TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Návrh zpracování a vizualizace dat z vodárenského přivaděče Bedřichov

Magisterský projekt

M1300136

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Lukáš Pelc**
Vedoucí práce: Ing. Jan Kolaja, Ph.D.
Konzultant: doc. Ing. Milan Hokr, Ph.D.

Liberec 2021

Zadání

Upravte stávající stav softwarového řešení sběru, ukládání a grafické prezentace dat z vodárenského přivaděče Bedřichov. Práce by měla splňovat následující body:

1. Analyzujte současný stav měření fyzikálních jevů v přivaděči Bedřichov.
2. Navrhněte řešení výpočtů a přenosů dat z databáze tak, aby byla možná jejich interpretace na webovém serveru.
3. Analyzujte použití různých programovacích jazyků, se kterými jste se seznámil během studia, pro generování dynamického obsahu a vyberte vhodný nástroj pro svou práci.
4. Vytvořte demonstrační aplikaci, která ukáže možnosti dalšího směřování vývoje nástroje pro grafickou prezentaci naměřených dat nezávisle na použitém databázovém systému.

Prohlášení

Prohlašuji, že svůj magisterský projekt jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mého magisterského projektu a konzultantem.

Jsem si vědom toho, že na můj magisterský projekt se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mého magisterského projektu pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li magisterský projekt nebo poskytnu-li licenci k jeho využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Beru na vědomí, že můj magisterský projekt bude zveřejněn Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

29. 4. 2021 Bc. Lukáš Pelc

Návrh zpracování a vizualizace dat z vodárenského přivaděče Bedřichov

Abstrakt

Tato zpráva popisuje návrh grafické reprezentace dat z vodárenského přivaděče Bedřichov. Cílem je rychlejší a přehlednější práce s daty získávanými v projektu měření fyzikálních jevů v horninovém masivu. V první části se věnuje měření v přivaděči Bedřichov a rozebírá nutné výpočty nad daty. Dále se věnuje výběru prostředků pro programování. Poukazuje na nedostatky současného stavu databáze. Výsledná aplikace demonstruje možnosti použitého řešení a nastiňuje směr dalšího vývoje.

Klíčová slova: měření fyzikálních veličin, zpracování dat, databáze, webová aplikace

Design of processing and visualization of data from the Bedřichov water supply

Abstract

This report describes the design of a graphic representation of data from the Bedřichov water supply. The aim is to work faster and more clearly with the data obtained in the project of measuring physical phenomena in the rock massif. The first part deals with measurements in the Bedřichov water feeder and discusses the necessary calculations over data. It is also dedicated to the selection of funds for programming. It points to shortcomings in the current state of the database. The resulting application demonstrates the possibilities of the solution used and outlines the direction of further development.

Keywords: measurement of physical quantities, data processing, databases, web application

Poděkování

Mé poděkování patří především Ing. Janu Kolajovi za podporu a čas který mi věnoval při vedení práce. Dále konzultantu panu doc. Milanu Hokrovi za trpělivé vysvětlení fyzikálního pozadí práce.

Obsah

Seznam obrázků

Seznam kódů

Seznam zkratek

AJAX	Asynchronous JavaScript and XML – Asynchronní JavaScript a XML
API	Application Programming Interface, aplikační rozhraní
CDN	Content Delivery Network – Síť pro doručování obsahu
CSRF	Cross Site Request Forgery
CSV	Comma-separated values – Hodnoty oddělené čárkami
CxI	Ústav pro nanomateriály, pokročilé technologie a inovace
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things – Internet věcí
JS	JavaScript
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LTS	Long-term Support – prodloužená podpora
MTV	Model, Template, View
MVC	Model, View, Controller
OMP	Oddělení modelování procesů
ORM	Object-relational mapping – Objektově relační mapování
OS	Operační systém
REST	Representational State Transfer
SVG	Scalable vector graphics – Škálovatelná vektorová grafika
TUL	Technická univerzita v Liberci
XML	Extensible Markup Language
XSS	Cross Site Scripting

1 Úvod

Cílem práce je zorientovat se ve stavu ukládání geologických dat naměřených ve vodárenském přivaděči v Bedřichově. Analyzovat vhodné prostředky pro vývoj administračního prostředí a prohlížení měřených dat. Identifikovat problematická místa současného řešení, navrhnout řešení tak, aby lépe odpovídalo potřebám jejich vhodného zobrazení a přehledné analýze. K tomu se vztahuje další část práce, která si klade za cíl demonstrovat přehledné, jednoduché, ale dostatečně komplexní webové rozhraní pro zobrazení a prezentaci dat, správu uživatelů a jejich rolí.

Vzniklá demonstrativní aplikace bude sloužit jako podklad k dalšímu vývoji a bude dále upravována dle požadavků autora námětu a budoucích uživatelů na NTI FM a OMP CxI.

Tato práce je členěna do tří částí. První se zabývá fyzikálními reáliemi a některými vlastnostmi měření. Druhá je věnována výběru aplikačních nástrojů a prostředků vývoje. Poslední pak vývoji samotné demonstrační aplikace.

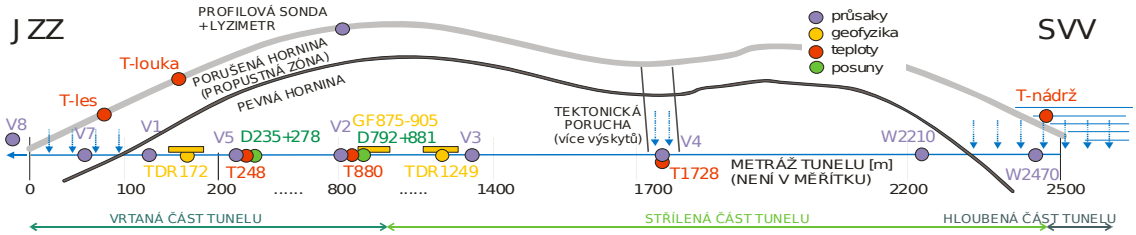
2 Měření

Data měření, která jsou předmětem zpracování v tomto projektu, jsou získávána výzkumnou skupinou na TUL pod vedením doc. Hokra, který je zároveň konzultantem této práce. Měření jsou prováděna ve štole neustále a data jsou z čidel odesílána na databázový server. V závislosti na typu senzoru (čidla) jsou data odesílána na server různými cestami, této části systému se práce nemá ambice věnovat. Omezí se pouze na vyjádření faktu, že z většiny senzorů přichází data v minutových rozestupech. Tento rozestup však není zaručen a u některých senzorů se může lišit nebo kolísat. Může se také stát, že dojde k výpadku měření či ukládání dat.

2.1 Přivaděč Bedřichov

Vodárenský přivaděč Bedřichov je z hlediska měření geologicky významné místo. Jedná se o šachtu, která je zčásti vrtaná a zčásti střílená, je dlouhá přibližně 2,6 km, vede až 150 m hluboko žulovým masivem, a slouží k přivádění pitné vody do Liberce.[1]

Obrázek 2.1 převzatý z dokumentu [2], zde znázorněné body měření pokrývají širší rozsah aktivit, než je zahrnuto ve zpracovávaných datech v tomto projektu. Zobrazené označení senzorů slouží pro interní komunikaci výzkumných týmů. Pro tuto práci jednotlivá označení nejsou důležitá, obrázek má však ilustrační hodnotu a lze si podle něho udělat hrubou představu, čím se měření zabývá.



Obrázek 2.1: Názorné schéma přivaděče[2]

2.2 Senzory

Práce se nebude zabývat detailní specifikací jednotlivých zařízení na měření fyzikálních veličin a odesílaní těchto dat na server. Pokud by případného čtenáře tyto informace zajímaly, nalezne je v dokumentu [3]. Zaměří se pouze na charakter takto získaných dat, případně na způsob jejich vyčištění a zpracování. Způsob měření zde bude uveden pouze v případě, kdy je tato informace nezbytně nutná k pochopení uložených hodnot v databázi. Protože právě zpřehlednění prohlížení dat, již naměřených a uložených, je hlavní náplní práce.

Následující podkapitoly blíže rozebírají jednotlivé veličiny měřené v přivaděči Bedřichov.

2.2.1 Průsaky vody – průtok

Jedna z veličin, která je měřena hned na několika místech v tunelu, je průtok prosakující vody. Zde jsou data měřena více způsoby, které budou níže rozebrány.

Měrný přeliv s profilem – výška hladiny

Výška hladiny je měřena ultrazvukovým snímačem, ten je umístěn nad přelivem a měří vzdálenost hladiny od bodu měření. Vyšší hodnoty proto znamenají nízký stav hladiny. Je náchylný na chyby způsobené odrazem od nechtěných objektů. Těmi mohou být například padající kapka vody nebo prach poletující vzduchem.

Opačná chyba, tedy chyba, kdy senzor zaznamená nižší hladinu než skutečně je, by naopak neměla nastat.

Z toho důvodu jsou surová data rozptýlena jedním směrem, mají ale dobře zřetelnou horní obalovou křivku. Měření výšky hladiny probíhá každou minutu, což je pro potřeby analýzy dostatečně často, a můžeme si dovolit snížit rozlišení tím, že vezmeme v potaz pouze maximální hodnotu ve vhodném časovém úseku. Tento časový úsek otevírá prostor pro dodatečné ladění. Bylo by například možné nechat vhodné parametry zvolit samotným koncovým uživatelem.

I přes to mají data velký rozptyl, který je způsoben kolísáním hladiny vody. Další možností shlazení průběhu je aplikování klouzavého průměru. I v tomto případě by bylo možné nechat volbu vhodného intervalu na uživateli.

Protože konkrétní pozice a označení senzorů nejsou pro tuto práci důležité, senzory budou rozlišeny jejich ID v databázi.

Následující vzorec je platný pro senzor průtoku ID 187,

$$Q = 7,95 \cdot 10^{-5}(310,4 - d)^{2,3} \quad [\mathrm{l/s}], \tag{2.1}$$

kde d je vzdálenost hladiny od senzoru v mm.

Danaidy – výška hladiny ve válci s otvorem

V případě tzv. danaidy je výška hladiny měřena pomocí sondy relativního tlaku (nebo absolutního s barometrickou kompenzací), tj. hydrostatického tlaku vyvola-

ného sloupcem vody nad sondou. Vzorce 2.2 až 2.4 přibližují zjednodušení vzorce pro efektivnější výpočet.

$$Q = S\sqrt{2gh} \quad [\mathrm{m}^{\frac{5}{2}}/\mathrm{s}] \tag{2.2}$$

$$K = S\sqrt{2g} \quad [\mathrm{m}^{\frac{5}{2}}/\mathrm{s}] \tag{2.3}$$

$$Q = K\sqrt{h} \quad [\mathrm{m}^3/\mathrm{s}] \tag{2.4}$$

Kde S je obsah výtokového otvoru v m^2, h potom v m a značí výšku vodního sloupce. Tato hodnota je měřena senzorem a ukládána do databáze.
K je empirický výtokový koeficient.

Pro stanoviště se senzorem ID 187 je $K = 34{,}5\ \mathrm{m}^{\frac{5}{2}}/\mathrm{s}$.

Senzor ID 172 je umístěn ve válci, kde je zapotřebí rovnici uvedenou v předchozím případě upravit pro více odtokových otvorů. Funkce sgn, použitá v následující rovnici, vrací 1 pro kladný výraz a -1 pro výraz záporný. Tato funkce zajišťuje „spínání" jednotlivých členů, podle toho jak stoupá výška vodní hladiny a jednotlivými otvory začíná vytékat voda. (Jak je ze vzorce patrné, ve výšce 38 mm ode dna nádobky jsou umístěny hned dva stejné otvory.)

$$\begin{aligned} Q = K \Bigg(& \sqrt{\frac{h}{1000}} \\ & + 2\sqrt{\frac{\mathrm{sgn}(h-38)+1}{2} \cdot \frac{h-38}{1000}} \\ & + \sqrt{\frac{\mathrm{sgn}(h-200)+1}{2} \cdot \frac{h-200}{1000}} \Bigg) \quad [\mathrm{ml/s}] \end{aligned} \tag{2.5}$$

Zde $K = 82{,}2\ \mathrm{m}^{\frac{5}{2}}/\mathrm{s}$. (Faktor již obsahuje převod z m^3/s na ml/s.)

Při implementaci aplikace je na zvážení, zda, místo složitého matematického výrazu, nevyužít konstrukci `if, else`.

Počet vylití sběrné nádobky

V následujících případech je měření průtoku realizováno nádobkou o známém objemu, která se vždy po zaplnění automaticky vyprázdní. Toto je prováděno tzv. čítačem sklopky a počet takto naměřených pulzů za dané časové období je následně možné triviálně přepočíst na průtok dle následujícího vzorce.

$$Q_i = \frac{N_i - N_{i-1}}{t_i - t_{i-1}} \cdot V \quad [\mathrm{ml/s}] \tag{2.6}$$

Kde N je uložený stav čítače sklopky. V je objem měrné nádoby v milimetrech a t pak značí časové razítko záznamu v sekundách.

Pro implementaci je zde zapotřebí upozornit na fakt, že čítač sklopky nepočítá do nekonečna. Je proto potřeba počítat s jeho vynulováním a tento případ řádně ošetřit.

Rovnice 2.6 je platná pro senzor ID 184. Zde je V rovno 5,66 ml. Výsledné hodnoty by měly být v rozmezí $0 - 0{,}02$ ml/s.

Sklopka čítače pulzů ID 168 má odlišnost v tom, že je konstruována pomocí dvou nádobek. Zde je pouze zapotřebí počítat s nesymetričností sklopky. Z toho důvodu se přeskuje jeden pulz tak, aby hodnota průtoku byla počítána vždy z jedné a té samé nádobky. V následují obecnější rovnici tedy volíme parametr $r = 2$ a $V = 15$ ml. Velikost průtoku by pak měla být v rozsahu $0{,}01 - 0{,}05$ ml/s.

$$Q_i = \frac{N_i - N_{i-r}}{t_i - t_{i-r}} \cdot V \quad [\text{ml/s}] \tag{2.7}$$

V obou těchto příkladech se nabízí možnost volby většího kroku pro výpočet. Pokud by byl zvolen parametr r například 20, dojde rovnou ke shlazení. Toto použití je obdobou aplikace klouzavého průměru na řadu Q_i. Zde není nutné data tzv. ředit, jako k tomu dochází při výběru maxima v uvedeném případě se senzorem 187. Pouze je nutné dodržet podmínku volby sudého parametru r.

2.2.2 Ostatní veličiny

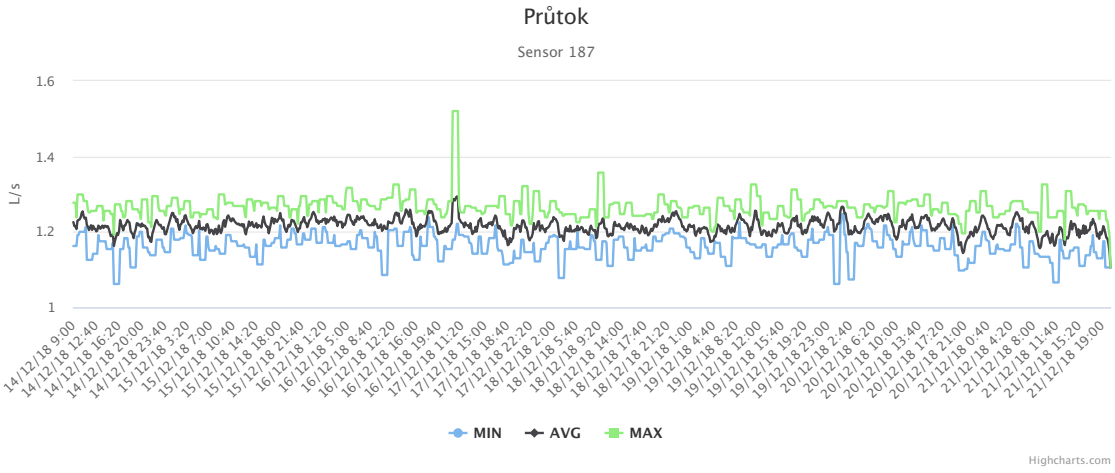
Všechny ostatní veličiny jsou uloženy v databázi přímo a není nutný jejich přepočet.

Jedná se o

- teplotu [°C],
- vlhkost [RH],
- pH,
- konduktivitu [uS/cm],
- redox potenciál [mV].

S těmito daty již je možné pracovat jednotně. V případě dlouhých časových řad je vhodné nabídnout uživateli agregaci průměrem, aby nebylo nutné načítat veškerá data. Pro příklad je možné uvést, že pokud by uživatel chtěl zobrazit průběh teploty za celý jeden rok, není nutné načítat všech 525 600 řádků tabulky. (Počet záznamů je přibližně roven počtu minut v roce.) Takovou jemnost dat by stejně nebylo možné efektivně zobrazit, protože horizontální rozlišení Full HD displeje je pouze 1920 zobrazovacích bodů. Agregací průměrných hodnot za každých 6 hodin je tak možné snížit počet přenášených řádků na 1460, což se při vhodné velikosti grafu zdá být rozumná hodnota.

Dalším námětem je zobrazení také minima a maxima. Výsledný graf by tak mohl mít například podobu zobrazenou na obrázku 2.2.



Obrázek 2.2: Příklad vykreslení agregovaných dat

3 Prostředky vývoje

Tato kapitola popisuje, jakým způsobem byly vybírány prostředky pro vývoj demonstrační aplikace s předpokladem jejího pozdějšího nasazení na produkční server. V kapitole jsou také shrnuty některé požadavky a možnosti dalšího vývoje a rozšiřování platformy v budoucnu.

3.1 Požadavky

V rámci problematiky měření veličin v přivaděči Bedřichov bylo již zpracováno více podobně zaměřených prací, přesto zadavatel není zcela spokojen. Problémy vidí v:

- převodu hodnot
  - ze surových dat na fyzikální veličinu
- rychlosti
  - aplikace
  - databáze
- přesnosti
  - výběru dat
  - rozlišení (jemnosti) dat
- pohodlnosti
  - manipulace s grafy
  - porovnání hodnot
- přístupnosti
  - pro správce
  - pro pověřeného návštěvníka
  - pro veřejnost

Dále klade důraz na bezpečnost a udržitelnost aplikace do budoucna, případně i na její pokud možno pohodlnou rozšiřitelnost. A to případně i dalšími studenty.

V dosavadních řešeních se vždy jednalo o webovou aplikaci. Tento přístup byl zvolen především kvůli přenositelnosti a univerzálnosti. Odpadají tak problémy s kompilací programu pro různé platformy a web bude fungovat jak v desktopovém prostředí, tak pro rychlou kontrolu na mobilním zařízení. Tento přístup je opodstatněný, a tato páce se ho bude ze stejných důvodů držet.

Ze zmíněných problémů přirozeně vyplývají požadavky, které budeme klást na vznikající systém.

3.2 Výběr programovacího jazyka

Klíčovým rozhodnutím pro návrh každého softwaru je zvolit vhodný programovací jazyk. V tomto kontextu je zřejmé, že se jedná o jazyk programovací, z toho důvodu se dále práce omezí na užívaní pouze termínu jazyk, bez přívlastku programovací. Již bylo zmíněno, že se bude jednat o webovou aplikaci, zvolený jazyk by měl být proto vhodný pro běh na serveru a servírování naší aplikace přes protokol HTTP.

Dále by měl být také vyučovaný na TUL, a to právě z důvodu možného rozšíření dalším studentem. Níže jsou rozebrány vhodné jazyky, se kterými se student v průběhu studia seznámí.

3.2.1 Java

Jedná se o jazyk předkompilovávaný do takzvaného bajtkódu, který je posléze interpretovaný programem *Java Virtual Machine* (*JVM*). Tento přístup umožňuje kódu psanému v jazyce Java výhodu multiplatformnosti, protože o přenositelnost se zde stará právě instalace odpovídající verze *JVM*.

Je to jazyk se silnou typovou kontrolou a automatizovanou správou paměti. Orientovaný na bezpečné a robustní aplikace a programátorovi přináší přístupnost tzv. vyššího jazyku.[1] Z toho důvodu je velmi oblíbený v korporátním sektoru. Společnost Google si tento jazyk zvolila jako hlavní nástroj pro programování nativních aplikací pro svůj mobilní sytém Android a podle indexu TIOBE[2] se jedná o dlouhodobě nejpoužívanější jazyk.[5]

Na Technické univerzitě v Liberci, dále jen TUL, se s ním student Informačních technologií, dále jen IT, setká v prvních dvou semestrech v rámci předmětu *Algoritmizace a programování 1* a *2* (*MTI/ALP1- 2*). Následně si potom může v šestém semestru zvolit předmět *Programování mobilních zařízení* (*MTI/PMZ*), kde se setká s Javou cílenou pro OS Android.

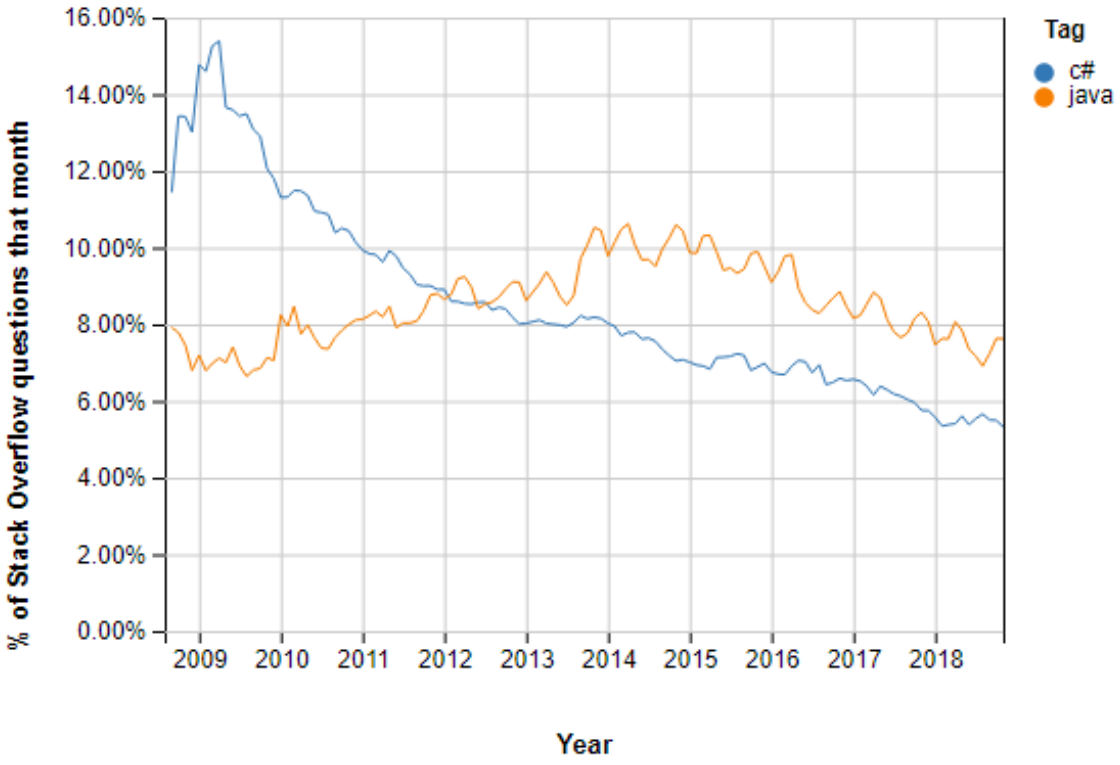
[1] Označení pro programovací jazyk s větší mírou abstrakce.

[2] Žebříček oblíbenosti založený na analýze vstupních dotazů z různých internetových vyhledávačů. Index zahrnuje statistiky od 25 vyhledávačů (tyto se v čase mírně mění), mezi které patří Google, Qq, Sohu, MSN, Yahoo!, Baidu, Wikipedia a Ebay.[4]

3.2.2 C#

C# je velmi podobný Javě. A to jak syntaxí, tak specifiky jazyka samotného. Na rozdíl od Javy, která byla vyvíjena společností Sun Microsystems od roku 1995, C# je o pět let mladší. Byl uvedem společností Microssoft v roce 2000.[6]

Na grafu 3.1 je vidět, že jeho oblíbenost prudce rostla do roku 2009 a od té doby zaznamenává mírný pád ve prospěch právě Javy a objevujících se nových jazyků. Na TUL se s ním student IT setká především ve čtvrtém semestru v rámci předmětu *Vývoj aplikací pro Windows* (*MTI/VAW*).



Obrázek 3.1: Java oproti C# podle dat serveru Stack overflow[7]

3.2.3 PHP

Odlišným jazykem je PHP. Jedná se o jazyk tzv. skriptovací, který není kompilovaný, nýbrž interpretovaný. Vznikl v roce 1995 za účelem programování dynamických webových stránek. Odtud také jeho zkratka, která v původním znění vychází z názvu *Personal Home Page*. Od té doby se stal jakýmsi webovým standardem a pro mnoho programátorů je jasnou volbou nehledě na zaměření konkrétní aplikace, tedy jak již se stalo vžitým termínem i pro české prostředí, tzv. *use casu*.

Student TUL se s ním mohl setkat v šestém semestru v rámci předmětu *Webové aplikace* (*NTI/WEAP*), kde je ale nyní již částečně nahrazen JavaScriptem.

3.2.4 Python

Python je, podobně jako PHP, multiparadigmatický[3] interpretovaný jazyk. Často se uvádí jako dobrá volba pro začínající programátory. Toho dosáhl především díky dobré čitelnosti. V kódu se nepoužívají pro uzavírání bloků závorky, ale pouze odsazení textu. To přispívá také ke krátkému a dobře srozumitelnému zápisu.

V roce 1991 ho navrhl programátor Guido van Rossum. Nejedná se tedy o módní trend posledních let. Python je jazyk, který prošel již dlouhým vývojem, jako projekt s otevřeným zdrojovým kódem, a pomalu získával na oblíbenosti. Dnes je jeho hlavní výhodou, mimo jiné, jeho univerzálnost. Hojně se používá ve vědeckých výpočtech, neuronových sítích, IoT[4] zařízeních nebo webových aplikacích.

Student TUL, oboru IT, se s ním mohl seznámit v pátém semestru v rámci předmětu *Programovací jazyk Python* (*NTI/PJP*).

V navazujícím studiu potom při předmětu *Pokročilé webové aplikace* (*NTI/PWA*).

3.2.5 Zvolený jazyk

Při výčtu vyučovacích předmětů, kde se student seznámí s daným jazykem, je vhodné zmínit, že vyučující některých zde neuvedených předmětů umožňují zvolit si jazyk dle uvážení studenta, byť je často některý z výše uvedených jazyků preferovaný.

Na obrázku 3.2 je možné vidět celosvětový zájem uživatelů o daný jazyk, který je měřený pomocí procentuálního zastoupení jeho vyhledávání na serveru Google. Podle této metriky je možné zhodnotit, že trvale velmi oblíbeným jazykem je Java. Ovšem jazyk Python získává na popularitě především v posledních pěti letech. JavaScript je zde uveden spíše pro zajímavost, protože je to jazyk primárně určený k interpretaci webovým prohlížečem. Přestože v posledních letech je čím dál víc používán také na straně serveru, v rešerši jazyků nebyl uveden. Důvodem je menší míra seznámení studentů TUL s tímto jazykem.

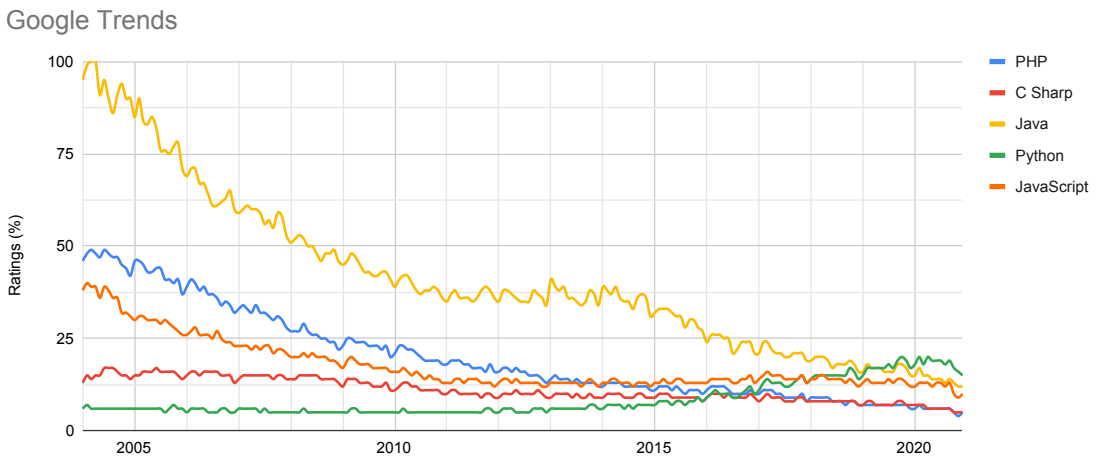
Na dalším obrázku 3.3 je graf podle již zmíněného indexu TIOBE. Tento je taktéž založený na vyhledávání, ale napříč různými platformami, proto je možné se domnívat, že jeho výpovědní hodnota má vyšší význam. Zde je vidět strmý nárůst oblíbenosti Pythonu až od roku 2018. Java má konstantně sestupný trend, ale přesto si drží nejméně padesátiprocentní náskok oproti C#.

IEEE Spektrum je internetový i tištěný časopis mezinárodní neziskové organizace jménem *Institut pro elektrotechnické a elektronické inženýrství* (IEEE). Ten vydává každoročně žebříček nejoblíbenějších jazyků. A to dokonce s přihlédnutím k jednotlivým odvětvím vývoje, které člení do čtyř kategorií, na aplikace webové, mobilní, desktopové a tzv. vestavěné. Tento žebříček je dohledatelný od roku 2014.

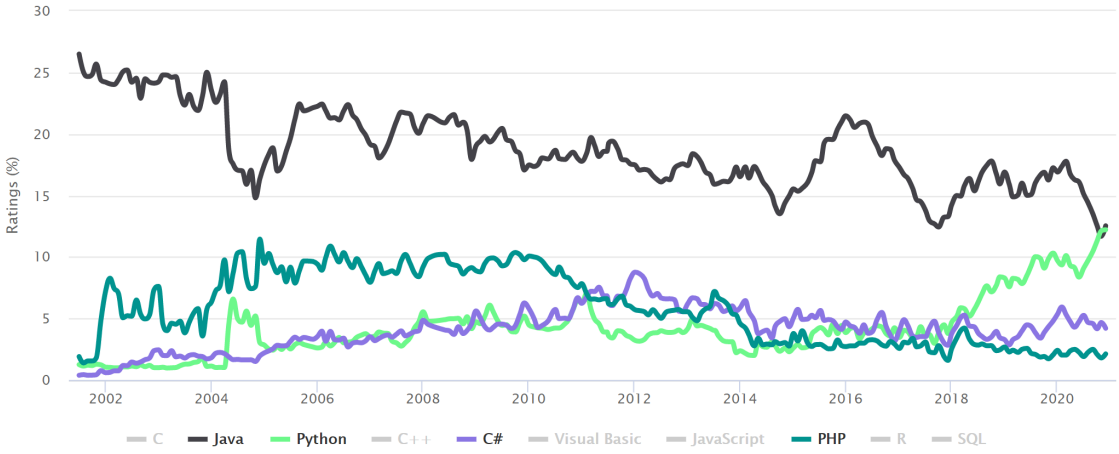
Obrázek 3.4 zobrazuje graf deseti nejpoužívanějších jazyků pro jednotlivé roky

[3] Podporuje více než jedno programové paradigma, jako je *objektové*, *procedurální (imperativní)*, či *funkcionální (deklarativní)*.

[4] Internet of Things – obvykle málo výkonná a energeticky úsporná zařízení zapojená do internetu. Mohou to být žárovky, termostaty, bezpečnostní prvky nebo další drobná domácí elektronika, stejně jako dopravní vozidla.

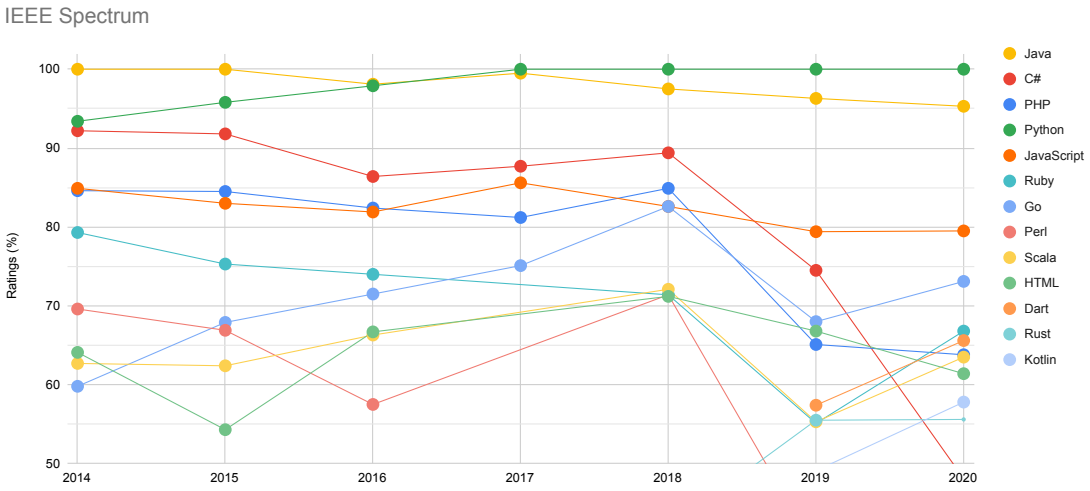


Obrázek 3.2: Trend ve vyhledávání[8]



Obrázek 3.3: Oblíbenost podle indexu TIOBE[5]

v kategorii webových aplikací. Z následujícího a výše uvedených důvodů je tento graf pro účely práce nejrelevantnější. Žebříček, ze kterého vychází, zahrnuje kromě dat z internetového vyhledávání také trendy na sociálních sítích, kterými jsou Twitter, Reddit, komunit serverů Stack Overflow a Hacker News, a v neposlední řadě z repozitáře GitHub, či volných pracovních pozic serveru CareerBuilder.



Obrázek 3.4: Index IEEE Spektrum – webový vývoj[9]

Z tohoto grafu vyplývá, že Python se začal vyrovnávat Javě v odvětví webového vývoje již v roce 2016, v roce 2017 pak získal převahu, kterou nadále zvyšuje.

V tomto ohledu je vhodné zmínit PHP, které je překvapivě, po dobu těchto sedmi měřených let, až na čtvrté, nebo horší pozici.

Ze všech tří grafů vyplývá nejperspektivněji jazyk Python. Jeho vhodnost je podpořena faktem, že se hojně využívá pro práci s daty a vědecké výpočty, i výukou na TUL.

V dalších částech se práce zaměří na vývoj aplikace s využitím jazyka Python.

3.3 Výběr webového frameworku

Na požadavek zadavatele, který vyjádřil přání umožnit další vývoj aplikace, částečně odpovídá také tato kapitola, která se zabývá výběrem webového frameworku.[5]

Udržitelnost aplikace zčásti závisí na schopnosti programátora psát čistý a přehledný kód. V tom mu může pomáhat framework, který poskytuje jasnou a pokud možno dobře dokumentovanou strukturu aplikace.[10] Český PHP framework Nette má na své propagační stránce uvedenou větu: „*Opravdoví programátoři nepoužívají*

[5] Aplikační rámec – slouží jako podpora při programování tím, že řeší některé obvyklé problémy a umožňuje tak soustředit se na specifika zadání. Může využívat různé knihovny, návrhové vzory, poskytovat API atd.

frameworky. Píší webové aplikace přes příkazovou řádku rovnou na server. Tímto jim vzdáváme hold. Nám ostatním Nette ohromným způsobem ulehčí a zpříjemní práci."[11] Práce se bude tohoto motta držet také a zvláštní důraz bude klást ještě na kvalitu dokumentace, která je klíčová k jeho dobrému pochopení a umožnění danému frameworku podávat optimální výkon. Při výběru bude tedy největší důraz kladen právě na kvalitu jeho dokumentace, její udržovanost a aktivní vývoj.

Použití frameworku odpovídá také na další bod požadavků, a tím je bezpečnost výsledné aplikace. Webové frameworky obsahují bezpečnostní vrstvy, které automaticky zajišťují ochranu například před XSS[6] útoky, vkládání *CSRF*[7] *tokenu* do požadavků POST, ochranu před *injektováním SQL* nebo proti *Clickjackingu*.

Pro to, aby mohl toto framework dělat správně, je zapotřebí držet se již zmiňované dokumentace a ideálně dodržovat tzv. *best practices* neboli osvědčené postupy.

Aby byla zajištěna co největší nezávislost aplikace na použité databázi, bude požadováno *objektově relační mapování* (ORM). Díky němu nebude nutné psát SQL dotazy typické pro různé databáze. Tento případ by mohl nastat, pokud by v navazující činnosti na tuto práci vyvstal požadavek otestovat jiné databázové systémy a posléze na některý z nich migrovat.

Bude tedy požadováno, aby vybraný framework byl především

- dobře dokumentovaný,
- aktivně vyvíjený
  - nebo alespoň udržovaný,
- neobsahoval známé bezpečnostní problémy,
- umožnil abstrakci použité databáze.

3.3.1 Pyramid

Pyramid sám sebe prezentuje jako malý framework, který roste spolu s požadavky. Malou aplikaci lze obsáhnout i jedním souborem, struktura a použití aplikačních nástrojů poskytuje programátorovi značnou volnost, což je možné považovat za výhodu, ale také nevýhodu. Zvláště začínající programátor tak může sklouznout k psaní velkých souborů a špatné strukturalizaci kódu.

Jeho tvůrci se při vývoji částečně inspirovali webovým frameworkem pro Python Django, viz kapitola 3.3.5. Zavázali se také k neustále aktuální a kvalitní dokumentaci[12], ta se ale subjektivně nezdá být tak přehledná a čtivá, jako je tomu například u zmíněného Djanga.

K napojení na databázi využívá ORM prostřednictvím sady nástrojů SQLAlchemy. Ta aktuálně podporuje SQLite, Postgresql, MySQL, Oracle, MS-SQL, Firebird, Sybase a další relační databáze prostřednictvím balíčků, které lze doinstalovat.

[6] Cross site scripting – útok založený na podstrčení škodlivého kódu na straně klienta. To se typicky provede vložením do databáze, například formou komentáře pod článkem blogu.

[7] Cross site request forgery – technika umožňuje uživateli se zlými úmysly provádět akce pomocí pověření jiného uživatele bez jeho vědomí nebo souhlasu.

Mezi webové stránky vytvořené pomocí Pyramid patří newcars.com, SurveyMonkey nebo PwnedList.[13]

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response


def hello_world(request):
    return Response('Hello World!')


if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

Listing 1: Příklad aplikace Pyramid

3.3.2 Bottle

Bottle je jednoduchý framework, často se o něm hovoří jako o mikroframeworku. Nemá žádné další závislosti, krom standardní knihovny Pythonu. Hodí se pro použití v jednodušších webových aplikacích. Výhodou je integrovaný testovací server, který zjednoduší vývojový proces.

```python
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

Listing 2: Příklad aplikace s Bottle

3.3.3 Flask

Flask si klade za cíl být jednoduchý, dobře srozumitelný a plně rozšiřitelný, podobně jako je tomu u frameworku Bottle. Ve výchozím nastavení tedy Flask neobsahuje vrstvu pro abstrakci databáze, ověřování formulářů, ani nic jiného. Na rozdíl od Bottle má ale ambice být součástí i rozsáhlých projektů, podobně jako je tomu u frameworku Pyramid, a taktéž obsahuje integrovaný vývojový server.

Zajišťuje kompatibilitu s mnoha balíčky a knihovnami. Například pro implementaci tzv. signálů doporučuje knihovnu *Blinker*. Součástí dokumentace je také použití těchto knihoven. Výhodou je, že programátor tak není odkázán pouze na dokumentaci každé z používaných závislostí třetí strany, ale je při integraci veden samotnou dokumentací pro framework Flask. Dokumentace je dobře udržovaná, přehledná a podrobná.

Mezi velké společnosti, které používají Flask, patří Pineterest a LinkedInd.[14, 15]

Ukázku minimálního projektu je možné spatřit v kódu 3.

```
1  import os
2  from flask import Flask
3
4
5  app = Flask(__name__)
6  app.config.from_object(os.environ['APP_SETTINGS'])
7
8  # Rendrování základní stránky
9  @app.route('/')
10 def hello():
11     return "Hello World!"
12
13 # Odkaz s parametrem
14 @app.route('/<name>')
15 def hello_name(name):
16     return "Hello {}!".format(name)
17
18 if __name__ == '__main__':
19     app.run()
```

Listing 3: Příklad aplikace Flask

3.3.4 FastAPI

FastAPI je framework určený k vytváření API. Je velmi dobře optimalizovaný a orientovaný na rychlost. Hodí se spíše pro mikroslužby zaměřené na odbavení velkého množství požadavků.

Tento progresivně se vyvíjející framework vznikl v roce 2019 a do dnešní doby získal již přes 27 tisíc hvězd v online repozitáři GitHub. Má dobře zpracovanou dokumentaci a rostoucí podporu komunity.

Další jeho výhodou je automatická validace dat, generování dokumentace (*Swagger UI*, *ReDoc*) a asynchronní běh.

Ve svých projektech ho používá Uber (interně), Explosion AI, Microsoft (interně) a také Netflix.[15, 16]

3.3.5 Django

Django je nejkomplexnější framework ze všech. Staví na principu DRY (*Don't repeat yourself*), což se pozitivně odráží na počtu řádků, které musí programátor napsat. Weby mají většinou podobné potřeby a Django je připraveno pokrýt je všechny již v základní instalaci. Obsahuje tak řešení pro správu uživatelů, oprávnění, ověřování formulářů, cachování, zasílání emailů, signály napříč aplikací, generování dokumentace, RSS i Atom zdrojů, a jeho velkou předností je také automatické generování uživatelského rozhraní správce – Django admin.

Jeho komplexnost může být v začátcích překážkou, ale při lepším seznámení naopak napomáhá opravdu rychlému prototypování webových aplikací. Sám se prezentuje jako „webový framework pro perfekcionisty s termíny."[17]

Jedná se o nejrozšířenější framework pro Python. Má velkou uživatelskou základnu a stabilní vývoj. Jeho dokumentace je rozsáhlá, ale přehledná, a dobře strukturovaná.

Mezi velké společnosti, které používají Django patří Disqus, Instagram, Pinterest, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Open Stack Coursera, Udemy.[13, 17]

3.3.6 Souhrn

Specifikace zde uvedených populárních webových frameworků jsou uvedeny v tabulce 3.1. Krom FastAPI jsou všechny výše uvedené webové frameworky postaveny na základě architektury MVC (Model, View, Controller), respektive jeho modifikace MTV (Model, Template, View).

Framework	Poslední stabilní verzce	Licence	MVC	ORM
Pyramid	9. listopadu 2020 (V1.10.5)	BSD	ano	ne*
Bottle	11. listopadu 2020 (V0.12.19)	MIT	ano	ne*
Flask	3. dubna 2020 (V1.1.2)	BSD	ano	ne*
FastAPI	20. prosince 2020 (V0.63.0)	MIT	**	ne*
Django	1. února 2021 (V3.1.6)	BSD	ano	ano

* Lze použít knihovnu třetí strany, například SQLAlchemy.

** Dostupná dokumentace informaci neuvádí.

Tabulka 3.1: Srovnání webových frameworků

Všechny zde uvedené frameworky mají vydání poslední verze v roce 2020, v případě Django dokonce 2021. U frameworku Bottle je ale zapotřebí upozornit, že V0.12.0 byla vydána již v roce 2013. Naopak u Django je cyklus aktualizací jasně daný a předem předvídatelný (každé dva roky vychází verze LTS, tedy verze s prodlouženou podporou).

Django bylo vybráno jako nejvhodnější Framework pro tuto práci. Důvodem je jeho dobrá udržovanost, kvalitní dokumentace a široká uživatelská komunita. Pro tuto práci poskytne vhodné prostředí pro prototypování aplikace a s využitím strojově generovaného rozhraní Django admin. Dále zajistí správu uživatelů a jejich oprávnění.

3.4 Vykreslování grafů

V aplikaci bude zapotřebí vykreslovat grafy. V ideálním případě by se mělo jednat a automatické generování responzivních a plně interaktivních grafů, tedy s možností přibližovat konkrétní části grafu a zobrazit funkční hodnotu v daném bodě. Obsah grafu by měl být plněn daty filtrovanými standardním prostředím Django admin.

Toho je možné dosáhnout několika způsoby. Bylo by možné vykreslit graf na serveru a odesílat jako bitmapu. To není vhodná varianta z důvodu většího datového přenosu. A nebylo by možné splnit ani požadavek na interaktivitu grafu. Z toho důvodu takové řešení není možné použít.

V případě vykreslování pouze sloupcových grafů je možné přistoupit k nejjednodušší formě, a tou je vykreslení pomocí samotného HTML, jak popisuje následující podkapitola.

3.4.1 Použití HTML

Využití HTML by mohlo mít pro koncového uživatele několik výhod. Z grafu bude možné kopírovat text nebo funkční hodnoty. Také bude možné texty strojově překládat či jinak zpracovávat. Graf může být také částečně interaktivní, aniž by bylo nutné do něj zasahovat. Můžeme tak dosáhnout zobrazení hodnot po najetí myši nad sloupec (atribut `href`).

Tato kapitola by také měla sloužit jako seznámení se s šablonovacím systémem Django a úpravami administračního rozhraní.

Šablonovací systém je velmi podobný systému Jinja2. Zápisu dominuje čisté HTML doplněné o funkční bloky a značky if, for a podobně. Tyto značky se uzavírají mezi znak procenta a složených závorek. Dále je možné v šabloně přistupovat ke kontextu daného pohledu jako k proměnné. Jednotlivé proměnné se uzavírají do dvou složených závorek za sebou.

Šablony je možné dědit, ale také importovat část šablony z jiného souboru. Tato vlastnost napomáhá znovupoužitelnosti kódu i přehlednosti.

V příkladu 4 je poděděna šablona (řádek 1), kterou Django používá ke generování výpisu entit (objektů) v administračním rozhraní. Zde je volný blok v patičce, který je možné využít pro účely vykreslení sloupcového grafu. Tento blok je shodou okol-

ností prázdný, ale není zaručeno, že v dalších verzích Djanga nebude použit, z toho důvodu je na řádku 4 umístěn blok `super`, ten zprostředkuje volání kódu z rodičovské šablony. Díky tomu je možné upgradovat další verze Django beze změny takto vytvořené šablony.

Šablona předpokládá data přepočtená na procenta vzhledem k nejvyššímu bodu grafu. Cyklus začínající na řádku 9 postupně prochází data, která bude vykreslovat jako sloupce pomocí bloků `div`. Tento kód je oddělen v samostatném souboru (kód 5) a vkládán s předáním příslušných hodnot.

```
1  {% extends "admin/change_list.html" %}
2
3  {% block footer %}
4      {{ block.super }}
5
6      <div id="footer" class="flex footer">
7          <h2>V čase</h2>
8          <div class="bar-chart">
9              {% for period in chart.data %}
10                 {% include 'admin/bar.html' with d=period
   ↪  period=period.period color=chart.color.0 %}
11             {% endfor %}
12         </div>
13     </div>
14 {% endblock %}
```

Listing 4: Soubor templates/admin/change_list_with_html_chart.html

```
1  {% load humanize filters_extras l10n %}
2
3  <div class="bar" style="height:{{ d.pct|unlocalize
   ↪  }}%;background-color:{{ color|default:'blue' }};">
4      <div class="bar-tooltip">
5          <b>{{d.value|default:0|intcomma }} {{chart.unit}}</b><br>
6          <small>{{period|date:"D,j.E.y"}}</small>
7      </div>
8      <span class="beak">{# zobáček obláčku #}</span>
9  </div>
```

Listing 5: Soubor templates/admin/bar.html

Šablony Django umožnují také práci s proměnnými za pomocí tzv. filtrů. Filtry se píší za symbol svislé čáry. Například použitý filtr `default`, po kterém za dvoujtečkou následuje hodnota, doplní tuto hodnotu místo obsahu proměnné, pokud je proměnná z hlediska logických výroků vyhodnocena jako `false`.

Obrázek 3.5: Ukázka sloupcového HTML grafu

V ukázce je také použita sekvence závorek `{# #}`, těmito znaky se uzavírá komentář na úrovni šablonovacího systému Django, text uvnitř není použit ve vyrendrovaném HTML.

Protože SVG je shodně s HTML formát založený na XML, bylo by možné analogicky postupovat i při tvorbě složitějších grafů generováním vektorové grafiky.

3.4.2 Balíček pro Django

Přestože se často uvádí, že jednou z předních výhod Django je veliké množství rozšiřujících balíčků, pro automatické generování grafů v prostředí Django admin žádný uspokojivý neexistuje.

Pro příklad je možné uvést udržovanou a populární JavaScriptovou knihovnu ChartJS. Ta doporučuje použití balíčku django-chartjs, který byl vytvořen v roce 2014 a od té doby je již pouze udržována kompatibilita s novými verzemi Django.[18] Navíc nezajišťuje automatickou integraci do administračního rozhraní.

3.4.3 JS knihovna

Nejlepší volbou pro další vývoj je vytvoření vlastního řešení za pomoci některé z JavaScriptových knihoven pro vykreslování grafů přímo na klientovi.

Mezi takové knihovny patří

- open source
  - D3.js – 95,6 K
  - ChartJS – 52,1 K
  - CHARTIST.JS – 12,5 K
  - APEXCHARTS – 9,9 K

- komerční s možností bezplatného použití
  - Highcharts JS – 9,9 K
  - amCharts – 880
  - Google Charts – 280
  - ZingChart – 235
- komerční s možností vyjednat licenci pro studijní účely
  - KOOL CHART

a ještě mnoho dalších. Všechny tyto knihovny jsou založené buďto na dynamickém vykreslování SVG nebo vykreslování na HTML5 Canvas. Pokud daný projekt má repozitář na GitHubu, pak za pomlčkou je uveden počet jeho hvězdiček.

Všechny zmíněné knihovny je možné doporučit a splňují uvedené požadavky. V testovací verzi byla pro účely dalšího vývoje vybrána knihovna ChartJS. Důvodem je aktivní open source vývoj, velká uživatelská základna a dobrá dokumentace.

4 Aplikace

Pomocí vybraného frameworku Django byl vytvořen základ aplikace. Také byl vytvořen klon originálního databázového serveru. Na tomto klonu je možné pracovat, aniž by bylo ovlivněno nebo ohroženo současné měření. Napojení Djanga na databázi předpokládá vytvoření modulů (objektů), které je možné následně mapovat na databázové entity.

Cílem této práce není suplovat dokumentaci, ani poskytnout podrobný popis tvorby demonstrační aplikace. Stručně popsány budou z toho důvodu pouze některé provedené kroky.

Standardní vytvoření aplikace používá přístup „code first", kdy se vytvoří nejprve potřebné modely, a poté se pomocí skriptu manage.py vytvoří příslušné entity automatizovaně. Django ale počítá i s možností, kdy je potřeba se připojit k již hotové a naplněné databázi. To je umožněno opět pomocí skriptu manage.py s přepínačem `inspectdb`. Příkaz vygeneruje hrubou kostru modelů, které je možné použít k dalšímu rozšiřování tak, aby odpovídaly potřebám Django aplikace.

Aby v prvních fázích projektu nebylo žádným způsobem zasahováno do současné databáze, byla oddělena databázová struktura potřebná k měření od databáze potřebné ke správě uživatelů. Za tímto účelem byla implementována třída `DatabaseRouter`, který směřuje dotazy na zvolené databázové servery.

Django poskytuje také *cache framework*, ten byl úspěšně testován v režimu ukládání do lokálního souborového systému. A následně také do operační paměti díky použití databáze Redis.

Administrační rozhraní je Djangem generováno pro registrované modely. Poskytuje jednoduché volby, kterými lze toto rozhraní upravovat. Například zvolit požadované řádky ve výpisu objektů, jejich řazení, prohledávání podle zvolených atributů nebo filtrování. Pokud nestačí základní filtrování, které je založené na atributech modelu, lze definovat vlastní. Tato možnost zatím v projektu nebyla využita, nýbrž je doporučena pro další vývoj. Dále je možné implementovat hromadné akce. Pokud v budoucnu vznikne požadavek na přepočítání a uložení přepočtených hodnot zpět do databáze (jak je popsáno v kapitole 4.2), jednou z možností je implementace tímto způsobem.

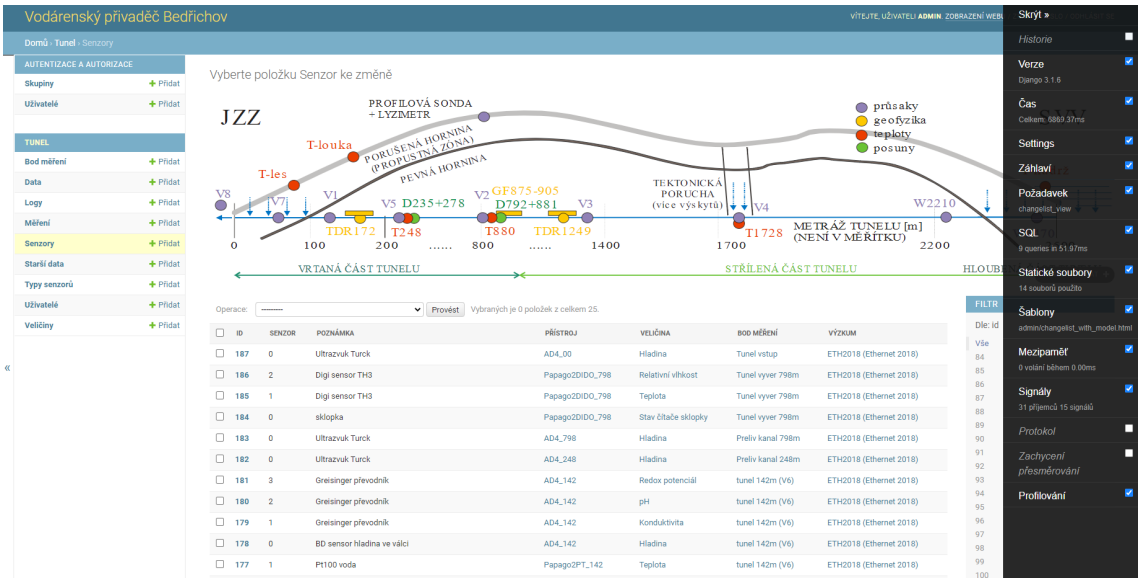
Byly také provedeny zásahy do základních šablon, podobně jako je popsáno v kapitole 3.4.1. Takto byla do šablon přidána knihovna ChartJS. Ta je nyní načítána z externích zdrojů, kterými jsou servery CDN[1]. Prostředí Django admin s integro-

[1] Content delivery network – Síť serverů rozmístěná v různých částech internetové infrastruktury tak, aby zvyšovala dostupnost dat.

vaným grafem je zobrazeno na obrázku 4.2.

Naplnění grafů daty je řešeno skrze předávání připravených dat šabloně již při renderování. Toto řešení je vhodné pro testování grafové knihovny, ale nehodí se do produkčního prostředí. Pro další vývoj je doporučeno načítat data dynamicky pomocí techniky AJAX[2].

Aby bylo možné přepočítávat hodnoty v databázi na reálné fyzikální veličiny, byla přepsána metoda `get_queryset`. Při implementaci vznikl problém popsaný v kapitole 4.2, z toho důvodů nyní aplikace nepřistupuje k datům jednotně, a pro některé senzory není načítání přepočtených hodnot funkční.



Obrázek 4.1: Ukázka z prototypové aplikace s panelem Django Debug Toolbar

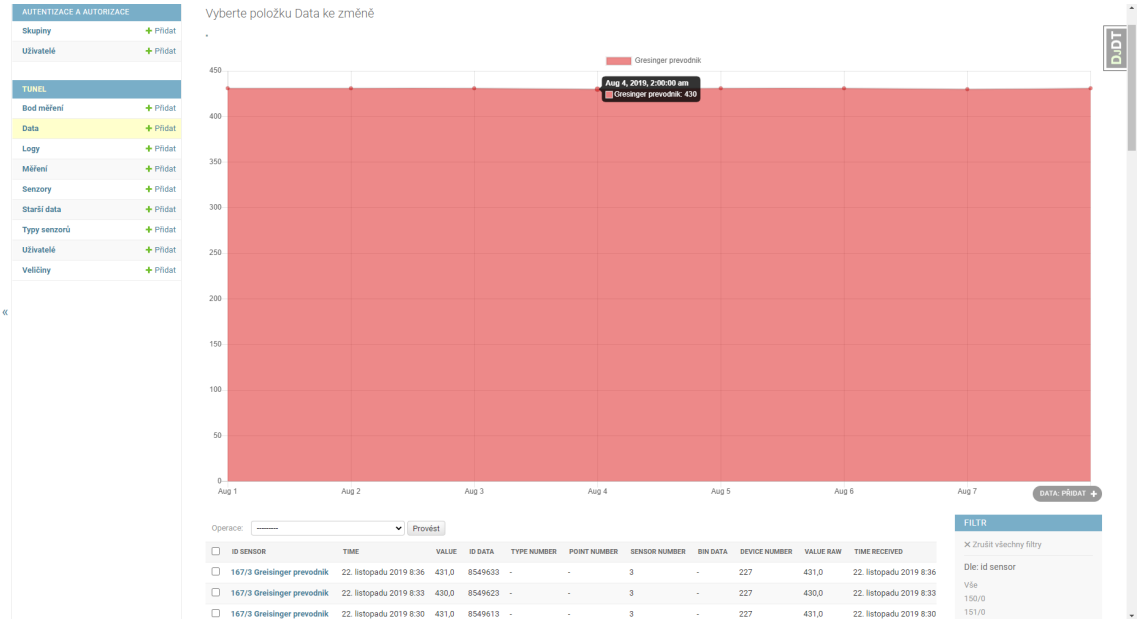
Základní instalace Djanga byla také rozšířena dvěma balíčky. Prvním je *django-import-export*, který zprostředkovává export dat v požadovaném (CSV, JSON, XLS a další) formátu pro účely analýzy pomocí externího softwaru.

Druhým je balíček *django-debug-toolbar*. Panel s možnostmi jeho výpisu je zobrazen na obrázku 4.1. Ten zásadním způsobem rozšiřuje možnosti ladění a optimalizování vykonávaných požadavků. Umožňuje zobrazení vykonaných dotazů na databázi a jejich časové zatížení celého požadavku. Jeho prostřednictvím lze také kontrolovat obsah paměti cache.

4.1 Databáze

Bylo zjištěno, že běžící databáze je PostgreSQL verze 9.2.24. Tato verze byla vydána v roce 2012 a její podpora byla ukončena v září roku 2017. Nyní je aktuální verze

[2]Asynchronous JavaScript and XML – označení technologií, které umožňují načítání dat bez nutnosti opětovného načtení celého obsahu stránky.



Obrázek 4.2: Ukázka z prototypové aplikace s grafem ChartJS

PostgreSQL 13.1.0. Od verze 9.2.X prošla tato databáze vývojem, který opravoval důležité bezpečnostní problémy, přidával nové funkce, ale také zásadním způsobem navyšoval výkon. Optimalizována byla efektivnost zpracování dotazů a zajištěno paralerní zpracování jednoho dotazu na více jádrech procesoru.

Z těchto důvodů je doporučena migrace aktuální databáze na nejnovější verzi PostgreSQL.

Schéma na obrázku 4.3 znázorňuje dominantní část databáze, která se týká ukládání dat z měření.

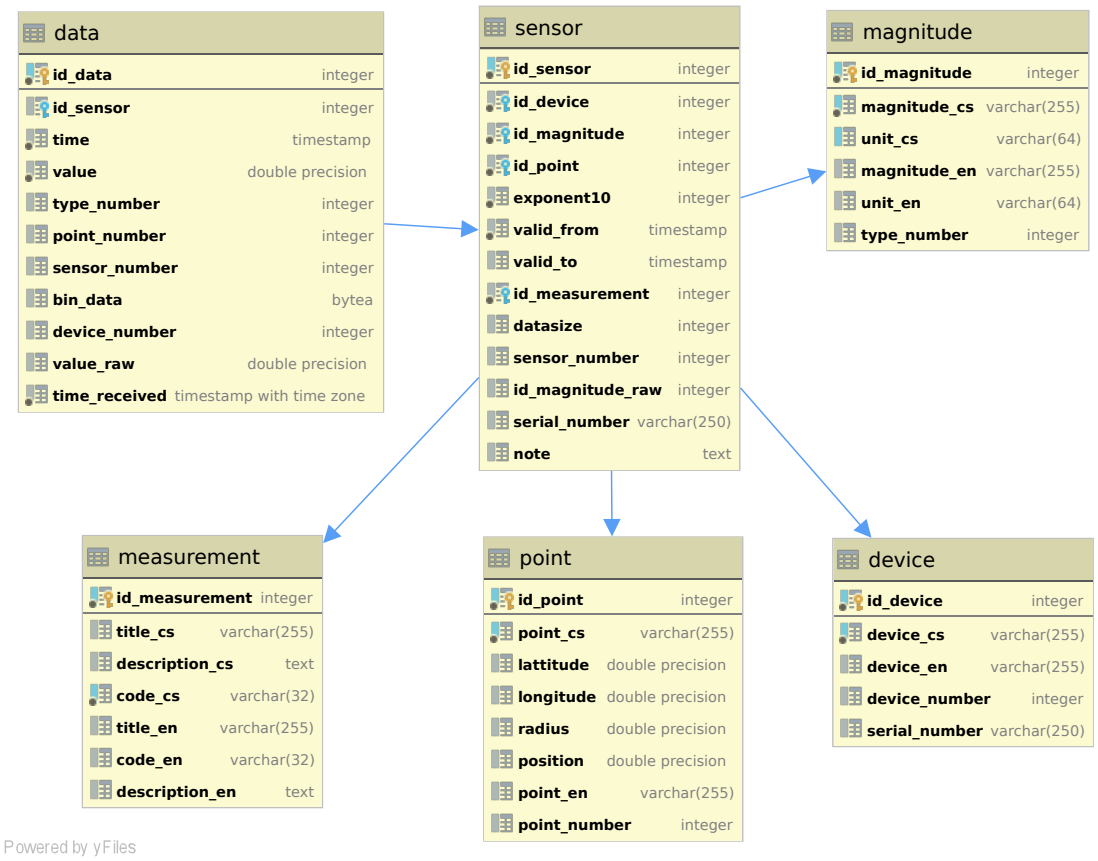
Slovy lze vazby mezi entitami volně popsat následovně. *Data* jsou měřena senzory (*sensor*). Každý senzor má definovaný rozměr (*magnitude*) veličiny, kterou měří.

Jednotlivé senzory jsou součástí nějakého měřícího zařízení (*device*) a měří na určeném místě v tunelu (*point*). Senzory lze také shlukovat podle toho, zda patří k nějakému výzkumu, v tomto případě říkejme měření (*measurement*).

4.2 Implementace přepočtů hodnot

V kapitole 2.2.1 je popsán výpočet, který předpokládá v každé iteraci přístup ke dvěma záznamům s určitým posunem. Jak již bylo řečeno, aplikace by měla využívat jednotný přístup k databázovému stroji na bázi ORM, a proto není možné sestavit komplexní dotaz, který by pro každý záznam počítal výsledek na straně databáze.

Django ORM od verze 2.2 implementuje funkci *window*, ta umožňuje vybírat z tabulky data a k nim přiřazovat data z té samé tabulky, ale s určitým posunem (*lag*). To je pro výpočet vhodné, ovšem pouze s omezením. Při použití funkce *window* totiž není možné na stejný dotaz aplikovat agregační funkce. Krom toho tato



Obrázek 4.3: UML diagram databáze

funkce má známé problémy s výkonem. Možností by bylo obejít ORM a napsat příslušný dotaz explicitně, i za cenu nesplnění předpokladu nezávislosti implementace na použité databázi.

Z hlediska výkonu dotazování se lepší možností jeví zásah do databáze. Přidáním atributu v tabulce *data* by bylo možné mít vypočítanou hodnotu připravenou pro přímé načtení. Výpočet této hodnoty je možné za pomoci *triggru* navázat na akci ukládání nové hodnoty a odstranit výpočetní prostředky potřebné při načítání, kdy je kritickou hodnotou čas potřebný k vykonání dotazu.

Pokud by byl zajištěn předpoklad, že každá měřená veličina je v tabulce uložena v základních jednotkách SI (aktuálně není), bylo by možné k nim přistupovat jednotně. To by značně ulehčilo práci programátorovi výsledné aplikace pro zobrazování a umožnilo generování administračního prostředí automaticky.

5 Závěr

V práci se podařilo nastudovat převody surových dat uložených v databázi na fyzikální veličiny. Z toho vyplynuly nároky na databázi a aplikaci. Byl identifikován problém s výpočtem napříč různými záznamy databáze a jeho dopadem na softwarové prostředky a výkon.

Byl zjištěn nedostatek neaktuální databáze a navrženo přidání atributu s přepočtenou fyzikální veličinou. Tato změna zásadně zjednoduší práci s databází a výkon prováděných dotazů za cenu jen malého navýšení ukládání dat.

Dále se v práci podařilo vybrat nástroje softwaru, kterými je možné splnit požadavky popsané v kapitole 3.1. Pro aplikaci byl vybrán jazyk Python a framework Django. Výhodou tohoto řešení je snadná tvorba administračního rozhraní, které je generováno z modelu spjatého s databází. Django také zajišťuje správu uživatelů a lze pomocí něho snadno zajistit uživatelské role a jejich oprávnění.

Zhotovená aplikace běží v testovacím režimu na lokálním počítači a testována byla také na serveru.

Použitá literatura

1. HOKR, Milan. *Poster Bedrichov tunnel - overview of partner institution contributions (in Czech)*. 2016.
2. HOKR, Milan. *Měření a vyhodnocení průsaků ve vodovodním přivaděči Bedřichov*. 2017. seminář TESEUS.
3. DOC. ING. MILAN HOKR, PH.D. *TUNEL 2011*. Liberec, 2014-03.
4. *Programming Languages Definition | TIOBE - The Software Quality Company* [online] [cit. 2020-12-06]. Dostupné z: https://www.tiobe.com/tiobe-index/programming-languages-definition/.
5. *TIOBE - The Software Quality Company* [online] [cit. 2020-12-06]. Dostupné z: https://www.tiobe.com/tiobe-index/.
6. C Sharp (programming language). In: *Wikipedia* [online]. 2020 [cit. 2020-12-16]. Dostupné z: https://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&oldid=992332648. Page Version ID: 992332648.
7. *Java vs C# - 10 Key Differences between Java and C#* [online] [cit. 2020-12-16]. Dostupné z: https://www.guru99.com/java-vs-c-sharp-key-difference.html.
8. *Google Trends* [Google Trends] [online] [cit. 2020-12-13]. Dostupné z: https://trends.google.cz/trends/explore?hl=cs&tz=-60&date=today+5-y&q=PHP,%2Fm%2F07657k,%2Fm%2F07sbkfb,%2Fm%2F05z1_,%2Fm%2F02p97&sni=3.
9. GMT, Stephen CassPosted 22 Jul 2020 {\textbar} 18:15. *Interactive: The Top Programming Languages 2020 - IEEE Spectrum* [IEEE Spectrum: Technology, Engineering, and Science News] [online] [cit. 2020-12-02]. Dostupné z: https://spectrum.ieee.org/ns/IEEE_TPL_2020/index/2020/1/0/0/0/1/50/1/50/1/50/1/30/1/30/1/20/1/20/1/5/1/50/1/100/1/50/.
10. BARÁŠEK, Jan. *Proč a jak používat frameworky a knihovny* [online]. 2020 [cit. 2020-12-16]. Dostupné z: https://php.baraja.cz/proc-pouzivat-frameworky.
11. 2008, Nette Foundation; c. *Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP* [online] [cit. 2021-02-01]. Dostupné z: https://nette.org/cs/.
12. *Vítejte v Pyramid, Python Web Framework* [online] [cit. 2021-02-05]. Dostupné z: https://trypyramid.com/.

13. TANEJA, Sheetal; GUPTA, Pratibha R. Python as a tool for web server application development. *JIMS8I-International Journal of Information Communication and Computing Technology*. 2014, roč. 2, č. 1, s. 77–83. Publisher: Jagan Institute of Management Studies, Rohini.

14. *Odpověď Steva Cohena na Jaké výzvy se Pinterest setkal s Flaskem? - Quora* [online] [cit. 2021-02-06]. Dostupné z: https://www.quora.com/What-challenges-has-Pinterest-encountered-with-Flask/answer/Steve-Cohen?srid=hXZd&share=1.

15. *Rachel Sanders: Developing Flask Extensions - PyCon 2014* [online]. Ve spol. s PYCON 2014. 2014 [cit. 2021-02-06]. Dostupné z: https://www.youtube.com/watch?v=OXN3wuHUBP0#t=46&ab_channel=PyCon2014.

16. JUAN CRUZ MARTINEZ. *Quickly Develop Highly Performant APIs with FastAPI & Python* [online]. 2020-08-05 [cit. 2021-02-10]. Dostupné z: https://livecodestream.dev/post/quickly-develop-highly-performant-apis-with-fastapi-python/.

17. *Django overview | Django* [online] [cit. 2021-02-08]. Dostupné z: https://www.djangoproject.com/start/overview/.

18. *peopledoc/django-chartjs* [GitHub] [online]. Ve spol. s RÉMY HUBSCHER [cit. 2021-02-09]. Dostupné z: https://github.com/peopledoc/django-chartjs.