preprocessor.
MVC	Model-View-Controler.
MVP	Model-View-Presenter.
HTML	Hyper Text Markup Language.
AJAX	Asynchronous JavaScript and XML.
XML	Extensible Markup Language.
BSD	Berkeley Software Distribution.
GPL	General Public License.
SEO	Search Engine Optimization.
SQL	Structured Query Language.
API	Application Programming Interface.
ODBC	Open Database Connectivity.
PDO	PHP Data Objects.
CNC	Počítačem řízený obráběcí stroj (Computer Numeric Control).
HTTP	Hypertext Transfer Protocol.
CSRF	Cross-Site Request Forgery.
CSM	Content Management System.
JSON	JavaScript Object Notation.
URL	Uniform Resource Locator.
XSS	Cross-Site Scripting.
TCP/IP	Transmission Control Protocol/Internet Protocol.
OEE	Overall Equipment Effectiveness.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: DIAGRAM DATABÁZOVÉ STRUKTURY

PŘÍLOHA P II: DIAGRAM MODELU TŘÍD