VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

PREDIKTIVNÍ ÚDRŽBA U AUTOMATIZOVANÝCH MONTÁŽNÍCH STROJŮ

PREDICTIVE MAINTENANCE ON AUTOMATED ASSEMBLY MACHINES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Vladimír Janík

AUTHOR

VEDOUCÍ PRÁCE — Ing. Václav Mecerod

SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Vladimír Janík ***ID:*** 186099
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Prediktivní údržba u automatizovaných montážních strojů

POKYNY PRO VYPRACOVÁNÍ:

Automatické montážní stroje a průmysloví roboti produkují velké množství dat, ze kterých je dalším zpracováním možno získat informace užitečné pro optimalizaci výroby. Navrhněte strojové zpracování těchto dat, které umožní zobrazení využití stroje v čase.
Vypracujte studii dostupných softwarových knihoven a frameworků, popište strukturu databáze, blokové schéma softwarového řešení, vytvořte schematický návrh uživatelského rozhraní. Zobrazovač implementujte jako webovou aplikaci, která bude data čerpat z databáze, a umožní uživateli volbu časového období, porovnání jednotlivých strojů mezi sebou, provádění statistických operací a možnost nastaveni limitních hodnot pro zvolené ukazatele.

DOPORUČENÁ LITERATURA:

[1] CUESTA, Hector. Analýza dat v praxi. Přeložil Jiří HUF. Brno: Computer Press, 2015. ISBN 9788025143612.

[2] GEMIGNANI, Zach, Chris GEMIGNANI, Richard GALENTINO a Patrick Jude SCHUERMANN. Efektivní analýza a využití dat. Přeložil Jiří HUF. Brno: Computer Press, 2015. ISBN 9788025145715.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Václav Mecerod
Konzultant: Ing. Dávid Genčanský (ALPS Electric Czech, s.r.o.)

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

Tato práce se zabývá problematikou analýzy dat získaných z automatizovaných montážních strojů a jejich rychlým a přehledným zobrazením ve formátu vhodném pro jednotlivé koncové uživatele. Součástí této práce je také srovnání webových frameworků, návrh struktury databáze a následného softwarového řešení. Data jsou načítána pomocí implementovaného modulu programovacího jazyka. Dále jsou analyzována a zobrazována uživateli prostřednictvím webové aplikace, ke které mohou mít uživatelé přístup z kteréhokoli zařízení připojeného k firemní síti.

Klíčová slova

Flask, prediktivní údržba, analýza dat, databáze, průmysloví roboti, webová aplikace

Abstract

This thesis deals with data analysis. Data obtained from automated assembly machines and their quick and well-arranged displaying in a format suitable for individual end users. In the thesis web frameworks are compared and database structure as well as final software solution is proposed. The data is loaded using the implemented programming language module. The data is further analyzed and displayed to a user through a web-based application accessible to end user from every device connected to the corporate network.

Keywords

Flask, predictive maintenance, data analysis, database, industrial robots, web application

Bibliografická citace:

JANÍK, V. *Prediktivní údržba u automatizovaných výrobních strojů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 24 s. Vedoucí práce: Ing. Václav Mecerod

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Prediktivní údržba u automatizovaných montážních strojů jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne **14. prosince 2018**

.....

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Václavu Mecerodovi a konzultantovi Ing. Dávidovi Genčanskému za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne **14. prosince 2018**

.....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Současná průmyslová výroba se vyznačuje množstvím zařízení, která pro svoji činnost potřebují informační technologie. Nejde jen o využití těchto technologií pro automatizaci výrobních procesů, ale i o sběr velkého množství dat, která jsou potřeba pro provádění analýz, řízení celého výrobního procesu a sběr dat pro potřeby zákazníků. Tím je možno vyhýbat se problémům se zpožděním dodávek, jejich zabezpečením a spolehlivostí. Stejně tak důležité je udržet výrobu v požadovaných parametrech a kvalitě. Prediktivní údržbou je možno v důsledku zabránit i škodám, které by mohly nastat nedodržením požadovaných parametrů.

Pro optimalizaci výrobních procesů je nutné získaná data analyzovat a vyjádřit je jako informace relevantní pro jednotlivé uživatele. Tyto informace umožní firmám optimalizovat výrobu dle politiky a potřeb firmy. Pokud budou vytvořeny správné datové modely, budou se v budoucnu systémy schopny samy učit a řešit jednoduché a postupně i složitější problémy. Je potřeba si uvědomit, že návratnost těchto projektů nebude okamžitá, ale ve výhledu je tato cesta jediná pro udržení konkurenceschopnosti na trhu. Jak [1] uvádí, náklady na údržbu mohou představovat 15% až 60% výsledné ceny vyrobeného zboží. Očekávaným výsledkem je také delší doba bezporuchového provozu zařízení, podstatně vyšší rychlost výroby a v neposlední řadě vyšší výnosy.

Tato práce se zabývá zpracováním dat získaných z automatizovaných výrobních strojů tak, aby měla pro odpovědné zaměstnance i pro management firmy vypovídací hodnotu a umožnila zjednodušení řízení průmyslového procesu. Pomocí průběhů a změn hodnot získaných z těchto dat bude možno predikovat potřebnou údržbu a další úkony ve vztahu k pracovním strojům.

2 ROZBOR PROSTŘEDKŮ PRO REALIZACI

Před zahájením práce na realizaci zadání je vždy potřeba zvolit si vhodný programovací jazyk, který bude umožňovat co možná nejefektivnější cestu k dosažení cílů řešeného problému. Při rozhodování o programovacím jazyku bylo potřeba vycházet z několika základních kritérií, která zásadně ovlivňují možnosti dosažení požadovaného výsledku. Jako hlavní kritéria pro výběr programovacího jazyka byla zvolena následující:

- Čitelnost kódu – Intuitivní syntaxe, která s sebou přináší přehlednost a snížení nároků na udržitelnost, jde o snahu vyhnout se pokud možno složitým, zdlouhavým a nepřehledným řešením.
- Využití knihoven – Možnost využití již existujících knihoven, které usnadňují řešení jednotlivých částí projektu.
- Používanost – Rozšířenost daného programovacího jazyka, která přináší dobrou dostupnost literatury a více diskusních odborných fór pro odstraňování možných problémů, které se mohou vyskytnout během řešení požadované úlohy.
- Multiplatformnost – Možnost použití na různých zařízeních, včetně na zařízení od různých výrobců.
- Dostupnost – Toto kritérium poskytuje možnost pracovat na počítači jak doma, tak i na počítačích na jiných místech bez nutnosti zakoupení licence (freewarové řešení).

Na základě výše uvedených kritérií, dále na základě doporučení pana konzultanta a dřívějších zkušeností s programováním jsem se rozhodl použít programovací jazyk Python.

2.1 Python

Jedná se o velice populární, freewarový, interpretovaný programovací jazyk. Byl navržen Guido van Rossumem, jeho první verze byla vydána roku 1991.

V současnosti existují dvě verze Pythonu a sice verze 2.x.x a 3.x.x, které nejsou vzájemně kompatibilní. Verze 3 vznikla pro odstranění chyb a změně některých částí návrhu, které se při používání verze 2 neosvědčily. Jistou nevýhodou je však poměrně vysoká výpočetní náročnost.
Tento programovací jazyk je dostupný pro většinu známých platforem (Microsoft Windows, macOS, Android, Raspberry Pi). Skriptovací jazyk Python si zakládá

především na čitelnosti kódu. Má obecně jednodušší syntaxi než systémové jazyky a pro tuto jednoduchost je často je i doporučován začátečníkům v programování. Jeho přirozenost tak umožňuje i začínajícím programátorům tvořit jejich projekty s menším úsilím na jejich realizaci. Existuje také obrovské množství knihoven, které umožní uživateli vyhýbat se složitým řešením, zlepší jeho čitelnost, podstatně kód zkrátí a v neposlední řadě výrazně zkrátí čas k vytvoření aplikace. Další výhodou je také jeho uživatelská rozšířenost a dostupnost kvalitních materiálů. Předpokládá se, že díky rostoucí důležitosti grafických uživatelských prostředí a neustálému rozšiřování Internetu bude stále více aplikací vytvářeno ve skriptovacích jazycích. Aplikace systémových jazyků se bude stále více omezovat na vytváření komponent. Je možno uvést, že Python jako skriptovací jazyk má asi nejblíže k jazykům systémovým.[3][4]

Obr 2.1 Logo Pythonu

2.1.1 Využití Pythonu

Python umožňuje řešení téměř jakéhokoliv problému, nejvíce aplikací se však používá zejména pro:

- Web development
- Strojové učení
- Komplexní analýzu dat
- Počítačové vidění
- Skriptování
- Web scraping

2.2 Databáze

[9] uvádí, že databáze jsou souhrny digitálních dat, která jsou uspořádána tak, aby s nimi bylo možno snadno a rychle pracovat. Slouží tak k vyhledávání potřebných údajů, k jejich ukládání do paměti, zobrazování a k dalším operacím. Daty jsou myšlena fakta, která je možno zaznamenat a která nesou určitou informaci (například jména, telefonní čísla a adresy lidí pracujících ve firmě, v průmyslové výrobě jsou to například časová známka, kód operace, název operace, délka výroby součástky a chybová hlášení atd.). Data v databázi by měla mít určitou souvislost. Každá databáze má určitý okruh uživatelů, kteří těmto datům rozumí a mohou je využívat pro své další činnosti. Z toho důvodu mají databáze a databázové systémy velký dopad na rozvoj využívání počítačů a v současnosti hrají významnou roli v každém odvětví lidské činnosti.

Podle [2] je možno databáze dělit podle různých hledisek jako:

Podle obsahu:

- Všeobecné databáze – obsahují velké množství informací, které není možno zařadit do malého množství oborů, jedná se například o různé encyklopedie a podobné soubory setříděných informací, jejichž úkolem je podávat velmi široké spektrum dat a informací pro široké spektrum uživatelů.

- Specifické databáze – obsahem těchto databází jsou soubory dat a informací, které patří pouze do určitého zaměření lidských vědomostí a činností, proto je tyto databáze možno označit jako jednooborové.

Podle formy:

- Elektronické databáze – jde o databáze uložené ve formě digitálních dat uložených na různých datových úložných médiích (datových nebo taky paměťových nosičích), která mohou být magnetická (např. pevné disky), optická (CD, DVD) nebo elektronická (USB flash, SD karty atd.).

- Analogové databáze – jejich typickými představiteli jsou papírové kartotéky.

Podle způsobu ukládání dat a vazeb mezi nimi:

- Hierarchické databáze – V těchto databázích jsou data uspořádaná do stromové struktury, kde každý prvek tvoří jeden uzel. Jednotlivé uzly mají mezi sebou vztahy nadřazené a podřazené. Nevýhodou tohoto uspořádání je složité vkládání a také mazání záznamů.

- Síťové databáze – k hierarchické struktuře připojují další již nehierarchické vztahy mezi jednotlivými prvky. To dává možnost vytvářet obousměrné vztahy mezi prvky. Přístup k záznamům je tak přímý bez nutnosti dalšího vyhledávání. Nevýhodou takové databáze je hlavně obtížná změna její struktury.

- Objektově-orientované databáze – jsou databáze, které jsou integrovány do programovacího jazyka. Informace jsou uloženy ve formě objektů, což umožňuje programátorům vyvíjet další objekty a modifikovat již existující. Využívají je společnosti Google, Amazon, Siemens apod.

- Relační databáze – jsou v současné době nejpoužívanější díky jednodušší struktuře. Data jsou pro zjednodušení zpracování uložena do tabulek. V těchto tabulkách, které jsou nazývané relace slouží řádky k uložení informací a sloupce určují datový typ. Využívají jazyka SQL a entitně-relační modelování.

Podle architektury

- Centrální architektura – data i DBMS jsou uloženy v centrálním počítači. Jde o typickou terminálovou síť, kde se údaje přenáší na terminál, ale vlastní zpracování proběhne na centrálním počítači, který může zpracovávat více úloh současně.

- Architektura file-server – je výsledkem rozšíření PC a sítí LAN. DBMS a databázové aplikace jsou provozovány na počítačích jednotlivých uživatelů, data jsou umístěna na file-serveru a mohou být sdílena.

- Architektura klient-server – je založena na síti LAN, PC a databázovém serveru. V jednotlivých PC je instalován program podporující např. vstup dat, formulaci dotazu atd. Ten se předá pomocí jazyka SQL na databázový server, který jej vykoná a vrátí výsledky na PC uživatele. Databázový server musí být dostatečně výkonný počítač, protože je v průběhu činnosti velmi zatížený. Tato architektura vyhovuje i náročným aplikacím, proto je hojně využívána databázovými firmami. Omezuje přenos dat po síti, vzhledem k tomu, že všechny dotazy jsou prováděny na databázovém serveru a po síti jsou posílány jen výsledky.

- Distribuovaná databáze – je tvořena datovými archívy. Tyto archívy mohou být fyzicky na zcela různých místech. Tento fakt nehraje pro uživatele žádnou roli, protože toto uložení archívů při své činnosti nezaznamená.

2.3 Jazyk SQL

Podle [15] a [16] je jazyk SQL dotazovací jazyk, sloužící k práci s relačními databázovými systémy vyvinutý společností IBM. Umožňuje definovat data uložená ve formě tabulek, dotazovat se na ně a manipulovat s nimi. Má svá syntaktická a sémantická pravidla, která jsou odvozena z jednoduchých anglických vět, což jej činí srozumitelným a snadno pochopitelným. Jeho příkazy vytváří nebo upravují strukturu databáze, získávají požadovaná data, která je možno následně ukládat a popřípadě mazat. Dále jeho příkazy řeší správu uživatelských práv a rolí jednotlivých uživatelů.

Z jazyka SQL vychází většina používaných databázových systémů, čímž vlastně tento jazyk vytváří standart, který je dalšími systémy v určité míře dodržován. Jeho další výhodou je možnost jej volat i z jiných programovacích jazyků jako C, PHP nebo Python.

2.4 Systém řízení báze dat (DBMS)

Jde o software určený k efektivní práci s velkým množstvím dat, která je schopen ukládat, měnit, mazat a provádět na ně dotazy. Tento systém je nedílnou součástí většiny aplikací. Vytváří rozhraní mezi aplikacemi a uloženými daty. Jeho úloha roste s velikostí databáze, množstvím uživatelů a množstvím přenášených dat.

2.4.1 MySQL

Byl vytvořen švédskou firmou MySQL AB. Nyní je vlastněn společností Oracle corporation.

MySQL je multiplatformní databáze, která je snadno implementovatelná. Je dostupná jak placená, tak i volná licence. Je to software, který je v současnosti velmi využívaný a populární. Tento systém mohou využívat virtuální servery, které pracují na platformách Linux i Microsoft Windows Server. Je optimalizován na rychlost za cenu zjednodušeného zálohování. Je k dispozici jak pod bezplatnou licencí, tak pod komerční placenou licencí určenou pro komerční projekty. Placená verze programu má výhody ve vyšší podpoře a větším množství nástrojů. Funkčně i konfiguračně je tento systém jednodušší než MS SQL Server. Pro práci je možné využít jejich nástroj MySQL Workbench. [10]

Obr 2.2 MySQL Workbench [27]

2.4.2 MS SQL

Byl vytvořen společným vývojem firem Microsoft a IBM. MS SQL nebo také Microsoft SQL Server je výkonný relační databázový systém, který mohou využívat virtuální servery platforem Linux i Microsoft Windows. Software MS SQL vyžaduje placenou licenci, v bezplatné verzi má řadu omezení jako jsou velikost databáze, paměti, nemožnosti plánovat úlohy a dalších. Systém má standardní grafické rozhraní Windows, má jednotné, přehledné ovládání a propojení na programy MS Office, export dat, umožňuje současné otevření více objektů atd. [11]

2.4.3 PostgreSQL

Jde o databázový systém vyvinutý na Kalifornské univerzitě v Berkeley.

Jak uvádí [12] a [13], je to komerční relační databázový systém, jedna z nejpopulárnějších open source databází. Primárně je vyvíjen pro Linux, ale podporuje i jiné platformy.

2.4.4 Oracle

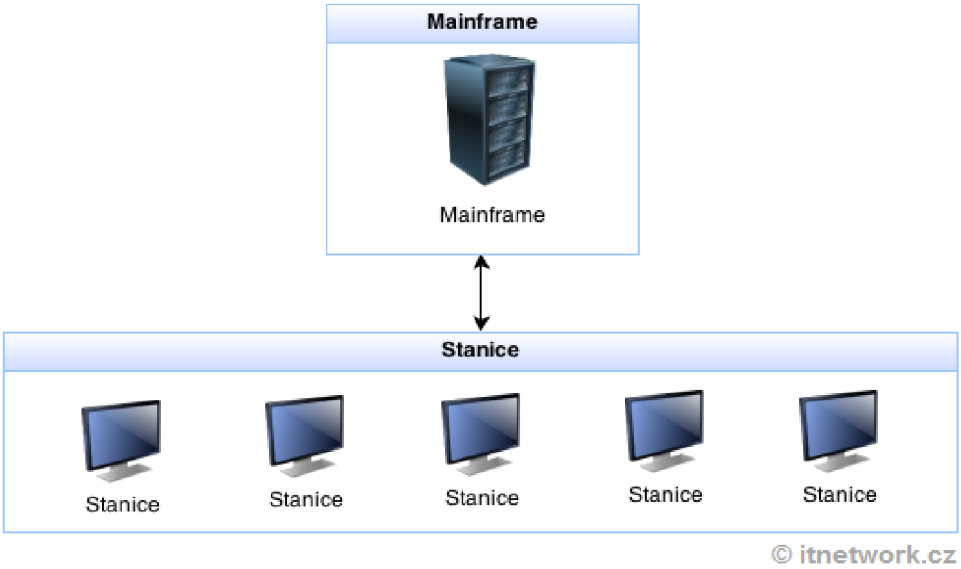
Oracle database (Oracle DB) je produktem společnosti Oracle Corporation.

Je to plně škálovatelný relační databázový systém, použitelný na většině běžných platformách. Představuje jednoho z největších konkurentů MS SQL na trhu enterprise. [14]

2.5 Historie Webových aplikací

2.5.1 Mainframe

Když se poprvé objevily samočinné počítače, byly velmi drahé a tím i těžko dostupné nejen pro jednotlivce, ale i pro firmy. Z toho důvodu firmy využívaly jeden na tehdejší poměry výkonný centrální počítač, ke kterému se připojovalo více uživatelů. Všichni uživatelé (zaměstnanci firmy) tedy pracovali současně na jednom počítači, který je zvlášť obsluhoval. V současnosti jde o velké sálové počítače pro velké firmy a společnosti (Volkswagen, Crysler, Česká spořitelna atd.) ke zpracování velkých objemů dat a pro kritické aplikace. Vyžívají se pro rozsáhlé statistické úlohy, finanční operace jako výběry z bankomatů apod. [17]



Obr 2.3 Architektura mainframe [17]

Hlavní výhody mainframu jsou:

- Snadná správa – Uživatelům postačuje pouze potřebná aplikaci, která je dostupná na centrálním počítači, nepotřebují data uchovávat na svém vlastním zařízení.
- Jednoduchá aktualizace – To, že všichni uživatelé pracují na centrálním počítači, zároveň znamená, že jakýkoli update aplikace na mainframu je okamžitě dostupný všem uživatelům.
- Vysoká bezpečnost – Veškerá data jsou uložena na mainframu, kam se připojují pouze oprávnění uživatelé, to znamená, že se k datům zvenku nikdo neoprávněný nedostane. Je umožněna funkce více aplikací vedle sebe,

přičemž jsou tyto naprosto izolované. Mainframe nebyl nikdy úspěšně napaden.

Nevýhody mainframu:

- Vysoké náklady, nízký výkon – Centrální počítač vyžaduje vysoký výkon, protože musí zpracovávat všechny úlohy aplikace požadované uživateli najednou. Tím se samozřejmě zároveň zvyšují i jeho provozní náklady.

2.5.2 Desktopové aplikace

Postupem času se počítače stávaly cenově dostupnějšími, každý zaměstnanec tak měl vlastní počítač a ve většině případů již nebylo třeba provozovat drahý centrální počítač. Všichni zaměstnanci tak měli uloženy na pevném disku svého počítače potřebné aplikace pro svoji činnost. [17]

Výhody desktopových aplikací:

- Vysoký výkon – veškerý výkon aplikace stojí na uživatelově počítači.
- Rychlé spuštění
- Jsou k dispozici i bez připojení na síť

Nevýhody desktopových aplikací:

- Nízká bezpečnost – Uživatel má k dispozici kompletní aplikaci, což mu dává možnost ji hackovat a tím například získat zdrojový kód nebo data, která mu nepřísluší.
- Správa – při updatu aplikace je potřeba ji aktualizovat u každého uživatele zvlášť.
- Je zde problematická týmová spolupráce.

Obr 2.4 Architektura desktopových aplikací [17]

2.5.3 Webserver

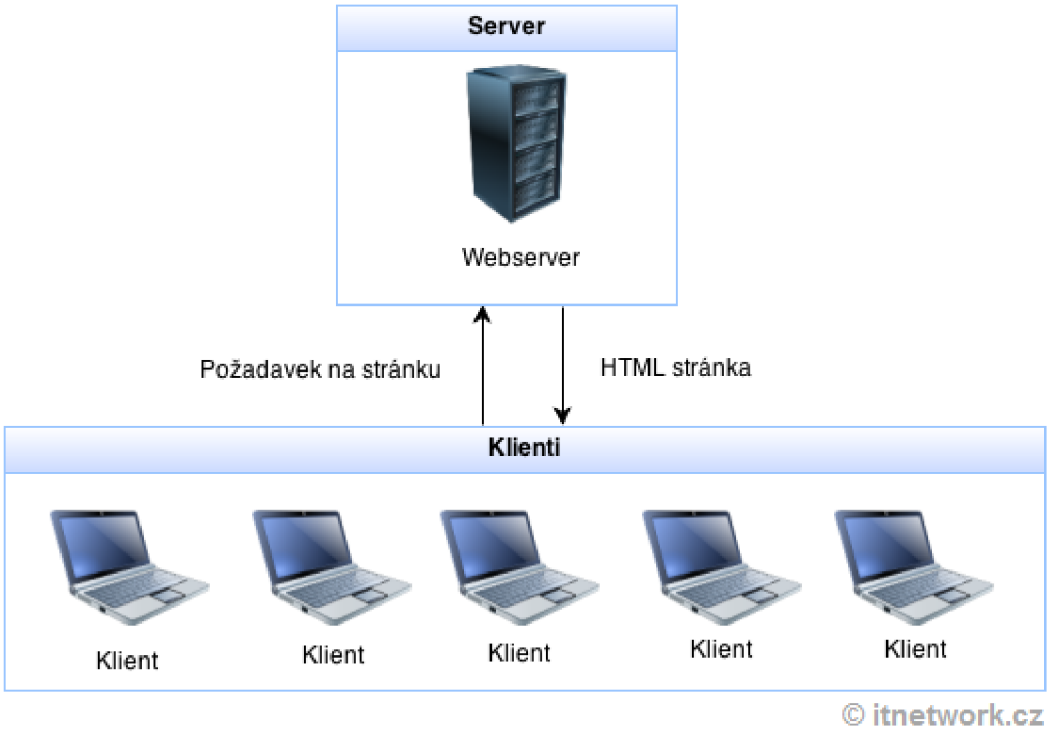
Jak uvádí [17], webové stránky se objevily s rozšířením internetu. Tyto stránky byly statické – jedná se v podstatě o textový soubor, který nemůžeme měnit, pouze jej číst.
Funguje na architektuře klient-server. Kód jazyku HTML je uložen na serveru. Ten na základě požadavku od klienta vrátí přesně tu stránku, kterou má uloženou.

Má následující výhody:

- Vysoká bezpečnost – Klient se dostane pouze k datům, které mu server pošle. O program se stará majitel serveru, na kterém jsou data uložena. A to se odráží na bezpečnosti systému.
- Jednoduchá správa – Stačí měnit soubory na serveru a změny budou okamžitě dostupné všem uživatelům.
- Nízká vytíženost – Server pouze odpovídá na požadavky klientů (nestará se o žádné vstupy klientů ani o zobrazení obsahu).
- Umožňují týmovou spolupráci.

Jeho nevýhodou je:

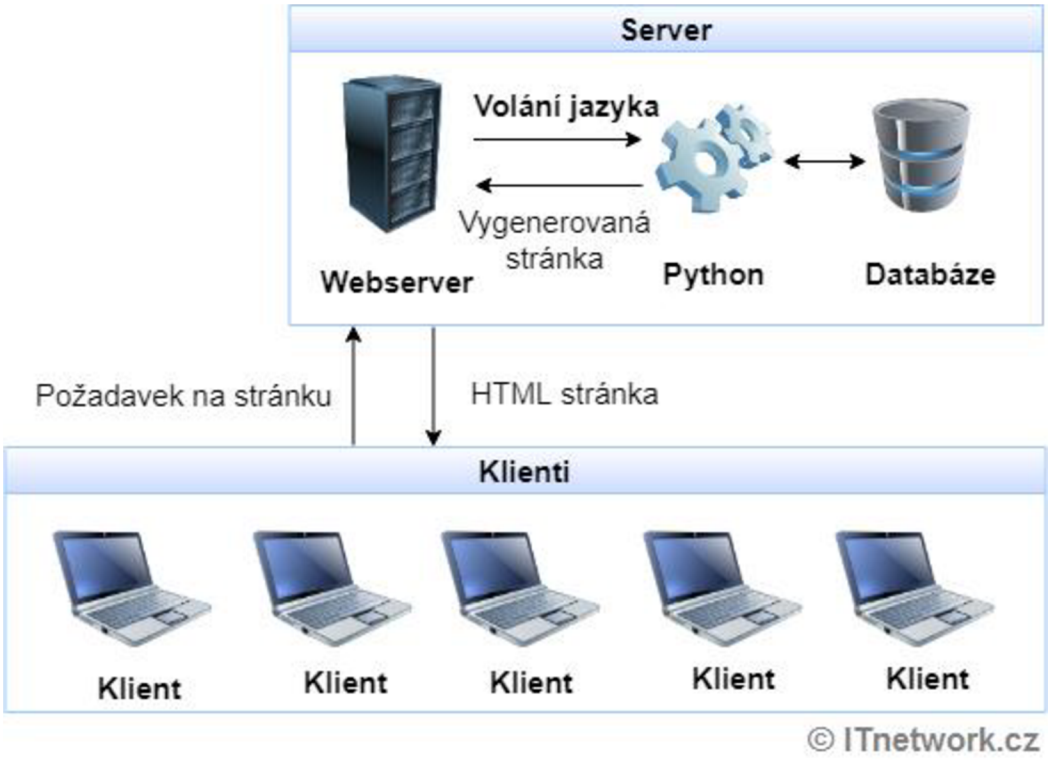
- Umí odesílat pouze statické HTML stránky, což je značně limitující. Uživatel je plně závislý na svém internetovém připojení.



Obr 2.5 Architektura webserveru [17]

2.5.4 Webové aplikace

Po rozšíření webových stránek a internetu obecně se hledal způsob, jak aplikacím dát možnost dynamizace. Vznikly tak webové aplikace, které fungují tak, že klient pošle požadavek na určitý dokument serveru, server vygeneruje stránku na základě nějakých parametrů – toho, co klient potřebuje a teprve poté ji pošle jako odpověď klientovi. Takové skripty v Pythonu zprostředkovávají webové frameworky. [17]



Obr 2.6 Architektura pro webové aplikace [17]

Obvykle je postup následující:

1. Uživatel do prohlížeče zadá URL adresu.
2. Server na základě požadavku zavolá použitý framework.
3. Framework zpracuje požadavek, připojí se k databázi, načte potřebná data, která klient vyžaduje a vygeneruje HTML kód.
4. Klientovi se zobrazí stránka podle HTML kódu. Stránka je sice statická, ale byla dynamicky vygenerovaná na základě požadavku.

Výhody webových aplikací:

- Bezpečnost – Aplikace je celá na serveru, k nevyžádaným datům by se uživatel neměl dostat, pokud nevyužije nějaké bezpečnostní chyby.
- Správa – Změna aplikace na serveru se okamžitě projeví i u všech klientů.
- Kompatibilita – Operační systém vůbec nehraje roli, přístup zprostředkovává pouze webový prohlížeč uživatele.
- Výkon – Za zobrazení zodpovídá webový prohlížeč na klientském počítači, což má za výsledek menší zatížení serveru.

Nevýhody webových aplikací:

- Vývoj – Webové aplikace jsou na vývoj náročnější než aplikace desktopové, ale s příchodem nových standardů se tato nevýhoda podstatně zmenšila. Složitější webové aplikace jsou doplňovány jazykem JavaScript, který běží u klienta a zpracovává zobrazování.
- I zde, jako v předchozím případě, je uživatel plně závislý na kvalitě svého internetovém připojení.

2.6 Webové frameworky

Jak uvádí [18], webové frameworky jsou knihovny, které umožňují vývojářům podstatně snadnější tvorbu webových aplikací. Poskytují základní šablony pro spolehlivé, snadno udržovatelné a škálovatelné webové aplikace. Jejich hlavním úkolem je především jednoduché a rychlé vytvoření webové aplikace, která většinou vychází z nějaké databáze.

Web frameworky využívají zkušeností pro budoucí použití na základě postupně nabytých znalostí a vědomostí, které developeři získali za posledních více než dvacet let. Frameworky umožňují jednodušší opakované používání kódu pro základní HTTP operace a strukturu projektu, takže se vývojáři se znalostmi frameworku dokáží lépe v daném projektu orientovat, což velmi výrazně zvyšuje efektivitu a rychlost jejich práce.

Dále frameworky poskytují funkcionality dostupné pomocí kódu nebo dalších rozšíření, která umožňují provádět základní operace vyžadované od webových aplikací. Například podle [19] operace zahrnují:

- URL směrování
- Zpracování formy vstupů a ověřování
- Možnost použití kódů HTML, XML a JSON
- Možnost konfigurace pro připojení do databáze a manipulaci s obsaženými daty

- Poskytnutí zabezpečení vůči CSRF, SQL Injection, XSS a jiným známým útokům
- Session storage – uchování dat pouze po dobu trvání relace

Všechny tyto operace však nemusí frameworky nezbytně obsahovat. K těm nejznámějším frameworkům patří například django, flask, bottle, pyramid, falcon atd.

2.6.1 Django

Django je komplexní webový framework psaný v jazyce Python. Má širokou škálu implementovaných nástrojů, se kterými je možno vytvářet aplikace bez nutnosti použití jiných rozšíření a programovacích jazyků. Podporuje čisté, pragmatické konstrukce pro rychlý vývoj webových aplikací.

Původně byl vyvinut pro novináře, kde se osvědčil a byl později vydán veřejně. Je to v současné době nejoblíbenější framework. Drží se principu „Don't repeat yourself" – neopakuj se.

To, že je Django opravdu výjimečný nástroj potvrzuje i fakt, že jej pro své webové stránky využívají společnosti jako například YouTube, Google, NASA, Instagram a mnoho dalších. [20][21]

2.6.2 Flask

[22] uvádí, že Flask je webový micro framework psaný v Pythonu. Původně byl vyvinut Arminem Ronacherem jako aprílový žert, ale stal se během chvíle velmi populární alternativou Djanga.

Má se za to, že je Flask Pythonu bližší než Django, protože v určitých situacích je obdobná aplikace psaná ve Flasku řešitelná kratším a jednodušším zápisem. Dále je také Flask jednodušší pro začátečníky. To je způsobeno především minimem kódu potřebného k rozběhnutí základní aplikace.

Příklad aplikace ve Flasku:

```
from flask import Flask
app = Flask(__name__)


@app.route('/')
def nazev aplikace():
return 'navracena hodnota'

if __name__ == '__main__':
app.run()
```

Výhodou Flasku bylo i to, že byl vydán až několik let po Djangu a byl tak schopen reagovat na názory uživatelů. Flask používá například Pinterest, Netflix nebo Reddit.

2.6.3 Bottle

Podle [23] je Bottle velice jednoduchý a rychlý micro web framework. Je tvořen jediným modulem a nemá návaznost na jiné knihovny než na základní knihovny Pythonu. Je volně rozšiřovatelný a má možnost využití šablon.

2.6.4 Pyramid

Je komplexnější framework. Nabízí řadu nástrojů k tvorbě složitějších aplikací. Je ovšem složitější než menší, jednoduché micro frameworky. Tyto potíže se snaží vyvážit kvalitními návody a tutoriály. Výhodou Pyramidu je podpora na všech verzích Pythonu. [24]

2.6.5 Falcon

Podle [25] je Falcon minimalistická knihovna sloužící jako nízkoúrovňový micro framework. Je velmi čistý a jednoduchý a zakládá si především na výkonu. V některých aplikacích vykazuje výborné výsledky.

3 NÁVRH A IMPLEMENTACE

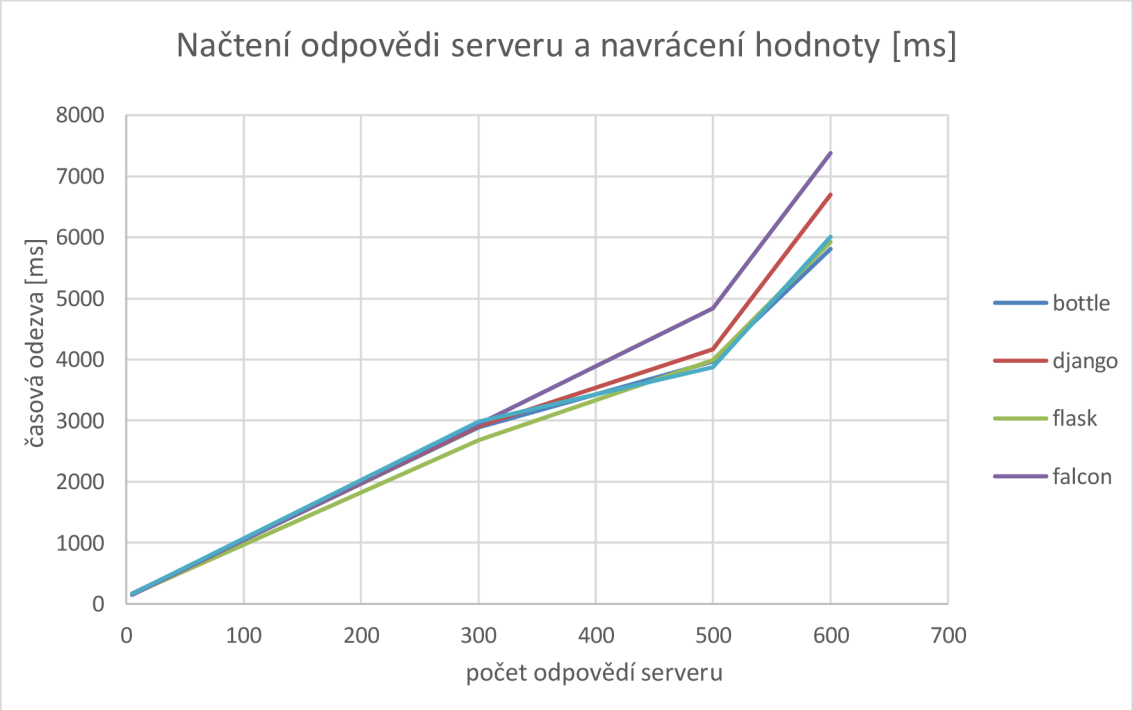
3.1 Požadavky

Úkolem tohoto semestrálního projektu bylo zpracovat studii dostupných SW knihoven a frameworků, navržení struktury databáze a vytvoření jednoduchého blokového schématu SW řešení. Výstupem této práce je webová aplikace zobrazující syrová data z databáze na základě uživatelem zadaného časového filtru.

3.2 Srovnání frameworků

Srovnání frameworků bylo založeno na měření doby provádění kódu. K tomu byl využit modul Time. Testy byly prováděny na osobním počítači, server byl použit firemní v místě zaměstnání.

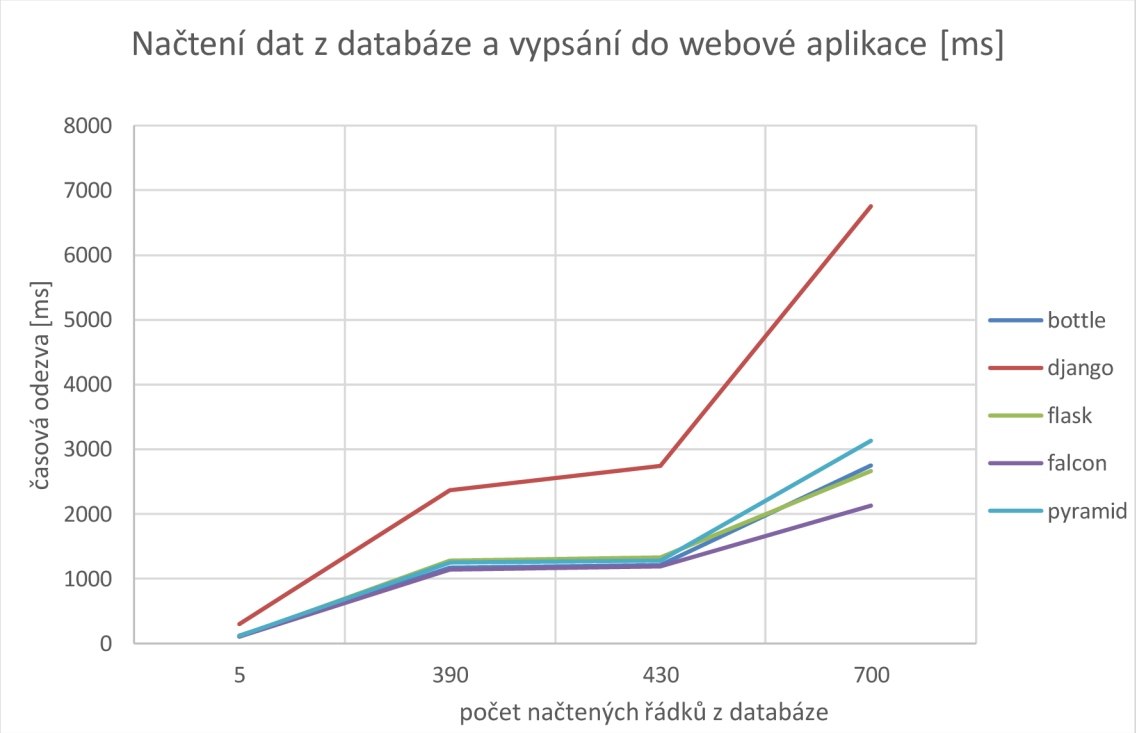
První provedené testování spočívalo v měření času potřebného ke komunikaci se serverem. Jde o závislost časové odezvy udávané v milisekundách na počtu odpovědí serveru. Kratší odezva znamená lepší výsledek. Výsledky jsou získány odečtením hodnot času začátku a konce vykonávání aplikace.



Graf 3.1 Načtení odpovědi z externího serveru a navrácení hodnoty

Dalším testem byla rychlost načtení hodnot z databáze a jejich navrácení na webovou stránku. Jde opět o závislost časové odezvy udávané v milisekundách na

počtu načítaných řádků z databázového systému MySQL. Kratší odezva znamená lepší výsledek. Výsledky jsou získány odečtením hodnot času začátku a konce vykonávání aplikace.



Graf 3.2 Načtení dat z databáze a jejich vypsání na webovou stránku

Z výsledných grafů vyplývá, že nejlepších výsledků při zohlednění všech posuzovaných kritérií dosáhl framework Flask. Dále je z grafů patrné, že lepší výsledky v testování mají micro frameworky, což je dáno tím, že komplexní framework je mnohem robustnější.

Po srovnání všech výsledků, po konzultacích s programátory, kteří se touto problematikou zabývají a po prověření dobré dostupnosti studijních materiálů a podkladů byl pro řešení úkolu vybrán micro framework Flask.

3.3 Návrh databáze

Na základě surových dat bylo třeba navrhnout správnou strukturu databáze tak, aby se do ní mohly záznamy smysluplně ukládat a byl tak možný co nejlepší přístup k datům požadovaným klientem.
Příklad surových dat ze dvou různých dnů:

Time	Code	Event	Error	Msg	Cycle
17/07/24 02:23:30	E03	Increment arb. counter	Case c/b frontside NG.	1	
17/07/24 02:23:30	D16	Log message	Case c/b frontside NG.	01224	
17/07/24 02:24:16	D16	Log message	Stopwatch End	01050	
17/07/24 02:24:42	E1	Increment quantity counter			
17/07/26 06:29:26	D06	1cycle operation is completed			32.468
17/07/26 06:30:12	D06	1cycle operation is completed			32.131

Tabulka 1 – Příklad surových dat v tabulce

Z dat uvedených v tabulce je zřejmé, že se části záznamů v určitém tvaru opakují, lze tedy na základě typu záznamu rozdělit jednotlivé složky do sloupců tabulky. Každý sloupec může mít vlastní datový typ. Například sloupec Time bude typu timestamp, sloupec Event typu text, sloupec Cycletime bude vždy datový typ double.

Délka záznamu v tabulce bohužel není konstantní, což by mohlo způsobovat problémy. To lze řešit rozdělením záznamů podle typu do více tabulek jako například tabulka s errory, výrobní tabulka a tabulka informační.

3.3.1 Sloupce navrhované databáze

- **ID (UDINT)** – Unikátní identifikátor záznamu sloužící především k lepší orientaci v databázi.
- **Time (Timestamp)** – Čas, ve kterém byl záznam pořízen.
- **Machine (Text)** – Informace o tom, ze kterého výrobního stroje záznam pochází.
- **Event (Text)** – Informace o tom, jaká operace byla strojem prováděna.
- **Cycletime (Double)** – Číslo, udávající dobu trvání operace Event.
- **Message (Text/UINT)** – Volitelná zpráva do message logu. Může být i třeba jen číselný kód zprávy.
- **Error (text/UINT)** – Název nebo kód chyby.
- **Čas zápisu (timestamp)** – Čas, ve kterém byl záznam zapsán do databáze.

Tato struktura byla navržena proto, aby byla data do tabulky uložena co nejpřehledněji. Rozdělení do sloupců umožní lepší SQL dotazování a volba správných datových typů může tyto dotazy výrazně zrychlit.

3.4 Použité knihovny

Pro řešení této semestrální práce byly využity knihovny, které umožňují zjednodušit realizaci jednotlivých částí projektu.

Pro instalaci jednotlivých knihoven (modulů) byl použit nástroj PIP, který slouží jako package manager. Instalace knihovny pomocí PIP v příkazovém řádku je velmi jednoduchá. Příklad pro instalaci modulu flask:

```
C:\Users\Janik>py pip install flask
```

3.4.1 PyMySQL

Podle [5] jde o modul (knihovnu), který zprostředkovává konektor pro připojení k MySQL serveru v jazyku Python. PyMySQL slouží jako náhrada pro starší knihovnu MySQLdb, která je určena pouze pro verzi Pythonu 2.x.x.

Zápis pro připojení k databázi a načtení požadovaných dat je podobný zápisu jiných databázových konektorů.

Nejdříve se vydefinuje adresa, na kterou se bude konektor připojovat, následuje přihlašovací heslo a jméno do databáze a jako poslední se udává název tabulky, ke které se bude přistupovat. Dále se definuje kurzorová třída cursor, pomocí kterého se vykonávají SQL dotazy do databázi.

```
conn = pymysql.connect(host='localhost',
                       user='root',
                       password='xjanik19',
                       database='db',)

cursor = conn.cursor()
cursor.execute("SELECT * FROM db.test5")
```

3.4.2 Time

Time je standardní knihovna Pythonu, která slouží k práci s časem. Tato knihovna byla využita v předchozí kapitole při testování výkonnosti frameworků.

Na začátku aplikace zjistí aktuální čas, který uloží do proměnné. Na konci aplikace zjistí aktuální čas, odečte od něj proměnnou s časem začátku a tím se dostane čas vykonávání. [7]

Příklad výpočtu času celkové doby provádění kódu:

```
start_time = time.time()                //čas začátku
duration = time.time() - start_time     //čas provádění
```

3.4.3 Flask

Z knihovny micro frameworku stačí importovat funkce Flask a render_template. V hlavním souboru main.py byly načteny data z databáze. Tato data obsahují hlavičku sloupců tabulky databáze a samotná data. Tato data jsou pomocí externího HTML souboru profile.html zpracována do tabulky zobrazované na webové stránce. Uživateli tak stačí zadat adresu do internetového prohlížeče a server mu navrátí dostupná data.

Zápis potom vypadá následovně:

```
@app.route("/profile/<name>")
def profile(name):
     return render_template("profile.html", name=name,
     header=header, data=data)
```

Na prvním řádku je uvedena cesta pro zobrazení určitého HTML souboru definovaného ve funkci. Za samotným názvem HTML souboru se definuje předávání proměnných z Pythonu pro použití v HTML. V HTML souboru je potom možno psát Python kód do zdvojených složených závorek.

3.4.4 Pandas

Podle [6] jde o volně dostupnou knihovnu pro Python, která umožňuje rychlé zpracování a analýzu dat. Je mnohými uznávána za nejlepší nástroj pro manipulaci s daty a jejich analýzou i mezi ostatními programovacími jazyky.

Dokáže pracovat s různými formáty dat, jako například:

- Tabulková data (SQL tabulky, CSV soubory)
- Uspořádaná a neuspořádaná časová data
- Tabulky s náhodným přístupem

Pandas umožňuje například:

- Snadné zpracování chybějících dat
- Slučování dat do skupin
- Snadnou konverzi do objektů
- Intuitivní spojování data setů

Data načtená z databáze se pomocí této knihovny uloží do základní struktury dat Panas – dataframe. Do dataframe jsou data ukládána v tabulkové podobě. Vyhledávání a filtrování dat v dataframe je rychlejší, spolehlivější a bezpečnější než dotazování do databáze.

Dalším příkladem využití knihovny Pandas v této práci je příkaz resample. Příkaz resample slouží k frekvenční konverzi a převzorkování dat v dataframe.

3.4.5 Bokeh

Podle [28] je Bokeh knihovna pro interaktivní vizualizaci dat, která se zaměřuje na moderní webové prohlížeče. Je prostředkem k poskytnutí stručné a elegantní konstrukce všestranné a interaktivní grafiky. Dokáže pracovat i s velmi velkými datovými sadami. Bokeh je vhodný pro všechny, kdo chtějí snadno a rychle tvořit interaktivní grafy nebo datové aplikace.
Bokeh je fiskálně sponzorovaný projektem neziskové organizace NumFOCUS zabývající se vývojem open-source zdrojů pro různé programovací jazyky.

3.4.6 JavaScript

JavaScript je programovací jazyk, používaný k tomu, aby byly webové stránky interaktivní – aby mohl klient zasílat specifické požadavky na zobrazení webové aplikace serveru.

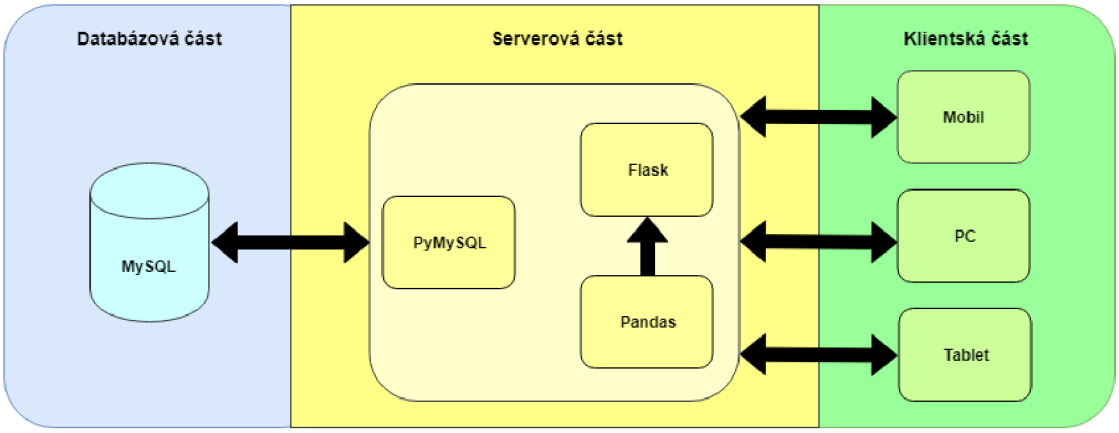
Umožňuje klientovi například vkládání textů do textových polí, sledování videí a mnoho dalších. Je jedním ze tří pilířů webových stránek společně s HTML a CSS. Pro použití skriptu jej stačí připsat na správné místo do HTML kódu. [8]

3.5 Praktické provedení

Pro řešení byla zadána data získaná z výrobního stroje. Na základě těchto dat byla navržena možná struktura databáze, která byla dále pro testovací účely vytvořena v systému MySQL. Do této databáze byla dále naimportována data dodaná firmou zadavatele. Tento databázový systém byl zvolen kvůli mým předchozím zkušenostem a freewarové licenci.

Dále byl vytvořen soubor main.py, na začátku kterého jsou importovány použité moduly. Do souboru main.py bylo zadáno připojení do databáze modulem PyMySQL dle předešlé kapitoly a byla otestována funkčnost navrácení korektních dat. Poté byl do souboru doplněn kód základní aplikace Flask. Pro správné fungování webové aplikace byl přidán soubor forms.html, do kterého byl zadán pouze testovací text.

Po odladění chyb byly soubory upravovány tak, aby se generovala tabulka řádek po řádku pomocí proměnných z Pythonu do tabulky v HTML. Nakonec byla přidána hlavička tabulky obsahující názvy jednotlivých sloupců. Tato část sloužila pouze pro ověření korektního načtení hodnot z databáze a není součástí konečného výsledku této práce.



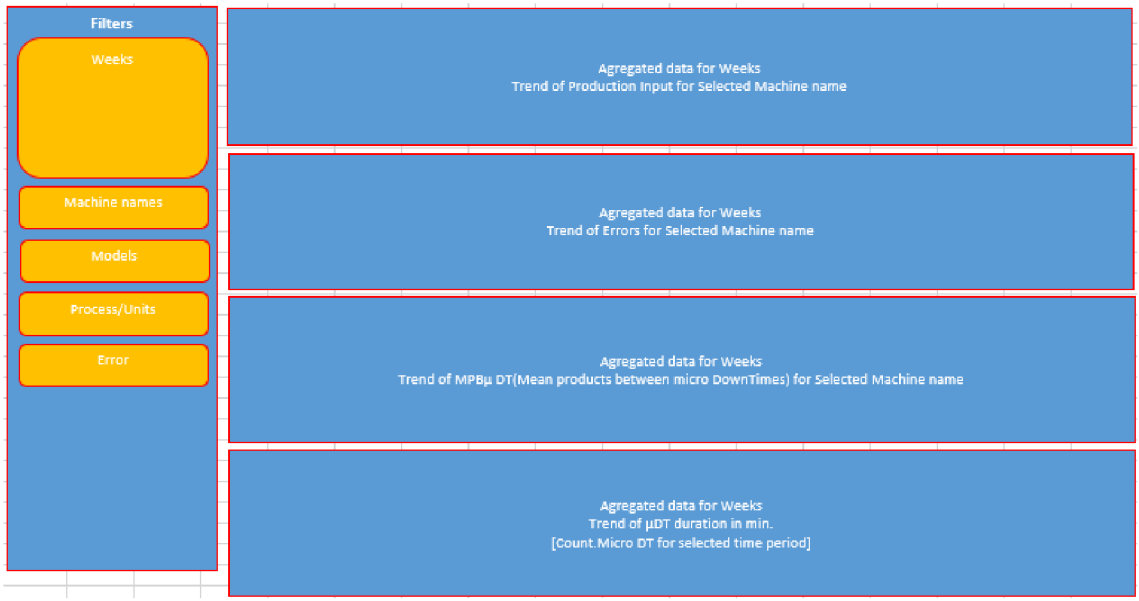
Obr 3.1 Blokové schéma řešení

V první části této práce (v části rozboru prostředků pro realizaci – kapitola 2) byla pozornost věnována především průzkumu dostupných knihoven, frameworků a obecně možných prostředků pro realizaci této práce. První částí této práce bylo pouze zobrazit data ve formě webové aplikace a ověřit tak jejich správnost načítání z databáze.
Následně byla pozornost věnována už konkrétnímu řešení a zpracování načítaných dat tak, aby tato data byla pokud možno co nejčitelnější a nejsrozumitelnější.

3.6 Uživatelské prostředí

Při tvorbě uživatelského prostředí byla snaha se držet doporučení firmy zadavatele a vytvořit tak prostředí, které by bylo pro koncového uživatele jednoduché, přehledné a aby si byl uživatel schopen rychle zobrazit potřebná data.

Obr 3.2 Návrh uživatelského prostředí pro denní zobrazení



Obr 3.3 Návrh uživatelského prostředí pro týdenní a měsíční zobrazení

Tyto dva obrázky (Obr 3.2 Návrh uživatelského prostředí pro denní zobrazení a Obr 3.3 Návrh uživatelského prostředí pro týdenní a měsíční zobrazení) obsahují doporučený vzhled uživatelského prostředí od zadavatele. První z obrázků (Obr 3.2 Návrh uživatelského prostředí pro denní zobrazení) je doporučením pro vzhled uživatelského prostředí denního zobrazení, druhý obrázek (Obr 3.3 Návrh uživatelského prostředí pro týdenní a měsíční zobrazení) pro zobrazení týdenní a měsíční.

Z obrázků je patrné, že si zadavatel přeje, aby bylo filtrační menu situováno vždy v levé části uživatelského prostředí a aby jednotlivé položky v této části byly skládány pod sebe. Jsou zde i vypsány položky, podle kterých by se měla data filtrovat. Filtrační menu pro denní zobrazení obsahuje zadavače:

- Času
- Shift = Pracovní směny
- Line = Výrobní linky
- Vyráběného modelu
- Výrobního procesu
- Vyskytnutého chybového kódu

Filtrační menu pro týdenní zobrazení obsahuje zadavače:

- Týdne/měsíce
- Výrobní linky
- Vyráběného modelu
- Výrobního procesu
- Vyskytnutého chybového kódu

Zbývající část prostředí slouží k zobrazení zpracovaných dat. V denním zobrazení se v horní části nachází tabulka, která obsahuje počty výskytů jednotlivých kódů událostí v databázi nebo třeba průměrné hodnoty čekací doby a doby výrobního cyklu v zadaném časovém okamžiku.
Vpravo od tabulky se nachází graf, zobrazující počet výskytů mikroprostojů (kódu E03) a objemu produkce (kód E0) v zadaném časovém období.
Níže jsou umístěny čtyři sloupcové grafy.

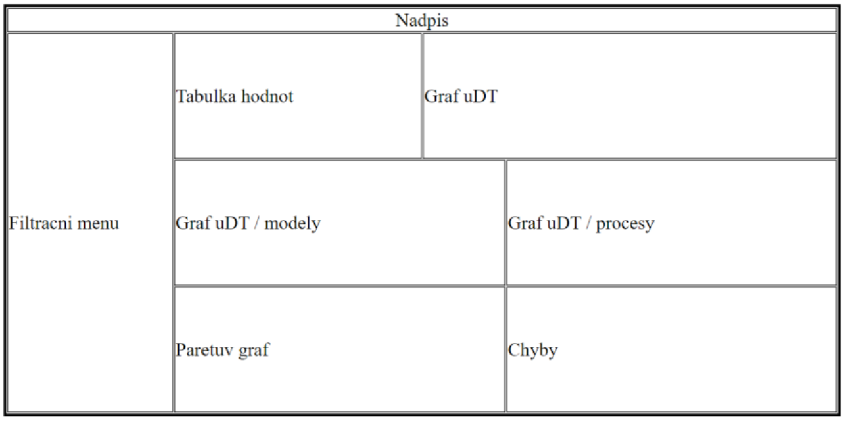
1. Graf zobrazující počet mikroprostojů a objemu produkce rozdělený podle vybraných vyráběných modelů
2. Graf zobrazující počet mikroprostojů a objemu produkce rozdělený podle vybraných výrobních procesů
3. Paretův graf zobrazující počet chybových kódů
4. Graf zobrazující počet výskytů uživatelem vybraného chybového kódu za daný časový okamžik.

V týdenním a měsíčním zobrazení část vpravo od filtračního menu obsahuje pouze grafy řazené pod sebe. Data pro grafy jsou týdně nebo měsíčně agregována. Jde o grafy:

1. Graf zobrazující objem produkce rozdělený podle vybraných vyráběných modelů
2. Graf zobrazující počet chybových kódů rozdělený podle vybraných výrobních linek
3. Graf zobrazující střední hodnotu vyrobených kusů mezi mikro prostoji pro jednotlivé vybrané výrobní linky
4. Graf zobrazující dobu mikroprostojů v minutách
5. Graf zobrazující dobu čekání na součástky v minutách

3.6.1 Tvorba uživatelského prostředí

K tvorbě uživatelského prostředí bylo využito tabulek, které rozdělují webovou stránku na předem definované bloky.



Obr 3.4 Tabulkové rozdělení pro denní zobrazení

Nevýhodou je, že formát pro denní zobrazení je jiný než pro zobrazení týdenní a měsíční.
Pro přehlednost byly vytvořeny tři HTML strany (forms, formsweek, formsmonth), kdy každá slouží pro zobrazení pouze svého obsahu, Flask k nim přistupuje ke každé zvlášť a každá z nich má tedy svou vlastní funkci v Pythonu.

3.6.1.1 Filtrační menu

Ve filtračním menu byl vytvořen formulář, pomocí kterého se odesílají data zadanou metodou. Data z formuláře jsou odesílána na odkaz udaný v parametru. V tomto případě se odesílají do denního zobrazení, které je na adrese http://localhost:5000/:

```
<form action = "http://localhost:5000/" method = "POST">
</form>
```

Do tohoto formuláře byly přidány veškeré zadavače pro filtraci dat.
Čas se zadává kombinací dvou zadavačů – data typu date a času typu time. Aby se k těmto zadavačům dalo přistupovat z Pythonu, je nutné jim přiřadit jméno. Do parametru pro hodnotu (value) zadavače jsou z Pythonu předávány proměnné, do kterých tyto zadavače samy zapisují. Když uživatel zadá hodnotu a potvrdí její odeslání zůstane tato hodnota vepsaná v zadavači – nezmění se na výchozí hodnotu hodnotu.

```
<p> <input type = "date" name = "d_from"
value="{{d_from}}"/></p>
<p> <input type = "time" name = "t_from"
value="{{t_from}}"/></p>
```

Další typ zadavače je single select, který umožňuje vybrat pouze jednu hodnotu. Při zvolení nové hodnoty se předchozí výběr vymaže.

```
<select name="event_code">
    {% for i in Event_codes %}
        {% if i == event_code %}
            <option selected value="{{i}}">
                {{i}}
            </option>
        {% else %}
            <option value="{{i}}">
                {{i}}
            </option>
        {% endif %}
    {% endfor %}
</select>
```

Podobným zadavačem jako single select je zadavač multiple select, který uživateli umožňuje vybrat více hodnot z předem vyplněného seznamu. V tomto případě musí být zadavač plněn cyklem pomocí Pythonu. To je zapříčiněno tím, že data se mohou lišit v různých časových intervalech. Při novém načítání Python zároveň kontroluje, zda je hodnota v hodnotách zadaných. Když tak tomu je, opět ji vybere – takže se zadané hodnoty po potvrzení zadání nevyprázdní.

```
<select multiple name = "line">
    {% for i in Lines %}
        {% if i in line %}
            <option selected value="{{i}}">
                {{i}}
            </option>
        {% else %}
            <option value="{{i}}">
                {{i}}
            </option>
        {% endif %}
    {% endfor %}
</select>
```

Všechny zadavače typu multiselect jsou nastaveny podobně, liší se od nich pouze zadavač Shift – pracovní směna. Ten je obsažen pouze v případě denního zadávání. Tento zadavač vybírá možnosti, které mají přidělenou číselnou hodnotu, se kterou se dále pracuje na straně Pythonu:
1 – ranní směna
2 – odpolední směna
4 – noční směna

```
<select multiple name = "shift">
    {% if '1' in shift %}
        <option selected value="1">
            Morning
        </option>
    {% else %}
        <option value="1">
            Morning
        </option>
    {% endif %}
    {% if '2' in shift %}
        <option selected value="2">
            Afternoon
        </option>
    {% else %}
        <option value="2">
            Afternoon
        </option>
    {% endif %}
    {% if '4' in shift %}
        <option selected value="4">
            Night
        </option>
    {% else %}
        <option value="4">
            Night
        </option>
    {% endif %}
</select>
```

Forma je ukončena jednoduchým tlačítkem pro potvrzení výběru. Po zmáčknutí je odeslán požadavek do Flasku http metodou POST:

```
<input type = "submit" value = "submit"/>
```

Pro navigaci mezi jednotlivými zobrazeními slouží odkazy v levé dolní části strany pod filtračním menu.

3.6.1.2 Grafy

Pro použití knihovny bokeh v HTML je třeba do hlavičky HTML souboru vložit následující kód obsahující číslo verze korespondující s verzí nainstalovanou v Pythonu:

```
<link href="https://cdn.pydata.org/bokeh/release/bokeh-
1.1.0.min.css" rel="stylesheet" type="text/css">
<link href="https://cdn.pydata.org/bokeh/release/bokeh-
```

```
widgets-1.1.0.min.css" rel="stylesheet" type="text/css">
<link href="https://cdn.pydata.org/bokeh/release/bokeh-
tables-1.1.0.min.css" rel="stylesheet" type="text/css">

<script src="https://cdn.pydata.org/bokeh/release/bokeh-
1.1.0.min.js"></script>
<script src="https://cdn.pydata.org/bokeh/release/bokeh-
widgets-1.1.0.min.js"></script>
<script src="https://cdn.pydata.org/bokeh/release/bokeh-
tables-1.1.0.min.js"></script>
```

Samotný graf a jeho komponenty jsou v Pythonu ukládány do slovníků. Tyto slovníky jsou předány do Flasku, následující kód v HTML vykreslí na stránku graf definovaný v Pythonu.

```
{{plot_div4|safe}}
{{plot_script4|safe}}
```

3.6.1.3 Tabulka hodnot

Tabulka zobrazovaná v horní části stránky obsahuje data za zadaný časový okamžik. Data jsou filtrována z dataframe a ukládána do listu, v tabulce se zobrazují jednotlivé indexy tohoto listu.

3.6.2 Serverová část

Načtení dat

Na samém začátku aplikace je definováno připojení do databáze pomocí knihovny PyMySQL.

```
conn = pymysql.connect(host='localhost',
                       user='root',
                       password='xjanik19',
```

Tato definice je dále využita v části, kde se načtená data předávají do Pandas dataframe. Objekt dataframe (df) knihovny Pandas je vytvořen metodou read_sql_query(). Argumenty této metody jsou SQL dotaz a definice připojení do databáze. V tomto případě je ještě doplněno označení časového sloupce Event_date a také volba sloupce Event_date jako sloupce pro indexování dataframe.

```
df = pd.read_sql_query(''' SELECT
                                `Event_date`, `Line`,
`Process`, `Model`, `Event_code`, `Std_msg`, `Usr_msg`,
```

```
`CT`, `Target_CT`, `Micro_DT`, `CT_below_target`, `Rate`,
`Time_consumption`
                                FROM db.data2
                            ''', conn,
parse_dates=['Event_date'], index_col='Event_date')
```

Definice Flask funkce, výchozí hodnoty

Dále bylo třeba definovat funkci Flasku pro denní zobrazení. V celé práci budou použity pouze metody GET a POST.

```
@app.route('/', methods=['POST', 'GET'])
def index():

return render_template('forms.html')


''' Run app '''
if __name__ == '__main__':
    app.run(debug=True)
```

Uvnitř funkce byly přidány dvě větve. Jedna větev slouží při vstupu do aplikace pomocí metody POST, druhá v případě vstupu pomocí metody GET.
Metoda GET slouží v tomto případě pro první načtení stránky.
Na začátku této metody bylo třeba zvolit výchozí hodnoty, podle kterých se data budou filtrovat.
Aby nebylo první vykonávání příliš dlouhé, výchozí časové rozmezí bylo stanoveno jako:

```
Od = čas před osmi hodinami, Do = současný čas
```

Toho bylo docíleno použitím knihovny datetime, kdy pro čas Do byl použit příkaz datetime.now(), který navrací aktuální čas a pro čas Od byl použit příkaz timedelta, který tak umožnil odečíst požadovaný časový interval.

```
dt = str(datetime.now() - timedelta(hours=8))
```

Jako výchozí hodnoty zadavačů typu multiple select byly zvoleny všechny hodnoty, kterých může zadavač nabývat.
Výchozí hodnota zadavače event code byla zadána po konzultaci jako hodnota E03, která představuje zastavení výrobního stroje.

Tyto proměnné byly následně přidány jako argumenty funkce render_template, která se nachází v návratových hodnotách funkce pro denní zobrazení. Tím jsou tato data předávána do Flasku a je možné je dále použít v HTML kódu jako {{proměnná}}. Tímto způsobem byly hodnoty ověřovány pro správnost výsledků.

Data pro tabulku

V této části se do listu t_data předávají hodnoty, které se v HTML vypisují do tabulky. Nejdříve se zadefinuje prázdný list, do kterého se následně pomocí příkazu append připisují data zaokrouhlené na 3 desetinná místa.

```
t_data = []
t_data.append(round((g[(g['Event_code'] ==
'E0')]).Event_code.count().tolist(), 3))
```

K jednotlivým položkám listu se v HTML přistupuje pomocí čísla indexu.

Shift – zadání pracovní směny

Zadavač pro pracovní směnu vrací pro každý výběr tři hodnoty. K zadání jsou tři možnosti, které se předávají do listu.

```
[ranní směna: odpolední směna, noční směna]
```

Ranní směna má při výběru přidělenou hodnotu 1, odpolední směna 2 a noční směna hodnotu 4.
Není-li možnost vybrána, nabývá hodnoty 0.
Pokud není vybráno nic (list nabývá hodnot [0, 0, 0], je automaticky přepsán na [1, 2, 4] tedy vybráno vše)

Při zpracování se hodnoty v listu sečtou a z výsledných hodnot 0 až 7 je podle výsledku z dataframe vybírán časový okamžik.

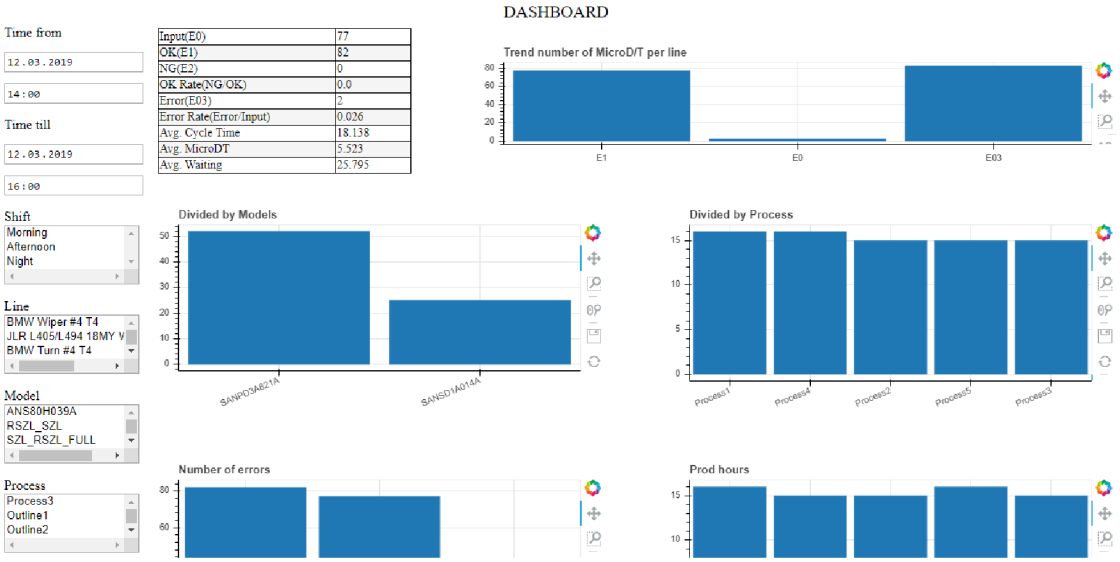
```
shift_sel = shift[0] + shift[1] + shift[2]

if shift_sel == 0:
    shift = [1, 2, 4]
if shift_sel == 1:
    g = (g[(g['Event_date'].dt.hour >= 6) &
(g['Event_date'].dt.hour < 14)])
```

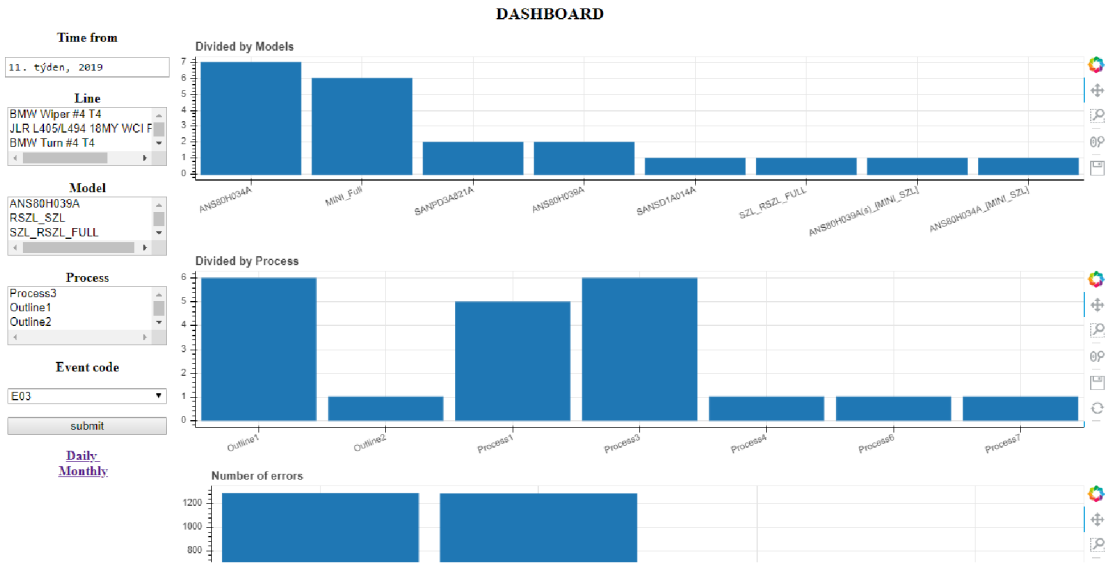
Zpracování dat pro grafy

Z dataframe jsou vybírány data pro jednotlivé grafy – vybraná data jsou převáděna do listů a předávána jako parametr funkci z knihovny Bokeh. Při tomto vytváření grafu se nejprve uloží popisky osy x grafu, výška a šířka grafu, popisek grafu a tooltip do proměnné. Na tuto proměnnou se potom použije funkce z knihovny Bokeh. Název této funkce udává, o jaký typ grafu půjde, v tomto případě se jedná o funkci vbar, tedy o sloupcový graf. Parametry funkce jsou potom hodnoty os a šířka čar.
Pomocí funkce xaxis.major_label_orientation byla změněna orientace na pí/8, aby se delší popisky nepřekrývaly a byly čitelné.
Na závěr jsou komponenty grafu předány do slovníku, který se odesílá do HTML.

```
x_axis = g1h
plot = figure(x_range=x_axis, plot_height=250,
plot_width=530, title="Divided by Models",
tooltips=TOOLTIPS)
plot.vbar(x=x_axis, top=g1, width=0.9)
plot.xaxis.major_label_orientation = pi / 8
plot_script1, plot_div1 = components(plot)
chart1 = {'plot_script1': plot_script1, 'plot_div1':
plot_div1}
```



Obr 3.5 Screenshot z vytvořené webové aplikace – denní zobrazení



Obr 3.6 Screenshot z vytvořené webové aplikace - týdenní zobrazení

4 ZÁVĚR

V této práci je řešeno zobrazení dat z databáze, která vzniká sběrem výstupů z automatizovaných výrobních strojů. Tato data, která jsou do databáze ukládána průběžně na základě ukončení předem definované operace nejsou pro cílovou skupinu uživatelů smysluplná a přehledná. Úkolem práce je získaná data převést do takového formátu, který by uživatelům poskytl možnost zobrazení přehledných a pro ně požadovaných dat.

Pro tento účel byl využit programovací jazyk Python s rozšířením micro frameworku Flask a připojovacího konektoru PyMySQL do databázového systému MySQL, zajišťující, že požadovaná data budou splňovat kritéria koncových uživatelů.

Pro získání uceleného přehledu o daném tématu jsou v této práci popsány základní pojmy databázových systémů, frameworků a Python modulů. To vytváří podmínky pro výběr vhodného řešení. Takové řešení má být co nejefektivnější a co nejjednodušší. Zároveň je zde snaha, aby byl kód aplikace co nejsrozumitelnější a nejčistší.

Vytvořená aplikace adresuje předchozí požadavky zadavatele semestrální práce. To je doloženo aplikací v příloze a obrázky výše (Obr 3.5 Screenshot z vytvořené webové aplikace – denní zobrazení a Obr 3.6 Screenshot z vytvořené webové aplikace - týdenní zobrazení). Tato aplikace umožňuje zobrazení dat ve webovém prohlížeči, aniž by koncoví uživatelé museli vstupovat přímo do databázového systému. Tím je zajištěna vyšší bezpečnost uložených dat a přístup k těmto datům je pro koncového uživatele mnohem jednodušší.

Data zobrazovaná v grafech byla zpracována podle doporučení zadavatele, avšak s ohledem na časovou náročnost nebylo možné splnit veškeré požadavky. Ty bude možné realizovat po dohodě se zadavatelem jako pokračování spolupráce nad rámec bakalářské práce.

Vzhledem k tomu, že firma zadavatele využívá placený software MS SQL, bylo veškeré testování prováděno na bezplatném databázovém systému MySQL a po dokončení testování byl konektor změněn na MS SQL.

Literatura

[1] MOBLEY, Keith. *An introduction to predictive maintenance*. 2nd ed. Woburn: Elsevier Science, 2002. ISBN 0-7506-7531-4.

[2] KOHUTOVÁ, Radka. *Databáze ve věku informační společnosti a jejich právní ochrana*. Praha: Beckova edice právní instituty, 2013. ISBN 9788074004933.

[3] Python. *Python.org* [online]. Python Software Foundation, 2018 [cit. 2018-12-12]. Dostupné z: https://www.python.org/doc/essays/blurb/

[4] Python introduction. *W3schools* [online]. 2018 [cit. 2018-12-12]. Dostupné z: https://www.w3schools.com/python/python_intro.asp

[5] Python database access. *Tutorialspoint* [online]. [cit. 2018-12-12]. Dostupné z: https://www.tutorialspoint.com/python3/python_database_access.htm

[6] Python data analysis library. *Pandas.pydata* [online]. [cit. 2018-12-12]. Dostupné z: https://pandas.pydata.org/pandas-docs/stable/

[7] Time access and conversions. *Python.org* [online]. [cit. 2018-12-12]. Dostupné z: https://docs.python.org/3/library/time.html

[8] Introduction to JavaScript. *Thoughtco.com* [online]. [cit. 2018-12-12]. Dostupné z: https://www.thoughtco.com/what-is-javascript-2037921

[9] ELMASRI, Ramez a Shamkant NAVATHE. *Fundamentals of Database Systems* [online]. 6th ed. Boston, Messachusetts: Addison-Wesley, 2011 [cit. 2018-12-12]. Dostupné z: http://iips.icci.edu.iq/images/exam/databases-ramaz.pdf

[10] *Promotic* [online]. Ostrava-Vítkovice [cit. 2018-12-13]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Subsystems/Db/MySql/MySql.htm

[11] *TechTarget* [online]. [cit. 2018-12-13]. Dostupné z: https://searchsqlserver.techtarget.com/definition/SQL-Server

[12] *Managementmania* [online]. [cit. 2018-12-13]. Dostupné z: https://managementmania.com/cs/postgresql

[13] *PostgreSQL* [online]. [cit. 2018-12-13]. Dostupné z: https://www.postgresql.org/

[14] *Technopedia* [online]. [cit. 2018-12-13]. Dostupné z: https://www.techopedia.com/definition/8711/oracle-database

[15] *Voho* [online]. [cit. 2018-12-13]. Dostupné z: http://voho.eu/wiki/sql/

[16] *Interval.cz* [online]. [cit. 2018-12-13]. Dostupné z: https://www.interval.cz/clanky/databaze-a-jazyk-sql/

[17] *ITnetwork.cz* [online]. [cit. 2018-12-13]. Dostupné z: https://www.itnetwork.cz/python/django/uvod-do-django-frameworku-a-webovych-aplikaci-v-pythonu

[18] *STEEL KIWI* [online]. [cit. 2018-12-13]. Dostupné z: https://steelkiwi.com/blog/best-python-web-frameworks-to-learn/

[19] *Full Stack Python* [online]. [cit. 2018-12-13]. Dostupné z: https://www.fullstackpython.com/web-frameworks.html

[20] *Django Česká Republika* [online]. [cit. 2018-12-13]. Dostupné z: http://www.djangoproject.cz/
[21] *Django* [online]. [cit. 2018-12-13]. Dostupné z: https://www.djangoproject.com/
[22] *Full Stack Python* [online]. [cit. 2018-12-13]. Dostupné z: https://www.fullstackpython.com/flask.html
[23] *Bottle* [online]. [cit. 2018-12-13]. Dostupné z: https://bottlepy.org/docs/dev/
[24] *Pyramid* [online]. [cit. 2018-12-13]. Dostupné z: https://trypyramid.com/
[25] *Falcon* [online]. [cit. 2018-12-13]. Dostupné z: https://falcon.readthedocs.io/en/stable/
[26] GEMIGNANI, Zach, Chris GEMIGNANI a Patrick SCHUERMANN. *Efektivní analýza a využití dat*. Brno: Computer Press, 2015. ISBN 9788025145715.
[27] *MacUpdate: MySQL Workbench* [online]. [cit. 2018-12-14]. Dostupné z: https://www.macupdate.com/app/mac/31829/mysql-workbench
[28] Bokeh: data visualization. *Bokeh.pydata* [online]. [cit. 2019-05-21]. Dostupné z: https://bokeh.pydata.org/en/latest/

Seznam symbolů, veličin a zkratek

SQL	-	Structured Query Language
DMBS	-	Database Management Systém
XML	-	Extensible Markup Language
URL	-	Uniform Resource Locator
JSON	-	JavaScript Object Notation
XSS	-	Cross Site Scripting
CSRF	-	Cross Site Request Forgery
Shift	-	Pracovní směna
Line	-	Výrobní linka

Seznam příloh