VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VZDÁLENÉ MONITOROVÁNÍ A DIAGNOSTIKA VÝROBNÍCH PROCESŮ

REMOTE MONITORING AND DIAGNOSTICS OF PRODUCTION PROCESSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB LIŠKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Liška Jakub**
Program: Informační technologie
Název: **Vzdálené monitorování a diagnostika výrobních procesů**
Remote Monitoring and Diagnostics of Production Processes
Kategorie: Informační systémy

Zadání:

1. Seznamte se s možnostmi vzdáleného monitorování výrobních prostředků PLC. Prostudujte režimy komunikačních modulů a způsoby získávání dat po síti.
2. Prostudujte softwarové prostředky pro tvorbu webových informačních systémů na platformě PHP.
3. Navrhněte architekturu informačního systému pro shromažďování provozních informací ze zařízení a jejich další vyhodnocení pomocí webového rozhraní.
4. Implementujte navržený systém pomocí vhodných technologií. Implementujte počítadla cyklů, sledování chyb a stavu, spotřebu energií, případně sledování dalších ukazatelů.
5. Proveďte testování implementovaného systému.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 16. října 2019

Abstrakt

Cieľom tejto bakalárskej práce je preskúmať možnosti vzdialeného monitoringu výrobných procesov v priemysle a následnej implementácii týchto možností do funkčnej aplikácie. Výslednú aplikáciu je možné rozdeliť na dve hlavné časti. Prvá časť je samotné získanie surových dát z výrobného procesu (výrobnej linky, priemyselného zariadenia) za pomoci komunikačných modulov a ich odoslanie na vzdialený server. V druhej časti dochádza k spracovaniu daných dát na servery, ich uloženiu do databázy a ich zobrazenie koncovému uživateľovi pomocou rozhrania vo webovom prehliadači. Aplikácia podporuje zobrazenie dôležitých ukazateľov v reálnom čase (s určitou limitáciou obnovovacej frekvencie použitých komunikačných modulov) ako aj zobrazenie historických dát.

Abstract

The goal of this bachelor thesis is to explore the possibilities of remote monitoring of production processes in industry and the further implementation of these possibilities into a functional application. The application can be divided into two main parts. The first part is the acquisition of raw data from the production process (production line, industrial equipment) using communication modules and send them to a remote server. In the second part, the data is processed on servers, stored in a database and displayed to the end user using an interface in a web browser. The application supports the display of important indicators in real time (with a certain limitation of the refresh rate of the used communication modules) as well as the display of historical data.

Kľúčové slová

Nette, výrobný proces, PLC, monitoring, Priemysel 4.0, IIoT, PHP7, MySQL, informačný systém

Keywords

Nette, production process, PLC, monitoring, Industry 4.0, IIoT, PHP7, MySQL, information system

Citácia

LIŠKA, Jakub. *Vzdálené monitorování a diagnostika výrobních procesů*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Vzdálené monitorování a diagnostika výrobních procesů

Prehlásenie

Prehlasujem, že túto bakalársku prácu som vypracoval samostatne pod vedením pána Ing. Radka Burgeta Ph.D. Ďalšie informácie mi poskytli páni Ing. Ján Tížik a Ing. Peter Fábry. V tejto bakalárskej práci som uviedol všetky využité literárne pramene, publikácie a zdroje, z ktorých som čerpal.

.....
Jakub Liška
16. júna 2020

Poďakovanie

Rád by som poďakoval vedúcemu mojej bakalárskej práce pánovi Ing. Radkovi Burgetovi, firme Rossum Integration s.r.o. za zápožičku hardwaru a prístup k developérskym účtom. Taktiež by som rád poďakoval rodičom a sestre za trpezlivosť a podporu.

Obsah

Kapitola 1

Úvod

V súčasnej dobe prebieha trend masívneho zavádzania informačných technológii do oblasti priemyslu, ktoré ešte pred niekoľkými rokmi boli v tomto odvetví neznáme. Priemysel sa tak postupne dostáva na novú úroveň, ktorá dostala prívlastok Industry 4.0 (v slovenskom preklade Priemysel 4.0), čo značí novú úroveň priemyslu.

Vo veľkej miere prebieha aj automatizácia procesov výroby, ktoré sa ešte do nedávna spoliehali iba na ľudskú pracovnú silu. Vďaka tomu vo výrobných halách pribúdajú komplikované stroje a redukuje sa počet operátorov výroby (obslužného personálu). To má za následok zefektívnenie produkcie, avšak, aby bola táto efektívnosť dlhodobo udržateľná, je potrebné výrobný proces neustále monitorovať. Práve za týmto účelom sú nové stroje a linky osádzané zadriadeniami pre vzdialený monitoring a zber užitočných dát. Tieto dáta sú veľmi užitočné nielen pre sledovanie činnosti stroja v reálnom čase bez nutnosti nachádzať sa fyzicky pri ňom, ale slúžia aj pre sledovanie efektívnosti, nákladov na prevádzku, ako napríklad spotreba energie, či náklady na servis. Sledovaním historických dát je možné u určitých prípadov odhaliť aj blížiacu sa vážnu poruchu, ktorú je možné odstrániť v rámci bežnej údržby, čím sa redukeje čas, kedy daný stroj neprodukuje a tvorí stratu.

Cieľom tejto bakalárskej práce bolo vytvorenie vlastnej aplikácie pre zber a zobrazenie cenných dát z výrobného procesu. Dáta sú získavané priamo z riadiaceho počítača výrobnej linky alebo jej časti za pomoci komunikačného modulu. Táto práca sa zameriava na získavanie dát pomocou univerzálnych komunikačných modulov, vďaka ktorým je možné získavať dáta z riadiacich počítačov rôznych výrobcov a zobrazovať ich v jednej univerzálnej aplikácii. Ak by sme chceli použiť riešenia konkrétnych výrobcov, komplikovala by sa implementácia použitými štandardmi, štruktúrou získaných dát a ďalšími špecifikami jednotlivých riešení. Získané dáta z riadiaceho počítača sú pomocou komunikačného modulu odosielané po sieti na server, kde dochádza k ich uloženiu do databázy pre ich neskoršiu analýzu. Prípadne je možné sledovať tieto dáta v reálnom čase. Zobrazenie dát bude možné v užívateľskej aplikácii vo webovom prehliadači.

Kapitola 2

Riadenie výrobného procesu a získavanie dát z neho

Zariadenie, stroj, na ktorom prebieha výrobný proces je riadené za pomoci hlavného riadiaceho počítača, v praxi označovaného skratkou PLC[1]. Ak je stroj zložitejšej konštrukcie, môže obsahovať viacero PLC rozdistribuovaných vrámci stroja, ktoré spolu komunikujú. Aktuálne je na trhu niekoľko výrobcov PLC, ako napríklad Siemens, Allen Bradley, Mitsubishi, Schneider a mnohé ďalšie menšie spoločnosti.

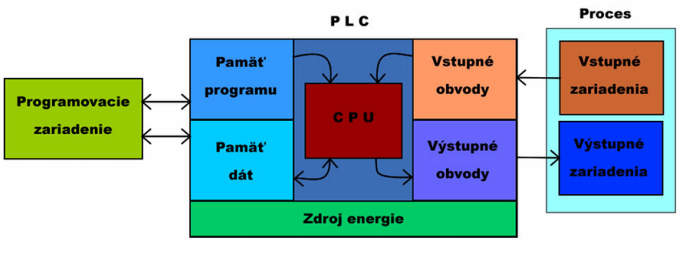
Obr. 2.1: Moderné PLC Siemiens Simatik S7-1500

Pre účely tejto bakalárskej práce som mal zapožičané PLC Siemens Simatic S7-1212–DC/DC/DC. Siemens je aktuálne lídrom na európskom trhu v oblasti automatizácie, z tohto dôvodu ho budem využívať aj v tejto bakalárskej práci.

2.1 Architektúra PLC

Súčasné PLC sa svojou základnou architektúrou zásadne nelíšia od prvých PLC. Každé PLC obsahuje niekoľko základných častí. Medzi tieto časti patrí centrálne procesorová jednotka (CPU), ktorá sa stará o beh firmwaru a vykonávanie programu. Ďalej obsahuje pamäť programu a pamäť dát. Neoddeliteľnou súčasťou sú potom hardwerové vstupy a výstupy pre jednotlivé signály. Postupným vývojom pribudli rôzne komunikačné rozhrania pre možnosť komunikácie s okolitými zariadeniami a rozširujúcimi modulmi. [2]

[1]Programmable Logic Controller (PLC), v preklade "Programovateľné Logické Riadenie"



Obr. 2.2: Bloková schéma popisujúca architektúru PLC

2.2 Operačný cyklus PLC

Princíp práce PLC sa v praxy nazýva operačným cyklom. Jedná sa o sled udalostí, ktoré sa neustále opakujú. Rovnako ako architektúra, tak aj operačný cyklus sa naprieč jednotlivými riešeniami rôznych výrobcov veľmi neodlišujú. Z toho dôvodu je ho možné popísať v jeho zjednodušenej forme. Celý cyklus začína vytvorením "Process image".[2]. [2]

Obr. 2.3: Diagram zjednodušeného operačného cyklu PLC

Process image sa vytvára zo špeciálnej časti operačnej pamäte, v ktorej sa nachádzajú aktuálne hodnoty vstupov a výstupov. Ďalej prebehne vykonanie programu nad týmto obrazom. Po vykonaní sú aktualizované hodnoty výstupnej časti obrazu. V ďalšom kroku sú hodnoty zapísané na hardwarové výstupy. Následne prebieha komunikácia so zariadeniami, ktoré sú určené na vizualizáciu a monitoring procesu. Jedná sa v tomto prípade predovšetkým o rôzne kontrolné a zobrazovacie panely. Posledná časť cyklu zahŕňa aktualizáciu systémových a stavových premenných. Následne, ak je systém pripravený, začína celý cyklus od začiatku. Dĺžka jedného cyklu sa bežne pohybuje v desiatkach, maximálne v stovkách milisekúnd, záleží na zložitosti programu a počte vstupov a výstupov.

[2]Proccess image – aktuálny obraz procesu

2.3 Komunikačné moduly

Takmer každé moderné PLC disponuje určítým typom komunikačného rozhrania, pomocou ktorého je možné pripojiť moduly rozširujúce jeho využitie. Pomocou týchto portov je však možné pripájať k PLC aj iné kompatibilné zariadenia, s ktorými môže PLC následne po úprave konfigurácie komunikovať. Vďaka tomu môže PLC komunikovať lokálne s iným PLC, ako aj s rôznymi inteligentnými kamerami, robotmi, manipulátormi a rôznymi inými zariadeniami, ktoré sú súčasťou daného stroja/linky.

Nemusí sa tak jednať iba o zariadenia, ktoré vykonávajú učitú činnosť na základe programu v PLC, môže sa jednať aj o zariadenie, ktoré dokáže byť pripojené k PLC a monitorovať jeho chovanie, respektíve dáta, s ktorými pracuje a následne tieto dáta zálohovať lokálne alebo odosielať do siete LAN, ale aj WAN, kde ich následne spracuje koncová aplikácia.

Obr. 2.4: Rodina komunikačných modulov a priemyselných routrov Ewon

Na trhu sú momentálne riešenia rôznych výrobcov. Môže sa jednať o zariadenia, ktoré vyrábajú priamo výrobcovia PLC, alebo je možné zakúpiť kompatibilné komunikačné moduly výrobcov tretích strán, napríklad HMS–Ewon, InSYS, KEB, IXON a iné. Tieto komunikačné moduly sú výhodné v situácii, ak výrobca priemyselných strojov využíva vo svojich produktoch rôzne modelové rady alebo výrobcov PLC. To všetko je možné vďaka rozšírenej podpore komunikačných štandardov pri týchto zariadeniach. Taktiež tieto zariadenia vo väčšine prípadov umožňujú spojenie pomocou VPN tunela. Následne je možné vzdialene pristupovať k pripojeným zariadeniam a vykonávať softwarové úpravy, čo šetrí čas, najmä ak sa jedná o poruchu, ktorú je potrebné odstrániť úpravou nastavenia alebo programu.

2.4 Komunikačné protokoly a štandardy

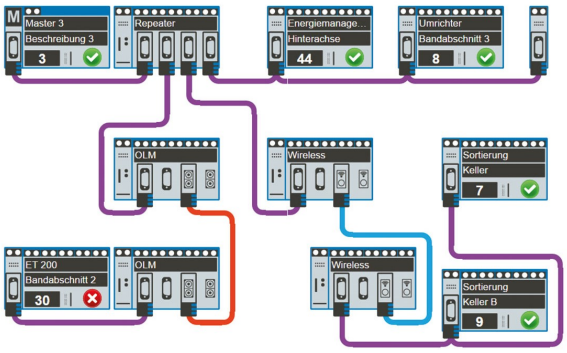
Komunikačné protokoly a štandardy v priemysle sa postupom času vyvíjali rovnako, ako sa vyvíjali rozhrania danej doby používané v PC. Z počiatku sa používali rôzne štandardy zbernicového typu. Výrobcovia HW bežne využívali najmä porty tybu DE–9 a DB–25. Na týchto portoch následne využívali sériové komunikačné štandardy RS232, RS422 a RS485. Medzi najrozšírenejšie komunikačné protokoly patrili napríklad Profibus[3], DF–1[4] alebo Modbus RTU a Modbus ASCII[5].

[3]Profibus – protokol predstavený v roku 1989, využívaný prevažne v zariadeniach Siemens
[4]DF–1 – Protokol používaný v zariadeniach Allen–Bradley, využíval štandard RS–232
[5]Modbus RTU a ASCII – Protokol fungujúci na princípe Master/Slave, predstavený v roku 1979

V súčasnosti sa zbernicový typ komunikácie pri výrobe nových zadiadení nevyužíva. Nahradili ho protokoly založené na Ethernete s využítím portu typu RJ–45 alebo v prípade náročných podmienok šrobovacie konektory typu M12 a M8. Medzi tieto protokoly patria napríklad Profinet[6], EtherCAT[7], ako aj Modbus TCP[8].

Komunikácia pomocou zbernice

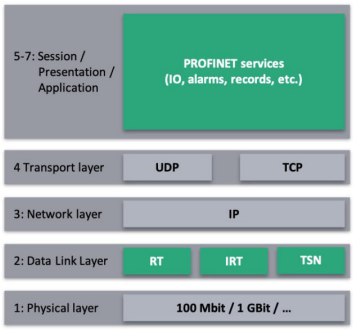
Princíp komunikácie po zbernici sa naprieč jednotlivými riešeniami príliš nelíši. V rámci komunikácie dvoch zariadení jedno vystupuje v úlohe Master (nadradené) a druhé Slave (podriadené). Jedno zariadenie typu Master môže komunikovať s viacerými zariadeniami Slave. Zariadenie typy Master posiela správy zariadeniam Slave, tie na dané správy odpovedajú, ak sa nejednalo o broadcast správu. Slave nemôže odoslať správu Mastrovi, ak predtým nedostal správu s požiadavkou. V praxi PLC pracuje v režime Master a ostatné zariadenia, ktoré riadi zase v úlohe Slave.



Obr. 2.5: Príklad topológie systému pri použití zbernice

Komunikácia pomocou Ethernetu

Priemyselné protokoly založené na bežnom Ethernete[9] využívajú ako základ podľa OSI modelu fyzickú a dátovú vrstvu, v ktorých sa zhodujú. Vďaka tejto vlastnosti je možné



Obr. 2.6: Mapovanie priemyselného štandardu Profinet na ý-vrstvý OSI model

[6]Profinet – vyvinutý rovnakou spoločnosťou ako Profibus, predstavený v roku 2003
[7]EtherCAT – vyvinutý firmou Beckhoff Automation
[8]Modbus TCP – Jedná sa o upravenú verziu Modbus RTU, rozšírenú o TCP interface
[9]Ethernet – štandard IEEE 802.3

využívať v priemyselných ethernetových sieťach aj komerčné zariadenia. Ostatné vrstvy sa pri konkrétnych riešeniach líša. Predovšetkým dochádza k úprave v aplikačnej vrstve.

Obr. 2.7: Konektory využíváne pri priemyselnom ethernete (M12 a RJ-45)

Priemyselná verzia ethernetu sa narozdiel od komerčnej verzie líši v zvýšenej odolnosti zariadení a kabeláže. Taktiež sa tu dbá na vysokú mieru determinizmu, vysokú spoľahlivosť a nízku latenciu. To všetko je potrebné zabezpečiť pre spoľahlivú činnosť a kontrolu procesu v reálnom čase. Predovšetkým je potrebné riadiť proces s niekoľko milisekundovými presnosťami.

U väčšiny priemyselných protokolov založených na Ethernete sa využívajú Statické IP adresy pri adresovaní jednotlivých zariadení v sieti.

2.5 Získavanie dát pomocou komunikačných modulov

Spôsob získavania dát z PLC pomocou komunikačného modulu, je do veľkej miery ovplyvnený použítým komunikačným protokolom. Ten je závislí na podporovaných komunikačných protokoloch použitého hardwaru, konktrétne PLC.

Využitie zbernicového typu komunikácie

Pri zbernicovom type komunikácie sa komunikačné moduly zbierajúce dáta konfigurují ako zariadenie typu Master. PLC v tomto prípade vystupuje ako Slave. Pri konfigurácii je potrebné nastaviť správnu adressu PLC v rámci danej zbernice, typ použitého protokolu a hodnotu Baudrate [10] a ďalšie vlastnosti komunikácie podľa zvoleného protokolu. Následne sa v komunikačnom module nastavujú adresy premenných, ktoré majú byť monitorované a aj s akou frekvenciou majú byť monitorované. Jedná sa o adresy v špeciálnej časti pamäte.

Pri nastavovaní adries je potrebné dodržať syntax zápisu adresy. V prípade chybne zapísanej adresy nemusí komunikačný modul získať žiadne dáta a reportovať chybu, alebo môže získať dáta, ktoré patria inej premennej.

Využite komunikácie založenej na Ethernete

Pri použití komunikácie na základe Ethernetu je nastavenie podstatne jednoduchšie. Pri konfigurácii je potrebné nastaviť správny komunikačný protokol a IP adresu PLC, ktoré má byť monitorované. V prípade niektorých výrobcov je potrebné doplniť číslo racku, v ktorom sa nachádza CPU daného PLC a skontrolovať nastavenie pravidiel konektivity a čítania pamäte. V prípade niektorých PLC je tiež potrebné skontrolovať nastavenie adresovania pamäte. Moderné PLC podporujú absolútne, ale aj symbolické adresovanie pamäte, no

[10] Baudrate – komunikačná rýchlosť zbernice

pre monitorovanie väčšina riešení využíva absolútne adresovanie. Následne je potrebné v monitorovaciom zarianení nastaviť adresy premenných, ktoré majú byť monitorované a s akou frekvenciou.

2.6 Prenos dát vzdialenej strane

Prenos dát na vzdialený server alebo do aplikácie každý výrobca implementoval vlastným spôsobom. Vždy sa jedná o bezpečný prenos šifrovaných dát. Väčšinou sa jedná o spojenie pomocou VPN so serverom výrobcu. Užívateľská aplikácia alebo server koncovej aplikácie následne komunikuje so serverom výrobcu, ktorý zastupuje úlohu prostredníka. Užívateľská aplikácia alebo koncový server nemusí udržiavať spojenie so všetkými zariadeniami, ktorých dáta spracováva. Túto úlohu zastáva server výrobcu, komunikuje so zariadeniami, riadi tok dát a zabezpečuje ich agregáciu pre následné jednoduchšie spracovanie. Pre prístup k dátam využívajú aplikácie tretích strán API daného výrobcu.

Obr. 2.8: Jednoduchý schématický popis bežne vyúžívaného riešenia vzdialeného prístupu v priemysle.

Kapitola 3

Prostriedky pre tvorbu webových informačných systémov

V tejto kapitole sú popísané nielen prostriedky pre tvorbu webových informačných systémov. Táto kapitola sa taktiež bližšie zaoberá webovým informačným systémom, aká je jeho úloha a architektúra.

Jednotlivým prostriedkom, architektúram a technológiam sú venované samostatné podkapitoly a sekcie.

3.1 Webový informačný systém

Informačný systém je systém otvorený, pracujúci s nehmotnými zdrojmi a informáciami. Práca s informáciami prebieha formou transformácií na základe vstupu a požadovaného výstupu. Informačné systémy prostredntvom vhodnej abstrakcie modelujú fyzický systém.

Webový informačný systém je formou webovej aplikácie, ktorá má funkcionalitu informačného systému.

3.2 Webová aplikácia

Webová aplikácia je software, určený pre beh na servery. Aby mohol užívateľ s danou aplikáciou pracovať, musí k danej aplikácii pristúpiť pomocou internetu. Užívateľ k tomuto účelu využíva webový prehliadač, rovnako ako keď pristupuje k webovej stránke.

Bežný užívateľ často nerozozná webovú aplikáciu od webovej stránky aj napriek tomu, že webové aplikácie často vyžadujú autentifikáciu užívateľa.

Webové stránky majú do veľkej miery statický obsah, ktorý je určený k prezentácii a je vyhľadatelný pomocou vyhľadávača. Naopak webová aplikácia pracuje s dátami, ktoré sú uložené v databáze alebo zadané užívateľom. Na základe požiadavky užívateľa sú tieto dáta spracované a výsledok spracovania zobrazí formou webovej stránky. Obsah výslednej stránky sa mení dynamicky na základe výstupných dát.

Výhody webových aplikácií

- Dáta potrebné k behu aplikácie sú uložené v databáze aplikácie. Pri využívaní aplikácie užívateľom na rôznych zariadeniach sú k dipozícii vždy aktuálne dáta.

- Pre beh aplikácie nie je potrebná inštalácia špeciálneho softwaru. Pre beh je potrebné aby dané zariadenie disponovalo bežným moderným prehliadačom.
- Webové aplikácie sú multiplatformné.
- Užívatelia vždy pracujú s aktuálnou verziou aplikácie.

Nevýhody webových aplikácií

- Potrebné pripojenie k internetu pre beh aplikácie.
- Nestabilné alebo pomalé pripojenie k internetu dokáže do veľkej miery lyvniť plynulosť chodu aplikácie.
- V prípade slabého zabezpečenia aplikácie môže útočník ľahko získať citlivé dáta alebo dáta poškodiť, znehodnotiť.

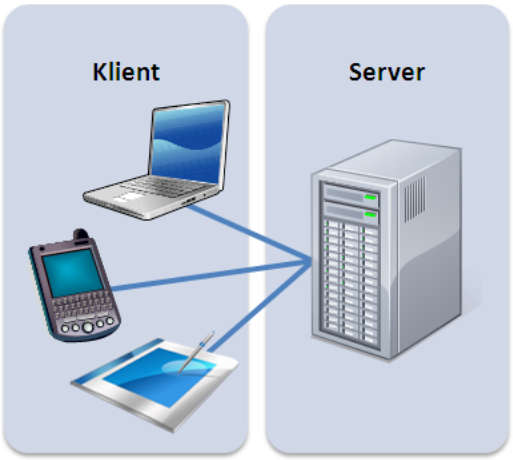
3.3 Architektúra informačného systému

Architektúra informačného systému určuje, ako daný systém pracuje a ako bol navrhnutý. Architektúra vychádza z rozdelania funkčnosti systemému na viacero častí. Podľa toho, na koľko častí je daný systém rozdelený a akú funkcionalitu tieto časti poskytujú, poznáme niekoľko typov informačných systémov.

Niektoré typy sú už historickým prežitkom, ako napríklad Mainframe[1], iné sú v súčasnosti hojne využívané. Medzi architektúry využívané v súčasnosti patria predovšetkým monolitické architektúry typu Klient-Server (2-vrstvá) a modernejšia 3-vrstvá architektúra. Využívajú sa taktiež rôzne podoby distribuovaných typov, tieto sa využívajú pre potreby rozsiahlých informačných systémov a pre potreby tejto práce sú nezaujímavé.

3.3.1 Architektúra Klient–Server

Informačný systém typu Klient-Server je 2-vrstvým typom architektúry. Celý systém je tak



Obr. 3.1: Architektúra Klient – Server

[1]Mainframe – jeden centrálny sálový počítač, užívatelia pristupovali pomocou terminálov

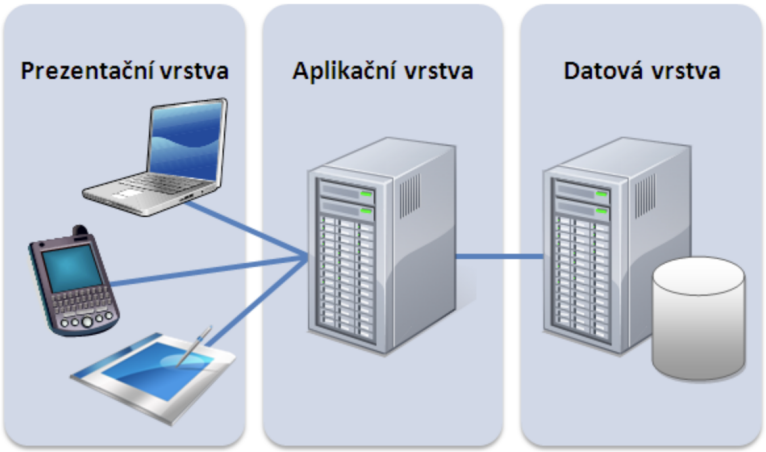
rozdelený na dva celky (klient a server), ktoré medzi sebou komunikujú a vymieňajú si dáta, pričom každý v sebe zahŕňa určitú časť potrebnej funkcionality.

Klientská časť v sebe zahŕňa užívateľské rozhranie a aplikačnú logiku. Klientská časť pomocou aplikačnej logiky prekladá požiadavky užívateľa do podoby, ktorej rozumie server a naopak. Serverová časť zabezpečuje beh a správu databázy. Serverová časť odosiela dáta z databázy na základe požiadavok od klientskej časti.

Klient-Server architektúra nie je príliš vhodná pre webové aplikácie z dôvodu vysokej náročnosti na koncové zariadenie pri rozsiahlejšej aplikačnej logike a zložitú implementáciu pre beh vo webovom prehliadači. Z tohoto dôvodu sa od tejto architektúry pri tvorbe webových aplikácií upúšťa a uprednostňuje sa 3-vrstvá architektúra.

3.3.2 3–vrstvá architekúta

V súčasnosti je 3-vrstvá architektúra hojne využívaná pri tvorbe informačných systémov a webových aplikácií. Vďaka rozdeleniu na 3 vrstvy lepšie rozdeľuje záťaž medzi zariadenie klienta a server.



Obr. 3.2: Trojvrstvá architektúra

Vrstvy architektúry:

- **Prezentačná vrstva** – vizualizácia informácií pomocou klienta, grafického rozhrania, zobrazeného vo webovom prehliadači. Prezentačná vrstva neobsahuje aplikačnú logiku, nespracováva dáta, môže kontrolovať správnosť vstupov. Implementovaná býva formou HTTP klienta.
- **Aplikačná vrstva** – obsahuje hlavnú logiku aplikácie. Spracováva dáta, vykonáva výpočty a funcie aplikácie. Komunikuje s prezentačnou a dátovou vrstvou aplikácie. Implementovaná býva pomocou skriptov, napríklad v jazykoch PHP, Python, Ruby, .NET a ďalších. Taktiež sa vo veľkej miere využívajú rôzne frameworky.[2]
- **Dátová vrstva** – zabezpečuje uchovanie a správu dát. Najčastejšie sa jedná o databázu, môže existovať aj v podobe súborového systému alebo inej služby.

[2]Framework – rámcové riešenie vyvinuté pre skriptovací jazyk. Obsahuje nástroje pre zjednodušenie vývoja aplikácie.

3.4 Technológie používané pri vývoji webových aplikácií

V tejto kapitole sú bližšie popísané technológie a jazyky, ktoré budú použité pri tvorbe webovej aplikácie. Tiež sa v tejto kategórii nachádzajú alternatívy, ktoré mohli byť využité pri tvorbe aplikácie.

HTML

HTML je štandardný značkovací jazyk určený pre tvorbu webových stránok. Skratka HTML vznikla z písmen plného znenia názvu v anglickom jazyku, konkrétne **H**yper **T**ext **M**arkup **L**anguage.

Ako štandard schválené 22. septembra 1995 s príchodom HTML verzie 3.0. Spolu s verziou 3.0 vznikla organizácia W3C - World Wide Web Consortium, ktorá udržiava špecifikáciu jazyka.

Jazyk bol odvodený od SGML jazyka, jeho hlavným účelom bola prezentácia informácií. Ako už názov jazyka napovedá (Markup - značka), jazyk pozostáva zo sady značiek (tagov) a pravidiel, pomocou ktorých sa text potrebný pre prezentáciu informácií upraví do potrebnej formy. Každú zobrazovanú stránku tvorí jeden dokument. Ak web pozostáva z viacerých podstránok, je potrebné pre každú vytvoriť samostatný zdrojový dokument.

Každý dokument musí na svojom začiatku obsahovať deklaráciu, o aký typ dokumentu sa jedná. V prípade zdrojového dokumentu obsahujúceho stránku vytvorenú pomocou HTML vo verzii 5, dokument obsahuje nasledovnú hlavičku `<!DOCTYPE HTML>`. Všetky prvky, ktoré sú súčasťou štandardu, pomocou ktorých je dokument formátovaný do výslednej polohy je potrebné korektne zapísať pomocou začiatočnej a ukončovacej značky. Značky sú zapisované vo formáte `<značka>` pre začiatočný tag a `</značka>` pre ukončujúci tag. Komentáre sú výnimkou z tohto pravidla. Komentár má formu jedného tagu v tvare `<!--text komentáru-->`.

CSS

CSS je štandardizovaný jazyk pre tvorbu kaskádových štýlov, pre rozšírené možnosti úpravy vizuálu webovej stránky. Podporuje možnosti úpravy rôznych značkovacích jazykov, predovšetkým sa využíva v spojení s HTML.

Skratka CSS je vytvorená zo začiatcných písmen originálneho názvu **C**ascading **S**tyle **S**heets. Vznik CSS sa datuje od roku 1996, kedy vznikla jeho prvá verzia označovaná ako Level 1. Aktuálne sa využíva verzia označená ako Level 3.

Pomocou CSS je možné vytvárať štýlové predpisi pre celé weby pomocou jedného externého súboru, taktiež je možné CSS definovať v hlavičke dokumentu HTML alebo ho zahrnúť priamo ako atribút konkrétneho tagu. Prvé dva spôsoby umožnujú jednoduho upravovať veľkú časť obsahu bez nutnosti upravovať jednotlivé prvky. Pre tvorbu pravidla konkrétneho prvoku alebo skupinu prvkov je potrebné zvoliť vhodný **selector**. Selector slúži k navigácii (výberu) prvku stránky, ku ktorému sa viaže štýlový predpis (pravidlo) pri vykreslení stránky.

PHP

PHP je open-source skriptovací programovací jazyk, označovaný aj Hypertext Preprocessor. PHP vytvoril Rasmus Lerdorf, pôvodne sa jednalo o sadu skiptov vytvorených pre jazyk Perl, neskôr pre jazyk C. Od vzniku bola táto sada skriptov určená k tvorbe webových

stránok, podľa tohto niesla názov Personal Home Page Tools. Z tohto názvu bola odvodená skratka PHP. V neskorších verziách pribudla podpora objektov.

V súčasnosti[3] existuje PHP v stabilnej vezii 7.4.6. Od verzie PHP3 jazyk beží na vlastnom jadre a od verzie PHP5 pribudla podpora objektovo orientovaného programovania. Vďaka vlastnému jadru je PHP nevávislé na použitej platforme. Pomocou PHP je tak možné vytvárať komplexné aplikácie s užívateľským rozhraním. Dynamicky dokáže nielen generovať výsledný HTML kód, ale taktiež dokáže aj spolupracovať s rôznymi relačnými databázami.

Vďaka svojej vysokej modularite vznikajú na základe PHP rôzne frameworky, aplikačné rámce, ktoré zjednodušujú vývoj webových aplikácií. Aktuálne je možné pre vývoj webovej aplikácie vyberať z veľkého množstva frameworkov, preto im je venovaná jedna z následujúcich kapitol. [6]

MySQL

Databáza má formu tabuliek, ktoré v sebe zlučujú dáta rovnakého významu. Každá tabuľka sa skladá z buniek usporiadaných do stĺpcov a riadkov. V jednotlivých stĺpcoch sú uchovávané dáta rovnakého typu. Jednotlivé riadky následne reflektujú jednotlivé záznamy štruktúrovaných dát. Aby bolo možné jednotlivé záznamy od seba spoľahlivo rozlíšiť, využíva sa v databázach primárny kľúč. Primárny kľúč sa aplikuje na jeden alebo kombináciu stĺpcov, v ktorých sa nachádza alebo čoho kombináciou vznikajú jedinečné hodnoty, ktoré sú vhodné pre jednoznačnú identifikáciu jednotlivých záznamov. Pre každú tabuľku musí existovať iba jeden primárny kľúč a taktiež primárny kľúč nikdy nesmie nadobúdať hodnoty NULL.

MySQL je open-source relačný databázový systém. Podporuje viacvláknový beh systému ako aj viacužívateľský prístup. Pri práci s MySQL databázou sa využíva jazyk SQL[4], pomocou ktorého sa zapisujú požiadavky, na základe ktorých systém riadenia databázy vykoná potrebné akcie, napríklad získanie alebo vloženie dát do databázy.

Medzi výhody MySQL databázového systému patrí široká podpora rôznych platforiem, ako aj implementácia do rôznych programovacích jazykov. Tiež existuje množstvo knižníc, ktoré umožňujú pracovať s databázou pohodlnejšie. Ako aj rôzne riešenia grafických rozhraní, ktoré dokážu s databázou pracovať bez nutnosti využívať textového rozhrania konzoly/príkazového riadku.[3]

3.5 Frameworky

Framework je pojem označujúci aplikačný rámec, ktorý slúži ako podpora pri vývoji softwaru. Jedná sa o softwarovú štruktúru, ktorá obsahuje podporné nástroje vo forme programov, knižníc alebo API. Pri využívaní frameworkov je potrebné dodržiavať návrhové vzory, ktoré jednotlivé frameworky vyžadujú pri ich používaní a pomáhajú pri vytváraní finálnej aplikácie.

V súčasnosti je na výber z veľkého množsta frameworkov pre vývoj webovej aplikácie. Záleží len na vývojárovi, aký návrhový vzor, architektúru, chce použiť a v akom programovacom jazyku chce aplikáciu vytvoriť. Podmienkou pri výbere môže byť aj typ databázy, platforma a mnohé ďalšie detaily. Mnohé moderné frameworky sú natoľko flexibilné, že nie je problém pokryť pri určitom jazyku a architektúre široké spektrum zvyšných požiadaviek.

[3] Informácia platná k 25.5.2020

[4] SQL - z angl. Structured Query Language, alebo štruktúrovaný dopytovací jazyk

Pre webové aplikácie, ktoré využívajú architektúru MVC (Model-View-Controler), sa jedná o 3-vrstvú architektúru, v kombinácii s jazykom PHP, je možné vyberať stále z veľkého počtu. V súčasnosti je svetovo najpoužívanejším frameworkom Laravel avšak v Česku a na Slovensku sa vysokej popularite stále teší framework Nette. Bližšie sú tieto frameworky porovnané v následujúcej kapitole.

Laravel

Laravel je open-source framework určený pre jazyk PHP. Využíva architektúru MVC a kladie dôraz na jednoduchosť kódu a ľahkú čitateľnosť. Taktiež umožňuje efektívnu prácu s databázami, rozširujúcimi knižnicami ako aj možnosť jednoducho nastavovať cesty. Pre správnu funkčnosť vyžaduje framework používanie nástroja Composer. Laravel je navrhnutý s dôrazom na efektívnosť ako aj urýchlenie vývoja webových aplikácií. [1]

Nette

Nette je rovnako ako Laravel open-source framework určený pre jazyk PHP. Taktiež využíva architektúru MVC a je v mnohom podobný frameworku Laravel. V prípade Nette je však kladený aj veľký dôraz na bezpečnosť výslednej aplikácie, preto v sebe zahŕňa aj niekoľko nástrojov, ktoré automaticky pri generovaný výsledného kódu ošetrujú najbežnejšie bezpečnostné riziká, ako aj užitočné nástroje pre vývoj a ladenie kódu.

Celosvetovo sa nejedná o najrozšírenejší framework, no lokálne v Česku a na Slovensku má vysoké zastúpenie a aktívnu česko-slovenskú komunitu. Prispieva k tomu aj fakt, že tento framework má svoje korene práve v Česku. Z toho vyplýva, že mnoho lokálnych úspešných firiem využíva tento framework pre svoju prezentáciu a poskytovanie služieb na internete. Preto bol tento framework zvolený aj pre vývoj tejto aplikácie. Nasledujúca kapitola je preto venovaná frameworku Nette a jednotlivým jeho komponentom.[5]

3.6 Nette

V tejto kapitole bude hlbšie popísaný framework Nette, ktorý bol zvolný pre implemntáciu tejto bakalárskej práce. Bližšie sa bude táto kapitola zaoberať jeho nástrojmi ako aj výhodami a možnými rozšíreniami.[5]

3.6.1 MVC - architektúra

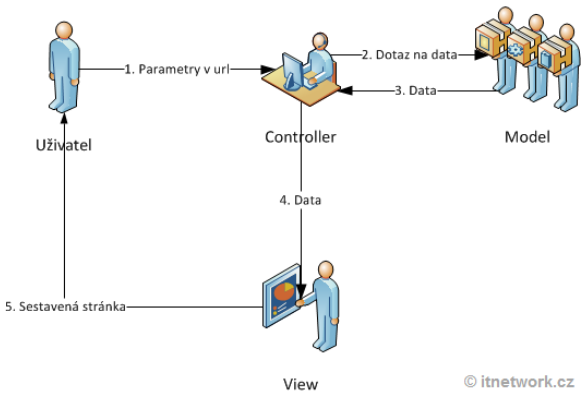
Nette je založené na trojvrstvej architektúre MVC. Cieľom tejto architektúry je oddeliť jednotlivé vrstvy aplikácie a rozdeliť ich na logické celky podľa funkcionality. Oddeľuje sa tak grafické rozhranie (view) od časti obsluhy (controler) a aplikačnej logiky (model).[7]

MVC - Model

Základom každej aplikácie architektúry MVC je Model. Model vykonáva akcie potrebné pre beh aplikácie, stará sa o výpočty, ale aj o prácu s dátami a o komunikáciu s databázou. Model nevie o existencii ostatných častí aplikácie, rovnako ako ostatné časti nevedia o Modely.

MVC - View

View je časť aplikácie, s ktorou prichácha do kontaktu užívateľ. View sa stará o zobrazenie dát užívateľom pomocou grafického rozhrania. Dáta, ktoré zobrazuje sú výsledkom práce Modelu, ktoré sú mu poskytnuté. Ak si dáta získa sám, je tým porušená architektúra MVC a ohrozená bezpečnosť celej aplikácie. Bežne sa dáta zobrazujú pomocou vopred pripravných šablón, rovnako ako je to v prípade Nette.



Obr. 3.3: Archytektúra MVC

MVC - Controler

Z vyššie popísaných View a Model vyplýva, že pre ich činnosť je potrebný prvok, ktorý sprostredkuje komunikáciu medzi zvyškom aplikácie. Controler, ako už jeho anglický názov napovedá má za úlohu riadenie. Riadi tok požiadaviek a odpovedí v aplikácii a stará sa, aby im jednotlivé prvky rozumeli.

Framework Nette popisuje svoju architektúru ako MVP. Je to z dôvodu, že v tomto frameworku je controler označovaný ako *Presenter*, bližšie je popisovaný v následujúcej kapitole.

Presenter

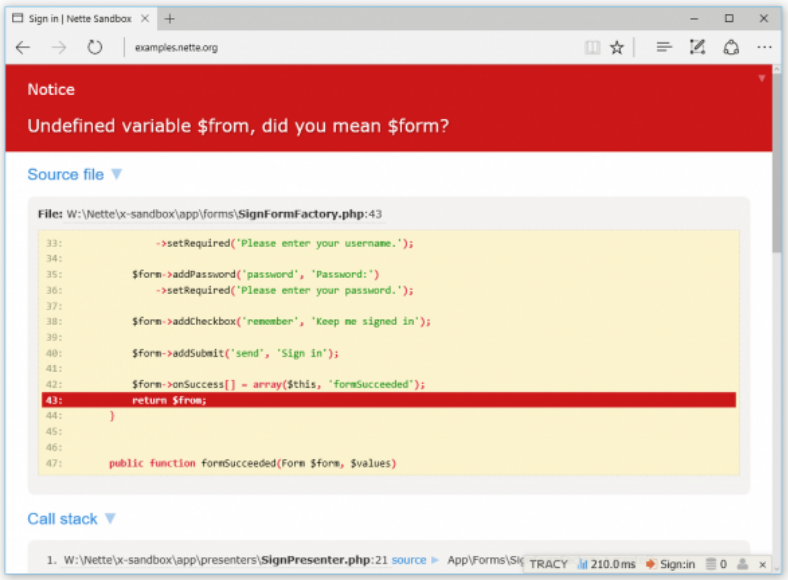
Presenter je objekt frameworku Nette, ktorý je potomkom triedy `Nette Aplication UI Presenter`. Jeho úlohou je spracovať prichádzajúce HTTP požiadavky, ktoré mu boli preložené routrom. Na základe prijatých požiadaviek generuje odpoveď pomocou sledu logických akcií a následne ju vykresluje za pomoci šablóny. Odpoveď môže byť v podobe HTML kódu pre vykreslenie stránky, no môže sa jednať aj o iné typy dokumentov.

Router

V predchádzajúcej kapitole bol spomínaný router, ktorý sa stará o preklad požiadaviek HTTP. Router sa stará o obojsmerný preklad požiadaviek a akcií. Prekladá tak HTTP požiadavky zapísané pomocou URL na akcie, ktoré je potrebné vykonať presenterom. V opačnom prípade generuje URL, ktoré sa pri konkrétnej akcii zobrazí užívateľovi v prehliadači.

3.6.2 Tracy

Tracy je nástroj vyvinutý vývojármi Nette, ktorý slúži na pohodlné a rýchle ladenie kódu. Pomáha tak pri odhalovaní a opravách chýb, ktoré programátor v kóde vytvoril.



Obr. 3.4: Nástroj tracy, príprad zachytenia chyby

Veľmi užitočná je aj funkcia výpisu chyby s detailným popisom a s možnosťou nechať danú chybu priamo vyhľadať na internete.

3.6.3 Latte

Pre tvorbu zobrazovacích šablón sa využíva v prípade frameworku Nette šablónovací systém Latte. Latte je systém určený pre PHP a poskytuje podporu pri vývoji aplikácie. Rieši do veľkej miery jej bezpečnosť a zraniteľnosť výstupu. Hlavnou úlohou je preklad vytvorených šablón na optimalizovaný kód. Pri tomto preklade dochádza k vyhodnocovaniu bezpečnostných rizík a ich zamedzeniu. Deje sa tak napríklad zmenou špeciálnych znakov, ktoré majú rôzny význam v jazykoch použitých pri tvorbe aplikácie, čím môže dôjsť k úprave kódu alebo požiadaviek. [4]

Šablóny v Latte sa zapisujú formou zmiešaného HTML a PHP zápisu, často doplneného o JavaScript.

3.7 Composer

Nástroj určený pre správu závislostí a balíčkov v rozsiahlejších projektoch v jazyku PHP. Do tejto kategórie spadajú sa PHP frameworky, ktoré v sebe obsahujú veľké množstvo balíčkov a podporných nástrojov. Tieto balíčky však majú na sebe určité závislosti a je potrebné ich dodržať, inak nebudú správne alebo vôbec pracovať. Pre tento účel vznikajú rôzny správcovia závislostí v projekte. Jedným z nich je aj Composer.

Jeho úlohou je udržať všetky požiadavky a závislosti v projekte počas jeho inštalácie ako aj počas dopĺňania dodatočných balíčkov. Ovláda sa z príkazového riadku, pomocou ktorého

mu je predaný názov požadovaného balíčku a ak je tento balíček dostupný, automaticky ho doinštaluje do projektu.

Dôležitou súčasťou Composera je jeho konfiguračný súbor, ktorý je umiestnený v koreňovom priečinku projektu. Jedná sa o súbor `composer.json`, v ktorom sú uložené záznamy popisujúce balíčky a závislosti v projekte.

3.8 JavaScript

Skriptovací objektovo orientovaný jazyk určený predovšetkým pre beh v prostredí webového prehliadača. Z toho vyplýva, že beží na strane klienta.

Pomocou JavaScriptu je tak možné webovej stránke pridať funkcionalitu, ktorú nie je možné, alebo veľmi zložité implementovať na servery. Jedná sa predovšetkým o úpravu zobrazenia výslednej stránky. Je tak možné u klienta ovplivniť spôsob vykreslenia stránky, pridať dynamické elementy alebo zabezpečiť podporu pre asynchrónne prekreslovanie dát pomocou AJAXu.

3.8.1 AJAX

Skratkou Ajax sa označuje súhrn aplikácií a technológií používaných pri vývoji webových aplikácií. Nosnými prvkami Ajaxu sú JavaScript v spojení s XML. Taktiež sa využívajú technológie DOM, CSS, HTML a mnohé ďalšie využívané pri tvorbe vybových aplikácií. Pomocou Ajaxu je možné asynchrónne komunikovať medzi klientom a zvyškom aplikácie.

Vďaka tomu je možné vykreslovať nové dáta aj v tom, ako už bola načítaná stránka a zobrazená užívateľovi. Deje sa tak zmenou určitej časti kódu výslednej stránky a nie je potrebné preto znovu načítať a vykreslovať celé zobrazenie.

Kapitola 4

Analýza požiadaviek

Pri vývoji aplikácie je potrebné dopredu vedieť, akú funkcionalitu má výsledná aplikácia spĺňať a aké požiadavky sú na ňu kladené. Čím kvalitnejšie je zadanie, tým kvalinejší môže byť aj finálny produkt.

Na základe zadania prebieha analýza požiadaviek. Je to jedna z kritických častí vývoja aplikácie, preto ju nesmie vývojár podceniť. Z výsledkov analýzy pramenia následujúce kroky podniknuté pri vývoji aplikácie.

V prípade aplikácie, ktorou sa zaoberá táto bakalárska práca je webový informačný systém, určený pre monitoring výrobných procesov. Výsledný systém má za cieľ zjednodušiť prácu programátorov a technikov pri ladení výrobného procesu, servisným technikom poskytnúť možnosť kontroly stavu zariadenia a vedeniu výroby pomôcť pri analýze efektivity. Predpokladom je, že s aplikáciou budú pracovať aj menej skúsení užívatelia počítačov, preto je potrebné zabezpečiť, aby bolo rozhranie aplikácie užívateľsky prívetivé.

4.1 Požiadavky

V tejto kapitole budú zhrnuté základné požiadavky zhromaždené od zamestnancov a technického manažéra firmy Rossum Integration.

Užívatelia:

- administrátor,
- zamestnanec firmy Rossum (technik, PLC programátor, ...),
- zákazník, zamestnanec zákazníka.

Požiadavky na funkcionalitu:

- správa užívateľov a ich práv,
- správa komunikačných modulov pripojených k aplikácii,
- zobrazenie stavu linky v reálnom čase,
- zber historických dát a ich uchovanie v databáze,
- možnosť sledovať počítadlá cyklov, spotrebu energie,
- možnosť monitorovať chybové hlášky.

4.2 Správa užívateľov a užívateľských práv

Do aplikácie smú mať prístup len prihlásení užívatelia. V systéme vystupujú užívatelia s troma typmi práv: administrátor, zamestnanec a zákazník.

Administrátor má v systéme najväčšie práva. Môže vytvárať, editovať a odstraňovať užívateľské účty. V rámci editácie môže zmeniť práva užívateľom. Užívatelia s právami zamestnanca a zákazníka môžu v nastaveniach svojho profilu zmeniť len vlastné prihlasovacie údaje a kontaktné údaje. Keďže sa jedná o typ aplikácie, ktorá je určená len vybraným užívateľom, ich registráciu a pridanie do systému zabezpečuje užívateľ s právami adminstrátora.

Podľa stupňa užívateľských práv sa budú líšiť aj práva v iných čatiach aplikácie. Bližšie budú tieto práva prebraté v kapitole

4.3 Správa komunikačných modulov

Keďže aplikácia má za úlohu monitorovať výrobné procesy, potrebuje k tomu komunikačné moduly. Pred samotným použitím je však potrebné jednotlivé moduly v aplikácii zaregistrovať, pričom je potrebné vyplniť potrebné údaje pre komunikáciu so serverom a nastaviť konfiguráciu pre získavanie dát.

Túto možnosť má opäť iba užívateľ s právami administrátora. Taktiež môže upravovať a odstraňovať jednotlivé zariadenia. Ostatný užívatelia môžu iba zobrazovať dané zariadenia.

4.4 Monitorig v reálnom čase

Medzi hlavné požiadavky na aplikáciu patrí možnosť zobrazovať dáta v reálnom čase alebo obnovovať zobrazenie v krátkych časový intervaloch.

Vďaka tejto funkcii tak buďe možné sledovať aktuálny priebeh výrobného procesu. Jednotlivé hodnoty sledovaných veličín a signálov sa majú zobrazovať formou grafu priebehu hodnoty v čase. Tým sa zabezpečí možnosť odsledovať náväznosť udalostí bez nutnosti sledovať trend súčasných udalostí pomocou historických záznamov, keďže súčasťou zobrazenia je aj krátke časové obdobie minulosti.

Monitoring v reálnom čase môžu využívať všetci prihlásení užívatelia, s obmedzením zobrazených dát pre zákazníka. Ten môže monitorovať len vlastné zariadenia. Administrátor a technik majú prístup ku všetkým zariadenam.

4.5 Historické záznamy

Rovnako ako monitoring v reálnom čase, tak aj historické záznamy sú veľmi dôležitou požiadavkou, ktorú má aplikácia spĺňať.

Užívateľ si po nastavení počiatočného a koncového času môže zobraziť priebeh jednotlivých hodnôt opäť formou grafu. Počiatočný a koncový čas je potrebné nastavovať jednoduchým a intuitývnym spôsobom. Taktiež je potrebné zamedziť užívateľovi v zadaní nekorektných časových údajov.

S historickými záznamami môžu opäť pracovať všetci prihlásení užívatelia, s obmedzením u zákazníka, ktorý môže zobraziť len dáta patriace k jeho zariadeniam.

4.6 Diagram prípadov použitia

Pomocou diagramu prípadov použitia (Use Case Diagram) sa modeluje výsledný systém tak, ako ho bude vnímať užívateľ pri práci s ním. Reflektuje sa tak poskytovaná funkcionalita pre jednotlivých užívateľov.

Pri vytváraní tohto diagramu sa využíva analýza špecifikovaných podiadaviek. Cieľom je vytvoriť diagram, ktorý bude popisovať systém a čo sa od neho očakáva. Popisuje, čo má výsledný systém obsahovať. Nešpecifikuje však, akým spôsobom to bude docielené.

Diagram prípadov použitia je zapisovaný pomocou modelovacieho jazyka UML (Jednotný modelovací jazyk) a tvoria tri základné prvky. Tieto prvky sú: užívateľ, vystupujíci v určitej úlohe, spôsob použitia systému a vzťah medzi týmito prvkami. Užívateľ je modelovaný ako jednoduchý nákres osoby, spôsob použitia je vložený do modelu formou ohraničeného textu a vzťah je modelovný ako šípka smerujúca od užívateľa smerom k činnosti, ktorú môže vykonať.

Nasledujúce kapitoly budú bližšie popisovať užívateľa podľa jeho práv na vykonávanie činností. Jednotlivé nadradené úlohy užívateľa v systéme rozširujú možnosti podradenej úlohy užívateľa, preto je táto skutočnosť modelovaná ako dedičnosť.

4.7 Neprihlásený užívateľ

Neprihlásený užívateľ nemá v prípade vytváraného systému takmer žiadne možnosti, ako môže s daným systémom pracovať, keďže je systém prioritne určený pre prácu prihláseného užívateľa.

Neprihlásený užívateľ sa preto môže iba prihlásiť do systému. Prihlásenie je však možné len po udelení prístupových údajov, prístupového práva. Tento prípad je zobrazený v obrázku

4.8 Užívateľ - Zákazník

Základná úloha užívateľa v systéme. Užívateľ s týmto stupňom opravávnení môže zobraziť všetky zariadenia, ktoré sú v jeho prevádzke inštalované. Tento užívateľ tiež môže prehliadať historické dáta z týchto zariadení a monitorovať tieto zariadenia v reálnom čase. Taktiež má možnosť zmeniť prihlasovacie údaje (login a heslo) a upraviť informácie o svojej osobe v užívateľskom profile.

4.9 Užívateľ - Zamestnanec/Technik

Táto úloha dedí všetky vlastnosti úlohy užívateť - zákazník. V tejto úlohe vystupujú zamestnanci dodávateľskej firmy, ktorá osadila u zákazníka jeho zariadenia. Tento užívateľ môže preto narozdiel od zákazníka pracovať so všetkými zariadeniami, ktoré sa nachádzajú v systéme. Môže pracovať s historickými záznamami, rovnako ako aj s monitorovaním v reálnom čase.

4.10 Užívateľ - Administrátor

Úloha administrátora je z pohľadu hierarche užívateľov na najvyššom stupni. Administrátor dedí možnosti na prácu so systémom od úlohy Zamestnanec/Technik a rozširuje ju o ďalšie možnosti.

Takýto užívateľ môže nielen zobrazovať všetky zariadenia, ale môže ich aj priraďovať ku konkrétnym zákazníkom/zákazkam. Taktiež môže vytvárať a odstraňovať užívateteľské účty, meniť ich úrovne a spravovať zákazníkov/zákazky, ku ktorým sú priradené zariadenia. Tiež môže spravovať jednotlivé zariadenia a ich historické dáta.

Kapitola 5

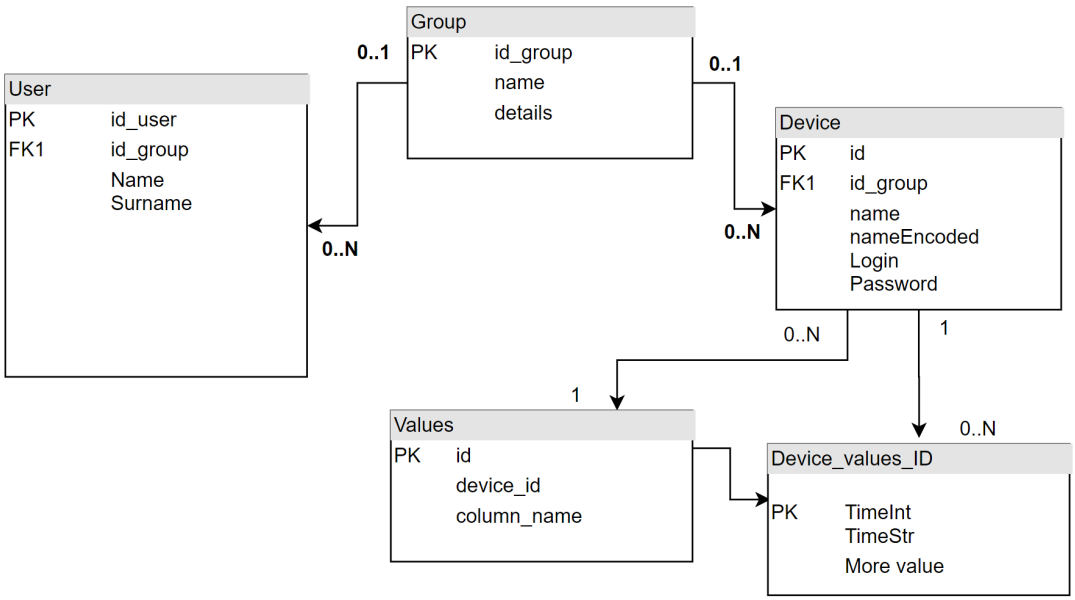
Návrh informačného systému

Pri dátovom návrhu aplikácie bol použitý ER-diagram. Tento typ diagramu sa vytvára na základe predchádzajúcej analýzy požiadaviek a poznatkov zistených vytvorením diagramu prípadov použitia. Vytvorený relačný model je potom možne transformovať na relačnú databázu a pristúpiť k návrhu užívateľského rozhrania.

Jednotlivé fázy tohto procesu sú bližšie popísané v nasledujúcich kapitolách.

5.1 ER-diagram

Pomocou ER diagramu (Entity-Relationship Diagram) sú znázornené dáta a ich vzájomné vzťahy vo výslednom informačnom systéme. ER-diagram je konceptuálnym modelom a pri jeho vytváraní sa rovnako ako pri Use Case Diagrame využíva jazyk UML. Výsledný model obsahuje opäť niekoľko základných prvkov:



Obr. 5.1: Er–diagram

- **Atribúty** – zabezpečuje uchovanie a správu dát. Najčastejšie sa jedná o databázu, môže existovať aj v podobe súborového systému alebo inej služby
- **Entita** – objekt reálneho sveta, o ktorom je potrebné uchovať informácie v databáze
- **Entitná množina** – množina obsahujúca entity s rovnakými vlastnosťami
- **Atribút** – zachytáva konkrétnu vlastnosť objektu, entity
- **Vzťahy** – popisujú vzájomné väzby medzi objektmi
- **Vzťahové množiny** - množina vzťahov rovnakého typu

Pri návrhu systému tejto bakalárskej práce bol vytvorený ER-diagram, ktorý je umiestnený na nasledujúcej stránke Popis jednotlivých entít sa nachádza v nasledujúcich kapitolách. Kurzívou sú uvádzané jednotlivé atribúty daných entitných množín.

5.2 Device – Zariadenie

Enititná množina zariadenie popisuje aké vlastnosti bude aplikácia uchovávať o jednotlivých zariadeniach, komunikačných moduloch. O zariadení je potrebné uchovávať informácie ako jeho názov zariedenia, ale aj prihlasovacie údaje (*login*, *password*) potrebné pre komunikáciu s daným zariadením.

Každé zariadenie vystupuje v systéme pod unikátnym číslom *id*, ktoré je jeho indentifikátorom aj z pohľadu výrobcu daného zariadenia a je elektronickou obdobou výrobného čísla daného zariadenia. Entitná množina taktiež obsahuje názov zariadenia v encoded tvare *encodedName*, ktorý je potrebný dodržať pri komunikácii. Odpadá tak nutnosť prekladať meno *name* pri každej požiadavke o zaslanie dát. Taktiež je vďaka tomu možné upraviť názov zariadenia, ktorý sa bežne zobrazuje v systéme a neznemožní sa tak ďalšia komunikácia.

Ako posledná vlastnosť ktorú môžeme uchovávať, je priradenie zariadenia k zákazníkovi/ zákazke. Táto vlastnosť sa zapisuje pomocou cudzieho kľuča odkazujúceho na identifikátor skupiny zariadení *group_id* v tabuľke skupina. Každé zariadenie môže byť priradené len do jednej skupiny podľa zákazníka alebo môže byť nepriradené.

5.3 Group – Skupina

Entitná množina skupina slúži k uchovaniu údajov o skupinách zariadení. Ako už bolo v predchádzajúcich kapitolách spomínané, jednotlivé zariadenia je možné zoskupovať ku konkrétnym zákazníkom. Skupina nesie v sebe informácie ako jedinečný identifikátor *id*, názov *name* a prípadné podrobnosti *detail*.

5.4 User – Užívateľ

Entitná množina užívateľ slúži na uchovanie údajov o užívateľovi. Pri užívateľoch je potrebné uchovávať základné informácie o užívateľovi, ako meno a priezvisko (*name*, *surename*), kontaktné údaje, email a telefónne číslo (*email*, *tel*).

Taktiež je potrebné uchovávať aj informácie potrebné pre samotnú prácu so systémom, preto táto entita musí obsahovať jednoznačný identifikátor *id*, prihlasovacie meno do systému *login*, heslo *password*, pridelené práva *credentials* a ak sa jedná o užívateľa v úlohe

zákazník, je potrebné mať možnosť uchovať informáciu, ku ktorej skupine zariadení môže pristupovať pomocou odkazu na identifikátor skupiny *group_id*.

5.5 Values – Hodnoty

Táto entitná množina slúži na mapovanie sledovaných veličin pomocou jednotlivých zariadení. Keďže sa pri každom zariadení monitorované veličiny líša, je potrebné uchovať ich názvy pre možnosť ďalšej práce s ich hodnotami. Táto entita mapuje názov hodnoty *value_name* k identifikátoru zariadenia *device_id*. Jednotlivé dvojice následne identifikujeme pomocou prideleného identifikačného čísla *id*.

5.6 Device Values – Hodnoty zariadenia

Táto entitná množina je slabou entitnou množinou. Jej existencia je podmienená existenciou záznamu pre dané zariadenie, ku ktorému táto množina prislúcha.

Nemá pevne stanovenú štruktúru a počet atribútov. Tie sa líša v závislosti od zariadenia. Každá takáto entita obsahuje iba dva pevne stanovené atribúty *TimeInt* a *TimeStr*. Jedná sa reprezentáciu časového razítka v dvoch formátoch. V prípade *TimeInt* sa jedná o reprezentáciu časového razítka vo formáte Unix Epoch, ktorý má hodnotu sekvečného čísla a je ho možné využit ako jednoznačný identifikátor záznamu.

5.7 Relačná databáza

Výsledná relačná databáza systému vznikne transformáciou ER–diagramu. Pri transformácii sa prevádzajú jednotlivé entity do podoby tabuliek relačnej databázy. Každá tabuľka musí obsahovať atribút, ktorý vystupuje v úlohe primárneho kľuča.

Jednotlivé vzťahy mezdi entitami sa prevedú do tabuliek formou pridania cudzích kľúčov podľa toho, akú kardinalitu má daný vzťah. Pri vzťahu 1:1 sa aplikuje cudzí kľuč do jednej z tabuliek, ktoré sú viazané týmto vzťahom. V tejto bakalárskej práci sa vzťah vyskytuje iba v jednom prípade, nejedná sa však o bežný vzťah medzi dvoma silnými entitnými množinami. Jedná sa o vzťah silnej a slabej entitnej množiny. V tomto prípade sa presúva cudzi kľúč do deskriptoru danej slabej entitnej množiny, čo sa v relačnej databáze premietne presunom do názvu tabuľky.

Pri vzťahoch 1:N sa pridáva cudzí kľúč do tabuľky, ktorej kardinalita má hodnotu N. Tento typ vzťahov sa nachádza vo zvyšku implementovanej databázy. Posledným prípadom je vzťah, kde je kardinalita N:N, tento typ sa v ER-diagrame tejto aplikácie nevyskytuje. Ak by sa nachál, bolo by potrebné pri transformácii vytvoriť prepojovaciu tabuľku, ktorá by obsahovala páry cudzích kľúčov.

5.8 Užívateľské rozhranie

Užívateľské rozhranie je prvok, s ktorým prichádza užívateľ pri práci s aplikáciou do kontaktu. Preto pri jej návrhu je potrebné myslieť na túto skutočnosť a snažiť sa o vytvorenie užívateľsky prívetivého prostredia.

V prípade aplikácie, ktorá vznikla ako súčasť tejto bakalárskej práce bol zvolený jednoduchý desing aplikácie. Inšpiroval sa do veľkej miery dizajnovým smerom, ktorý aplikuje štýl Material Design, vytvorený spoločnosťou Google Inc..

Design aplikácie preto využíva farby, ktoré nie su rušivé. Plochy používajú jednofarebné výplne bez prechodov, tieňov a iných štýlovacích efektov.

Hlavná obrazovka

Hlavná obrazovka aplikácie je navrhnutá spôsobom, ktorý iné aplikácie nazývajú aj ako Dashboard - prístojový panel. Jedná sa o zobrazenie, kedy sú jednotlivé položky zobrazované formou dlaždíc. Jednotlivé prvky majú tvar štvorcov alebo obdĺžnikov. Obdobný štýl využíva aj MetroUI, ktoré obsahovali operačné systémy Microsoft Windows 8 a 8.1 . Podľa typu úlohy prihláseného užívateľa sa na tejto stránke zobrazuje sekcia Nepriradené zariadenia. Jednotlivé dlaždice zastupujú jednotlivé zariadenia. Farba dlaždice charakterizuje status pripojenia zariadenia k aplikácii. Po kliknutí na dlaždicu konkrétneho zariadenia je užívateľ presmerovaný na ďalšiu stránku, zobrazujúcu dáta v reálnom čase.

Obrazovka vykreslovania dát

V ľavej časti obrazovky sa nachádza zoznam aktuálne monitorovaných veličín a tlačidlo pre návrat späť na hlavnú obrazovku. Vo zvyšnej časti sa zobrazujú grafy priebehu monitorovaných veličín. V pravej hornej časti zobrazovacieho poľa nad prvým zobrazeným grafom sa nachádza tlačidlo, ktorým je možné prepnúť zobrazenie z aktuálneho na historické.

Pri historických záznamoch je vyhľad stránky obdobný ako pri monitoringu v reálnom čase. Hlavnou zmenou je pridanie Datepickrov, kalendárov, pomocou ktorý užívateľ vyberie požadovaný časový úsek, ktorého dáta sa majú zobraziť.

Kapitola 6

Implementácia

Táto kapitola popisuje implementáciu samotného informačného systému, ako aj implementáciu logiky, ktorá zabezpečuje zber dát z monitorovaných zariadení.

Ako prvej časti celku výslednej aplikácie sa bude táto kapitola zaoberat zberom dát, v druhej časti sa bude venovať informačnému systému, ktorý tieto dáta využíva pri svojej činnosti.

6.1 Zber dát

Zber dát zo zariadení prebieha pri tejto aplikácii pomocou komunikačných modulov od značky Ewon. Podmiekou zberu a odosielania dát do aplikácie je nutné pripojenie daného zariadenia do siete Internet. Taktiež je potrebné, aby monitorované zariadenie bolo podporované a podporovalo niektorý z komunikačných protokolov monitorovacieho zariadenia.

Pre účely tejto bakalárskej práce som mal zapožičaný komunikačný modulo Ewon Flexy 205 rozšírený o wifi kartu. Ako monitorované zariadenie slúžilo zapožičanie PLC značky Siemens, model S7-1200. Toto PLC bolo spojené so zariadením Ewon pomocou Profinetu. Profinet je priemyselným ethernetovým komunikačným protokolom, podľa toho bude ovplyvnená aj voľba nastavení pri konfigurácii. V PLC bol nahratý jednoduchý program, ktorý bežal v slučke a upravoval hodnoty niekoľkých premenných.

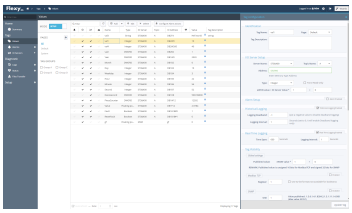
Konfigurácia zariadenia Ewon

Základná konfigurácia zariadenia Ewon nevyžaduje žiaden špecialny software, celé nastavenie sa vykonáva pomocou rozhrania vo webovom prehliadači.

Pri konfigurácii je potrebné poznať podporované štandardy zo strany PLC ako aj zariadenia Ewon. V prípade zariadení Siemens je možné zvoliť priamo v zariadení Ewon komunikačný protokol, ktorý je určený pre komunikáciu s týmito zariadeniami.

Pri implementácii bol využívaný práve tento dedikovaný protokol, ktorý umožnil komunikáciu so zariadením bez zbytočne zložitého nastavovania parametrov v PLC. Pre nastavenie bolo potrebné poznať IP adresu PLC a zabezpečiť aby boli obe zariadenia v rovnakej sieti. IP adresu je potrebné v prípade PLC Siemens doplniť o správnu hodnotu TSAP (Transport Service Access Point) prístupového bodu. Tieto hodnoty sa dajú jednoducho zistiť pomocou konfiguračného software TIA portál. Po nastavení TSAP je možné pristupovať do pamäte zariadenia PLC. Pre čítanie hodnôt je nutné v PLC povoliť možnosť pracovať s dátami v tejto pamäti externým zariadeniam. Pre dokončenie konfigurácie stačilo už len zadať čas, ako často sa majú načítať dáta z pamäte PLC.

Následne bolo možné pristúpiť k nastaveniu samotných hodnôt, ktoré majú byť sledované. Tieto sa nastavujú v menu pre správu hodnôt. Tu si užívateľ našpecifikuje, ktoré hodnoty a ako často chce monitorovať. To má však jeden háčik. Ewon síce má prístup do pamäte PLC, no je len pozorovateľom a nevie, kde a aké sa v pamäti nachádzajú hodnoty. Preto je pri konfigurácii hodnôt potrebné vedieť adresy jednotlivých hodnôt v pamäti. Úžívateľ neznalý problematiky by mohol dospieť k záveru, že stačí len nastaviť selektor a vypočítať ofset podľa doznamu hodnôt v PLC. To však nie je správne riešenie.



Obr. 6.1: Webové rozhranie pre konfiguráciu hodnôt

Aktuálne modely PLC značky Siemens majú vo východzom nastavení zapnutú optimalizovanú pamäť. To znamená, že si riadia a optimalizujú pamäť automaticky a podľa svojí potrieb. Túto možnosť je preto potrebné vypnúť v nastaveniach, následne sa v zozname premenných zobrazia absolútne adresy jednotlivých premenných v pamäti.

Tieto adresy sa následne použijú pri nastavení do jednotlivých sledovaných hodnôt. Pri nastavení dohnoty je tiež potrebné poznať dátový typ, s ktorým PLC pracuje pri danej premennej. Pre následné monitorovanie je potrebné hodnoty označiť, pretože je potrebné ich ukladať do pamäte. To sa jednoducho nastaví pri vytváraní monitorovacieho Tagu a povolenia historického logovania v určitom intervale.

Aby bolo možné zozbierané dáta zo zariadenia pomocou API výrobcu stiahnuť do aplikácie tretej strany, napríklad ako je aj táto bakalárska práca, je potrebné nastaviť zariadeniu Ewon prístup pomocou sprievodcu v menu, kde sa vyplnia potrebné údaje. Po pripojení je možné komunikovať so zariadením pomocou cloudu, ktorý sprostretkováva službu prostredníka.

Komunikačné API

Pre komunikáciu tretej strany so zariadeniami Ewon sa využíva komunikačné API, ktoré je síce voľne prístupné, no nie je možné ho využívať bez registrácie developéra. Pri každej požiadavke je potrebné udávať developéreské číslo a údaje bežného užívateľského konta. Pri niektorých volaniach, ak je potrebné dané hodnoty získať zo zariadenia Ewon, je potrebné udať aj prihlasovacie údaje k zadanému zariadeniu.

Pomocou API sa odosielajú požiadavky na cloud T2M firmy Ewon a tie sú následne zabezpečeným spojením predané zariadeniu. Zariadenie túto požiadavku spracuje a vráti odpoveď, ktorá putuje opäť cez cloud ku aplikácii, ktorá odoslala pôvodnú žiadosť. Model tejto komunikácie zobrazuje obrázok

Loggovací nástroj

Pre účely získavania dát za pomoci API bol vytvorený nástroj, pozostávajúci z dvoch skriptov v jazyku PHP. Skripty majú názov `DataLogger.php` a `logUpdate.php` Tieto skripty nevyžadú nejaké zvláštne nároky pre svoju činnosť. Je potrebné v oboch správne nastaviť

cestu k databáze a vyplniť údaje potrebné pre prihlásenie pomocou API. Skripty pre svoj beh vyžadujú PHP vo verzii 7.

Skript `DataLogger.php` je hlavný riadiaci skript, ktorý pracuje v nekonečnej slučke. Skript počas svojho behu volá opakovane `pcntl_fork()`, čím vytvára svojích potomkov, ktorých hlavnou úlohou je volať skript `logUpdate.php` s parametrom `$id`. Tento parameter obsahuje identifikačné číslo zariadenia, od ktorého majú byť získané dáta. Tým sa skracuje čas medzi volaniami jednotlivých zariadení v porovnaní so sekvenčnými volaniami.

Dáta získané pomocou oboch skriptov sú pri každej iterácii ukladané do databázy.

6.2 Informačný systém

Informačný systém je implementovaný na základe analýzy požiadavkov a návrhu informačného systému.

V nasledujúcich častiach tejto kapitoly budú popísané hlavné časti informačného systému. Taktiež tu bude spomenutá komunikácia s databázou.

6.2.1 Adresárová štruktúra

Použitá adresárová štruktúra informačného systému vychádza z odporúčanej adresárovej štruktúry webového projektu frameworku Nette, ktorá je vytvorená nástrojom composer pri vytváraní nového projektu.

V koreňovom priečinku webovej aplikácie sa nachádza 5 priečinkov a 4 súbory.

Súbory

- **.htaccess** – tento súbor obsahuje nastavenia potrebné pre prácu servera
- **composer.json** – konfiguračný súbor nástroja composer, tento súbor v sebe obsahuje zápis naištalovaných balíčkov
- **composer.lock** – konfiguračný súbor nástroja composer, tento súbor v sebe obsahuje zápis naištalovaných balíčkov a ich závislostí
- **readme.md** – automaticky vygenerpvané readme popisujúce použitie frameworku vo webovej aplikácii

app (priečinok)

V priečinku app sa nachádzajú všetky zdrojové kódy webovej aplikácie, ako aj prístupové údaje do databázy, ktoré sú súčasťou týchto súborov. Tieto súbory je preto potrebné chrániť a zamedziť prístup k nim bežnému návštevníkovi pomocou prehliadača. Toto zabezpečuje práve správne nastavenie servera.

V tomto priečinku sa nachádza súbor `bootstrap.php` a niekoľko podpriečinkov, v ktorých sa nachádzajú potrebné zdrojové kódy a konfiguračné súbory vytvoreného informačného systému.

Súbor `bootstrap.php` slúži ako spúšťací súbor frameworku.

Podpriečinky

- **config** – nachádzajú sa tu konfiguračné súbory local.neon a common.neon, ktoré obsahujú konfiguráciu projektu a prístupové údaje do databázy

- **Model** – tento priečinok obsahuje súbory, modely, zabezpečujúce komunikáciu s databázou
- **Presenters** – tento pričinok obsahuje v sebe súbory a podpriečinky, obsahujúce jednotlivé presentre a šablóny potrebné pre tvorbu užívateľskeho rozhrania
- **Router** – tento podpriečinok obsahuje súbor potrebný pre preklad URL na požiadavy, ako bolo popisované v kapitole vyššie

log (priečinok)

Do tohto priečinka sa ukladajú záznamy o výskyte chýb pri aplikácii v produkčnom režime, kedy je zakázané zobrazenie ladiaceho nástroja Tracy.

temp (priečinok)

Do tohto priečinka si aplikácia ukladá potrebné dočasné súbory do podpriečinka Cache.

vendor (priečinok)

V tomto priečinku sa nachádzajú zdrojové kódy použitých knižníc, ktoré sú súčasťou frameworku Nette. Tieto knižnice sem boli nainštalované pomocou nástroja Composer.

WWW (priečinok)

Táto zložka je ako jediná prístupná pre všetkých užívateľov pomocou webového prehliadača. V tomto priečinku sa nachádzajú dodatočné súbory informačného systému.

Tieto súbory sú prevažne podporného charakteru, preto že sa jedná o súbory typu CSS, obrázky a skripty napísané v jazyku JavaScript. Nachádza sa tu však aj veľmo potrebný súbor `index.php`. Tento súbor je potrebný pre spustenie aplikácie.

6.2.2 Práca s databázou

Prístup k dátam v databáze je nevyhnutný pre prácu informačného systému. Je potrebné zabezpečiť, aby bolo možné do databázy zapisovať nové dáta a zobrazovať existujúce.

V prípade frameworku Nette túto úlohu zabezpečujú modely. Modely (jednotlivé súbory) v prípade frameworku Nette pokrývajú vrstvu Model, ktorá je súčasťou trojvrstvej architektúry MVC.

V prípade frameworku Nette je odporúčané nevytvárať jeden veľký model, ale niektoľko menších, ktoré budú pokrývať určitú špecifickú časť práce s dátami a databázou. V prípade tejto aplikácie bol využítý tento prístup a preto je model rozdelený na niekoľko súborov v prečinku Model.

Pre vytváranie modelov a prácu s nimi obsahuje framework Nette podporu v podobe rôznych pripravených metód. Pri tvorbe modelu framework do veľkej miery zjednodušuje vytvorenie pripojenia k databáze. Taktiež pri tvorbe modelu je možné vytvoriť jednotlivé požiadavky na databázu pomocou zjednodušujúcich zápisov, ktoré sú následne preložené na SQL požiadavky. Vďaka tomu je možné vytvárať bezpečné požiadavky na databázu. Požiadavky je potrebné zapisovať mierne odlišným spôsobom ako v pripade priameho zápisu SQL, kde môže dôjst bezpečnostnému riziku, ktoré spôsobý programátor nevhodnbým použítím premennych, do ktorých môže byť následne injektovaný škodlivý kód.

Kapitola 7

Testovanie

Testovanie je súčasťou vývoja každého softwaru. Pri testovaní sa ohhaľujú nedostatky aplikácie, ktoré je možné ďalším vývojom zlepšiť. Taktiež sa pomocou testovania odhaľujú chyby, ktoré sa nesmú dostať do produkčného módu aplikácie.

Testovanie webových aplikácie väčšinou prebieha vo viacerých fázach. Naplánované bolo dvojfázové testovanie. Prvou Fázou malo byť testovanie prográtorské, druhou fázou malo byť testovanie na užívateľoch.

Druhá fáza testovania nenastala z dôvodu neúplnej implementácie do tej miery, aby mohla byť predvedená užívateľom a otestovaná.

7.1 Programátorská fáza

Programátorská fáza testovania prebieha neustále počas vývoja aplikácie. Skaždou novo pridanou funkciou je potrebné ju otestovať.

Taktiež prebehlol test budu loggovacieho skriptu, keď daný skript bežal bez chyby 24 hodín.

7.2 Užívateľska fáza

K užívateľskej fáze nedošlo z dôvodu časového sklzu a neúplnej implementácii.

Kapitola 8

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť funkčný informačný systém pre analýzu a monitorovanie dát z výrobných procesov. Dôraz mal byť kladený na jednoduchosť používania ako aj strabilitu loggovacieho nástorja.

V tejto technickej správ sú najskôr popísané samotné výrobné procesy a ich riadenie, aby bol čitateľ tejto technickej správy oboznámený s kontextom vzniku tejto aplikácie. Boli tu bližšie popísané riadiace počitače PLC, ďalej komunikačné moduly pre získavanie dát a komunikačné protokoly potrebné pre komunikáciu medzi PLC a komunikačným zariadením.

Následovala rozsiahla kapitola, ktorá obsahovala informácie o aktuálne dostupných prostriedkoch pri vývoji webových aplikácii. V tejto kapitole boli bližie špecifikované pojmy ako informačný systém a vebová aplikácia. Boli tu popísané povuživané architektúry, ich vlastosti, výhody a nevýhody. Taktiež táto kapitola obsahoje popis jednotlivých technológii ktoré boli použité pri vývoji ako aj ich alternatívy.

Ďaľšou v poradí bola kapitola obsahujíca analýzu požiadaviek na výsledný systém. Taktiež sú tu uvedené postupy, ktoré sa pri analýze požiadaviek používajú. Boli tu bližsie popísané jednotlivé diagrami, pomocou ktorých sa modelujú výsledné systémy.

V kapitole venovanej návrhu aplikácie sa popisuje postup použitý pri vývoji. Táto kapitola taktiež obsahuje dátový model výslednej aplikácie a jeho bližší popis. Taktiež tu je popísané užívateľské rozhranie.

Kapitola o implementácii nie je úplná z dôvodu neúpnosti fyzickej implementácie.

Literatúra

[1] Laravel. *Laravel Docs* [online]. [cit. 21-4-2020]. Dostupné z: https://laravel.com/docs/7.x.

[2] Leo, M. PLC a ich programovanie – 1. Čo je to PLC ? *POSTERUS* [online]. [cit. 20-4-2020]. ISSN ISSN 1338-0087. Dostupné z: http://www.posterus.sk/?p=6903&output=pdf.

[3] MySQL. *Why MySQL?* [online]. [cit. 26-4-2020]. Dostupné z: https://www.mysql.com/why-mysql/.

[4] Nette. *Latte* [online]. [cit. 21-4-2020]. Dostupné z: https://latte.nette.org/cs/guide.

[5] Nette. *Proč používat Nette?* [online]. [cit. 21-4-2020]. Dostupné z: https://doc.nette.org/cs/3.0/why-use-nette.

[6] PHP. *PHP Manual* [online]. [cit. 26-4-2020]. Dostupné z: https://www.php.net/manual/en/.

[7] Čápka, D. *MVC architektura* [online]. [cit. 25-4-2020]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

Príloha A

Obsah přiloženého paměťového média

- Zložka rossum_app obsahujúca implementáciu v sanboxovej štruktúre Nette frameworku
- Zložka datalogger obsahujúca skripty v jazyku PHP pre sťahovanie potrebných dát
- Subor localhost.sql obsahujúci export databázy
- Elektronickú podobu tohto dokumentu.