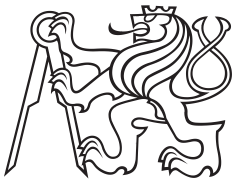
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

# Dolování znalostí z bezdrátových senzorových sítí

Bc. Jan Zídek

Vedoucí: Ing. Stanislav Vítek, Ph.D.
Obor: Otevřená informatika
Zaměření: Datové vědy
Květen 2019

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zídek** Jméno: **Jan** Osobní číslo: **423320**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Datové vědy**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Dolování znalostí z bezdrátových senzorových sítí**

Název diplomové práce anglicky:

**Data mining in wireless sensor networks**

Pokyny pro vypracování:

Cílem diplomové práce je vytvoření systému, který umožní uživateli získat představu o kvalitě životního prostředí ve svém okolí a to s co nejvyšším časovým a prostorovým rozlišením. Hlavním cílem systému je vytvoření prostředí pro sběr a fůzi dat z různých zdrojů, sekundárním cílem je návrh modelu pro krátkodobou predikci trendu vývoje kvality ovzduší.
1. Zabývejte se možnostmi získávání environmentálních dat z bezdrátových senzorových sítí (WSN) se zaměřením na kvalitu ovzduší.
2. Prozkoumejte možnosti fúze dat z WSN s daty z jiných zdrojů, např. data ze satelitů zaměřená na atmosférickou chemii (mise ESA Sentinel, NASA Terra, ...).
3. Navrhněte a implementujte serverovou část platformy pro online zpracování dat a GraphQL rozhraní pro prezenční vrstvu.
4. Navrhněte a implementujte model, který na základě dostupných dat vytvoří predikci stavu ovzduší v lokalitě uživatele v následující hodině. Navrhněte vhodný testovací režim a měřítka kvality predikce.

Seznam doporučené literatury:

[1] APPICE, Annalisa, et al. Data mining techniques in sensor networks: Summarization, interpolation and surveillance. Springer Science & Business Media, 2013.
[2] FOUAD, Mohamed Mostafa, et al. Data mining and fusion techniques for WSNs as a source of the big data. Procedia Computer Science, 2015, 65: 778-786.

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Stanislav Vítek, Ph.D., katedra radioelektroniky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **14.02.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

| Ing. Stanislav Vítek, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc.<br>podpis děkana(ky) |
|---|---|---|

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

# Poděkování

V první řadě bych chtěl poděkovat svému vedoucímu Ing. Stanislavu Vítkovi, Ph.D. za vedení této práce. Hned vzápětí děkuji Centru Znalostního Managementu za trpělivost, podporu a prostor pro tvorbu této práce.

Nesmím také opomenout vyjádřit velké díky prof. Matasovi za jeho konzultaci.

V poslední řadě chci poděkovat všem mým blízkým a to hlavně mojí rodině a přítelkyni, protože bez jejich nekončící trpělivosti a obrovské podpory by tato práce pravděpodobně ani nemohla vzniknout.


# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2019

Jan Zídek

# Abstrakt

Tato práce se zaměřuje na oblast kvality ovzduší. Především se zabývá návrhem a implementací systému na sběr a zpracování dat o kvalitě ovzduší na území hlavního města Prahy.

Součástí této práce je také rešerše znečišťujících látek a hlavně jejich vlivu na životní prostředí, především na kvalitu ovzduší. Součástí rešerše jsou také dopady na lidské zdraví.

Nedílnou součástí je také rešerše různých zdrojů dat o kvalitě ovzduší. Především jsou to senzory Českého hydrometeorologického ústavu, vlastní přenositelné senzory a satelitní data vlastněné NASA a ESA.

Jako sekundární cíl této práce je návrh a implementace modelu pro předpověď kvality ovzduší z hlediska polétavého prachu $PM_{10}$ na následující hodinu.

Vzniklá aplikace je tak stavebním kamenem pro vznik komplexního systému pro sběr a zpracování dat o kvalitě ovzduší a následné možné prezentaci obyvatelům Prahy.



**Klíčová slova:** dolování znalostí, Java EE, Payara, Mongo, kvalita ovzduší, znečištění ovzduší, znečišťující látky, $PM_{10}$, $PM_{2,5}$, lineární regrese

**Vedoucí:** Ing. Stanislav Vítek, Ph.D.



# Abstract

This thesis focuses on the field of study of air quality. The prime goal is design and implementation of a system for collecting and processing data of air quality of capital city Prague.

The part of this thesis is also a research of pollutants and their impact on the Environment, especially the effect on air quality. There are also mentioned impacts on human health as a part of this research.

The next important part is the research of possible data sources of air quality data. The main data sources are sensors of Czech Hydrometeorological Institute, own small movable sensors and satellite sensors owned by NASA and ESA.

The second goal of this thesis is to design and implement a model for prediction of the concentration of particulate matter $PM_{10}$ for the next hour.

This thesis is a good starting point for developing a complex system for collecting and processing air quality data. The next step is to present this data to the citizens of Prague.



**Keywords:** data mining, Java EE, Payara, Mongo, air quality, air pollution, pollutants, $PM_{10}$, $PM_{2,5}$, linear regression

**Title translation:** Data mining in wireless sensor networks

# Obsah

# Obrázky



# Tabulky

# Kapitola 1
# Úvod

V této kapitole se snažím uvést čtenáře do problematiky, kterou se práce zabývá. Dále zde představuji motivaci a cíl této práce. V neposlední řadě zde také uvádím strukturu práce.

## 1.1 Předmluva

Ve své diplomové práci se zabývám tvorbou základu systému, který bude sloužit pro zpracování dat kvality ovzduší na území Prahy. Data budou sbírána z různých zdrojů a díky tomu bude možné dávat přesnější informace pro některé části Prahy. Tomu by měly napomoci i malé, přenosné senzory na měření polétavého prachu.

Kromě sběru informací se také zabývám jejich zpracováním. Konkrétně jejich interpolace, aby v každé části Prahy byla dostupná alespoň přibližná hodnota. Mimo interpolace se zabývám i extrapolací dat pro polétavý prach pro konkrétní lokace, kde jsou umístěny profesionální senzory od Českého hydrometeorologického ústavu.

## 1.2 Motivace a cíl

V dnešní době je životní prostředí a především jeho kvalita velice diskutované téma. Moderní civilizace nezanedbatelně zatěžuje přírodu spalováním například fosilních paliv, používáním agresivní chemie v zemědělství, nadužíváním léků, těžbou, atd. To pochopitelně vede ke znečišťování půdy, vody i ovzduší a může následně vyvolávat zdravotní potíže. Především ve velkých městech dochází k nadměrnému znečišťování ovzduší a je tedy na místě se touto problematikou zabývat.

Tato práce má za cíl vytvořit systém pro informování obyvatel Prahy o aktuálním stavu kvality ovzduší. Výhodou takového systému bude především využití několika dostupných zdrojů informací obohacených o data z malých přenosných senzorů. Sekundárním cílem je také vytvořit predikční model a informovat tak uživatele systému o kvalitě ovzduší v následující hodině. Uživatel se tak bude moci rozhodnout, zda je zdravé si jít zaběhat nebo zda je lepší zůstat doma a nevětrat.

## 1.3 Struktura práce

Tato práce je dělena na 3 hlavní celky. Na teoretickou část, praktickou část a na závěr s přílohami.

### 1.3.1 Teoretická část

V teoretické části jsou veškeré potřebné informace pro seznámení se tématem kvality ovzduší, shrnuty zdroje dat a jejich následné zpracování.

V první kapitole se věnuji látkám, které ovzduší znečišťují, jak vznikají a jaké mají vliv na životní prostředí a především na naše zdraví. Dále zde rozebírám výpočet indexu kvality ovzduší.

V druhé kapitole představuji možné zdroje dat kvality ovzduší. Postupně zde představím profesionální senzory Českého hydrometeorologického ústavu, možnost rozšíření této senzorové sítě malými přenositelnými senzory a na závěr taky možnosti využití satelitních dat od evropské a americké kosmické agentury.

V poslední kapitole této části se věnuji zpracování výše zmíněných dat. Rozebírám zde metody interpolace i extrapolace těchto dat pomocí metody lineární regrese.

### 1.3.2 Praktická část

V praktické části se zabývám návrhem implementací systému pro sběr dat a tvorbou predikčního modelu pro hodnoty polétavého prachu.

V první kapitole se blíže zaměřuji na návrh výsledného systému od požadavků na nově vznikající systém, přes výběr vhodných technologií až po architekturu aplikace.

V další kapitole se pak věnuji už nějakým konkrétním zajímavým částem aplikace z hlediska jejich implementace. Zaměřuji se hlavně na popis možnosti rozšíření systému o další funkcionality a zabezpečení.

Dále v této části věnuji samostatnou kapitolu návrhu modelu pro předpověď. V této kapitole se také věnuji experimentům a vyhodnocení kvality modelu.

V poslední kapitole jsou výsledky jednotlivých experimentů a jejich následná diskuze a vyhodnocení.

### 1.3.3 Závěr a přílohy

V poslední částí sumarizuji výsledky této práce. Na úplném konci je pak v přílohách k nalezení seznam použité literatury, seznam zkratek, seznam použitých technologií a nástrojů, návod k instalaci aplikace a obrázky z instalace.

# Část I

# Teoretická část

# Kapitola 2
# Kvalita ovzduší

V této kapitole se zabývám otázkou, co je to vlastně kvalita ovzduší a které látky ovzduší znečisťují. V neposlední řadě se také věnuji definici indexu kvality ovzduší, který slouží ke klasifikaci.

## 2.1 Znečišťující látky

V této sekci krátce představím jednotlivé látky, které způsobují znečištění ovzduší. U každé z nich uvádím, jak vzniká a jakou hraje roli v otázce kvality ovzduší. Primárně se zaměřuji na látky, které poskytují možné datové zdroje. Těm se blíže věnuji v kapitole 3. Zvláště se zaměřím na prachové částice $PM_X$, které plánujeme měřit i vlastními přenosnými senzory.

### 2.1.1 $PM_X$

Následující odstavec je kompilací informací ze zdrojů [1, 2, 3].

$PM_X$ je označení pro částice nebo také polétavý prach; zkratka pochází z anglického Particulate Matter (PM). Mohou být pevné, kapalné i směsné. Písmeno $X$ v dolním indexu označuje maximální velikost částice v $\mu m$. $PM_{10}$ jsou tedy částice menší nebo rovno $10\,\mu m$. Hodnota $PM_X$ vyjadřuje průměrný počet částic za časové období, které je obvykle 1 h nebo 24 h.

Běžně měřenými hodnotami jsou hrubé $PM_{10}$, jemné $PM_{2,5}$ a výjimečně se také měří $PM_1$, na které je potřeba už velmi citlivých přístrojů. Některé zdroje pracují i s ultra-jemnými částicemi označované jako $PM_{0,1}$. Obecně platí, že čím menší částice, tím je pro člověka nebezpečnější. Vlivem těchto částic na lidské zdraví se blíže věnuji v podsekci 2.1.1.

#### $PM_{10}$

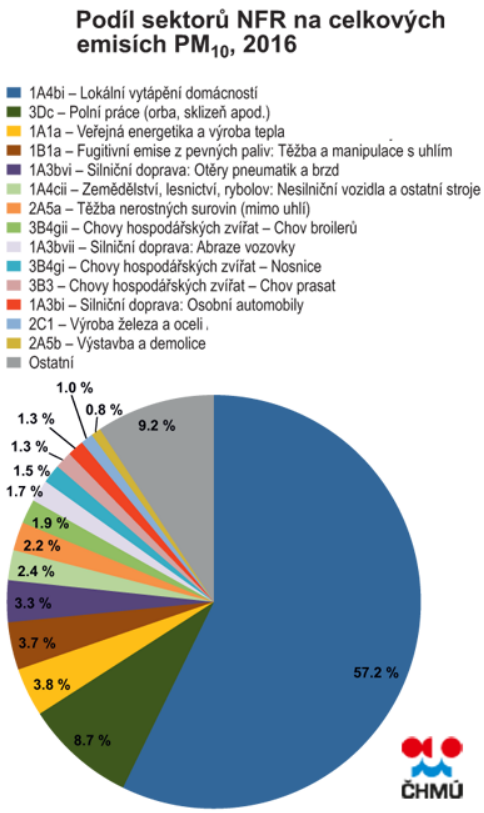
Tato pod-sekce je kompilací informací ze zdrojů [1, 2, 3, 4].

Hrubé prachové částice $PM_{10}$ jsou prachové částice o velikosti $\leq 10\,\mu m$. Přirozeně se v atmosféře nachází díky sopečné činnosti, z lesních požárů, odpařováním slané mořské vody a eroze. Za vznik těchto částí může ale také člověk a to hlavně spalováním. Mezi nejčastější zdroje tedy patří topení v domácnosti a doprava, ale také těžba, zemědělství a samozřejmě i průmysl. U

dopravy je ještě významná tzv. sekundární prašnost. Ta vzniká tak, že vozidla svou jízdou zvedají usazený prach na vozovce. Jak se jednotlivé činnosti podílejí na vniku $PM_{10}$ v České republice je vidět na grafu 2.1.

Množství částic v atmosféře se mění podle ročního období. To je dané hlavně venkovní teplotou, protože se při nízkých teplotách více topí.

$PM_{10}$ mají kromě zdravotních dopadů, kterým se blíže věnuji v samostatné pod-sekci 2.1.1, také dopady na životní prostředí. Mezi ně patří například snižování viditelnosti, smog nebo okyselování půd a vody.

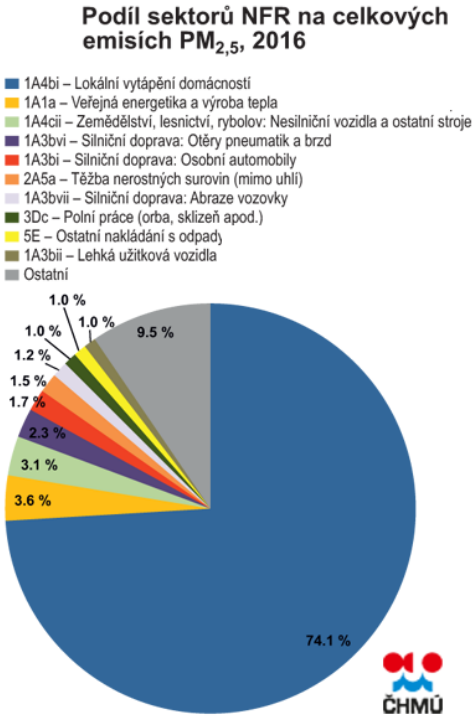



**Obrázek 2.1:** Podíl sektorů NFR na celkových emisích $PM_{10}$ v roce 2016, zdroj: [4]

### $PM_{2,5}$

Tato pod-sekce je kompilací informací ze zdrojů [3, 5].

$PM_{2,5}$ se, na rozdíl od $PM_{10}$, označují jako "jemné" a jedná se o částice o velikosti $\leq 2,5\,\mu m$. Stejně jako $PM_{10}$ vznikají hlavně spalováním, a proto jsou významnými zdroji spalovací motory, elektrárny, vytápění a požáry. Jednotlivé podíly těchto jevů jsou v grafu 2.2.




**Obrázek 2.2:** Podíl sektorů NFR na celkových emisích $PM_{2,5}$ v roce 2016, zdroj: [5]

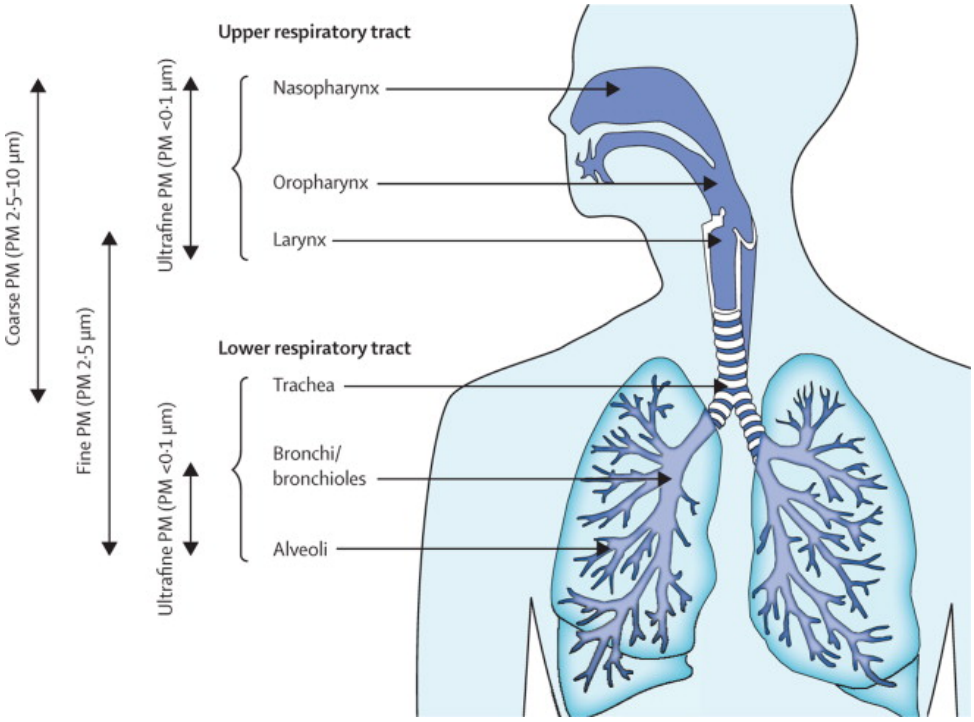
### Vliv na lidské zdraví

Tato pod-sekce je kompilací informací ze zdrojů [1, 3, 6, 7].

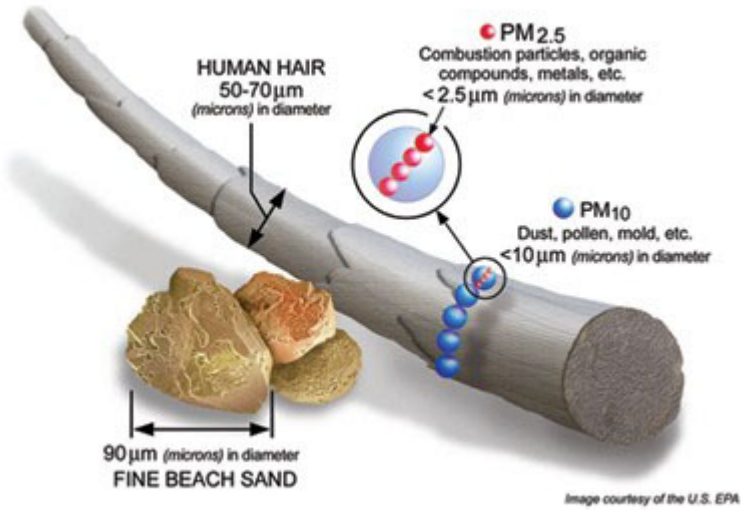
Krátkodobé vystavení organismu znečištěnému ovzduší způsobuje hlavně dušnost, astmatické záchvaty a dochází k podráždění očí, nosu a krku. Dlouhodobé vystavení může vést k chronickým zánětům dýchacích cest, infarktu, změnám genomu a rakovině.

Čím menší částice, tím horší následky může mít pro lidské zdraví. Velké viditelné znečištění dráždí hlavně oči, nos a krk. Částice velikosti $PM_{10}$ se zachytávají v hlavě a v horních cestách dýchacích. Způsobují tak hlavně podráždění, dušnost a při dlouhodobém vystavení záněty. Jemné částice velikosti $PM_{2,5}$ se usazují v průduškách a částice $PM_1$ se dostávají až do plicních sklípků a způsobují nejenom plicní problémy, ale taky onemocnění kardiovaskulární soustavy, včetně infarktu. Tyto částice na sebe vážou další pro člověka nebezpečné látky, například těžké kovy, a vytvářejí prostředí pro chemické reakce. Mohou tak způsobovat i rakovinu. Nejnebezpečnější jsou pro člověka částice $PM_1$ a menší. Ty se mohou dostat až do krevního řečiště a následně tak do celého těla. Pronikání jednotlivých velikostí do organismu je znázorněno na obrázku 2.3. Pro lepší představu je na obrázku 2.4 porovnání velikosti částic a lidským vlasem.

Větší riziko ze znečištěného ovzduší je hlavně pro starší lidi a malé děti.




**Obrázek 2.3:** Pronikání prachových částic do organismu, zdroj: [6]




**Obrázek 2.4:** Porovnání $PM_{10}$ a $PM_{2,5}$ s lidským vlasem, zdroj: [3]

### 2.1.2 $NO_X$

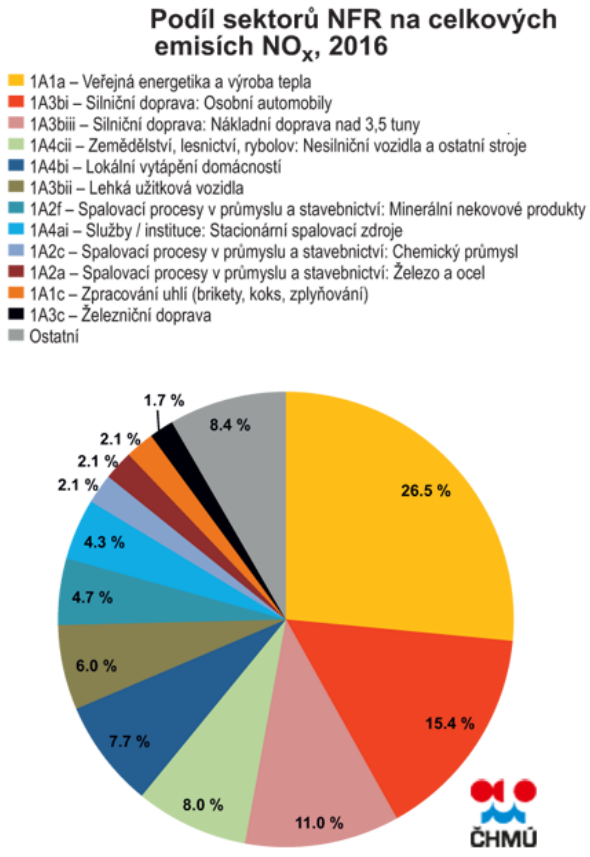
Jako $NO_X$ jsou označovány oxidy dusíku, konkrétně oxid dusnatý ($NO$) a oxid dusičitý ($NO_2$). Mohou reagovat s kyslíkem a vytvářet aerosoly, které způsobují kyselé deště, a ty poškozují velké zemědělské plochy i přírodní

ekosystémy, včetně těch vodních. Na vniku $NO_X$ má největší podíl energetika a doprava, jak je vidět na grafu 2.5. Díky $NO_X$ dochází i k sekundárnímu znečištění, protože mohou vytvářet směsné $PM$.

U lidí způsobuje dýchací obtíže a zvyšuje riziko chronických plicních onemocnění, včetně zánětů a astmatu. Při nízkých koncentracích způsobuje podráždění očí a horních cest dýchacích. Při vyšších koncentracích způsobuje popálení dýchací soustavy, nevolnostem, křečím, snížení okysličení organizmu a může nastat i smrt. Nejrizikovější skupinou jsou děti.

$NO_2$ vzniká oxidací $NO$ jako sekundární znečištění. Emise $NO_2$ tvoří asi 10 % emisí $NO_X$. $NO_2$ také přispívá k vzniku přízemního ozonu za pomoci UV záření. Více o ozonu píšu v pod-sekci 2.1.5.

Zdroje: [8, 9, 10, 11, 12]




**Obrázek 2.5:** Podíl sektorů NFR na celkových emisích $NO_X$ v roce 2016, zdroj: [10]

### 2.1.3 $SO_2$

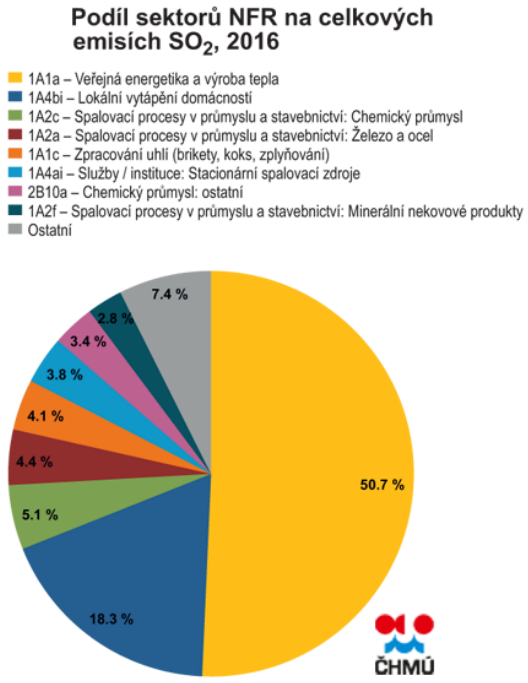
Oxid siřičitý ($SO_2$) je bezbarvý, štiplavě páchnoucí plyn. Stejně jako $NO$, tak i $SO_2$ mohou reagovat s kyslíkem za vzniku kyselých aerosolů, které následně způsobují kyselé deště. A nejenom to, stejně tak mohou tvořit směsné $PM$.

Kromě přirozených zdrojů, jako je vulkanická činnost nebo požáry lesů, má největší podíl na vniku $SO_2$ energetika a vytápění domácností. Jednotlivé podíly těchto sektorů jsou v grafu 2.6.

U rostlin reaguje s chlorofylem a narušuje tak jejich schopnost fotosyntézy. Lidem dráždí oči a dýchací cesty a podporuje vznik zánětů a astma. Vysoké koncentrace mohou vést k vážným poškozením plic a dýchacích cest. Citlivější skupinou jsou starší a chronicky nemocní lidé.

Oxid siřičitý se také používá v potravinářství jako konzervant sušených meruněk nebo vína.

Zdroje této kompilace: [9, 13, 8, 14]




**Obrázek 2.6:** Podíl sektorů NFR na celkových emisích $SO_2$ v roce 2016, zdroj: [13]

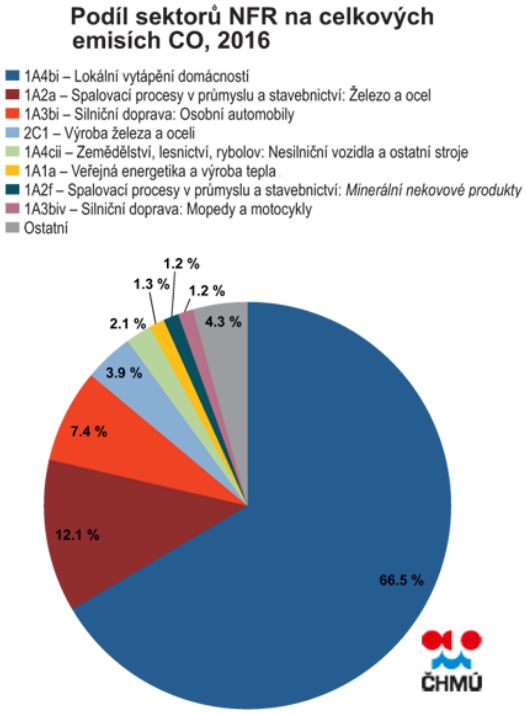
### 2.1.4 *CO*

Tato pod-sekce je kompilací zdrojů: [7, 8, 15, 16].

Oxid uhelnatý ($CO$) je bezbarvý plyn bez zápachu, který vzniká především při nedokonalém spalování fosilních paliv (za přítomnosti malého množství kyslíku, nízké teploty). Největšími zdroji jsou vytápění, průmysl a doprava. Jednotlivé podíly zdrojů na znečištění oxidem uhelnatým jsou vyobrazeny na grafu 2.7. Přirozeně vzniká například vulkanickou činností a dalšími procesy, které souvisejí s hořením. Nachází se také v cigaretovém kouři. Pro člověka je jedovatý a jeho nebezpečí je hlavně v tom, že se dostává do krve, kde se váže na hemoglobin. Tato vazba je silnější než s kyslíkem. Oxid uhelnatý tak

zabraňuje hemoglobinu vázat na sebe kyslík a rozvádět ho tak po těle.

Při nižších koncentrací může člověk pociťovat únavu. Při vyšších koncentracích dochází k poruchám vidění a koordinace, bolestem hlavy, závratím, zmatečnému chování a může být pociťována žaludeční nevolnost. Velmi vysoké koncentrace jsou smrtelné.




**Obrázek 2.7:** Podíl sektorů NFR na celkových emisích $CO$ v roce 2016, zdroj: [16]

### 2.1.5 $O_3$

Pod-sekce je kompilace ze zdrojů [9, 17].

Ozon ($O_3$) nemá žádný významný přirozený zdroj. Vzniká jako sekundární látka při fotochemických reakcí. Prekurzory pro vznik $O_3$ jsou oxidy dusíku ($NO_X$), nemetanogenní těkavé organické látky ($NMVOC$), metan ($CH_4$) a oxid uhelnatý ($CO$). Důležitou reakcí je fotolýza $NO_2$ pomocí ultra-fialového záření, při které vzniká $NO$ a atomární kyslík $O$. Ten pak reaguje za přítomnosti katalyzátoru s molekulárním kyslíkem ($O_2$) a vzniká tak ozon ($O_3$). Jedná se tedy o sekundární znečištění. Kromě již zmíněných fotochemických reakcí se také může při bouřkách dostat ozon do troposféry ze stratosféry, kde se vyskytuje přirozeně.

Při vdechnutí způsobuje u lidí podráždění dýchacích cest a bolesti hlavy. U rostlin způsobuje poškození listů, dále poškozuje lesy a obilí.

Důležité je zmínit fakt, že za znečišťující látku se považuje pouze troposférický, též nazývaný přízemní ozon, nikoliv stratosférický. Ten je naopak pro

naši planetu prospěšný, protože brání vstupu UV záření ze Slunce.

## 2.2 Index kvality ovzduší

V této sekci se zaměřuji na to, jak se na základě naměřených hodnot dá kvalita ovzduší klasifikovat.

Po světě se používá několik různých škál, jak definovat index kvality ovzduší. Zde ve své práci uvádím 2 definice. Jednu, kterou používá Český Hydrometeorologický ústav a Common Air Quality Index (zkráceně CAQI). V Evropě převažuje klasifikace kvality ovzduší právě pomocí CAQI. Avšak ve světě existuje celá řada systémů indexace kvality ovzduší. Například v USA se využívá AQI systém od U.S. EPA (zdroj [18]).

### 2.2.1 Definice dle Českého Hydrometeorologického ústavu

Definice indexace kvality ovzduší je volně převzata ze zdroje [19].

Český Hydrometeorologický ústav (CHMI) pro klasifikace kvality ovzduší využívá naměřené hodnoty látek $PM_{10}$, $NO_2$, $SO_2$, $CO$ a $O_3$. U každé látky se pro zadanou lokalitu vypočítá klouzavý průměr za určité období. Ten se spočítá tak, že pro jednu měřenou látku se vezmou data pro určitou lokalitu za časové období a z těchto dat se udělá aritmetický průměr. Vyjde nám tak průměrný počet částic dané látky.

Běžná hodnota časového období pro měřené látky je 1h. U látek $PM_X$ se obvykle počítá i průměr za 24h, avšak pro výpočet indexu kvality ovzduší u CHMI se nevyužívá. Speciálně pak průměr pro $CO$ se počítá za 8h období.

Pro každou látku zvlášť se spočítá index kvality ovzduší dle tabulky 2.1. Výsledný index kvality ovzduší v dané lokalitě je maximální index napříč všemi měřenými látkami.

| Index | Kvalita ovzduší | $PM_{10}$ 1h | $NO_2$ 1h | $SO_2$ 1h | $CO$ 8h | $O_3$ 1h |
|---|---|---|---|---|---|---|
| 1 | velmi dobrá | 0<br>20 | 0<br>25 | 0<br>25 | 0<br>1000 | 0<br>33 |
| 2 | dobrá | 20<br>40 | 25<br>50 | 25<br>50 | 1000<br>2000 | 33<br>65 |
| 3 | uspokojivá | 40<br>70 | 50<br>100 | 50<br>120 | 2000<br>4000 | 65<br>120 |
| 4 | vyhovující | 70<br>90 | 100<br>200 | 120<br>350 | 4000<br>10000 | 120<br>180 |
| 5 | špatná | 90<br>180 | 200<br>400 | 350<br>500 | 10000<br>30000 | 180<br>240 |
| 6 | velmi špatná | >180 | >400 | >500 | >30000 | >240 |

**Tabulka 2.1:** Definice indexu kvality ovzduší tabulkou. Hodnoty jsou v $\mu g/m^3$, zdroj: [19]

### 2.2.2 Common Air Quality Index

Stejně jako index kvality od CHMI i zde se využívá klouzavého průměru jednotlivých látek za určité období v dané lokalitě. Index kvality ovzduší se klasifikuje pro každou kategorii látek zvlášť a výsledný index kvality ovzduší je potom maximální index napříč všemi měřenými látkami. Určení dané kategorie kvality ovzduší se provádí tabulkou 2.2.

Na rozdíl od CHMI ale přidává do svého výpočtu i $PM_{2,5}$, a to jak klouzavý průměr za 1 h, tak 24 h. U $PM_{10}$ se pak navíc kromě klouzavého průměru za 1h využívá i 24h klouzavý průměr.

Zdroje: [20, 21]

| Index | Znečištění | $PM_{10}$ 1h | $PM_{10}$ 24h | $PM_{2,5}$ 1h | $PM_{2,5}$ 24h | $NO_2$ 1h | $SO_2$ 1h | $CO$ 8h | $O_3$ 1h |
|---|---|---|---|---|---|---|---|---|---|
| 1 | velmi nízké | 0<br>25 | 0<br>15 | 0<br>15 | 0<br>10 | 0<br>50 | 0<br>50 | 0<br>5000 | 0<br>60 |
| 2 | nízké | 25<br>50 | 15<br>30 | 15<br>30 | 10<br>20 | 50<br>100 | 50<br>100 | 5000<br>7500 | 60<br>120 |
| 3 | střední | 50<br>90 | 30<br>50 | 30<br>55 | 20<br>30 | 100<br>200 | 100<br>350 | 7500<br>10000 | 120<br>180 |
| 4 | vysoké | 90<br>180 | 50<br>100 | 55<br>110 | 30<br>60 | 200<br>400 | 350<br>500 | 10000<br>20000 | 180<br>240 |
| 5 | velmi vysoké | >180 | >100 | >110 | >60 | >400 | >500 | >20000 | >240 |

**Tabulka 2.2:** Definice CAQI tabulkou. Hodnoty jsou v $\mu g/m^3$, zdroj: [20, 21]

## 2.3 Shrnutí kapitoly

V této kapitole jsem v jednotlivých sekcích představil látky, které se podílejí na znečištění ovzduší. U každé látky jsem popsal, které oblasti lidské činnosti se na vzniku této látky podílí a v neposlední řadě jsem zmínil dopady na lidské zdraví i na životní prostředí.

V druhé části kapitoly jsem definoval CAQI index kvality ovzduší, který se v Evropě používá ke kategorizaci kvality ovzduší.

# Kapitola 3
# Zdroje dat

V kapitole "Zdroje dat" se věnuji zdrojům, ze kterých by aplikace mohla čerpat data o kvalitě ovzduší.

## 3.1 Pozemní senzory

V této sekci věnuji pozornost pozemním senzorům od Českého Hydrometeorologického ústavu a možnosti zpřesňování výsledků v některých lokalitách pomocí vlastních přenositelných senzorů.

### 3.1.1 Stacionární senzory

V České republice je hlavním zdrojem informací ohledně počasí, stavu vody a kvality ovzduší Český hydrometeorologický ústav[1] (zkráceně ČHMÚ nebo CHMI z anglického názvu). Jeho zřizovatelem je Ministerstvo životního prostředí České republiky.

CHMI poskytuje informace o kvalitě ovzduší z několika profesionálních stacionárních senzorů, které jsou rozmístěny po celé České republice, převážně však ve velkých městech. Nás budou zajímat jen ty, které jsou na území Prahy. Těch je v době psaní práce celkem 17.

Aktuální situace kvality ovzduší v Praze je poskytována hlavně v lidsky čitelné podobě na mapě[2] nebo na stránkách jednotlivých senzorů[3]. To ale není moc vhodný formát pro strojové zpracování. CHMI také poskytuje API[4] k datům o kvalitě ovzduší. Bohužel data v archivu jsou jen od června 2018.

Náš soukromý archiv obsahuje data od února 2017. Ten je realizován jen jako jednoduché stahování stránek jednotlivých senzorů CHMI. Bude tedy potřeba implementovat parsování dat z HTML stránek. Do budoucna ale bude rozhodně lepší implementovat aktualizaci dat skrze již zmiňované API.

[1] `http://portal.chmi.cz`

[2] `http://pr-asu.chmi.cz:8080/IskoPollutionMapView/faces/viewMapImages.xhtml`

[3] Přehled všech senzorů: `http://portal.chmi.cz/files/portal/docs/uoco/web_generator/actual_hour_data_CZ.html`

[4] Dokumentace k CHMI API: `https://golemio.docs.apiary.io/#reference/0/meteostanice-chmi/airquality-report-lastx-v2`

### 3.1.2 Přenositelné senzory

Dalším plánovaným zdrojem dat jsou naše vlastní a přenositelné senzory. Tyto senzory budou vybavené snímači prachových částic velikostí 2.5 $\mu m$ a 10 $\mu m$, vlhkosti a teploty. Samozřejmě budou mít také GPS lokátor pro určení polohy zařízení. Přední výhoda tohoto řešení bude jednoznačně v tom, že naše zařízení budou dostatečně malá na to, aby byla přenositelná. Tím půjde docílit snadného rozšíření již vybudované sítě stacionárních senzorů od CHMI a zlepšit tak přesnost měření v určitých lokalitách.

Tyto senzory budou odesílat data skrze LoRa síť na centrální server. Z toho serveru pak bude aplikace data stahovat a zpracovávat. Jako centrální server využijeme služby thethingsnetwork.org. Tato služba umožňuje zdarma připojit senzory přes LoRa síť a tyto data pak poskytovat přes RESTové rozhraní. Blíže se LoRa síti věnuji v následující podsekci.

Součástí této práce není tvorba těchto senzorů. V této práci implementuji pouze připojení těchto senzorů skrze již zmiňované rozhraní služby thethingsnetwork.org.

#### LoRa síť

Tato podsekce je kompilací informací ze zdrojů [22, 23, 24, 25, 26] a má za cíl ve zkratce čtenáře seznámit, co je LoRa síť a k čemu se využívá.

LoRa (Long Range) je bezdrátová síť určená pro malé IoT zařízení. Její přední výhoda je, že umožňuje zařízení připojit na vysokou vzdálenost s nízkou spotřebou. Což je u IoT zařízení velká výhoda.

Zatímco LoRa je "Lower Layer Protocol" a zajišťuje komunikaci mezi zařízeními, v síti se ještě využívá "Upper Layer Protocol" jménem LoRaWAN protokol. Ten umožňuje obousměrnou komunikoci nízko-energetické zařízení v síti s centrálním serverem. Tím je dále umožněno spojení se s aplikacemi v internetu. Specifikuje tak rychlost přenosu, frekvenci aktualizace dat ze zařízení nebo například FOTA aktualizace a zároveň optimalizuje výdrž baterie zařízení. Protokol definuje a spravuje LoRa Alliance, jejímž členy jsou například firmy Cisco, Fujitsu, Motorola Solutions nebo TKD.

## 3.2 Satelitní senzory

Kromě pozemních senzorů existují i projekty NASA i ESA, které měří kvalitu ovzduší z družic na oběžné dráze. Snímkovací frekvence však není tak častá jako u pozemních senzorů. O jednotlivých projektech blíže píšu v jednotlivých podsekcích níže.

### 3.2.1 ESA Sentinel

Sentinel[5] je projekt Evropské vesmírné agentury, který se zaměřuje na měření faktorů ovlivňujících životní prostředí. V rámci projektu Sentinel je realizováno

[5] `https://sentinel.esa.int/web/sentinel/home`

celkem 6 misí a každá z nich se zaměřuje na nějakou část. Sentinel-1 a Sentinel-2 mají primární úkol snímkovat planetu. Sentinel-3 sleduje oceán, barvu souše i oceánu a stejně tak i teplotu.

Pro tuto práci jsou nejzásadnější data z mise Sentinel-4, Sentinel-5 a Sentinel-5P. Ty sbírají data o látkách ovlivňujících kvalitu ovzduší jako jsou $O_3$, $NO_2$, $SO_2$ nebo aerosoly. Veřejně dostupná data jsou však pouze z mise Sentinel-5P a jsou dostupná na webu copernicus.eu v lidsky čitelné podobě ve formě mapy[6] i jako strojově zpracovatelná data skrze API[7]. Pro přístup k API je potřeba registrace. Celý postup je popsán v dokumentaci[8].

Pro lepší filtraci dat ESA zavedla orbitální čísla. Česká republika se nachází na absolutním orbitálním čísle 7534.

Jednotlivé datové produkty jsou ve formátu NetCDF. Jedná se o binární formát a k prohlížení dat lze využít nástroj ToolsUI[9], který je dostupný zdarma. Na stránkách www.unidata.ucar.edu, kde je nástroj dostupný, lze najít i knihovnu pro práci s formátem NetCDF v jazyku Java. V případě použití satelitních dat v budoucnu by se tato knihovna dala použít.

### 3.2.2 NASA Terra

NASA má také svůj projekt na zjišťování stavu klimatu pojmenovaný Terra[10]. Projekt se dělí na dalších 5 pod-projektů. ASTER[11] pořizuje snímky Země ve vysokém rozlišení nejenom ve viditelném spektru světla. CERES[12] měří tepelnou energii, kterou Země emituje a energii, kterou reflektuje. MISR[13] dokáže zjistit velikost a typ mraků, množství a typ částic aerosolů a typ povrchu, včetně struktury vegetativního porostu. MODIS[14] měří aerosoly v atmosféře, vypařování vody, sníh a led, mraky, barvu a teplotu oceánů. MOPITT[15] pozoruje $CO$ v troposféře, které může pocházet z dopravy, továren nebo požárů.

Z hlediska kvality ovzduší jsou nejzajímavější informace ohledně aerosolů. Atmosférická data jsou k dispozici na stránkách LAADS DAAC [16]. Konkrétní jsou dostupná po vyplnění "objednávky" přes formulář na stránce `https://ladsweb.modaps.eosdis.nasa.gov/search/`. Pro strojové zpracování je lepší využívat API[17].

Další zajímavý zdroj by mohla být mise MOPITT, ale bohužel v době psaní této práce nebyly webové stránky této mise funkční.

[6] Po přihlášení (s5pguest/s5pguest) na adrese `https://s5phub.copernicus.eu/dhus/#/home`

[7] `https://scihub.copernicus.eu/`

[8] `https://scihub.copernicus.eu/userguide/WebHome`

[9] `https://www.unidata.ucar.edu/downloads/netcdf/netcdf-java-4/index.jsp`

[10] `https://terra.nasa.gov/`

[11] `https://terra.nasa.gov/about/terra-instruments/aster`

[12] `https://terra.nasa.gov/about/terra-instruments/ceres`

[13] `https://terra.nasa.gov/about/terra-instruments/misr`

[14] `https://terra.nasa.gov/about/terra-instruments/modis`

[15] `https://terra.nasa.gov/about/terra-instruments/mopitt`

[16] `https://ladsweb.modaps.eosdis.nasa.gov/`

[17] Dokumentace k API: `https://ladsweb.modaps.eosdis.nasa.gov/tools-and-services/lws-classic/api.php`

NASA poskytuje data v HDF formátu. Stejně jako formát NetCDF je binární a k jeho otevření je potřeba speciálních nástrojů. Potřebné nástroje poskytuje přímo NASA na stránkách `https://eosweb.larc.nasa.gov/tools/`. Z jazyka Java lze formát HDF číst pomocí knihovny, která je k nalezení na `https://support.hdfgroup.org/products/java/`.

### 3.2.3 Data o počasí

Kvalita ovzduší je úzce spjata s vývojem počasí. Proto jsem se rozhodl, že se toho pokusím využít při tvorbě predikčního modelu. Tomu se blíže věnuji v samostatné kapitole 7.

Bohužel archiv dat kvality ovzduší od CHMI, který jsem popisoval v podsekci 3.1.1 obsahuje pouze hodnoty koncentrací jednotlivých znečišťujících látek. Archiv s daty o počasí nemáme k dispozici.

Archivní informace ohledně počasí jsou k dohledání na webových stránkách `https://en.tutiempo.net/`. Tyto stránky nabízí informace o teplotě, rychlosti větru, vlhkosti i tlaku několikrát do hodiny. Pro strojový přístup k datům jsem využil nástroje `curl` pro linuxovou příkazovou řádku.

Jelikož by takto stažená HTML stránka byla hůře strojově zpracovatelná, využil jsem ještě nástroje `xmllint` pro separaci tabulky s daty.

Celý příkaz vypadá takto:

```
curl -d "date=01-01-2019" --url "https://en.tutiempo.net/records/lkpr"
| xmllint --html --xpath '//div[@id = "HistoricosData"]' -
2>/dev/null 1>"01-01-2019.html"
```

Pro stažení kompletního archivu těchto dat za celé období sběru dat o kvalitě ovzduší jsem implementoval skript v jazyku bash, který přikládám jako přílohu.

# Kapitola 4
# Zpracování dat

V této kapitole věnuji pozornost způsobu, jakým budu data ze senzorů zpracovávat. V první části této kapitoly se věnuji interpolaci naměřených dat pro libovolnou lokaci včetně několika možných metod interpolace v prostoru.

## 4.1 Interpolace

Interpolace je úloha, která se snaží najít přibližné hodnoty uvnitř měřeného intervalu pro diskrétní data.

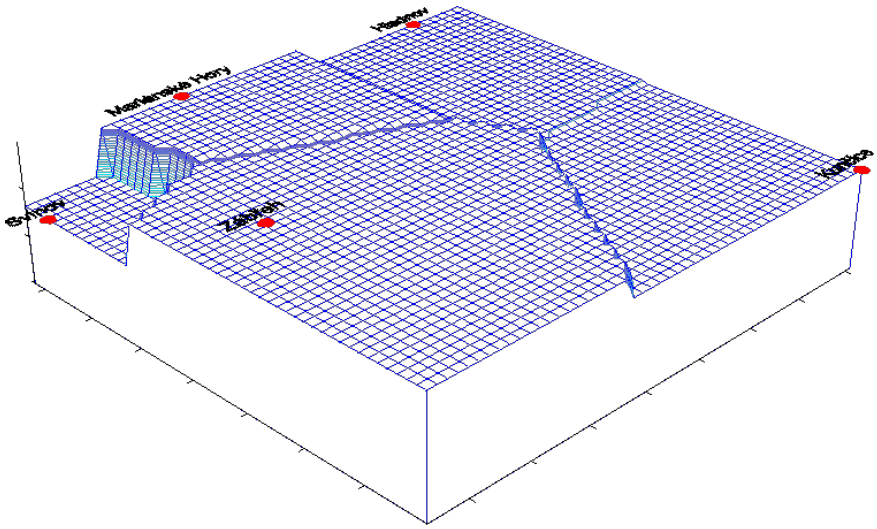
Data získaná ze zdrojů (popsaných v kapitole 3) jsou prostorová. Respektive jedná se o údaj naměřený v nějaké lokaci na Zemi. Pro jejich interpolaci tedy využiji metodu, která je vhodná pro prostorová data. Těch existuje celá řada jako například metoda nejbližšího souseda, trojúhelníková metoda nebo metoda inverzní vzdálenosti.

Metody interpolace v prostoru jsem čerpal ze zdrojů [27, 28, 29].

### 4.1.1 Metoda nejbližšího souseda

Nejjednodušší metodou interpolace v prostoru je metoda nejbližšího souseda. Ta jednoduše pro zadanou lokaci vyhledá nejbližší bod v 2D rovině, pro který zná hodnotu naměřených dat. Celý prostor se tak rozdělí na plochy polygonálních tvarů s konstantní hodnotou. Výsledek hodnoty je ukázán na obrázku 4.1.

Informace o metodě nejbližšího souseda jsem čerpal ze zdrojů [27, 28].



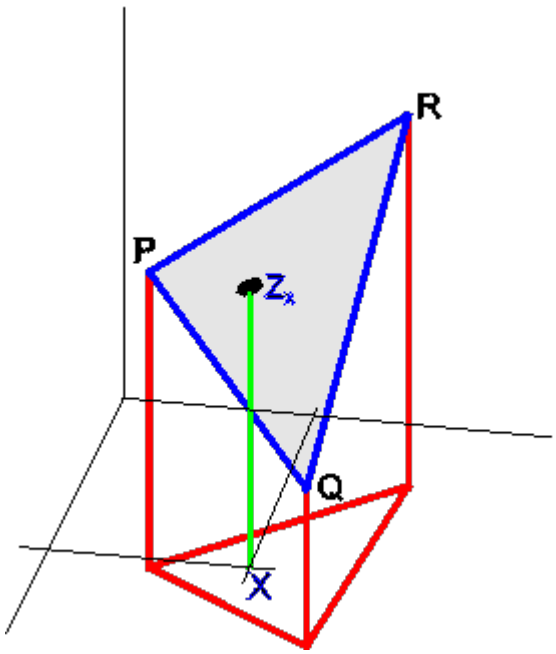
**Obrázek 4.1:** Ukázka metody nejbližšího souseda, zdroj: [27]

### 4.1.2 Trojúhelníková metoda

Trojúhelníková metoda je o něco sofistikovanější. Ta vezme 3 body, pro které známe naměřené hodnoty a pro tyto hodnoty sestrojí rovinu. Interpolovaná hodnota pak leží na této rovině.

Je zřejmé, že záleží na tom, jaké body pro tuto interpolaci zvolíme. Nejlepší tedy je zvolit 3 nejbližší sousedy pro žádanou lokaci.

Zdroje: [27, 28].




**Obrázek 4.2:** Ukázka trojúhelníkové metody, zdroj: [27]

### 4.1.3 Metoda inverzní vzdálenosti

Pro interpolaci dat kvality ovzduší jsem se rozhodl použít metodu inverzní vzdálenosti, protože největší váhu na tom, jak budou data v žádané lokaci vypadat, má nejbližší senzor. S rostoucí vzdáleností pak klesá relevance naměřených dat. Grafické znázornění interpolace je na obrázku 4.4.

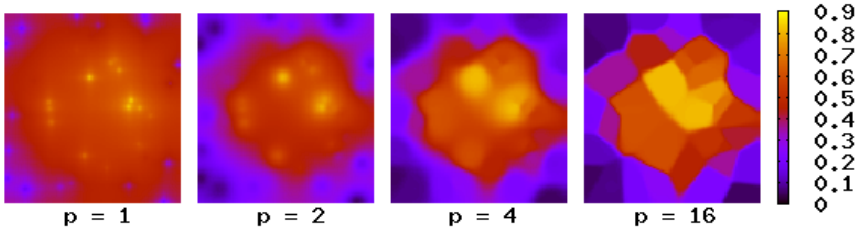
Vzorec pro výpočet interpolace vážené inverzní vzdálenosti:

$$u(x) = \begin{cases} u_i & pokud\, \delta(x_i, x) = 0\, pro\, nějaké\, i \\ \frac{\sum_i^n w_i(x) \cdot u_i}{\sum_i^n w_i(x)} & jinak \end{cases}$$

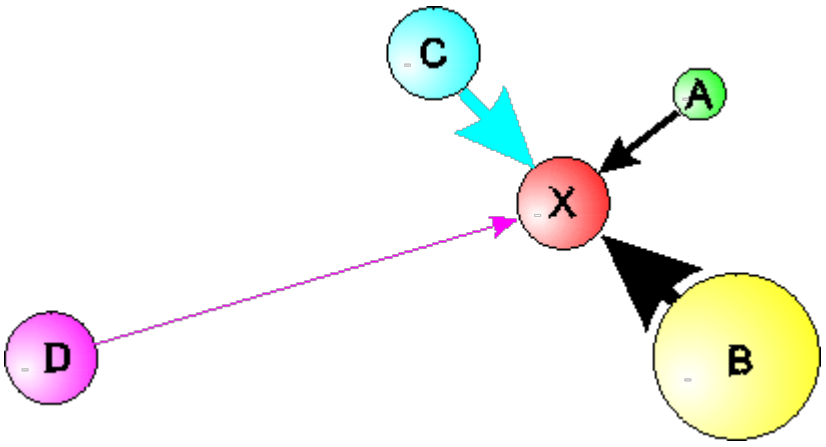
$$w_i(x) = \frac{1}{\delta(x_i, x)^p}$$

Jedná se tedy o vážený aritmetický průměr, kde váhy jsou dány jako $\frac{1}{\delta^p}$, kde $\delta$ je vzdálenost žádané pozice od bodu, ve kterém známe hodnotu. Parametrem $p$ se pak dá upravovat vliv vzdálených bodů. S rostoucím parametrem $p$ se zvětšuje vliv nejbližšího známého bodu. Pro hodně vysoké $p$ se tedy metoda inverzní vzdálenosti blíží metodě nejbližšího souseda. Vliv parametru $p$ je vidět na obrázku 4.3.

Zdroje: [27, 28, 29, 30].




**Obrázek 4.3:** Vlivu parametru $p$ na interpolovaná data pomocí metody inverzní vzdálenosti, zdroj: [30]




**Obrázek 4.4:** Ukázka metody inverzní vzdálenosti, zdroj: [27]

### Vzdálenost dvou souřadnic

Metoda inverzní vzdálenosti využívá vzdálenosti dvou bodů v rovině. Pro krátké vzdálenosti postačí délka úsečky $|AB| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$. Pro delší vzdálenosti už ale bude tento přístup velice nepřesný, protože se začne projevovat zakřivení Země.

Haversinova rovnice počítá vzdálenost dvou bodů na povrchu koule. Nezohledňují se tedy nerovnosti na Zeměkouli. Nic méně stále se jedná o přesnější odhad vzdálenosti než vzdálenost dvou bodů na rovině.

Definice Haversinovy vzdálenosti pro body $x_1$ a $x_2$, kde $\phi$ je latitude v radiánech a $\lambda$ je longitude také v radiánech:

$$x_1 = (\phi_1;\, \lambda_1)$$

$$x_2 = (\phi_2;\, \lambda_2)$$

$$\Delta\phi = \phi_2 - \phi_1$$

$$\Delta\lambda = \lambda_2 - \lambda_1$$

$$\delta(x_1,\, x_2) = 2 \cdot R \cdot sin^{-1}\left(\sqrt{sin^2\left(\frac{\Delta\phi}{2}\right) + cos(\phi_1) \cdot cos(\phi_2) \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

$R$ je poloměr Země; $R = 6378\,km$.
Zdroje Haversinovy rovnice: [31, 32].

## 4.2 Model pro předpověď

Součástí této práce je také vytvoření predikčního modelu na 1 hodinu dopředu. V této sekci se zabývám možnostmi extrapolace. Tedy nalezení přibližné hodnoty vně měřeného intervalu.

Pro účely extrapolace je vhodné použít regresní metody. Pro predikci hodnot prašnosti jsem se rozhodl použít lineární regresi.

### 4.2.1 Lineární regrese

Mějme množinu měření $X$ a k ní naměřené hodnoty $y$. Úkolem regrese je nalezení takové funkce $f(\vec{x_i})$, která nalezne přibližnou hodnotu $y'$ pro hodnoty $\vec{x_i} \in X$, která se od té skutečné liší co nejméně. V případě, kdy funkce $f(\vec{x_i})$ je lineární kombinací hodnot $\vec{x_i}$ hovoříme o tzv. lineární regresi.

$$f(\vec{x}) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

Pokud si zadefinujeme rozšíření vektoru $\vec{x}$ o 1 jako $\vec{x_{ext}}$, dá se zápis zkrátit na skalární součin.

$$f(\vec{x}) = \beta^T \vec{x_{ext}}$$

Obvykle je nalezení parametrů lineární regrese řešené metodou nejmenších čtverců. Tedy optimalizační úlohou minimalizace kvadratického rozdílu skutečné a aproximované hodnoty.

$$\vec{\beta_{opt}} = argmin_{\vec{\beta}} \sum_{i=0}^{m} (y_i - f(\vec{x_i}))^2 = argmin_{\vec{\beta}} \sum_{i=0}^{m} \left(y_i - \beta^T \vec{x_i}\right)^2$$

### Generalizace lineární regrese

Výše popsanou metodu lineární regrese můžeme zobecnit a hledat parametry $\beta$ pro transformovaná data předem známými funkcemi $\varphi_i(\vec{x})$. Nově tak hledáme funkci $f(\vec{x})$, která je lineární kombinací $n$ funkci $\varphi_i(\vec{x})$.

$$f(\vec{x}) = \beta_0\varphi_0(\vec{x}) + \beta_1\varphi_1(\vec{x}) + ... + \beta_n\varphi_n(\vec{x})$$

Definice lineární regrese jsem čerpal ze zdrojů [33, 34].

## 4.3 Shrnutí kapitoly

V této kapitole jsem v první části diskutoval metody interpolace dat v prostoru. Jako vhodnou metodu interpolace pro data kvality ovzduší jsem zvolil metodu inverzní vzdálenosti. V druhé části kapitoly jsem se věnoval regresním metodám pro extrapolaci dat a jako vhodného kandidáta jsem zvolil metodu obecné lineární regrese na transformovaných datech.

# Část II

# Praktická část

# Kapitola 5
# Návrh aplikace

V této kapitole se zabývám návrhem aplikace od výčtu požadavků až po výběr technologií, pomocí kterých hodlám dosáhnout cíle. V neposlední řadě zde diskutuji, jak by měla vypadat výsledná architektura aplikace. Hodlám se také věnovat možnostem budoucího rozšíření této aplikace.

## 5.1 Výčet funkčních požadavků na aplikaci

- Dlouhodobý sběr dat týkajících se kvality ovzduší.
- Jednotlivé záznamy dat budou pocházet z různých senzorů od různých poskytovatelů.
- Možná budoucí rozšiřitelnost aplikace o další zdroje.
- Možnost nahrání dat z vlastního archivu.
- Poskytování API pro možnost implementace uživatelského rozhraní.
- Z naměřených dat vytvářet model pro krátkodobou předpověď kvality ovzduší pro vybranou lokalitu.
- Jednoduché grafické webové rozhraní se statistickými údaji o datech.

## 5.2 Výběr technologií

V této sekci představuji technologie, které jsem vybral s ohledem na implementaci nástroje, jehož funkční požadavky jsem představil v předchozí sekci.

### 5.2.1 Java EE

Java Enterprise Edition[1] je sada specifikací definující framework založený na platformě Java SE. Jedná se o jednu z nejrozšířenějších platforem pro vývoj podnikových a webových aplikací na světě. Součástí této specifikace jsou

[1] `https://www.oracle.com/technetwork/java/javaee/overview/index.html`

další knihovny, například Java Servlet API, Java Security API, Enterprise JavaBeans, Contexts and Dependency Injection, Java Messaging Services, Java Server Faces, atd.

Zkráceně se dá najít pod označením Java EE, ale taky JEE. Dříve se Java EE označovala jako Java 2 Platform, Enterprise Edition (zkráceně J2EE). Současná verze je Java EE 8, nic méně následující vývoj má zajišťovat Eclipse Foundation pod označením Jakarta EE[2].

Mezi nejznámější implementace specifikace Java EE patří aplikační servery GlassFish, Payara, WildFly nebo WebShpere. Částečnou implementaci také nabízí Apache Tomcat.

### 5.2.2 Payara Server

Payara Server[3] je open source aplikační server, který je založený na serveru GlassFish. Poskytuje plnou implementaci specifikace Java EE. Kromě toho je ke stažení i verze Payara Micro, která implementuje tzv. Micro Profile, což je minimální podčást specifikace Java EE, obsahující ty nejdůležitější věci pro spuštění Java EE aplikace.

### 5.2.3 MongoDB

MongoDB[4] je NoSQL databáze JSON dokumentů a je zdarma k použití. Kromě databází známých vlastností jako je dotazování, indexace nebo agregace nabízí také řadu dalších výhod jako je například podpora GeoJSON nebo horizontálního škálování. Především podpora GeoJSON je výhoda této databáze, jelikož data naměřená ze senzorů budou vázána na lokaci.

Pro Javu nabízí ovladač[5], který zprostředkovává nejenom komunikaci s databází, ale do Javy také přidává podporu pro BSON.

### 5.2.4 GraphQL

GraphQL[6] je dotazovací jazyk pro API. Na rozdíl od známějšího RESTového rozhraní nabízí GraphQL mnohem komplexnější možnosti dotazování. Neposkytuje totiž pro každý dotaz separátní zdroj, ale naopak poskytuje jedno schéma, které plně popisuje data a možnosti dotazování. Klient si tak v jednom dotazu může vyžádat data z různých zdrojů a upravit si formát dat dle potřeby. Klient tak dostane přesně a jenom to, co skutečně potřebuje. Generování a úpravu dat přenechá serveru.

Pro snadnou implementaci GraphQL v Javě vznikl projekt GraphQL Java Kickstart[7]. Ten nabízí knihovnu GraphQL Java Tools[8], která výrazně zjed-

[2] https://jakarta.ee/
[3] https://www.payara.fish/
[4] https://www.mongodb.com/
[5] https://mongodb.github.io/mongo-java-driver/
[6] https://graphql.org/
[7] https://www.graphql-java-kickstart.com/
[8] https://www.graphql-java-kickstart.com/tools/

nodušuje implementaci rozhraní. Pro Javu EE také poskytuje knihovnu GraphQL Java Servlet[9].

### 5.2.5 Apache Maven

Apache Maven[10] je nástroj pro management projektu. Zajišťuje nastavení kompilace, buildu, balíčkování a to včetně automatického stažení závislostí z repozitáře.

## 5.3 Architektura aplikace

Program na správu dat jsem se rozhodl udělat jako serverovou vícevrstvou aplikaci, která bude zajišťovat aktualizace dat z různých datových zdrojů, výpočty nad daty a v poslední řadě také poskytování dat pro uživatelské rozhraní. V jednotlivých podsekcích se zaměřím na popis jednotlivých vrstev/částí. Výsledná architektura je zobrazena na obrázku 5.1.

Maven dovoluje snadno rozdělit aplikaci do modulů pro lepší oddělení jednotlivých vrstev/částí. Rozhodl jsem se, že aplikaci rozdělím na 3 moduly.

První EJB modul bude obsahovat perzistentní, datovou a servisní vrstvu. Druhý WEB modul bude obsahovat části, které budou zprostředkovávat komunikaci s vnějším světem, tedy API vrstvu s GraphQL a REST rozhraním a v neposlední řadě taky jednoduché webové rozhraní, které bude obsahovat statistické informace. To celé obalí EAR modul, který v sobě ponese jak EJB, tak WEB modul.

### 5.3.1 Datové zdroje

V současné implementaci jsou podporovány 2 datové zdroje a to data ze senzorů CHMI a data z vlastních senzorů. Do budoucna je samozřejmě možné rozšířit datové zdroje například o satelitní data.

### 5.3.2 Perzistentní vrstva

Ukládání dat zajišťuje MongoDB. Pro jednodušší práci s daty jsou jednotlivé JSON dokumenty mapovány do POJO objektů.

### 5.3.3 Datová vrstva

Tato vrstva zajišťuje základní práci s daty. Budou se zde nacházet DAO třídy pro jednotlivé databázové kolekce a DTO objekty pro jednotlivé entity.

### 5.3.4 Servisní vrstva

Servisní vrstva bude obsahovat třídy, které budou pracovat s daty, které agregují data i z více zdrojů.

[9] `https://www.graphql-java-kickstart.com/servlet/`
[10] `https://maven.apache.org/`

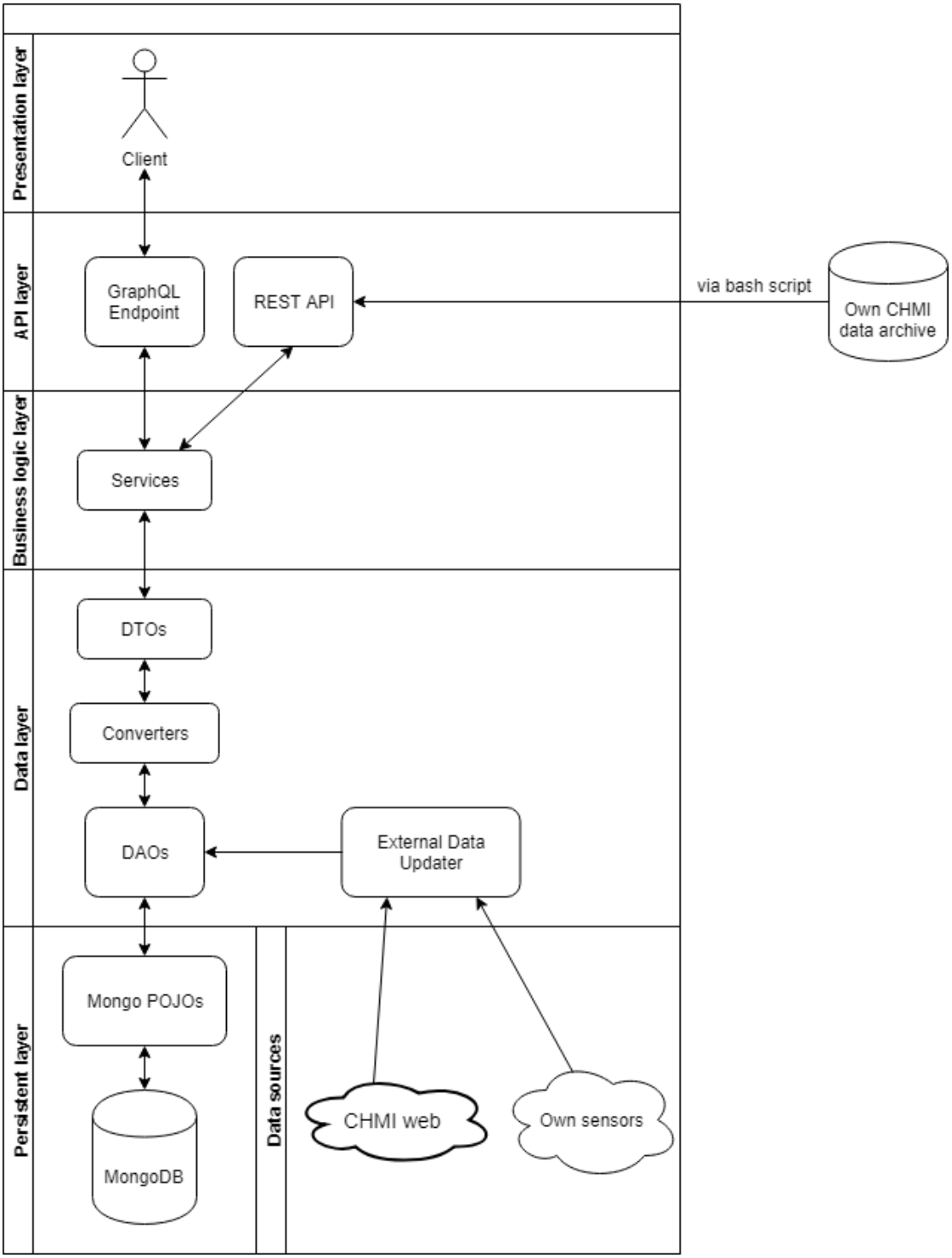
### 5.3.5 API vrstva

Pro komunikaci s vnějším světem bude aplikace poskytovat dvě různá API. Jednak GraphQL pro připojení webového grafického rozhraní a zároveň REST. Ten bude primárně sloužit pro manuální aktualizaci dat z vlastního archivu. Sekundární účel tohoto rozhraní bude export dat do CSV pro tvorbu predikčního modelu v jazyku Python.

### 5.3.6 Prezenční vrstva

V této vrstvě implementuji jednoduché webové rozhraní, které bude obsahovat statistické informace o systému.

### 5.3.7 Top level model diagram




**Obrázek 5.1:** Top level architektura aplikace, zdroj: autor

## 5.4 Shrnutí kapitoly

V této kapitole jsem definoval funkční požadavky na aplikaci. Dále jsem vybral technologie k naplnění těchto cílů a navrhl výslednou architekturu serverové, vícevrstvé aplikace. Jednotlivé vrstvy aplikace jsem krátce popsal v jednotlivých podsekcích.

# Kapitola 6
# Implementace aplikace

V této kapitole se věnuji některým zajímavým implementačním řešením aplikace s ohledem na to, aby bylo jasné, jak je možné aplikaci do budoucna rozšířit.

## 6.1 Spuštění a inicializace

Spuštění aplikace je realizováno "@Startup" "@Singleton" třídou *Starter*, takže "@PostConstruct" metoda této třídy se spustí hned po spuštění aplikace. V této metodě se pak spouští metody třídy *Initializer*. V nich se do DB nahrají inicializační data, pokud už v databázi nejsou. Samozřejmě je možné zde přidat další inicializační data a další metody.

## 6.2 Aktualizace z externích zdrojů

Aktualizaci z externích zdrojů jsem implementoval tak, aby bylo přidání dalšího zdroje co nejsnadnější. Celá aktualizace je realizována třídou *MainUpdater*. V této třídě jsou pomocí CDI injektovány všechny instance tříd typu *AbstractUpdater*. Na každé této instanci je zavolána metoda *update()*.

Přidání dalšího zdroje je tedy velice snadné. Stačí pouze naimplementovat třídu, která bude potomkem třídy *AbstractUpdater*. Veškerá logika aktualizace musí být implementována v metodě *update()*. Je silně doporučeno během aktualizace provést taky aktualizaci statistiky o průběhu aktualizace zavoláním metody *updateStatistics(boolean)*. Ta slouží k informaci, jestli aktualizace probíhají v pořádku, aby systém nepřicházel o data. Do budoucna lze posílat například emailové notifikace pro administrátory aplikace.

Aktualizace může být náročná procedura a zároveň není potřeba, aby byla prováděna příliš často. Například nová data z CHMI jsou dostupná každou hodinu. Nedává tedy smysl, aby aktualizace probíhala víckrát než jednou za hodinu. Proto každá třída typu *AbstractUpdater* vrací hodnotu *UpdateFrequencyEnum* enumu, která reprezentuje, jak často je potřeba zdroj aktualizovat. Třída pak na základě informace poslední aktualizace a hodnoty enumu zkontroluje, zda je vůbec potřeba aktualizace spustit.

## 6.3 Implementace GraphQL

Rozhraní GraphQL jsem implementoval pomocí knihovny . Veškerou komunikaci přes toto rozhraní zprostředkovává servlet *GraphqlEndpointServlet*. Zde je taky implementováno zabezpečení rozhraní. To je realizováno pomocí tokenu, který musí externí aplikace poslat přes HTTP Authorization header.

Je pravděpodobné, že GraphQL rozhraní bude využívat jen grafické webové rozhraní a to bude pravděpodobně jen jedno. Lze tedy token pro tuto aplikaci vytvořit již při inicializaci aplikace ve třídě *Initializer*. Další externí aplikace se dají přidat jako záznam do DB.

### 6.3.1 Načtení a úprava schématu

Načítání a nastavení schématu pro běh knihovny opět implementuje třída *GraphqlEndpointServlet*, konkrétně v metodě *createSchema()*. Za pozornost stojí určitě přidání tříd typu *GraphQLResolver* do schématu. Každý "type" musí mít vlastní implementaci resolveru.

Úprava schématu tedy znamená přidání metody do již stávajícího resolveru daného typu. Pro přidání dalšího typu do schématu stačí implementovat třídu typu *GraphQLResolver* pro daný typ a jeho instanci nastavit v metodě *createSchema()*.

### 6.3.2 Návrh schématu

```
scalar OffsetDateTime

type Query {
    providers: [Provider]
    provider(id: String): Provider
    providersCount: Int
    sensors: [Sensor]
    sensor(id: String): Sensor
    sensorByCode(code: String): Sensor
    sensorsCount: Int
    sensorData(sensorId: String, from: OffsetDateTime, to: OffsetDateTime): [Sen
    interpolateData(latitude: Float, longitude: Float, time: OffsetDateTime): [S
    commonAirQualityIndex(sensorId: String): Int
    interpolatedCommonAirQualityIndex(latitude: Float, longitude: Float): Int
}

type Provider {
    id: String!
    name: String
    abbr: String
    web: String
    sensors: [Sensor]
```

```
}

type Sensor {
    id: String!
    code: String
    latitude: Float
    longitude: Float
    altitude: Float
    web: String
    provider: Provider
    sensorData(from: OffsetDateTime, to: OffsetDateTime): [SensorData]
}

type SensorData {
    id: String
    from: OffsetDateTime
    to: OffsetDateTime
    pollutant: String
    value: Float
    hourAvg: Int
    sensorIds: [String]
}
```

## 6.4 Implementace REST

Vedle rozhraní GraphQL jsem také implementoval RESTové rozhraní pomocí knihovny JAX-RS. Rozhraní je tedy velice snadno rozšiřitelné. V první verzi se využívá k nahrání dat z archivu a k exportu dat obsažených v databázi v CSV formátu. Do budoucna by RESTové rozhraní mělo sloužit spíše jako vstupní brána do aplikace a pro exporty dat pro další aplikace.

Ačkoliv GraphQL kromě Query dotazů zvládá i tzv. Mutatation, které slouží k odesílání dat od klienta k serveru, vyžaduje GraphQL přesnou strukturu requestu. Narozdíl od RESTového rozhraní, kde se celý obsah dá poslat v těle requestu. Jeho použití je tedy mnohem snazší při sekvenčním zpracování pomocí scriptů v bashi, apod.

### 6.4.1 Zabezpečení

Zabezpečení RESTového rozhraní jsem si implementoval sám pomocí návrhového vzoru interceptor. Její implementace je realizována v třídě *RestSecurityInterceptor*. Pro spojení třídy RESTového rozhraní s tímto interceptorem stačí dané třídě přidat anotaci *@SecuredRestEndpoint*. Pro autorizaci dotazu je potřeba přidat hlavičku Authorization s příslušným tokenem.

Aby bylo možné rozlišit různé úrovně zabezpečení metod rozhraní, implementoval jsem ještě anotaci *@PermissionNeeded*, která jako parametr přijímá pole potřebných oprávnění. *RestSecurityInterceptor* tedy kromě tokenu také

zkontroluje, zda klient má dostatečná oprávnění k zavolání metody. Pokud zjistí, že na volání tohoto endpointu nemá klient práva, rovnou vrátí HTTP response 404 Unauthorized a k volání metody vůbec nedojde. V opačném případě standardně spustí volanou metodu.

## 6.5 Testování

V rámci implementace aplikace jsem také implementoval celou řadu unit testů. K tomu jsem využíval knihovnu TestNG. Většina testů je založena na metodě testování mezních hodnot formou parametrizovaných testů.

Zároveň jsem na projektu využíval verzovací systém GIT a fakultní systém Gitlab. V něm jsem nastavil Gitlab CI tak, aby se testy spustily při každém nahrání nové verze do repozitáře. V budoucnu bude možné nastavení Gitlab CI rozšířit i pro automatizované nasazování aplikace na aplikační server.

## 6.6 Shrnutí kapitoly

V této kapitole jsem představil nejdůležitější části aplikace z hlediska jejich implementací. Hlavně jsem se zaměřoval na oblasti, kde a jak se dá aplikace do budoucna rozšířit.

# Kapitola 7

# Model pro předpověď

Tuto kapitolu věnuji návrhu a implementaci modelu pomocí lineární regrese pro předpověď koncentrace $PM_{10}$. Dále je v této kapitole k nalezení definice vlastní funkce pro vyhodnocení chyby modelu a následné experimenty s navrženými modely.

## 7.1 Vstupní data

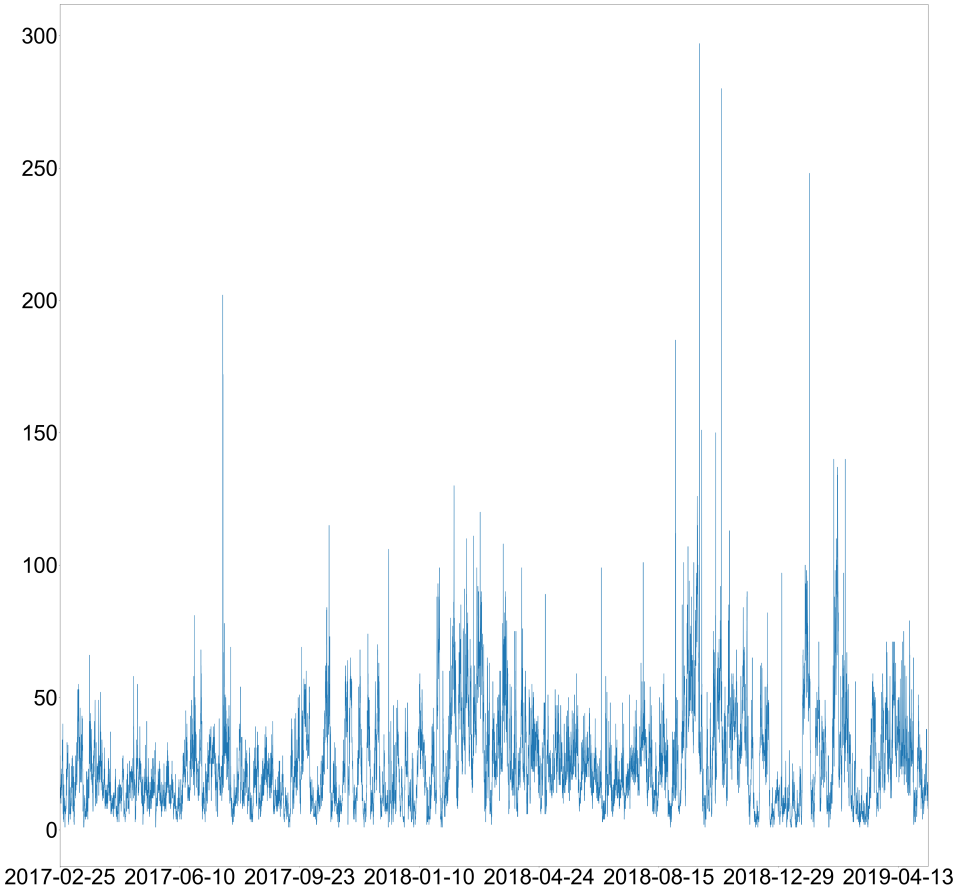
V této sekci se věnuji blíže datům, které hodlám použít pro učení i testování regrese. Popisuji zde přípravu datasetů a jednoduchou ruční analýzu, kterou jsem provedl před implementací lineární regrese.

### 7.1.1 Příprava

Pro naučení i testování jsem zvolil datasety s koncentrací $PM_{10}$ a to jak s hodinovými, tak 24-hodinovými průměry. Rozhodl jsem se použít pro experimenty v rozsahu této práce pouze koncentrace $PM_{10}$, jelikož tyto data poskytuje naprostá většina CHMI senzorů. Navíc v budoucnu budou přenositelné senzory (zmiňované v datových zdrojích 3.1.2) také měřit primárně $PM_{10}$. Nejvíce dat bude tedy pro polétavý prach.

Tyto data jsem si nechal vygenerovat jako CSV datasety pomocí RESTového rozhraní hlavní aplikace. Rozhodl jsem se použít všechna dostupná data z 5 náhodně vybraných senzorů. Konkrétně šlo o senzory od Českého hydrometeorologického ústavu s kódovými označeními AKALA, ALEGA, AREPA, ARIEA a ASMIA. V době přípravy dat to bylo rozmezí od 25.2.2017 do 9.5.2019. Jednotlivé datasety jsou dostupné jako přílohy.

Dataset vždy obsahuje unikátní id, od kdy a do kdy byla data měřena, znečišťující látka (v případě experimentu této práce pouze $PM_{10}$), za jak dlouhý hodinový interval jsou data průměrovány a samozřejmě samotná hodnota koncentrace. Data za celé zmiňované období pro senzor ALEGA jsou k vidění na grafu 7.1.



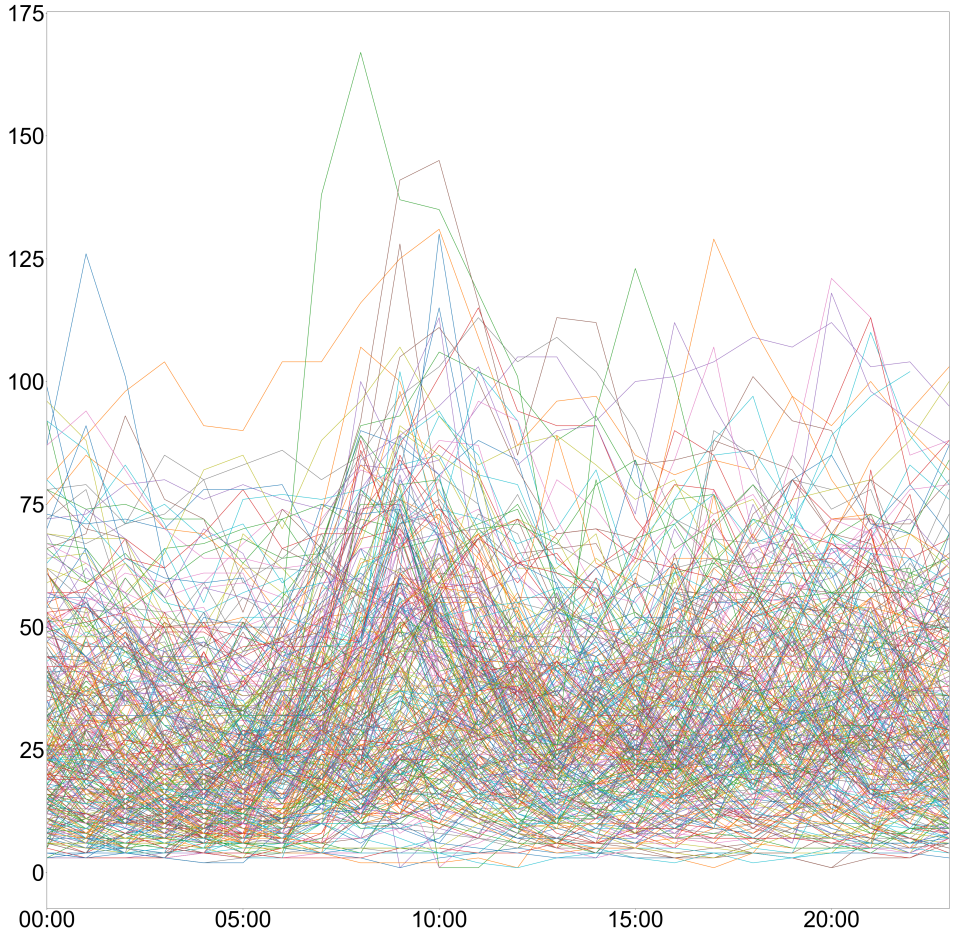

**Obrázek 7.1:** Graf koncentrace $PM_{10}$ (1h průměr) ze senzoru CHMI s kódem ALEGA za časové období 25.2.2017 - 9.5.2019, zdroj: autor

### Příprava dat počasí

Data o počasí ze zdroje, který jsem popsal v sekci 3.2.3, jsou ve formě HTML tabulky. Je třeba jejich scrapování. K tomu jsem využil knihovnu `BeautifulSoup4` pro Python. Data jsem přetransformoval do listu vektorů, kde jeden vektor na pozici $i$ reprezentuje záznam o počasí v hodině $i$.

## 7.1.2 Ruční analýza

Než jsem se pustil do implementace lineární regrese, zkusil jsem data různě vykreslit jako graf. Jeden z takových grafů je možno vidět na obrázku 7.2. Konkrétně se jedná o graf průměrných koncentrací za jednu hodinu pro $PM_{10}$ pro každý den roku 2018 zvlášť od půlnoci do půlnoci. Tyto dny jsou pak vykresleny jako jednotlivé křivky přes sebe. Data pochází ze senzoru od Českého hydrometeorologického ústavu s kódovým označením AKALA.



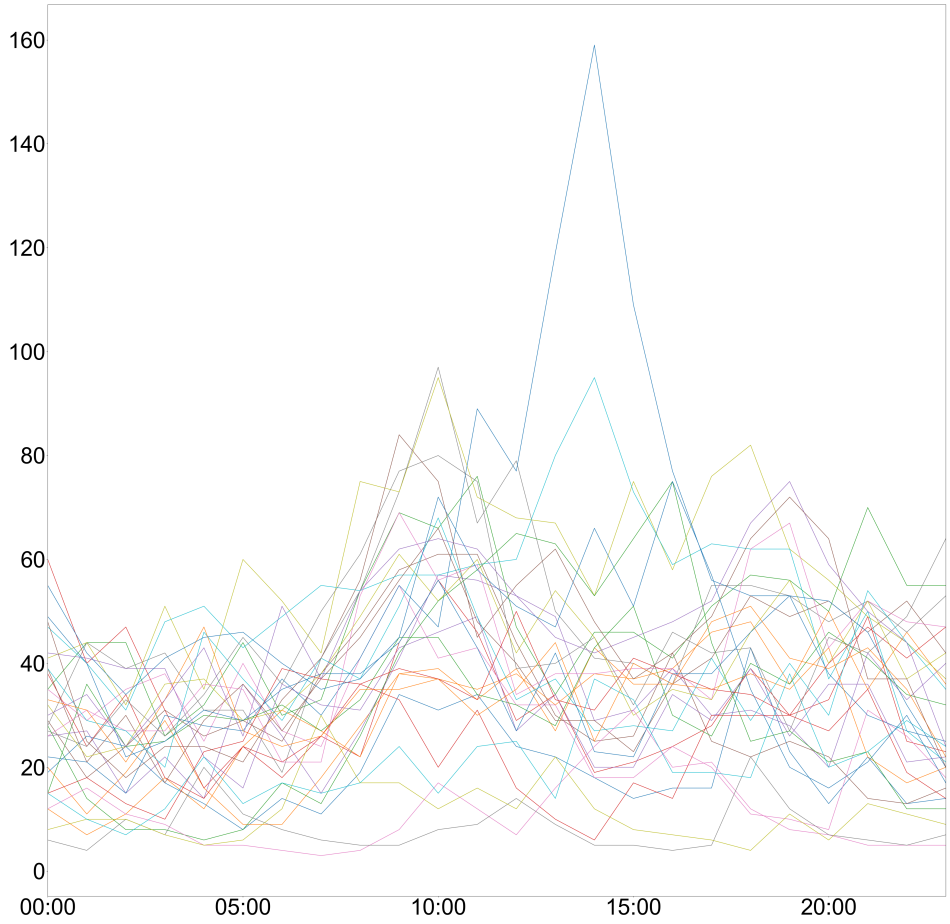

**Obrázek 7.2:** Graf koncentrace $PM_{10}$ (1h průměr) všech dní v roce 2018 pro jeden CHMI senzor (kód AKALA), vykreslené přes sebe, zdroj: autor

Na tomto grafu je patrné, že se například pravidelně opakuje zvýšení koncentrace každý den dopoledne / okolo poledne. Moje hypotéza tedy byla, že při učení modelu může hrát zásadní roli i koncentrace naměřená ve stejnou denní dobu předchozí den.

Další zajímavé pozorování z grafu 7.2 také je, že ani v noci není koncentrace nijak malá. Přitom v noci v Praze nejezdí téměř žádná doprava. Vytápění pak připadá v úvahu pouze v zimních měsících a čekal bych tedy, že budou v noci 2 hlavní linie koncentrací. To však zde není pozorováno.

Pro bližší zaměření zde přikládám i graf 7.3 za měsíc březen. Opět se jedná o hodinové průměry koncentrací $PM_{10}$ ze senzoru AKALA. Na tomto grafu je blíže vidět zvýšená koncentrace během poledne, kterou jsem zmiňoval výše.

Pro generování grafů jsem použil knihovnu `matplotlib` pro jazyk Python.




**Obrázek 7.3:** Graf koncentrace $PM_{10}$ (1h průměr) všech dní v březnu roku 2018 pro jeden CHMI senzor (kód AKALA), vykreslené přes sebe, zdroj: autor

## 7.2 Návrh modelu

V této sekci popisuji návrh modelu. V první části popisuji knihovny, pomocí kterých jsem model implementoval. V části druhé pak definuji metodu vyhodnocování kvality vytvořeného modelu.

### 7.2.1 Jazyk a knihovny

Implementaci lineární regrese jsem se rozhodl využít z knihovny `scikit-learn` pro jazyk Python. Ta poskytuje nejenom implementaci lineární regrese, ale i dalších metod jako například SVM, EM nebo rozhodovacích stromů. Jazyk Python je pro datovou analýzu vhodný, jelikož pro něj existuje velké množství knihoven implementujících metody strojového učení, data miningu, datové analýzy i vykreslování grafů.

Dále jako pomocné knihovny jsem se rozhodl použít `pandas` pro práci s datasety v CSV a `numpy` pro vektorové výpočty.

### 7.2.2 Vyhodnocení kvality modelu

Pro vyhodnocování kvality naučeného modelu se obvykle používají metody Mean Squared Error (MSE) nebo Mean Absolute Error (MAE). Nic méně pro kvalitu ovzduší není ani tak klíčové, jak vzdálená bude reálná hodnota od té predikované, ale informace jaký bude výsledný index kvality ovzduší. Tedy jaké budou mít zdravotní dopady pro koncového uživatele. Rozhodl jsem se tedy vytvořit vlastní chybovou funkci, která bude přísněji zohledňovat fakt, jak moc se liší predikovaný index kvality od toho skutečného.

Mnou navržená chybová funkce Mean Air Quality Index Error (MAQIE), kde $y$ je skutečná koncentrace sledované látky spolu s jejím indexem kvality $i$. Poté $y'$ je predikovaná koncentrace a $i'$ zní vycházející index kvality. Hodnoty pro výpočet indexu kvality vychází z CAQI popisovaném v 2.2.2.

$$MAIQE = \frac{1}{n}\sum_{k=0}^{n}\left((y_k - y'_k)^2 + 100^{|i_k - i'_k|}\right)$$

Kromě této mnou navržené chybové funkce budu pro porovnání uvádět i hodnoty MSE a MAE.

## 7.3 Experimenty

V podsekci experimenty se věnuji přípravě funkcí $\varphi_i(x)$ pro naučení generalizované lineární regrese popsané v 4.2.1. Poté nechám na těchto datech naučit model a vypočítat hodnoty MSE, MAE a MAQIE výsledného modelu.

Vstupní dataset rozdělím na trénovací a testovací část. Pro testovací část zvolím 4500 hodnot, což je zhruba půl roku měření.

Výsledky pro jednotlivé senzory a diskuzi pro větší přehlednost uvádím v samostatné kapitole 8.

Veškeré Python skripty implementované pro realizaci experimentů jsou v samostatné příloze.

### 7.3.1 Transformace dat

V této podčásti definuji jednotlivé transformace vstupních dat a definuji tak možné experimenty.

Mějme dataset koncentrací polétav0ho prachu $PM_{10}$ $X$, kde $X_i$ je hodnota $PM_{10}$ v $i$-té hodině. Vstupní vektor pro regresi v následující hodině označuji jako $\vec{T_i}$.

#### Nekompletní data

Data ze CHMI senzorů občas nejsou kompletní. Někdy chybí jen jedna hodina, jindy třeba celý den. Všechna tato data přeskakuji. Kolik záznamů je třeba přeskočit záleží však na povaze experimentu, které definuji níže.

#### Expetiment 1

Pro první experiment definuju vstupní vektory jako známé hodnoty koncentrace před hodinou a před 2 hodinami.

$$\vec{T_i} = (X_{i-1}, X_{i-2})$$

#### Experiment 2

Protože se domnívám, že důležitější je hodnota koncentrace před hodinou, zkusím přidat kvadratickou závislost této hodnoty.

$$\vec{T_i} = (X_{i-1}^2, X_{i-1}, X_{i-2})$$

#### Experiment 3

Při ruční analýze jsem položil hypotézu, že důležitá mohou být data ve stejnou dobu předchozí den, zkusím tedy přidat data ze stejné hodiny dne předem a dne předem a jedné hodiny.

$$\vec{T_i} = (X_{i-1}^2, X_{i-1}, X_{i-2}, X_{i-24}, X_{i-25})$$

#### Experiment 4

Pro tento experiment využiji stejná transformovaná data jako v předchozím experimentu jenom s tím rozdílem, že budu model učit online. Tzv. po každé predikované hodině naučím zcela nový model s nově příchozí hodnotou pro aktuální hodinu, resp. v tu dobu již pro hodinu předchozí.

### 7.3.2 Přidání dat o počasí

K datům pro učení jsem se také pokusil přidat data o počasí v předchozí hodině. Konkrétně jsem zkoušel teplotu ($t$), vlhkost ($h$) a tlak ($p$). Aktuální hodinu označuji indexem $i$. Aktuální hodnota teploty, tedy v čase $i$ bude $t_i$.

#### Experiment 5

K modelu z experimentu 4 zkusím přidat i data o počasí hodinu před zkoumanou hodinou.

$$\vec{T_i} = (X_{i-1}^2, X_{i-1}, X_{i-2}, X_{i-24}, X_{i-25}, t_{i-1}, h_{i-1}, p_{i-1})$$

## 7.4 Shrnutí

V kapitole jsem shrnul v první části shrnul informace o datech a nastínil jejich charakter za pomocí grafů. Druhé části jsem definoval 4 různé experimenty. Jejich výsledky a následnou diskuzi uvádím v samostatné kapitole 8.

# Kapitola 8

# Výsledky a diskuze

V této kapitole uvádím výsledky experimentů definovaných v předchozí kapitole v sekci 7.3.

## 8.1 Výsledky pro datasety o průměru za 1h

V této části prezentuji výsledky po jednotlivých experimentech pro datasety 5 senzorů. Data obsahují koncentrace $PM_{10}$ průměrované za 1h.

### 8.1.1 Sensor AKALA

#### Výsledky experimentu 1

MSE: 75.19
MAE: 6.58
MAQIE: 324.15

#### Výsledky experimentu 2

MSE: 69.69
MAE: 6.14
MAQIE: 318.26

#### Výsledky experimentu 3

MSE: 67.00
MAE: 6.00
MAQIE: 315.41

#### Výsledky experimentu 4

MSE: 65.96
MAE: 5.91
MAQIE: 252.08

#### Výsledky experimentu 5

MSE: 66.03
MAE: 5.94
MAQIE: 252.24

### 8.1.2 Sensor ALEGA

#### Výsledky experimentu 1

MSE: 70.10
MAE: 3.66
MAQIE: 325.24

#### Výsledky experimentu 2

MSE: 58.06
MAE: 3.43
MAQIE: 86.51

#### Výsledky experimentu 3

MSE: 56.36
MAE: 3.39
MAQIE: 80.22

#### Výsledky experimentu 4

MSE: 56.61
MAE: 3.44
MAQIE: 27.63

#### Výsledky experimentu 5

MSE: 56.29
MAE: 3.44
MAQIE: 27.82

### 8.1.3 Sensor AREPA

#### Výsledky experimentu 1

MSE: 60.90
MAE: 5.52
MAQIE: 87.09

#### Výsledky experimentu 2

MSE: 60.88
MAE: 5.50
MAQIE: 87.03

#### Výsledky experimentu 3

MSE: 64.51
MAE: 5.71
MAQIE: 91.22

#### Výsledky experimentu 4

MSE: 64.43
MAE: 5.60
MAQIE: 31.76

#### Výsledky experimentu 5

MSE: 64.65
MAE: 5.64
MAQIE: 31.80

### 8.1.4 Sensor ARIEA

#### Výsledky experimentu 1

MSE: 41.90
MAE: 4.83
MAQIE: 61.93

#### Výsledky experimentu 2

MSE: 41.83
MAE: 4.81
MAQIE: 61.73

#### Výsledky experimentu 3

MSE: 41.15
MAE: 4.78
MAQIE: 60.87

#### Výsledky experimentu 4

MSE: 39.70
MAE: 4.66
MAQIE: 24.16

#### Výsledky experimentu 5

MSE: 39.44
MAE: 4.68
MAQIE: 23.91

### 8.1.5 Sensor ASMIA

#### Výsledky experimentu 1

MSE: 117.00
MAE: 7.53
MAQIE: 169.37

#### Výsledky experimentu 2

MSE: 118.21
MAE: 7.62
MAQIE: 168.31

#### Výsledky experimentu 3

MSE: 111.28
MAE: 7.37
MAQIE: 162.88

#### Výsledky experimentu 4

MSE: 109.92
MAE: 7.24
MAQIE: 56.66

#### Výsledky experimentu 5

MSE: 109.25
MAE: 7.25
MAQIE: 56.60

## 8.2 Výsledky pro datasety o průměru za 24h

V této části prezentuji výsledky po jednotlivých experimentech pro datasety 5 senzorů. Data obsahují koncentrace $PM_{10}$ průměrované za 24h.

### 8.2.1 Sensor AKALA

#### Výsledky experimentu 1

MSE: 0.23
MAE: 0.35
MAQIE: 2.04

#### Výsledky experimentu 2

MSE: 0.24
MAE: 0.35
MAQIE: 2.09

#### Výsledky experimentu 3

MSE: 0.20
MAE: 0.33
MAQIE: 2.10

#### Výsledky experimentu 4

MSE: 0.20
MAE: 0.33
MAQIE: 2.16

#### Výsledky experimentu 5

MSE: 0.20
MAE: 0.33
MAQIE: 2.16

### 8.2.2 Sensor ALEGA

#### Výsledky experimentu 1

MSE: 0.19
MAE: 0.22
MAQIE: 1.87

#### Výsledky experimentu 2

MSE: 0.19
MAE: 0.22
MAQIE: 1.87

#### Výsledky experimentu 3

MSE: 0.17
MAE: 0.22
MAQIE: 1.77

#### Výsledky experimentu 4

MSE: 0.17
MAE: 0.22
MAQIE: 1.84

#### Výsledky experimentu 5

MSE: 0.17
MAE: 0.22
MAQIE: 1.82

### 8.2.3 Sensor AREPA

#### Výsledky experimentu 1

MSE: 0.21
MAE: 0.32
MAQIE: 1.96

#### Výsledky experimentu 2

MSE: 0.21
MAE: 0.32
MAQIE: 1.96

#### Výsledky experimentu 3

MSE: 0.21
MAE: 0.33
MAQIE: 2.09

#### Výsledky experimentu 4

MSE: 0.21
MAE: 0.32
MAQIE: 2.24

#### Výsledky experimentu 5

MSE: 0.21
MAE: 0.32
MAQIE: 2.18

### 8.2.4 Sensor ARIEA

#### Výsledky experimentu 1

MSE: 0.13
MAE: 0.27
MAQIE: 2.12

#### Výsledky experimentu 2

MSE: 0.13
MAE: 0.27
MAQIE: 2.12

#### Výsledky experimentu 3

MSE: 0.13
MAE: 0.26
MAQIE: 2.05

#### Výsledky experimentu 4

MSE: 0.12
MAE: 0.26
MAQIE: 2.14

#### Výsledky experimentu 5

MSE: 0.12
MAE: 0.26
MAQIE: 2.12

### 8.2.5 Sensor ASMIA

#### Výsledky experimentu 1

MSE: 0.34
MAE: 0.40
MAQIE: 2.70

#### Výsledky experimentu 2

MSE: 0.34
MAE: 0.40
MAQIE: 2.70

#### Výsledky experimentu 3

MSE: 0.32
MAE: 0.39
MAQIE: 2.59

#### Výsledky experimentu 4

MSE: 0.31
MAE: 0.38
MAQIE: 2.63

#### Výsledky experimentu 5

MSE: 0.31
MAE: 0.38
MAQIE: 2.63

## 8.3 Diskuze výsledků

U první části experimentu, kdy jsem model učil na datech průměrovaných za 1 hodinu, je vidět zlepšení chyby každým dalším experimentem. K výraznému zlepšení chyby přispělo přidáním kvadrátu hodnoty koncentrace hodinu zpět do vstupních dat. K dalšímu výraznému zlepšení došlo, když se model učil online. Tedy vždy se vytvořil nový model, když mu přišla nová data. Naopak velkým překvapením pro mě bylo, že data o počasí model nijak zvlášť nezlepšila. Vítězný návrh modelu je jednoznačně online učící se model z experimentu 4.

Výsledky druhé části experimentu s daty průměrovanými za 24 hodin jsou neméně překvapivé. Pro naprostou většinu senzorů stačí výrazně jednodušší transformace. Výsledky se téměř neměnily napříč všemi experimenty. Pro tato data tedy postačí i ten nejjednodušší model z experimentu 1.

Výsledky druhé části experimentu budou daná pravděpodobně tím, že data průměrovaná za 24 hodin nemají tolik extrémů a tedy i těžko předvídatelných výchylek. Z dat je tak vidět trend vývoje koncetrací.

# Kapitola 9

# Závěr

V rámci této práce vznikla serverová aplikace na shromažďování dat o kvalitě ovzduší. Zároveň data poskytuje skrze externí API buď přes GraphQL pro možnou implementaci klientské aplikace pro koncového uživatele nebo přes RESTové API formou CSV exportů. Ty jsem využíval pro experimenty s predikčním modelem.

Rád bych také zmínil, že na základě této diplomové práci vznikla již bakalářská práce. Ta měla za cíl vytvořit moderní webovou stránku, která uživateli přehledně zobrazí informace o aktuální kvalitě ovzduší v Praze. Data získává právě z mnou implementovaného systému skrze GraphQL API. Do budoucna by se tato webová prezentace měla rozšířit i o předpověď kvality ovzduší.

Nedílnou součástí práce je také rešerše znečišťujících látek a jejich vlivu na životní prostředí, především na kvalitu ovzduší. Dále pak rešerše možných zdrojů těchto dat. V samostatné kapitole jsem shrnul možnosti využití dat ze soukromého archivu z profesionálních stacionárních senzorů od Českého hydrometeorologického ústavu, dále pak možnost malých přenositelných senzorů a v neposlední řadě taky možnost využití dat ze satelitů NASA i ESA.

V druhé části práce jsem se zabýval možnostmi predikce kvality ovzduší z hlediska znečištění polétavým prachem $PM_{10}$. V několika experimentech jsem ověřil různé možnosti implementace modelu lineární regrese. Z výsledků těchto experimentů jsem určil vhodné kandidáty pro predikční model jak pro data průměrovaná za 1 hodinu, tak pro data průměrovaná za 24 hodin.

Na úplný závěr bych ještě rád podotkl, že v obou částech práce je otevřená možnost budoucího rozvoje a zdokonalení. Například vylepšení automatického sběru dat, sběr dat z přenosných senzorů a ze satelitů a v neposlední řadě také vylepšování předpovědního modelu.

Tuto práci jsem dělal jako součást většího projektu, který bude pokračovat i za rámec této práce. Budeme se tedy i na dále snažit dosáhnout cíle zpřesnění dat kvality ovzduší, jejich předpovědi a následné prezentaci obyvatelům Prahy.

Celkově tak tato práce položila základní kámen pro vybudování komplexního systému pro sběr a zpracování dat o kvalitě ovzduší.

# Přílohy

# Příloha A

## Literatura a zdroje


1. *Polétavý prach - PM10* [online]. Arnika [cit. 21. 04. 2019]. Dostupné z: https://arnika.org/poletavy-prach-pm10.
2. *Co jsou to částice PM10 a proč se vůbec měří?* [online]. Město Nový Jičín [cit. 21. 04. 2019]. Dostupné z: https://www.novyjicin.cz/co-jsou-to-castice-pm10-a-proc-se-vubec-meri/.
3. *Particle Pollution (PM)* [online]. Air Now [cit. 21. 04. 2019]. Dostupné z: https://airnow.gov/index.cfm?action=aqibasics.particle.
4. *Podíl sektorů NFR na celkových emisích PM10 v roce 2016* [online]. Český Hydrometeorologický ústav, 2017 [cit. 21. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/png/oIV1-20.png.
5. *Podíl sektorů NFR na celkových emisích PM2,5 v roce 2016* [online]. Český Hydrometeorologický ústav, 2017 [cit. 21. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/png/oIV1-22.png.
6. GUARNIERI, Michael; BALMES, John R. Outdoor air pollution and asthma. *The Lancet* [online]. 2014, roč. 383, č. 9928, s. 1581–1592 [cit. 21. 04. 2019]. ISSN 01406736. Dostupné z DOI: 10.1016/S0140-6736(14)60617-6.
7. *Common pollutants from household heating, cooking and lighting* [online]. World Health Organization [cit. 21. 04. 2019]. Dostupné z: https://www.who.int/airpollution/household/pollutants/combustion/en/.
8. *Látky znečisťující ovzduší* [online]. Arnika [cit. 23. 04. 2019]. Dostupné z: https://arnika.org/latky-znecistujici-ovzdusi.
9. KOOLEN, Cedric D.; ROTHENBERG, Gadi. Air Pollution in Europe. *ChemSusChem* [online]. 2018-12-11, roč. 12, č. 1, s. 164–172 [cit. 23. 04. 2019]. ISSN 1864-5631. Dostupné z DOI: 10.1002/cssc.201802292.
10. *Podíl sektorů NFR na celkových emisích NOx v roce 2016* [online]. Český Hydrometeorologický ústav [cit. 22. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/png/oIV3-9.png.

11. *Nitrogen oxides (NOx) emissions* [online]. European Environment Agency [cit. 24. 04. 2019]. Dostupné z: https://www.eea.europa.eu/data-and-maps/indicators/eea-32-nitrogen-oxides-nox-emissions-1.
12. *Oxidy dusíku* [online]. Arnika [cit. 25. 04. 2019]. Dostupné z: https://arnika.org/oxidy-dusiku.
13. *Podíl sektorů NFR na celkových emisích SO2, 2016* [online]. Český Hydrometeorologický ústav, 2017 [cit. 21. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/png/oIV7-8.png.
14. *Oxidy síry* [online]. Arnika [cit. 24. 04. 2019]. Dostupné z: https://arnika.org/oxidy-siry.
15. *Oxid uhelnatý* [online]. Arnika [cit. 23. 04. 2019]. Dostupné z: https://arnika.org/oxid-uhelnaty.
16. *Podíl sektorů NFR na celkových emisích CO, 2016* [online]. Český Hydrometeorologický ústav, 2017 [cit. 21. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/png/oIV8-2.png.
17. *PŘÍZEMNÍ OZON: Vznik přízemního ozonu* [online]. Český Hydrometeorologický ústav [cit. 23. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/17groc/gr17cz/IV4_O3_CZ.html.
18. KANCHAN, Kanchan; GORAI, Amit Kumar; GOYAL, Pramila. A Review on Air Quality Indexing System. *Asian Journal of Atmospheric Environment* [online]. 2015-06-30, roč. 9, č. 2, s. 101–113 [cit. 25. 04. 2019]. ISSN 1976-6912. Dostupné z DOI: 10.5572/ajae.2015.9.2.101.
19. *Informace o kvalitě ovzduší v ČR* [online]. Český Hydrometeorologický ústav [cit. 25. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/web_generator/actual_hour_data_CZ.html.
20. *Indices Definition* [online]. Air Quality Now [cit. 25. 04. 2019]. Dostupné z: http://airqualitynow.eu/about_indices_definition.php.
21. ELSHOUT, Sef van den; LÉGER, Karine; HEICH, Hermann. CAQI Common Air Quality Index — Update with PM2.5 and sensitivity analysis. *Science of The Total Environment* [online]. 2014, roč. 488-489, s. 461–468 [cit. 25. 04. 2019]. ISSN 00489697. Dostupné z DOI: 10.1016/j.scitotenv.2013.10.060.
22. *What is LoRaWAN?* [online]. LoRa Alliance [cit. 19. 04. 2019]. Dostupné z: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.
23. *What is the LoRaWAN™ Specification?* [online]. LoRa Alliance [cit. 19. 04. 2019]. Dostupné z: https://lora-alliance.org/about-lorawan.

24. *What is LoRa?* [online]. Semtech [cit. 21. 04. 2019]. Dostupné z: https://www.semtech.com/lora/what-is-lora.

25. *Background information about LoRaWAN* [online]. The Things Network [cit. 21. 04. 2019]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/.

26. *Member directory* [online]. LoRa Alliance [cit. 19. 04. 2019]. Dostupné z: https://lora-alliance.org/member-directory.

27. HOMOLA, Vladimír. *Metody lokálních odhadů v prostoru* [online] [cit. 04. 05. 2019]. Dostupné z: http://homel.vsb.cz/~hom50/SLBGEOST/LOD/GS09.HTM.

28. *Interpolating in a Triangle* [online]. CodePlea [cit. 04. 05. 2019]. Dostupné z: https://codeplea.com/triangular-interpolation.

29. LIANG, Qinghan; NITTEL, Silvia; WHITTIER, John C.; BRUIN, Sytze de. Real-time inverse distance weighting interpolation for streaming sensor data. *Transactions in GIS* [online]. 2018, roč. 22, č. 5, s. 1179–1204 [cit. 04. 05. 2019]. ISSN 13611682. Dostupné z DOI: 10.1111/tgis.12458.

30. NIENHUYS, Han-Kwang. *Shepards interpolation method, sampling a smooth function, exp(-$x^2 - y^2$)*. [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 05. 05. 2019]. Dostupné z: https://en.wikipedia.org/wiki/File:Shepard_interpolation_2.png.

31. CHOPDE, Nitin R.; NICHAT, Mangesh K. Landmark Based Shortest Path Detection by Using A* and Haversine Formula. *International Journal of Innovative Research in Computer and Communication Engineering Vol. 1*. 2013. ISSN 2320 – 9801.

32. *Haversine formula to find distance between two points on a sphere* [online]. Geeks For Geeks [cit. 05. 05. 2019]. Dostupné z: https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/.

33. KLÉMA, Jiří. *Multiple (non) linear regression* [online] [cit. 20. 05. 2019]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/b4m36san/san_regression.pdf. Přednáška. Fakulta elektrotechnická, ČVUT.

34. WERNER, Tomáš. *Optimalizace: Elektronická skripta předmětu A4B33OPT* [online]. 2018 [cit. 20. 05. 2019]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a4b33opt/opt.pdf. Skripta. Fakulta elektrotechnická, ČVUT.

35. *Seznam zkratek* [online]. Český Hydrometeorologický ústav [cit. 21. 04. 2019]. Dostupné z: http://portal.chmi.cz/files/portal/docs/uoco/isko/grafroc/groc/gr12cz/zkratky.html.

# Příloha B

## Seznam zkratek

| | |
|---|---|
| $CH_4$ | Metan |
| $CO$ | Oxid uhelnatý |
| $NMVOC$ | Nemetanogení těkavé organické látky |
| $NO$ | Oxid dusnatý |
| $NO_2$ | Oxid dusičitý |
| $NO_X$ | Oxidy dusíku ($NO$ a $NO_2$) |
| $O$ | Atomární kyslík |
| $O_2$ | Molekulární kyslík |
| $O_3$ | Ozon |
| $PM_{10}$ | Částice / polétavý prach do velikosti $10\,\mu m$ |
| $PM_1$ | Částice / polétavý prach do velikosti $1\,\mu m$ |
| $PM_{2,5}$ | Částice / polétavý prach do velikosti $2,5\,\mu m$ |
| $SO_2$ | Oxid siřičitý |
| API | Application Programming Interface |
| BSON | Binary JSON |
| CAQI | Common Air Quality Index |
| CHMI | Czech Hydrometeorological Institute |
| DAO | Data Access Object |
| DTO | Data Transfer Object |
| ESA | European Space Agency |
| FOTA | Firmware Over-The-Air |

| | |
|---|---|
| Java EE | Java Enterprise Edition |
| Java SE | Java Standard Edition |
| JEE | Java Enterprise Edition |
| JSON | JavaScript Object Notation |
| NASA | National Aeronautics and Space Administration |
| NFR | Nomenklatura pro podávání zpráv, dle [35] |
| POJO | Plain Old Java Object |
| ČHMÚ | Český hydrometeorologický ústav |

# Příloha C

## Seznam použitých technologií, knihoven a nástrojů

- Technologie
  - Payara 5.191
  - MongoDB 4.0.5 Community
  - Java 1.8u202
    - jsoup 1.11.3
    - mongodb-driver-sync 3.10.1
    - bson-codecs-jsr310 3.4.0
    - TestNG 6.14.3
  - Java EE 8
    - Java Server Faces 2.4
      - PrimeFaces 6.2
    - graphql-java 11.0
    - graphql-java-servlet 7.3.3
    - graphql-java-tools 5.5.1
  - Maven 3.3.9
    - lesscss-maven-plugin 1.7.0.1.1
  - Less
  - Python 3.7.3
    - pandas 0.24.2
    - matplotlib 3.0.3
    - numpy 1.16.3
    - scikit-learn 0.20.3
    - BeautifulSoup4
  - Anaconda 4.6.14
- Nástroje
  - MongoDB Compass Community 1.16.3

- IntelliJ IDEA 2019.1
- IntelliJ PyCharm 2019.1
- GraphiQL 0.7.2
- Insomnia 6.3.2
- ToolsUI 4.6

# Příloha D

# Instalace a spuštění

V této kapitole popisuji, jak nainstalovat potřebné běhové prostředí pro aplikaci a jak ji nasadit na aplikační server Payara.

## D.1 Příprava běhové prostředí

V této podsekci se věnuji instalaci všech důležitých technologií, které jsou potřeba pro spuštění této práce.

### D.1.1 Stažení aplikačního serveru Payara

Aplikace je postavena na frameworku Java EE a pro její spuštění je zapotřebí aplikačního serveru. Pro svojí práci jsem zvolil server Payara[1]. Tuto práci jsem vyvíjel na verzi 5.191, která se dá stáhnout z archivu[2]. Je třeba používat plnou verzi serveru, na verzi Micro by aplikace nemusela fungovat. Stažený ZIP archiv je rozbalte na libovolné místo. Pozor, na OS Windows se nedoporučuje umisťovat složku aplikačního serveru do složky *Program Files*.

Samotný server funguje na prostředí Java Runtime Environment[3] (zkráceně JRE). Bohužel podporuje pouze Java verze 8.

### D.1.2 Databáze MongoDB

Jako databázi aplikace využívá MongoDB verze 4.0.5. Aktuální verzi i archivní verze můžete stáhnout z oficiálních stránek[4]. Nic dalšího pro spuštění MongoDB nepotřebujete.

Dále je potřeba přidat administrátorský účet a databázi[5]. Otevřete složku, kde máte MongoDB nainstalované a spusťte `mongo.exe`. Vytvořte databázi

[1] Aktuální verze se dá stáhnout z oficiálních stránek `https://www.payara.fish/software/payara-server/`.

[2] `https://search.maven.org/artifact/fish.payara.distributions/payara/5.191/zip`

[3] `https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`

[4] `https://www.mongodb.com/download-center/community`

[5] Stručný návod: `https://zocada.com/setting-mongodb-users-beginners-guide/`

*admin* pomocí `use admin`. Pomocí příkazu `db.createUser()`[6] vytvořte uživatele *admin*, nastavte mu heslo a roli `userAdminAnyDatabase`. Program `mongo.exe` ukončete.

Celý příkaz bude vypadat nějak takto:

```
db.createUser({
     user: "admin",
     pwd: "xxx",
     roles: [{role: "userAdminAnyDatabase", db: "admin"}]
})
```

Nyní znova spusťte `mongo.exe` a pomocí přepínačů `-u`, `-p` a `-authenticationDatabase` se přihlašte za nově vytvořeného uživatele *admin*. Vytvořit databázi a uživatele *air2day* stejným způsobem pomocí `use ari2day` a `db.createUser()`. Akorát jako roli nastavte `readWrite`.

### D.1.3 Propojení Payary a MongoDB

Aby se Payara k MongoDB mohla vůbec připojit, potřebuje *mongo-java-driver*. Aktuální verze je ke stažení na GitHubu[7] projektu. V projektu využívám ovladač mongodb-driver-sync verze 3.10.1, který se dá stáhnout z mvnrepository.com[8]. Stažený JAR archív vložte do podadresáře *payara5/glassfish/lib* v instalačním adresáři Payara serveru. Stejnou proceduru opakujte s JAR archivy mongodb-driver-core[9] a bson[10], bez kterých nebude ovladač korektně fungovat.

Ještě než aplikaci nasadíme a spustíme, je potřeba aplikaci sdělit přihlašovací údaje do MongoDB. Vytvoření uživatele je popsáno v podsekci výše. V této práci je uchování hesla k databázi vyřešené jako "custom resource" přímo v serveru Payara.

Otevřete administrační rozhraní na `http://localhost:4848/` a přejděte do sekce *Resrouces -> JNDI -> Custom Resources*. Vytvořte novou resource. Finálně vyplněný formulář je na obrázku E.1. *Pozn.: Heslo k MongoDB musí odpovídat heslu uživatele air2day vytvořeného v podsekci D.1.2*

## D.2 Deploy a první spuštění

V tuto chvíli máte připravené prostředí pro spuštění programu. Otevřete opět v prohlížeči administrátorské prostředí Payara serveru na `http://localhost:4848/`. Přejděte do sekce *Applications* a zde klikněte na tlačítko Deploy.

V nově otevřeném okně vyberte EAR balíček *air2day.ear* s již zabaleným programem, klikněte na tlačítko OK a počkejte na nasazení aplikace. Poté, co je aplikace úspěšně nasazena ji lze zobrazit na adrese `http://localhost:8080/`.

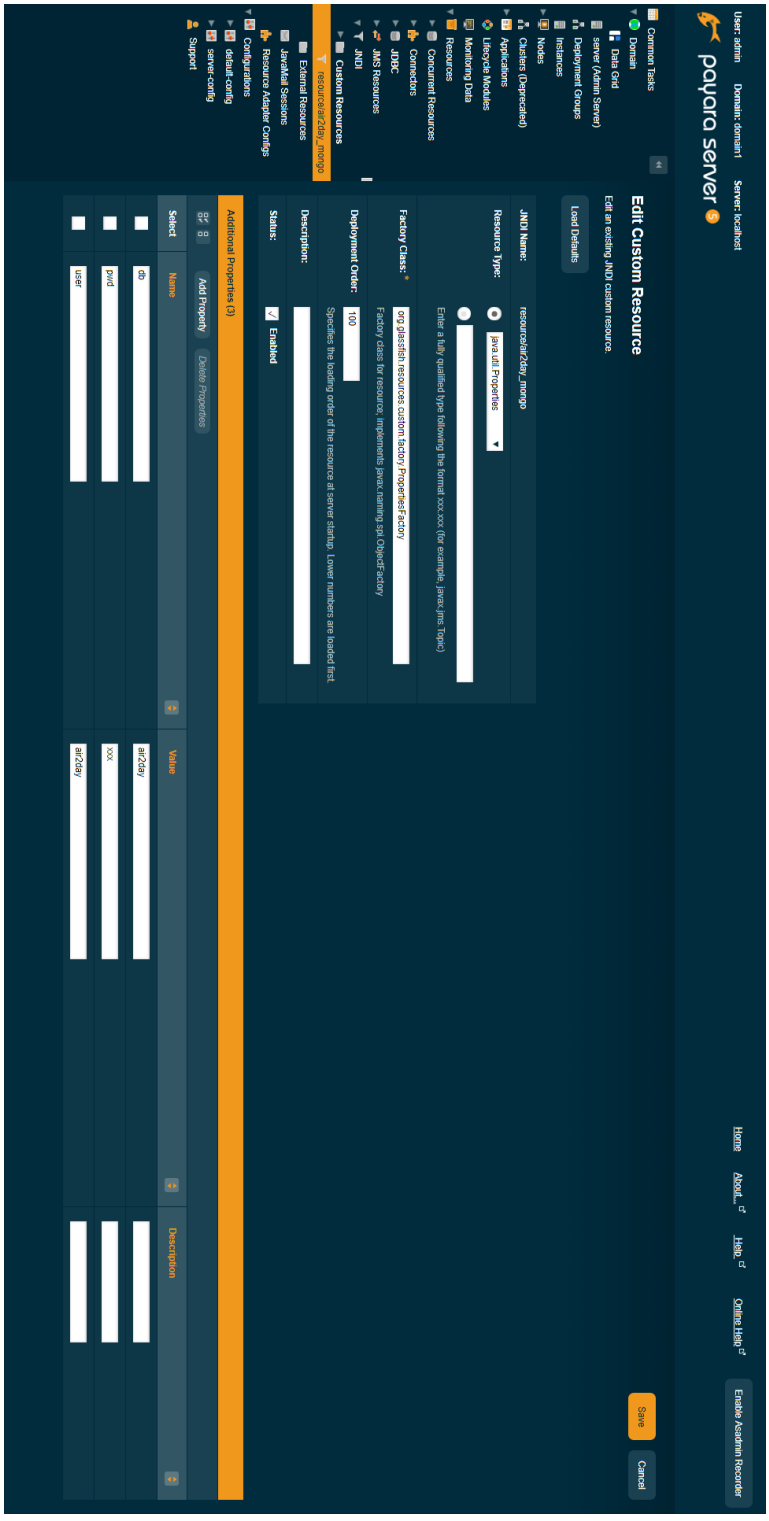
[6] `https://docs.mongodb.com/manual/tutorial/create-users/`
[7] `https://mongodb.github.io/mongo-java-driver/`
[8] `https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-sync/3.10.1`
[9] `https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-core/3.10.1`
[10] `https://mvnrepository.com/artifact/org.mongodb/bson/3.10.1`

## D.3 Shrnutí

1. Nainstalujte prostředí Java SE verze 8 ze stránek Oracle `https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`.
2. Stáhněte Payara server verze 5.191 z archivu `https://search.maven.org/artifact/fish.payara.distributions/payara/5.191/zip` ve formě ZIP archivu a rozbalte ho do libovolné složky. *Pozn.: Na OS Windows se nedoporučuje umísťovat Payara server do složky Program Files.*
3. Nainstaluje databázi MongoDB verze 4.0.5 ze stránky `https://www.mongodb.com/download-center/community`.
4. Stáhněte JAR java ovladače pro MongoDB verze 3.10.1 z `https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-sync/3.10.1` a umístěte jej do složky aplikačního serveru Payara *payara5/glassfish/lib*. To stejné udělejte s mongodb-drive-core `https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-core/3.10.1` a bson `https://mvnrepository.com/artifact/org.mongodb/bson/3.10.1`, které jsou potřebné pro správné fungování ovladače.
5. Spusťte `mongo.exe` a vytvořte administrátorský účet s rolí `userAdminAnyDatabase`. Návod k nalezení na `https://zocada.com/setting-mongodb-users-beginners-guide/`.
6. Znova spusťte `mongo.exe` s přihlaste se jako nově vzniklý *admin*.
7. Vytvořte databázi *air2day* a uživatele *air2day*, kterému nastavte libovolné heslo a roli `readWrite`. na nově vytvořenou databázi *air2day*. Pro vytvoření uživatele použijte funkci *db.createUser()*, návod najdete na `https://docs.mongodb.com/manual/tutorial/create-users/`.
8. Spusťte Payara server pomocí příkazové řádky spuštěním příkazu *asadmin start-domain domain1* a otevřete v prohlížeči `http://localhost:4848/`. *Pozn.: Program asadmin se nachází ve složce payara5/glassfish/bin/.*
9. Přejděte v menu na nastavení Resources -> JNDI -> Custom Resources, vytvořte resource a nastavte podle obrázku E.1. Položku *password* nahraďte Vámi vybraným heslem.
10. Přejděte do sekce Applications, klikněte na Deploy a vyberte přibalený air2day.ear balíček.
11. Aplikace je přístupná na adrese `http://localhost:8080/`.

# Příloha E

## Obrázky z instalace




**Obrázek E.1:** Nastavení custom resource s údaji o připojení k MongoDB, zdroj: autor