**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**


**Bakalářská práce**

# PHP knihovna na získání validních dat z MySQL databáze portálu Včelstva online

**Milan Jelínek**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Milan Jelínek

Systémové inženýrství a informatika
Informatika

Název práce

**PHP knihovna na získání validních dat z MySQL databáze portálu Včelstva on-line**

Název anglicky

**PHP library for retrieving valid data from MySQL database of the Včelstva on-line web portal**

---

**Cíle práce**

Cílem práce je navrhnout a implementovat knihovnu v jazyce PHP pro použití v rámci portálu Včelstva on-line. Tato knihovna má za úkol načíst kvantitativní data jednotlivého úlu z MySQL databáze (pořízených senzory) s možností volby časového okna či všech hodnot úlu. Knihovna vyhodnotí a vyřadí nevalidní hodnoty. Výstupem budou filtrovaná využitelná data.

**Metodika**

Bakalářská práce sestává ze dvou částí. Metodika zpracování teoretické části práce je založena na studiu odborný a vědeckých informačních zdrojů. Na základě syntézy zjištěných požadavků budou formulována teoretická východiska pro zpracování praktické části práce.

Praktická část práce spočívá v návrhu a implementaci knihovny v jazyce PHP, které bude využívána v rámci portálu Včelstva on-line. Knihovna bude sloužit k validaci a filtraci kvantitativních dat o jednotlivých včelstvech, získaných prostřednictvím IoT senzorů umístěných v úlech. Při návrhu a implementaci bude využito standardních metod a nástrojů softwarového inženýrství. Knihovna bude prakticky otestována a budou shrnuty zásadní poznatky zjištěné při její implementaci a testování a budou navrženy možnosti případného dalšího budoucího rozšíření.

**Doporučený rozsah práce**

35-40 stran

**Klíčová slova**

PHP, MySQL, SQL, MySQLi, knihovna, Včelstva on-line

---

**Doporučené zdroje informací**

Baron Schwartz, Peter Zaitsev, Vadim Tkachenko: High Performance MySQL, O'Reilly Media, ISBN-10: 1449314287
Dayle Rees: PHP Pandas
Padraic Brady: Survive the Deep End: PHP Security, ReadTheDocs.org
Phil Sturgeon, Josh Lockhart: PHP: The Right Way, PhpTheRightWay.com

---

**Předběžný termín obhajoby**

2019/20 LS – PEF

**Vedoucí práce**

Ing. Jiří Brožek, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 2. 2020

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 28. 2. 2020

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 07. 03. 2020

**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "PHP knihovna na získání validních dat z MySQL databáze portálu Včelstva online" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 20.3.2020 _____

**Poděkování**

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi, Ph.D. za vedení práce, poskytnuté rady a konzultace. Dále bych rád poděkoval doc. Ing. Janu Bartoškovi, Ph.D. a Ing. Pavlovi Junkovi za poskytnuté data.

# PHP knihovna na získání validních dat z MySQL databáze portálu Včelstva online

**Abstrakt**

Bakalářská práce „PHP knihovna na získání validních dat z MySQL databáze portálu Včelstva on-line“ se zabývá získáním a filtrací strojově pořízených dat. Práce začíná analýzou jednotlivých softwarových komponent úlohy a následným návrhem a popisem spirálového vývoje knihovny. Knihovna vrací všechny validní data nebo data mezi určenými datumy pro zadané včelstvo. Všechny takové data jsou seřazena podle datumu a veličiny senzoru, kterým byli pořízeny. Knihovna také umožňuje eliminaci odlehlých hodnot měření dle uživatelovi zadané míry.

**Klíčová slova:** PHP, MySQL, SQL, MySQLi, knihovna, Včelstva on-line

# PHP library for retrieving valid data from MySQL database of the Včelstva on-line web portal

**Abstract**

The bachelor thesis „PHP library for retrieving valid data from MySQL database of the Včelstva on-line web portal“ deals with obtaining and filtering machine data. The thesis begins with the analysis of individual software components of the task and subsequent design and description of the spiral library development. The library returns all valid data or data between specified dates for the specified hive. All such data are sorted by date and by the type of the sensor they were acquired. The library also allows the removal of outlying measurement values according to the user-specified rate.

**Keywords**: PHP, MySQL, SQL, MySQLi, library, Včelstva on-line

# Obsah

# Seznam obrázků

# Seznam kódů

# 1 Úvod

Žijeme v době, kdy chytrá zařízení prostřednictvím senzorů měří od délky spánku až po hladinu cukru v krvi. Tyto měření jsou následně použity k doporučením či k ovládání dalších prvků, které nám usnadňují život. Ne vždy ovšem tyto senzory fungují na sto procent.

My se v této práci podíváme na konkrétní výsledky měření hodnot portálu Včelstva on-line senzory umístěnými v jednotlivých úlech. Tyto senzory měří hmotnost, procentuální vlhkost a teplotu úlu. Vzhledem k faktu, že tato databáze obsahuje i nevalidní data, tak se tato práce zaměřuje na jejich odstranění a vrácení validních dat pro další využití.

Práce je koncipována teoretickou částí, kde jsou popsány jednotlivé softwarové komponenty. Těmito komponenty jsou skriptovací jazyk PHP a MySQL. V následné praktické části je popsán základní návrh knihovny (první cyklus) a postupný vývoj dalších cyklů, které shrnují nejdůležitější změny. Poslední cyklus je popsán podrobněji, kdy každá důležitá metoda je rozebrána a popsána.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Cílem práce je navrhnout a implementovat knihovnu v jazyce PHP pro použití v rámci portálu Včelstva on-line. Tato knihovna má za úkol načíst kvantitativní data jednotlivého úlu z MySQL databáze (pořízených senzory) s možností volby časového okna či všech hodnot úlu. Knihovna vyhodnotí a vyřadí nevalidní hodnoty. Výstupem budou filtrovaná využitelná data.

## 2.2 Metodika

Bakalářská práce sestává ze dvou částí. Metodika zpracování teoretické části práce je založena na studiu odborný a vědeckých informačních zdrojů. Na základě syntézy zjištěných požadavků budou formulována teoretická východiska pro zpracování praktické části práce.

Praktická část práce spočívá v návrhu a implementaci knihovny v jazyce PHP, které bude využívána v rámci portálu Včelstva on-line. Knihovna bude sloužit k validaci a filtraci kvantitativních dat o jednotlivých včelstvech, získaných prostřednictvím IoT senzorů umístěných v úlech. Při návrhu a implementaci bude využito standardních metod a nástrojů softwarového inženýrství. Knihovna bude prakticky otestována a budou shrnuty zásadní poznatky zjištěné při její implementaci a testování a budou navrženy možnosti případného dalšího budoucího rozšíření.

# 3 Teoretická východiska

## 3.1 PHP

PHP (Hypertext Preprocesor, jde o rekurzivní zkratku) je skriptovací jazyk se syntaxem podobným C, Javě a Perlu. Myšlenkou je umožnit webovým vývojářům rychle psát dynamické webové stránky. Hlavním rozdílem, například proti JavaScriptu na straně klienta je, že kód PHP je na straně serveru a vygeneruje HTML, které se pošle zpět klientovi. [1]

### 3.1.1 Historie

Za začátky PHP se musíme podívat do roku 1994, kdy Rasmus Lerdorf napsal jednoduchý soubor CGI v programovacím jazyce C. [2] (Common Gateway Interface je „Program napsaný v libovolném programovacím jazyce (název skript je zažitý, ale poněkud zavádějící). Přesnější by bylo CGI aplikace.
Aby mohl být program použit jako CGI skript, musí splňovat jedinou podmínku: po spuštění musí na výstup vygenerovat HTTP hlavičku Content-Type (když chce, může i další) následovanou prázdným řádkem, a pak vlastní obsah dokumentu (typicky HTML stránku, ale může se jednat o libovolný typ plain text, obrázek... jen je potřeba nastavit hlavičku Content-Type na odpovídající hodnotu“) [3]) Rasmus toto CGI používal na sledování návštěvnosti svého online životopisu a pojmenoval ho "Personal Home Page Tools" , nicméně známější pod zkratkou PHP Tools.

Postupem času Rasmus přepsal PHP Tools, kdy nová verze získala schopnost práce s databází nebo poskytovala framework pro tvorbu jednoduchých dynamických webových aplikací. V červnu 1995 Rasmus vypustil kód PHP Tools do světa, čímž umožnil vývojářům používat, upravovat a vylepšovat kód.

V září stejného roku Rasmus na chvíli upustil od názvu PHP Tools a začal používat název FI (Forms Interpreter). Nová verze už byla podobná PHP, jak ho známe nyní. Ve spoustě věcí si byla podobná s Perlem (Interpretovaný programovací jazyk. Populární nástroj pro tvorbu CGI skriptů). Popularita FI stále rostla. Ne zatím, jako programovacího jazyka, ale spíše jako CGI nástroje.

To se změnilo v říjnu 1995 s vydáním celého nově přepsaného kódu a novým jménem "Personal Home Page Construction Kit". Jazyk tohoto skriptovacího jazyka byl schválně navržen, aby se strukturou podobal jazyku C.

Kód byl znovu celý předělán a v dubnu 1996 vydán pod názvem PHP/FI. Tato verze implementace se už nebyl pouhý nástroje, nicméně programovací jazyk. Zahrnovala podporu databází DBM, Postgres95 a mSQL. Podporovala také cookies či funkce definované uživatelem a spoustu dalších vylepšení. V červnu PHP/FI přešlo na verzi 2.0 a při přechodu z bety se předělal celý virtuální procesor. [2]

Během dalších let si PHP prošlo řadami 3,4,5. Řada 6.x, která měla podporovat Unicode se ovšem nikdy nevydala. Aby se předešlo zmatkům poslední aktuální řada se nazývá PHP 7.x.

### 3.1.2 **PHP 7**

V této aktuální řadě byli předělány datové struktury a refaktorován kód. Nicméně nezůstalo pouze u těchto změn. My si ty nejdůležitější projdeme.

První změnou je možná typová kontrola skalárních vstupů. Další související změnou je možná definice návratových hodnot přímo do definice metody. Změnou prošlo také vyhazování errorů, kdy dříve PHP vyhazovalo fatální errory namísto výjimek. Byl tedy vytvořen nový typ výjimek EngineException a nový abstraktní předek pro všechny typy výjimek.

```
BaseException (abstract)
 +- EngineException
 +- ParseException
 +- Exception
     +- ErrorException
     +- RuntimeException
         +- ...
     +- ...
```

**Obrázek 1: Abstraktní třída BaseException potomci [4]**

V této řadě jsou také přidány méně důležité funkce, které zlepšují či zjednodušují implementaci. Za všechny asi operátor spaceship, anonymní třídy či možnost výrazu return v generátorech. [4-7]

### 3.1.3 **Druhy využití PHP**

#### 3.1.3.1 Skripty na straně serveru

Obvyklé použití PHP. Pro funkčnost je potřeba mít zajištěné 3 věci. První je PHP parser (CGI nebo server modul). Druhou je web server propojený s PHP instalací.
Třetím a posledním požadavkem je web browser.

V případě, že jsou všechny součásti zajištěné v podobě pronájmu či služby stačí nahrát PHP skript a je hotovo. V opačném případě jsou na výběr 2 možnosti propojení PHP a serveru. PHP má moduly známější pod názvem SAPI (Server Application Programming Interface). Příkladem může být DLL soubor(php5apache2.dll). Tento modul krom dalších funkcí poskytuje rozhraní mezi PHP 5 a Apache 2.0. Dále PHP podporuje ISAPI, což je rozhraní od Microsoftu. I v moment, kdy PHP nepodporuje daný typ serveru, tak se PHP dá použít jako CGI. [8]

#### 3.1.3.2 Skripty v příkazové řádce

PHP a CLI (Command Line Interface) přišlo ve verzi 4.2.0 jako experimentální funkce. Ve verzi 4.3.0 už CLI bylo oficiální jako SAPI. Lépe řečeno CLI mělo svůj oddělený spustitelný soubor. Ve většině případů se CLI v příkazové řádce používá pro administraci serveru (zálohování, čištění a archivování starých dat). Výhodou skriptu do příkazové řádky je, že je jednoduchý na napsání a použití. [9-10]

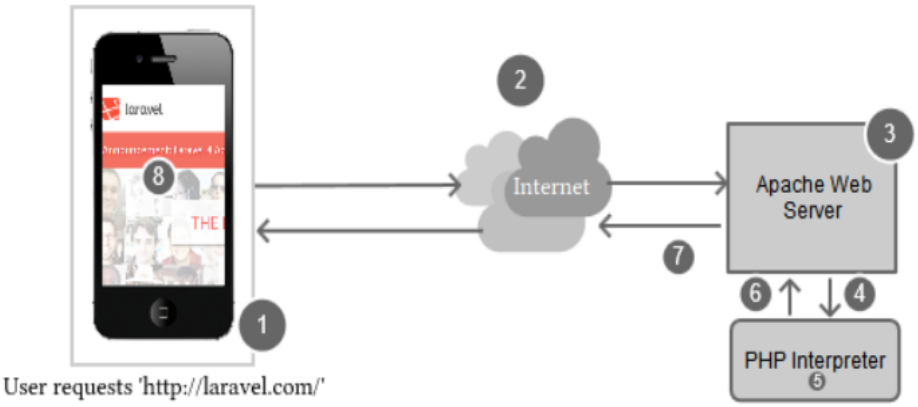
#### 3.1.3.3 Počítačové aplikace

PHP umožňuje pomocí rozšíření PHP-GTK (GIMP Tool Kit) psaní aplikací pro počítač. PHP-GTK je složena z PHP jazyka a GTK knihovny. Zjednodušeně je to PHP s více funkcemi. Výstupem není tedy webová stránka nicméně standartní program s GUI. Spojením těchto dvou komponent umožňuje spuštění jak na Windows, tak na Linuxu. Výhoda oproti normálnímu PHP je, že má program přístup jak k databázím, tak má práva k počítači (Protože je na straně klienta). Toto rozšíření ovšem není součástí oficiální PHP distribuce. [11]

### 3.1.4 Schéma průběhu PHP skriptu

PHP je interpretovaný jazyk. Ve zkratce to znamená, že až když je daný požadavek na danou stránku je PHP kód načten a zpracován. Tento průběh se tedy liší od klasických jazyků, jako Java nebo C#.

Pojďme se podívat na životní cyklus požadavku na webovou stránku. Vše začne u uživatele, který vyvolá požadavek na webovou stránku například ve svém vyhledávači. Pro náš případ používáme Apache a ten tedy zachytí tento požadavek a podívá se, jestli takový soubor existuje. Pokud existuje a má příponu .php předá soubor PHP interpretru. PHP interpreter si přečte celý soubor a provede veškeré instrukce. Když je hotov předá výslednou vygenerovanou stránku zpět, aby ji Apache vrátil uživateli na obrazovku.




**Obrázek 2: Průběh požadavku uživatele na webovou stránku. [12]**

1. Uživatel zadá například http://laravel.com do prohlížeče.
2. Uživatel potvrdí svůj požadavek. Vyhledávač pošle požadavek web serveru.
3. Web server přijme a analyzuje požadavek. Uvědomí si, že jsme neurčili specifický soubor, a tak se bude dívat po souboru index.php
4. Apache rozezná příponu .php a pošle soubor na zpracování PHP
5. V tomto kroku PHP skript se zpracovává. Takže komunikuje s databází, vykresluje, zkrátka probíhá kód napsaný v PHP.
6. PHP skončil s prováděním skriptu a posílá vygenerovaný výstup zpět web serveru
7. Apache převezme tento výstup a pošle ho zpět uživateli do prohlížeče.

8. Prohlížeč zobrazí výstup převzatý od Apache na zařízení. [12]

## 3.2 MySQL

MySQL je open source relační databáze (RDBMS) založená na strukturovaném dotazovacím jazyce (SQL). MySQL běží prakticky na všech platformách od Linuxu, UNIX až po Windows. Přestože je MySQL použitelný v celé řadě aplikací, je nejčastěji spojován s webovými aplikacemi.

Například MySQL je důležitou součástí LAMP. LAMP je platforma pro vývoj webových aplikací, která používá operační systém Linux, Apache jako webový server, MySQL (nebo MariaDB) jako systém pro správu relačních databází a PHP jako objektově orientovaný skriptovací jazyk. (Někdy se místo PHP používá Perl nebo Python.)

Dnes je MySQL RDBMS za mnoha předními webovými stránkami na světě a bezpočtem firemních a spotřebitelských webových aplikací. Příkladem jsou společnosti YouTube nebo Facebook. [13-14]

### 3.2.1 Historie

MySQL byl vymyšlen švédskou společností MySQL AB založenou Davidem Axmarkem, Allanem Larssonem a Michaelem Wideniusem. Vývoj začal v roce 1994 a první verze z roku 1995 byla vytvořena pro osobní použití z mSQL na základě nízko-úrovňového jazyka ISAM, který tvůrci považovali za pomalý. Tím, že zachovali API z mSQL hodně vývojářů používali MySQL namísto mSQL, který byl tehdy proprietárním(placeným) softwarem.

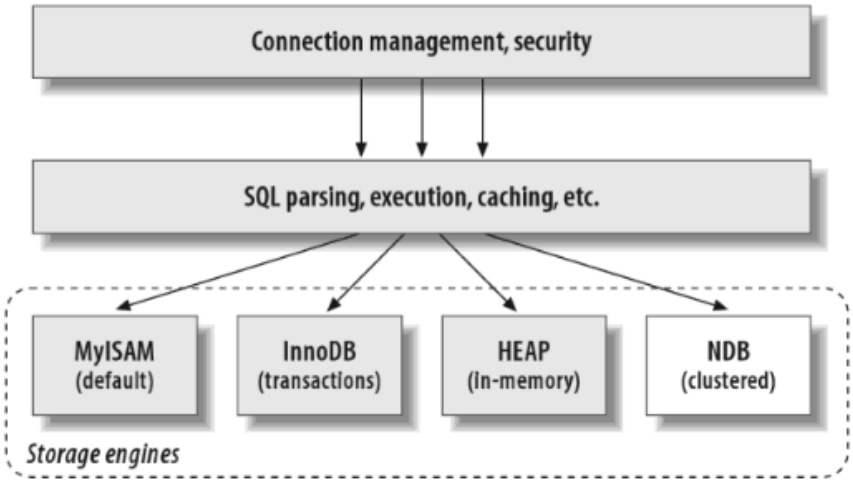
Později byl MySQL koupen společností Sun Microsystems v roce 2008 a poté Oraclem, v roce 2010. Vývojáři mohou MySQL používat pod GNU General Public License (GPL). Pro komerční účely je potřeba si zakoupit licenci od Oraclu. [14]

### 3.2.2 Logické schéma

MySQL se dost liší od ostatních databázových serverů. MySQL je dost flexibilní, aby fungovala ve velmi náročných prostředích, jako jsou webové aplikace. Současně MySQL můžeme potkávat ve vestavěných aplikacích, datových skladech či v indexování obsahu.

Abychom získali maximum z MySQL, musíme porozumět jeho designu. Jak už bylo zmíněno o odstavec výše MySQL je v mnoha ohledech flexibilní. Je schopná běhu na široké škále hardwaru a podporuje různé typy dat. Nejneobvyklejší a zároveň nejdůležitější vlastností MySQL je odlišnost storage enginů. Jejich design odděluje zpracování dotazů a další úlohy serveru od ukládání a načítání dat. Díky této odlišnosti si můžete zvolit, jak mají být data ukládána a jaké další funkce či charakteristiky chcete.

Abychom lépe pochopili, kam přesně storage enginy zapadají musíme si rozebrat logické schéma. Vybral jsem zjednodušenou verzi, která úplně neodpovídá implementaci, nicméně pro naše účely je vhodnější a přehlednější.




**Obrázek 3: Logické schéma databáze. (zjednodušená verze) [16]**

Horní vrstva se skládá ze služeb, které nejsou pro MySQL unikátní. Jsou to běžné síťové klient/server nástroje, či z pohledu serveru tam patří správa spojení, autorizace, bezpečnost a další.

V prostřední části se nachází skoro všechny důležité části. Kód pro analýzu dotazů, analýza, optimalizace, ukládání do paměti i všechny vestavěné funkce (datumy, matematika, šifrování a další). Veškerá funkčnost poskytovaná storage enginy je na této úrovni: například uložené procedury, spouštěče a zobrazení.

Spodní vrstva obsahuje storage enginy. Ty se starají o ukládání a načítání všech dat uložených pomocí MySQL. Není velkým překvapením, že každý tento storage engine má své přednosti a nevýhody. Server s nimi komunikuje pomocí rozhraní API uložiště, které není specifické k danému storage enginu. API se skládá z okolo 20 funkcí, které provádí operace typu „spusť“ transakci či „vrať řádku s tímto primárním klíčem“ atd. Storage

enginy nepracují s SQL a ani nekomunikují mezi sebou. Pouze odpovídají na požadavky z vyšší vrstvy MySQL. [15-17]

### 3.2.3 **Storage enginy**

Než si ukážeme jednotlivé druhy storage-enginů, respektive jak fungují, a jaké mají přednosti i nevýhody podíváme se na teorii společnou pro všechny. [15,16]

#### 3.2.3.1 Druhy storage-enginů

Nejpoužívanějším a doporučeným storage enginem krom specifických případů je InnoDB. InnoDB podporuje ACID (atomicitu, konzistenci, izolovanost, trvalost) a jako jediný podporuje omezení referenční integrity cizího klíče.

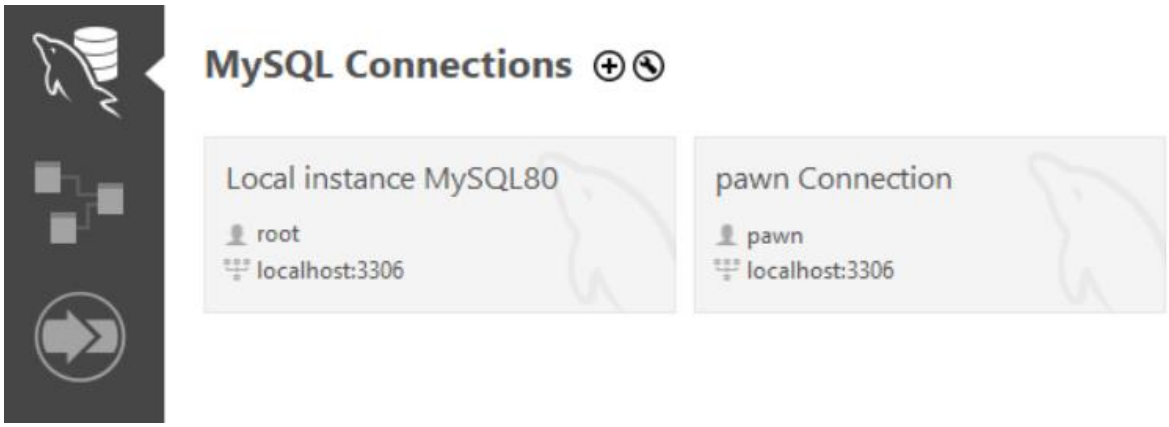
Nejrychlejší variantou je Memory storage engine. Sice nepodporuje transakce, ale je ideální pro vytváření dočasných tabulek nebo rychlého vyhledávání. Data jsou nenávratně ztracena, pokud je databáze restartována.

Dalším je Archive. Archive je optimalizovaný na rychlost insertu dat. Při příjmu dat jsou hned komprimována. Nepodporuje transakce. Tento storage engine je ideální pro zřídka odkazovaná historická archivovaná data. [17]

### 3.2.4 **MySQL Workbench**

MySQL Workbench je grafický nástroj pro práci s MySQL servery a databázemi. MySQL Workbench je kompatibilní s verzemi MySQL serveru 5.x výše až k 5.6, od které je plně funkční.

MySQL Workbench umožňuje připojení a jejich správu do různých serverů.




**Obrázek 4: Náhled do MySQL Workbench. Zobrazené 2 připojení**

Dále podporuje modelování databází a správu dat. MySQL Workbench umožnuje provádět i administrační část, takže správa uživatelských práv, zálohy, migrace a další úkony. [18]

# 4 Vlastní práce

## 4.1 Shrnutí požadavků

Prvním požadavkem knihovny je schopnost načítat data z MySQL databáze. Tyto data byla poskytnuta přes pohled do ostré databáze. Práce pracuje s daty z pohledu, jako s reálnou architekturou databáze.

Druhým požadavkem je, že knihovna musí být napsaná v jazyce PHP. Tento požadavek je z důvodu interní struktury portálu Včelstva online.

Knihovna musí být schopna vyřadit nevalidní data jednotlivého uživatelem zadaného včelstva. Po profiltrování dat vrátí knihovna validní data zpět uživateli knihovny pro budoucí použití.

## 4.2 Začátek a první cyklus

Aby knihovna mohla být testována v průběhu implementace, muselo se nasimulovat reálné budoucí řešení. Začalo se MySQL databází a stáhnutím dat poskytnutými administrátorem včelstva online přes view. Data viz obrázek struktury dat.

```
id,cas,hodnota,stanoviste_id,vcelstvo_id,senzor_id,nazev_senzoru,popis_senzoru,velicina_id,nazev_veliciny,jednotka_veliciny
747237,2019-01-01T17:02:47+01:00,36.388,102,245,36,Hmotnost,Hmotnost úlu,1,Hmotnost,kg
747238,2019-01-01T17:02:47+01:00,5.5,,,37,Vnitřní teplota 1,Teplota uvnitř úlu.,2,Teplota,°C
747239,2019-01-01T17:02:47+01:00,87,102,245,39,Vlhkost vzduchu,Vlhkost vzduchu uvnitř úlu,3,Vlhkost vzduchu,%
747241,2019-01-01T17:02:47+01:00,5.6,102,,40,Vnější teplota,Venkovní teplota na stanovišti,2,Teplota,°C
```

**Obrázek 5: Zobrazení 4 řádků vstupních dat z poskytnutého view**

Byl nainstalován a nastaven MySQL server a pro správu MySQL Workbench. Je spousta jiných nástrojů, které by pro daný úkol posloužili stejně dobře, nicméně byl použit MySQL Workbench z důvodu předchozí zkušenosti s nástrojem. Dalším krokem je zbavení se prázdných řetězců. Toto není nutné, nicméně zastávám myšlenku, že prázdná hodnota je nastavena na null, nikoliv na prázdný řetězec.

Tento problém jsem vyřešil konzolovým programem v jazyce C#, kdy program načetl data po řádcích ze souboru a místo prázdných řetězců dosadil null. Výsledek viz obrázek.

```
747237,2019-01-01T17:02:47+01:00,36.388,102,245,36,Hmotnost,Hmotnost úlu,1,Hmotnost,kg
747238,2019-01-01T17:02:47+01:00,5.5,null,null,37,Vnitřní teplota 1,Teplota uvnitř úlu.,2,Teplota,°C
```

**Obrázek 6: Předělaná data na null místo prázdných řetězců.**

Druhou částí je PHP komponenta. K vývoji byl zvolen JetBrainsPhpStorm verze 2019.2.3 a na webserver byl použit Apache v rámci programu XAMPP ve verzi v3.2.4. Na propojení PHP knihovny a MySQL databáze bylo využito MySQLi. Dá se použít knihovna PDO, ale podle benchmarku je MySQLi rychlejší.

### 4.2.1 Analýza dat z databáze

Data ze souboru byla načtena do databáze Thesis tabulka thesis. Dalším krokem byla analýza dat. Bylo potřeba určit kolik a jakých sloupců je potřeba na jednoznačné určení dat jednotlivého včelstva. Z dat vyplynulo, že je potřeba pouze sloupec vcelstvo_id. Pokud známe hodnotu sloupce vcelstvo_id, tak není problém si dohledat příslušné senzory a jejich data. Tento postup musí být dodržen z důvodů nekonzistence číslování senzorů (senzor_id 32 měří pro více včelstev naráz. Přitom jde o dva odlišné senzory). Senzory dokážou měřit a zaznamenávat 3 typy dat. Těmi jsou hmotnost, teplota a vlhkost vzduchu uvnitř úlu.

| | senzor_id | popis_senzoru | vcelstvo_id |
|---|---|---|---|
| | 36 | Hmotnost úlu | 887 |
| | 37 | Teplota uvnitř úlu. | 887 |
| ▶ | 39 | Vlhkost vzduchu uvnitř úlu | 887 |

**Obrázek 7: Data získaná selectem z databáze**

Z analýzy vzešlo také několik problémů, které vyžadovaly řešení. Prvním byly null hodnoty v některých případech sloupce stanoviste_id a vcelstvo_id. Další problém nastal při analýze hodnot jednotlivých typů měření. Vyskytovaly se záporné hodnoty hmotnosti, jasně nevalidní hodnoty -1000 stupňů u teploty, stejně jako nesmyslné hodnoty procent.

### 4.2.2 Prvotní analýza nevalidních dat

Před tím, než bude použit jakýkoliv složitější algoritmus se musíme zbavit všech nevalidních dat popsaných v odstavci výše. Bylo zvoleno vyřešení tohoto problému

pomocí přídavných podmínek do selectů. Tyto selecty budou volány přes PHP, a proto se jimi budeme zabývat později.

Tímto se přesouváme na druhý problém a tím je vcelstvo_id = null nebo stanoviste_id = null. Tento problém má 2 řešení. Prvním je pouhé zanedbání takových hodnot. V takovém případě by v databázi ubylo 500 tisíc záznamů z 3 milionů. Číslo je aktuální v době analýzy. Vzhledem k aktivnímu stavu serveru se množství takovýchto nevalidních data stále zvětšuje, jak přibývají další a další záznamy ze senzorů.

### 4.2.3 **Prvotní návrh knihovny**

První navrženou třídou byl singleton (návrhový vzor jedináček) MySQL_Database_Handler. Tato třída má za úkol zpracování požadavků na selecty knihovny a následnou komunikaci s MySQL databází.

Druhá třída Row_Of_Data byla navržena jako abstraktní třída, od které jsou odvozeny 3 specializované třídy pro každý typ dat. Každá taková třída má své metody pro správu dat.

Další třídou je Solver. Solver byl navržen jako singleton třída, která se stará o veškeré výpočty a odstranění nevalidních hodnot z načtených dat z databáze. Metody této třídy jsou postupně volány na nefiltrovaná data.

Poslední třídou byl Facade (návrhový vzor fasáda). Tato třída sloužila ke zjednodušení rozhraní a zavolání tříd a jejich metod pro vykonání požadavků dle uživatelova přání.

## 4.3 **Vývojové cykly 2 a 3**

Knihovna prošla čtyřmi vývojovými cykly. V každém cyklu byla provedena analýza problémů a navržena příslušná řešení. S každým cyklem byli přidány funkcionality a implementovány opravy.

Následující dvě kapitoly se zabývají těmito změnami ve struktuře knihovny a postupnému doplňování funkcionality návrhu, který byl implementován v prvním cyklu. Vzhledem k nízké relevanci nepoužitého či špatného kódu jsou oba tyto cykly obecně shrnuty největšími změnami. Veškeré finální metody a řešení jsou proto popsány až u konečné implementace.

### 4.3.1 **Cyklus 2**

V druhém cyklu byl změněn přístup třídy Facade k požadavkům na přístup dat. Třída Facade už nepovolovala výstup jednotlivých hodnot senzorů pomocí zadání vcelstvo_id a typu veličiny. Po změně vypisovala všechny validní data seřazená dle času a senzor_id pro zadané včelstvo.

V tomto cyklu byla přidána kontrola sousedních hodnot nejdříve obousměrná, kdy každá hodnota byla porovnávána s hodnotami okolními. Když se tyto hodnoty lišili na jednu ze stran o dvojnásobek průměru, kdy pro zamítnutí hodnoty byla potřeba neshoda 3 sousedních hodnot z 5 v daném směru, byli smazány. Tato metoda poté byla předělána na jednosměrnou, kdy se metodě snížil čas na projití na polovinu, nicméně na konci cyklu byl její vývoj pozastaven s potřebou najití lepšího a efektivnějšího algoritmu.

Další důležitou funkcí bylo zpětné párování hodnot k včelstvu, kde se vcelstvo_id rovnalo null. Tato třída byl nápad zachránit alespoň některé nevalidní data. Tato třída dokázala bezpečně zachránit 30 procent nevalidních dat (vcelstvo_id null nebo stanoviste_id null). Nevýhodou byl jeden dlouhý select, který ovlivňoval rychlost i paměť knihovny. Navíc zpětné párování není funkce, která by měla běžet při každém spuštění. Ideálně by tento problém měl být vyřešen SQL skriptem přímo v databázi, a proto byl vývoj této metody pro tento cyklus pozastaven.

### 4.3.2 **Cyklus 3**

Během třetím cyklu byla předělána třída na ukládání dat Row_Of_Data, kdy se ukázalo, že nejsou potřeba 3 specializovaní potomci. Výsledkem byla změna z abstraktní třídy Row_Of_Data na klasickou třídu a vynechání jejích potomků. V rámci tohoto kroku byla přepsána velká část knihovny. Důvodem pro přepsání bylo využívání polymorfismu a s tím související určování typu selectů a metod dle potomka této třídy.

V tomto cyklu byla přidána metoda na hledání malých ostrůvků dat v souboru a jejich označení a vynechání. Tato metoda v tomto cyklu byla pouze experimentální a spočívala ve vypočtení průměru (průměr je vyšší než modus změny dat) a vynásobením průměru 5. Toto číslo se porovnávalo s reálnými změnami hodnot. Myšlenka této metody byla v odstranění takových ostrůvků dat, která měli velikost menší jak 2 hodnoty. V průběhu zpětné analýzy a plánování dalšího cyklu tato metoda ukázala potenciál, a díky tomu byla označena a předělána do čtvrtého cyklu.

## 4.4 Konečná implementace

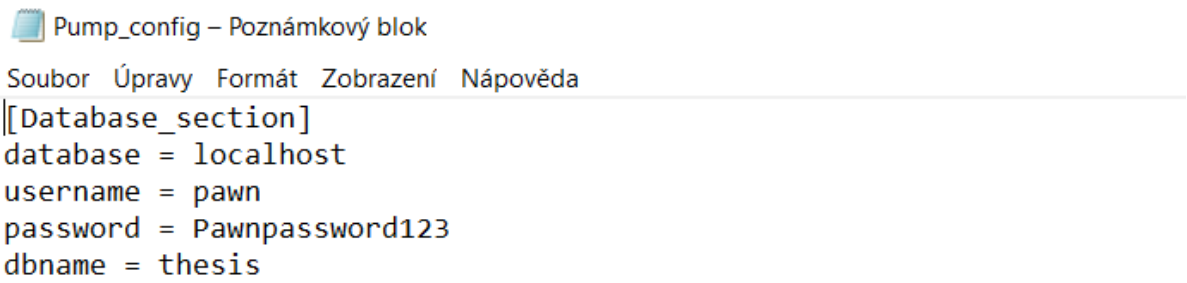
Tato kapitola popisuje konečnou implementaci této knihovny. V každé kapitole najdete seznam nejdůležitějších funkcí setříděného podle jednotlivých tříd. V každém takovém odstavci je teoretická myšlenka za danou metodou a popsaný kód metody.

### 4.4.1 Třída MySQL_Database_Handler

MySQL_Database_Handler je singleton třídou, která se stará o komunikaci se serverem. Třída si připraví select dle požadavku, poté si ověří, jestli spojení s databází je či není navázáno a následně provede select. V případě zamítnutí přihlášení či jiného problému vyhodí výjimku.

#### 4.4.1.1 Metoda open_Con

Metoda open_Con slouží k načtení adresy databáze, přihlašovacího jména, hesla a jména databáze, ke které se knihovna bude připojovat. Metoda načte přihlašovací údaje ze souboru Pump_config.ini. Výhoda tohoto provedení nastane v případě, kdyby se útočníkovi podařilo dostat k zdrojovému kódu PHP knihovny a zobrazit si jej. Při takové situaci útočník nezíská přístupové údaje k databázi, protože ty jsou uložené jinde na serveru (Klasicky mimo složku webového serveru).




**Obrázek 8: Zobrazení syntaxe souboru Pump_config.ini.**

Metoda určí dle hodnoty privátní proměnné $conn, jestli vrátí již existující spojení nebo si nejprve musí vytvořit nové. V případě nového spojení si metoda nejdříve zkontroluje, jestli požadovaný konfigurační soubor existuje. Pokud takový soubor neexistuje vyhodí chybovou hlášku. Pokud takový soubor existuje, načte výše uvedené

hodnoty ze souboru Pump_config.ini a s jejich pomocí se pokusí přihlásit do databáze. Když je metoda úspěšná vrátí spojení. V opačném případě vrací chybovou hlášku.

```
private function open_Con()
 {
     if ($this->conn == false) {
         $file_Path='C:\xampp\Pump_config.ini';
         if(!file_exists($file_Path)){
             throw new Exception('Neexistující soubor Pump_config.ini ');
         }Else{
             $config = parse_ini_file($file_Path);
             $this->conn=mysqli_connect($config['database'],
$config['username'],$config['password'],$config['dbname']);
             if ($this->conn==false){
                 throw new Exception('Přihlášení do databáze se
nezdařilo(zkontrolujte Pump_config.ini) a zkontrolujte přístupnost serveru');
             }

         }
     }
     return $this->conn;
 }
```

**Kód 1: Metoda open_Con**

K této funkci existuje opačná metoda close_Con, která slouží k uzavření existujícího spojení mezi knihovnou a databází. V souvislosti s uzavřením spojení nastavuje $conn zpět na false.

#### 4.4.1.2 Metoda get_Senzors_From_Vcelstvo

Metoda přijímá 3 argumenty. Metoda slouží k získání identifikátorů senzorů a druhu veličiny měřené daným senzorem včelstva. V případě volání z metody get_Data jsou argumenty $date_start a $date_end nastaveny na null. V opačném případě je tato metoda volána z metody get_Data_Between_Dates a tyto parametry jsou nastaveny na konkrétní dny. Select vrací číslo senzoru, včelstvo a jednotku, ve které senzor měří, případně senzory ohraničené datumy. Posledním krok této metody spočívá v zavolání metody open_Con a

provedení selectu, jehož výsledek je výstupem této metody. Případná nulová hodnota výsledků je řešena v nadřazených metodách.

```
public function get_Senzors_From_Vcelstvo($vcelstvo_id, $date_start,
$date_end)
{
    if($date_start==null and $date_end==null)
    {
        $sql = "select distinct senzor_id ,vcelstvo_id,velicina_id  from
thesis where vcelstvo_id= $vcelstvo_id";
    }
    else{
        $sql = "select distinct senzor_id ,vcelstvo_id, velicina_id  from
thesis where vcelstvo_id= $vcelstvo_id  and cas>='$date_start' and
cas<='$date_end'";
    }
    return $this->open_Con()->query($sql);
}
```

**Kód 2: Metoda get_Senzors_From_Vcelstvo**

#### 4.4.1.3 Metoda get_Data_From_Senzor

Metoda slouží k získání všech dat z databáze patřících ke konkrétnímu senzoru či data senzoru omezená 2 datumy.

Do metody vstupuje 5 argumentů, které ovlivňují výsledný select spuštěný nad databází. První 2 argumenty $vcelstvo_id, $senzor_id slouží pouze ke specifikování dat požadovaných uživatelem knihovny v selectu.

Argument $velicina_id určuje podmínky druhu selectu. U hmotnosti požadujeme hodnoty větší jak nula kilogramů. Teplotní hodnoty včelstva se skládají z hodnot větších jak -40 °C a menších jak 70 °C. V případě procentuální vlhkosti chceme hodnoty větší jak 10 % a menší jak 100 %. Následující blok kódu else zabezpečuje knihovnu proti momentu, kdy by se do databáze začali nahrávat hodnoty jiného typu měření. Myšlenkou je, že se takové data nezpracují nicméně ve výstupu knihovny bude vidět tento senzor. Tyto podmínky selectů vznikly podle analýzy dat z databáze.

```
public function get_Data_From_Senzor($vcelstvo_id, $senzor_id,$velicina_id
,$date_start, $date_end){
    if ($velicina_id==1){ //Hmotnost(kilogramy)
        $sql = "select cas, hodnota, stanoviste_id from thesis
            where senzor_id=$senzor_id and vcelstvo_id=$vcelstvo_id and
hodnota >0";
    }
    elseif($velicina_id==2){//Teplota (stupňe celsia)
        $sql = "select cas, hodnota, stanoviste_id from thesis
            where senzor_id=$senzor_id and vcelstvo_id=$vcelstvo_id and
(hodnota >-40 and hodnota<70)";
    }
    elseif ($velicina_id==3){//Vlhkost (procenta)
        $sql = "select cas, hodnota, stanoviste_id from thesis
           where senzor_id=$senzor_id and vcelstvo_id=$vcelstvo_id  and
(hodnota >10 and hodnota <100)";
    }
    else{
        return null;
    }
```

**Kód 3: První část metody get_Data_From_Senzor**

Poslední 2 argumenty $date_start, $date_end slouží k časovému omezení a v případě jejich nenulových hodnot se tyto doplňující podmínky připojí k už existujícímu selectu drženého v proměnné $sql. Následný příkaz zajistí získání otevřeného spojení a vrátí výsledek složeného selectu.

```
if($date_start!=null and $date_end!=null)
    {
        $sql.=" and cas>='$date_start' and cas<='$date_end'";
    }
    return $this->open_Con()->query($sql);
}
```

**Kód 4: Druhá část metody get_Data_From_Senzor**

### 4.4.2 Třída Pump

Třída Pump implementuje ochrany vstupů knihovny a po provedení ochran vstupů použije instance ostatních tříd pro návrat validních hodnot senzorů.

#### 4.4.2.1 Metoda get_Data

Metoda get_Data vrací všechny naměřené validní hodnoty pro uživatelem zadané včelstvo. První krok této metody je ověření validity vstupu proměnné $vcelstvo_id. Toho dosáhne ověřením, jestli zadaná hodnota je číslo nebo číselný řetězec. Dalším krokem této metody je načtení senzorů patřících k danému včelstvu pomocí instance třídy MySQL_Database_Handler.

```
function get_Data($vcelstvo_id)
{
    $data_Array= array();
    $data_Temp=null;
    $database_Handler=MySQL_Database_Handler::getInstance();
    $solver=Solver::getInstance();

    try {
        if(!is_numeric($vcelstvo_id)){
            throw new Exception('Hodnota $vcelstvo_id není číslo');
        }
        $senzor_ids=$database_Handler-
>Get_Senzors_From_Vcelstvo($vcelstvo_id,null,null);
```

**Kód 5: První část metody get_Data**

Metoda dále využije seznam senzorů a jejich typů získaných z databáze a ověří jejich počet. V případě nulového počtu je vrácena errorová hláška V opačném případě jsou pro každý senzor navráceny hodnoty záznamů z databáze. Dalším krokem je zavolání instance Solveru a provedení hledání nevalidních bloků dat pro každý senzor. Catch v této metodě chytá potenciální chyby v programu, vždy s popisem problému.

```
    if ($senzor_ids->num_rows > 0)
        {
            while($row = $senzor_ids->fetch_assoc()) {
                $data_Temp= $this-
>call_Base_Select_Function_According_To_Type($row,null, null);
                $data_Temp= $solver->go_Thru_Remove_Invalid_Blocks($solver-
>countDifference($data_Temp));
                array_push($data_Array,$data_Temp);
            }
            $database_Handler->close_Con();
        } else {
            throw new Exception('Pro zadané včelstvo není žádný senzor');
        }
    }catch(Throwable $e){
        return   $e->getMessage();
    }
    return $data_Array;
}
```

**Kód 6: Druhá část metody get_Data**

#### 4.4.2.2 Metoda get_Data_Between_Dates

Implementace této metody je podobná get_Data. Rozdílem jsou 2 extra parametry na ohraničení časového okna záznamů, tyto inputy od uživatele jsou prověřeny metodou validate_Date. validate_Date nejdříve nahradí veškeré potenciální hrozby speciálních znaků v zadaném stringu pro budoucí select a potom ověří pomocí regulárního výrazu správnou strukturu. Následně zavolá stejnou obslužnou metodu, jako metoda get_Data s parametry časového omezení.

```
public function validate_Date($input_str) {
    $return_str = str_replace( array('<',';','|','&','>',"'",'"',')','('),
array(''), $input_str );
    if (preg_match('/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])$/',
$input_str)) {
        return $return_str;
    } else {
        throw new Exception('Zadaný datum $input_str neprošel kontrolou ');
    }
}
```

**Kód 7: Metoda validate_Date**

#### 4.4.2.3 Metoda get_Data_Standard_Deviation

Metoda je slouží k odstranění extrémních hodnot pozorování ze souboru. Metoda si nejdříve zkontroluje validnost vstupu $numberOfStandardDeviation. Proměnná $vcelstvo_id je ošetřená v metodě get_Data. V případě vadné hodnoty $vcelstvo_id by se z metody get_Data vrátil string s chybou. Tento string by byl dále poslán nahoru.

```
function get_Data_Standard_Deviation($vcelstvo_id,
$numberOfStandardDeviation){
    $solver=Solver::getInstance();
    try{
        if(!is_numeric($numberOfStandardDeviation)){
            throw new Exception('Hodnota $vcelstvo_id není číslo');
        }else{
            if ($numberOfStandardDeviation<0 and 4<$numberOfStandardDeviation
)
            {
                throw new Exception('Hodnota $vcelstvo_id není v rozmezí 0-
4');
            }
        }
        $allRows=$this->get_Data($vcelstvo_id);
    } catch(Throwable $e){
        return   $e;
    }
    if (is_string($allRows)){
        return   $allRows;
    }
```

**Kód 8: První část Metody get_Data_Standard_Deviation**

V případě validního vstupu a úspěšného proběhnutí metody get_Data je každý ze senzorů podroben metodě třídy Solver standard_Deviation. Standard_Deviation vyřadí veškeré hodnoty podle zadaného množství směrodatných odchylek od střední hodnoty.

```
    for ($x=0;$x<count($allRows);$x++){
        $allRows[$x]= $solver-
>standard_Deviation($allRows[$x],$numberOfStandardDeviation);
    }
    return $allRows;
```

**Kód 9: Druhá část Metody get_Data_Standard_Deviation**

### 4.4.3 Třída Row_Of_Data

Třída Row_Of_Data implementuje uložení řádku dat načteného z databáze. Třída obsahuje metody na manipulaci s vlastněnými atributy.

```
class Row_Of_Data
{
    private $dateTime;
    private $hodnota;
    private $difference=0;
    private $senzor_id=0;
    private $velicina_id=0;

    public function __construct($cas, $hodnota, $senzor_id,$velicina_id)
    {
        $this->dateTime = new DateTime(substr($cas, 0, 19));
        $this->hodnota = $hodnota;
        $this->senzor_id = $senzor_id;
        $this->velicina_id = $velicina_id;
    }
```

**Kód 10: Třída Row_Of_Data**

### 4.4.4 Třída Solver

Solver je singleton třída sloužící k veškerým výpočtům knihovny. Tato třída implementuje počítání směrodatných odchylek a hledání bloků hodnot, které mají neúměrně velký skok svých krajních rozdílů hodnot.

#### 4.4.4.1 Metoda count_Difference

Tato metoda slouží k spočtení rozdílů hodnot jednotlivých objektů Row_Of_Data mezi sousedními hodnotami. V naší situaci už máme dané objekty seřazené podle času.

```
public  function count_Difference($allTheNumbers )
{
    $valueDifference=0;
    $difference=0;
    $allTheNumbersCount=count($allTheNumbers)-2;
    for ($x = 0; $x <=$allTheNumbersCount ; $x++) {
        $valueDifference = abs($allTheNumbers[$x]->getHodnota() -
$allTheNumbers[$x + 1]->getHodnota());
        $allTheNumbers[$x+1]->setDifference(round($valueDifference,3));
        $difference+=$valueDifference;
    }
    $allTheNumbers[$allTheNumbersCount+1]-
>setDifference(round($difference/$allTheNumbersCount,2));
    return $allTheNumbers;
}
```

**Kód 11: Metoda count_Difference**

Cyklus for projde všechny prvky pole, porovná hodnoty a uloží výsledek do atributu $difference příslušného objektu. Při procházení si také sčítá průběžný součet hodnot rozdílů, které po skončení cyklu vydělí počtem objektů v poli. Dále číslo zaokrouhlí a uloží do posledního objektu pole. Metoda vrátí takto upravené pole objektů zpět.

#### 4.4.4.2 Metoda go_Thru_Remove_Invalid_Blocks

Metoda go_Thru_Remove_Invalid_Blocks je vylepšenou verzí z třetího vývojového cyklu. Myšlenkou této metody je najít všechny hodnoty, které jsou ohraničeny neúměrnými skoky hodnot. Prvním rozhodnutím je, kolik hodnot může být v takovém bloku, abychom je zamítli. Z testů na datech z databáze a poznatcích z třetího vývojového cyklu bylo rozhodnuto o hledání bloků čísel o velikosti 3, 2 a 1. Přejděme od teorie na příklad. Nechť jsou diference hodnot [1 1 1 10 1 1 10] a průměrná diference je 0,8. Po vyhodnocení knihovna vyřadí poslední tři hodnoty vzhledem k tomu, že spočtené diference

metodou count_Difference ukazují na následnou hodnotu. První 4 hodnoty jsou spolu v bloku validních čísel.

Čas přejít od teoretického základu k reálné implementaci. V prvním kroku si inicializujeme pomocné proměnné. $numbersInBlock udává počet čísel v bloku. $start je pomocná proměnná, která udává, jestli jsme začali nový blok. Průměrnou diferenci načteme z posledního objektu Row_Of_Data, kam si ji knihovna uložila v metodě countDifference.

```
function go_Thru_Remove_Invalid_Blocks($allTheRows)
{
    $numbersInBlock=1;
    $start=true;
    $allTheNumbersCount=count($allTheRows)-2;
    $averageDifference=$allTheRows[$allTheNumbersCount+1]->getDifference();
    $allValidRows= array();
```

**Kód 12: První část metody go_Thru_Remove_Invalid_Blocks**

Cyklem for projde celé pole. První if rozhodne, jestli difference dvou po sobě hodnot je nižší, jak 10krát průměrná diference spočtená podle souboru dat. Úmyslně je tam zvolené takovéto absurdní číslo, což má za následek, že opravdu jen neúměrně velké skoky jsou zaregistrovány jako konce bloků. V případě, že je skok větší, je velikost bloku resetována a start bloku nastaven na true. V opačném případě, pokud projde podmínkou, přičte knihovna k velikosti bloku čísel jedničku a pokračuje k další if podmínce. Tentokrát rozhoduje o velikosti bloku větší jak 3.

V tento moment přijde na řadu ochrana pro začátek bloku. Pokud if ($numbersInBlock>3) neprojde, tak nevadí, protože pokud další čísla budou splňovat nadřazenou podmínku, tak se k této hodnotě knihovna vrátí. Představme si, že jsme prošli touto podmínkou if o 3 hodnoty dále. Knihovna má proměnnou $start rovnou true a proměnnou $numbersInBlock rovnou 4 v takovém případě projdeme podmínkou. Další podmínka se ptá jestli $start je nastavená na true. V našem případě ano, a proto musíme do validních hodnot uložit i 2 předchozí hodnoty. Pokud by i v další iteraci foru hodnota splnila diferenci, tak už je proměnná $start nastavena na false a validuje se pouze zkoumaná hodnota.

```
    for ($x = 1; $x <=$allTheNumbersCount ; $x++) {
        if ($allTheRows[$x]->getDifference() <  $averageDifference*10) {
            $numbersInBlock++;
            if ($numbersInBlock>3) {
                if ($start==true){
                    array_push($allValidRows, $allTheRows[$x-2],
$allTheRows[$x-1]);
                    unset($allTheRows[$x-2],$allTheRows[$x-1]);
                    $start=false;
                }
                array_push($allValidRows,$allTheRows[$x]);
                unset($allTheRows[$x]);
            }
        }
        else{
            $numbersInBlock=1;
            $start=true;
        }
    }
    return $allValidRows;
}
```

**Kód 13: Druhá část metody go_Thru_Remove_Invalid_Blocks**

#### 4.4.4.3 Metoda standard_Deviation

Standard deviation neboli směrodatná odchylka v češtině. Směrodatná odchylka je míra statistické variability souboru. Tato metoda tedy na tuto znalost navazuje pravidlem tří sigma. Pravidlo tří sigma nebo také známé pod názvem 68-95-99,7 říká, že pokud máme přibližně normální rozdělení souboru, tak všechny relevantní hodnoty jsou vzdálené do 3 směrodatných odchylek od průměru. Tato metoda má za úkol eliminovat odlehlé pozorování (extrémy) na obou stranách grafu.

Do metody vstupují 2 parametry, prvním je $allTheRows. $allTheRows obsahuje pole objektů Row_Of_Data. Druhým parametrem je míra volnosti (počet směrodatných odchylek) nastavená uživatelem v proměnné $allowedDistance (Jsou povoleny hodnoty 0-4). Metoda si připraví pomocné proměnné a sečte všechny hodnoty daného senzoru do proměnné $sum. V dalším kroku si vypočítá aritmetický průměr.

```
function standard_Deviation ($allTheRows, $allowedDistance )
{
    $sum=0;
    $allTheRowsInRange= array();
    $allTheNumbersCount=count($allTheRows)-1;

    for($x = 0; $x <=$allTheNumbersCount ; $x++){
        $sum+=$allTheRows[$x]->getHodnota();
    }
    $arithmeticMean=$sum/($allTheNumbersCount+1);
```

**Kód 14: První část metody standard_Deviation**

```
$sum=0;
for($x = 0; $x <=$allTheNumbersCount ; $x++){
    $sum+=pow(($allTheRows[$x]->getHodnota()-$arithmeticMean),2);
}
$standardDeviation=pow($sum/($allTheNumbersCount+1),0.5);
$allowedDistance=$allowedDistance*$standardDeviation;
```

**Kód 15: Druhá část metody standard_Deviation**

Metoda má připravený průměr, od kterého bude počítat směrodatné odchylky. Směrodatnou odchylku spočte tím, že znovu projde pole a od každé hodnoty odečte vypočtený průměr. Tento rozdíl musí před přičtením do pomocné proměnné $sum mocnin na druhou. Tím vyřeší potenciální záporné hodnoty rozdílů. Dalším krokem je vydělení součtu těchto mocnin množstvím hodnot. Směrodatnou odchylku pak získá druhou odmocninou.

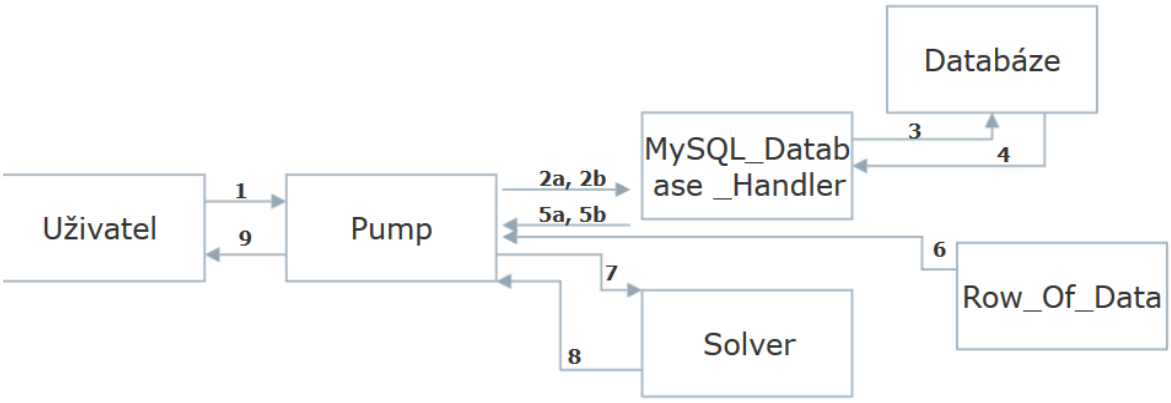
Když metoda získala hodnotu směrodatné odchylky spočte horní a dolní limit. Zde využije uživatelem zadanou hodnotu $allowedDistance a vynásobí jí směrodatnou odchylkou. Přičte k průměru pro horní limit a odečte pro spodní limit. Dalším krokem je znovu projít všechny hodnoty a porovnat je oproti hranicím. Metoda vrací pouze takové objekty Row_Of_Data, jejíž hodnoty jsou v tomto rozmezí.

```
        $bottomLimit=$arithmeticMean-$allowedDistance;
        $topLimit=$arithmeticMean+$allowedDistance;
        for($x = 0; $x <=$allTheNumbersCount ; $x++){
            $value=$allTheRows[$x]->getHodnota();
            if($topLimit>=$value and $value>=$bottomLimit){
                    array_push($allTheRowsInRange,$allTheRows[$x]);
            }
            unset($allTheRows[$x]);
        }
        return $allTheRowsInRange;
    }
```

**Kód 16: Třetí část metody standard_Deviation**

## 4.5 Průchod požadavku knihovnou

Následné schéma popisuje průchod požadavku knihovnou, kdy uživatel požaduje všechny data daného včelstva metodou get_Data. V rámci této kapitoly se nebudeme zabývat možnými errory a nastavováním konfiguračního souboru. Podíváme se na úspěšný požadavek.




**Obrázek 9: Průchod požadavku na získání všech dat knihovnou.**

1. Uživatel požaduje po knihovně data metodou get_Data.

2a. Třída Pump převezme požadavek a zavolá třídu MySQL_Database_Handler a požaduje seznam senzorů pro včelstvo a jakou veličinu měří.

3 MySQL_Database_Handler vyhodnotí požadavek, složí si select a ověří připojení do databáze. Následně pošle select na vyhodnocení do databáze.

4. Databáze vyhodnotí select a MySQL_Database_Handler příslušné záznamy.

5a. MySQL_Database_Handler vrátí výčet senzorů a veličin třídě Pump.

(Následující body (2b-8) jsou prováděny v cyklu tolikrát, kolik existuje pro včelstvo senzorů.)

2b. Pump požaduje po MySQL_Database_Handleru všechny naměřené hodnoty senzorem.

3. MySQL_Database_Handler si složí select dle druhu veličiny měřené senzorem. Následně pošle select na vyhodnocení databázi.

4. Databáze vyhodnotí select a vrátí příslušné záznamy.

5b. MySQL_Database_Handler přijme a vrátí záznamy třídě Pump.

6. Pump si uloží záznamy, jako instance třídy Row_Of_Data, do datové struktury.

7. Pump pošle data ve struktuře na zpracování třídě Solver.

8. Solver vrátí validní data zpátky třídě Pump.
(Konec cyklu)

9. Třída Pump vrátí validní data uživateli.

Pro požadavek uživatele s časovým omezením je průchod knihovny stejný krom změny počtu parametrů ve volání jednotlivých metod.

Pro požadavek uživatele na vrácení dat a odstranění vzdálených pozorování třída Pump nejprve zavolá jednu z metod pro získání dat (get_Data nebo get_Data_Between_Dates). Následně na profiltrovaná data zavolá na každý senzor metodu třídy Solver, která odstraní vzdálená pozorování. Zbylá data vrátí uživateli.

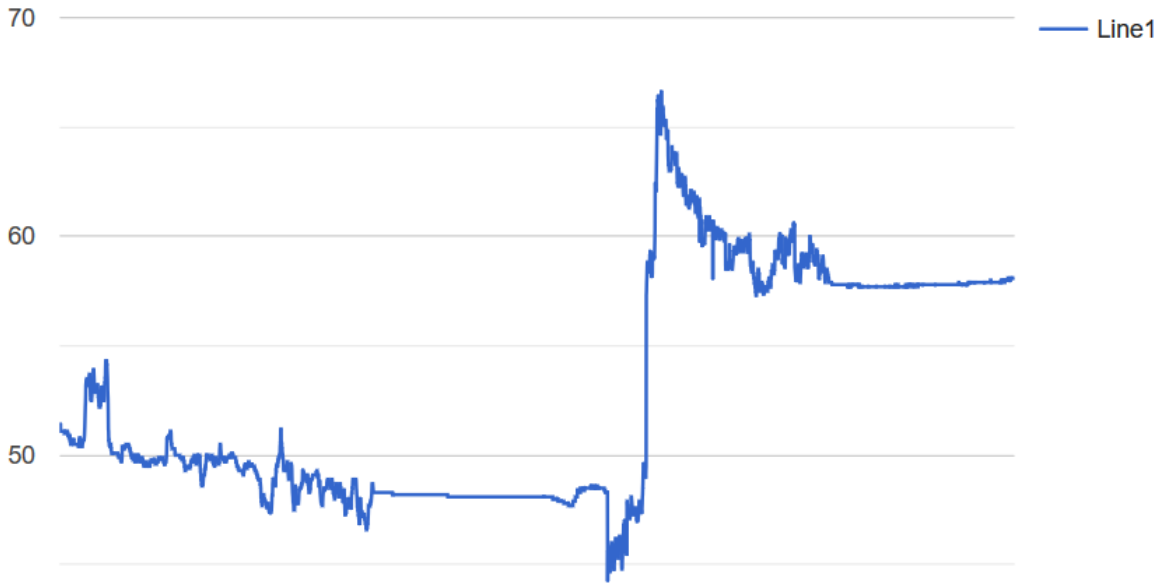
## 4.6 Knihovna v grafech

V této kapitole jsou zobrazena vyřazená data v rámci reálných grafů. Grafy byly vytvořeny na základě reálných dat. Na zobrazení grafů nejsou použity všechny hodnoty daného senzoru. Vždy jde o výřez dat z důvodu lepší viditelnosti.

Tato kapitola neobsahuje konkrétní číselné údaje v grafech, kdy při několikatisícových grafech není možné u každé hodnoty mít popisek s přesnou hodnotou a s časovým razítkem. Nicméně na vybraných grafech jsou vizuálně ukázány změny knihovny provedené na datech.

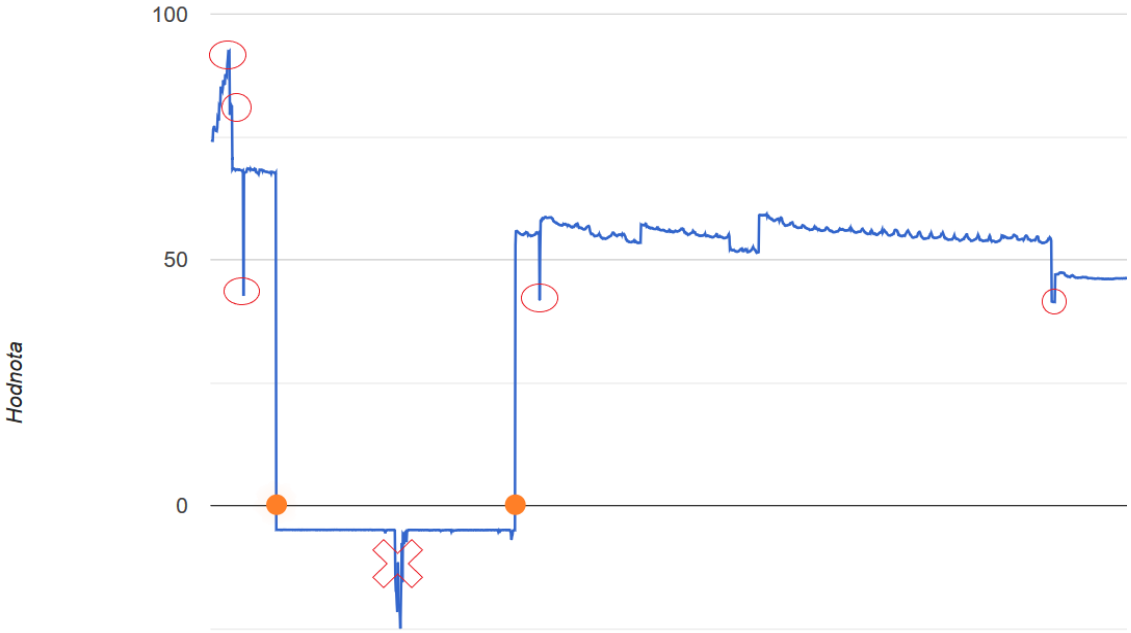
### 4.6.1 Váha (čas / kg)

Nejdříve si ukážeme senzor, který správně funguje a jeho hodnoty se řádně ukládají do databáze. Příkladem je tento graf, kdy na zhruba 30 tisíc záznamů kontrolou nesmyslných hodnot (selecty) se nevyřadila žádná hodnota. Po kontrole skokových bloků čísel bylo vyřazeno 73 hodnot.



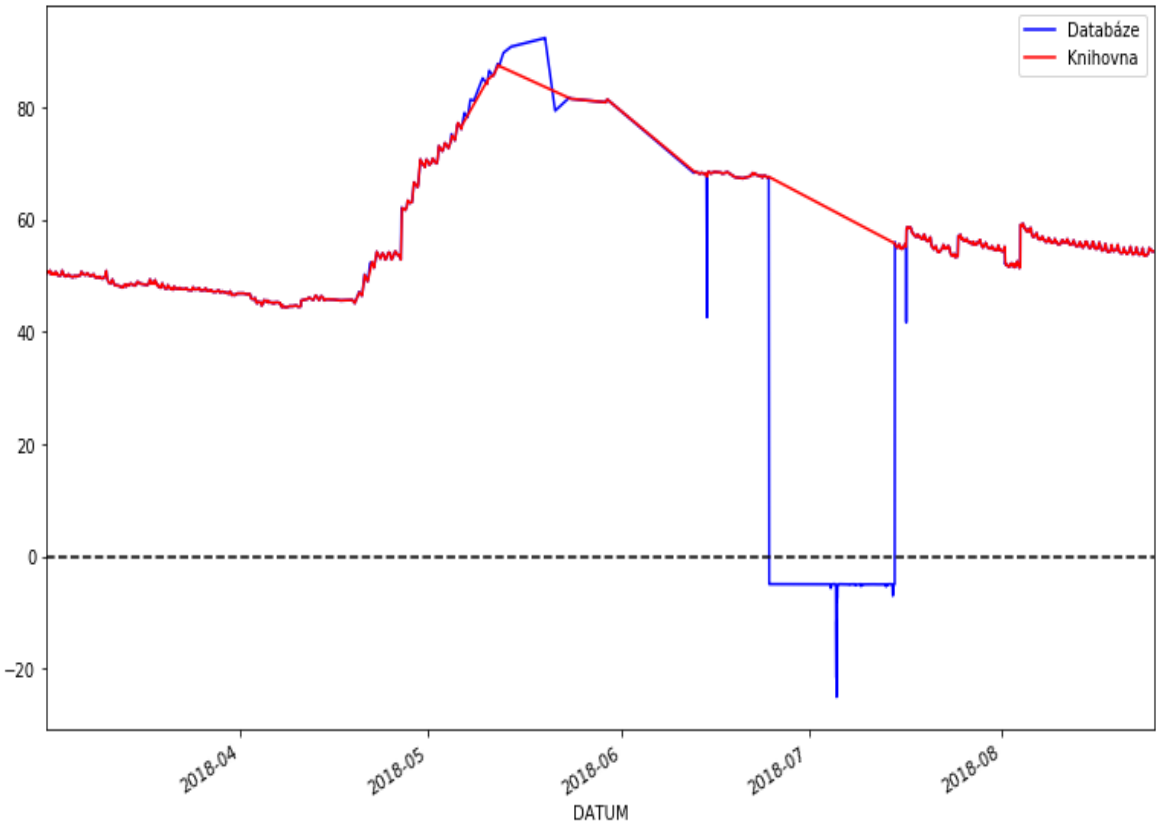

**Obrázek 10: Graf včelstva (čas / kg)**

Přejděme na jiné včelstvo. Toto včelstvo má zhruba 12 tisíc záznamů pro senzor měřící váhu. Na grafu je nicméně zaznamenáno 2000 záznamů hodnot. Když si necháme vyhodnotit tento graf naší knihovnou, tak selecty vyřadíme hodnoty menší 0. Tento interval je označen místem mezi oranžovými kruhy. Místa označená červenými kružnicemi jsou místa, která budou vyřazena metodou skokových bloků čísel, vzhledem k tomu, že jim chybí sousední podobné hodnoty. Když se podíváme na oblast globálního maxima tohoto grafu, bude vyřazeno, vzhledem k chybějícím podobným hodnotám okolo (vlevo i vpravo od globálního maxima chybí několik spojujících hodnot).




**Obrázek 11: Graf včelstva (čas / kg)**

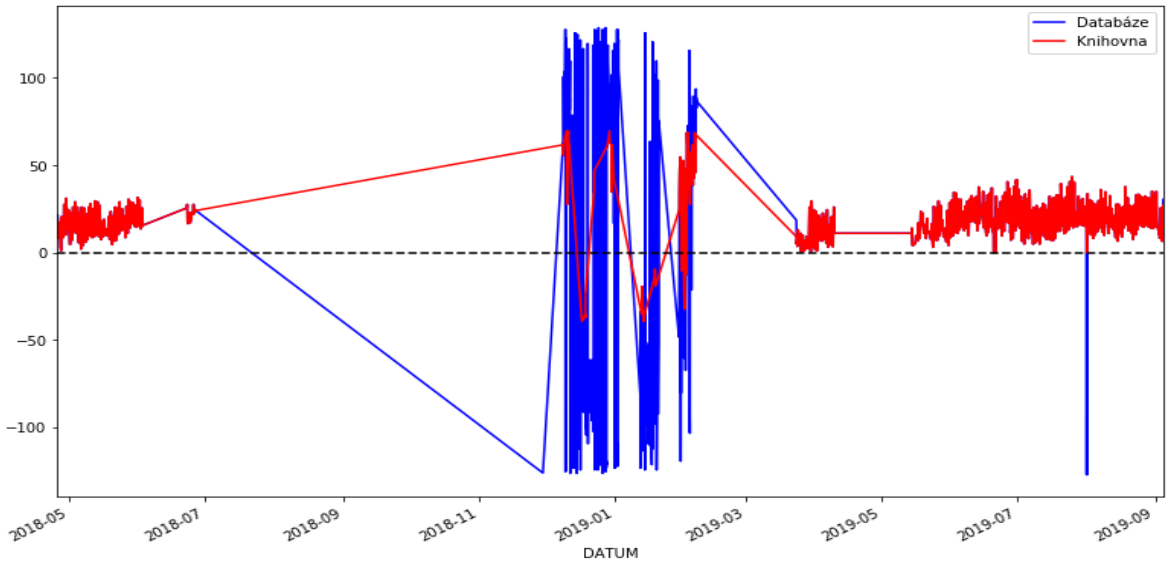
Nyní porovnáme křivku poskládanou z hodnot v databázi s křivkou vygenerovanou z hodnot výstupu knihovny. Aby byly vidět všechny souvislosti odříznutí vrcholu je graf posunutý po časové ose vlevo. Modrá křivka představuje data z databáze a červená křivka představuje data z výstupu knihovny.




**Obrázek 12: Srovnání grafů z dat Databáze vs Knihovny**

### 4.6.2 Teplota (čas / °C)

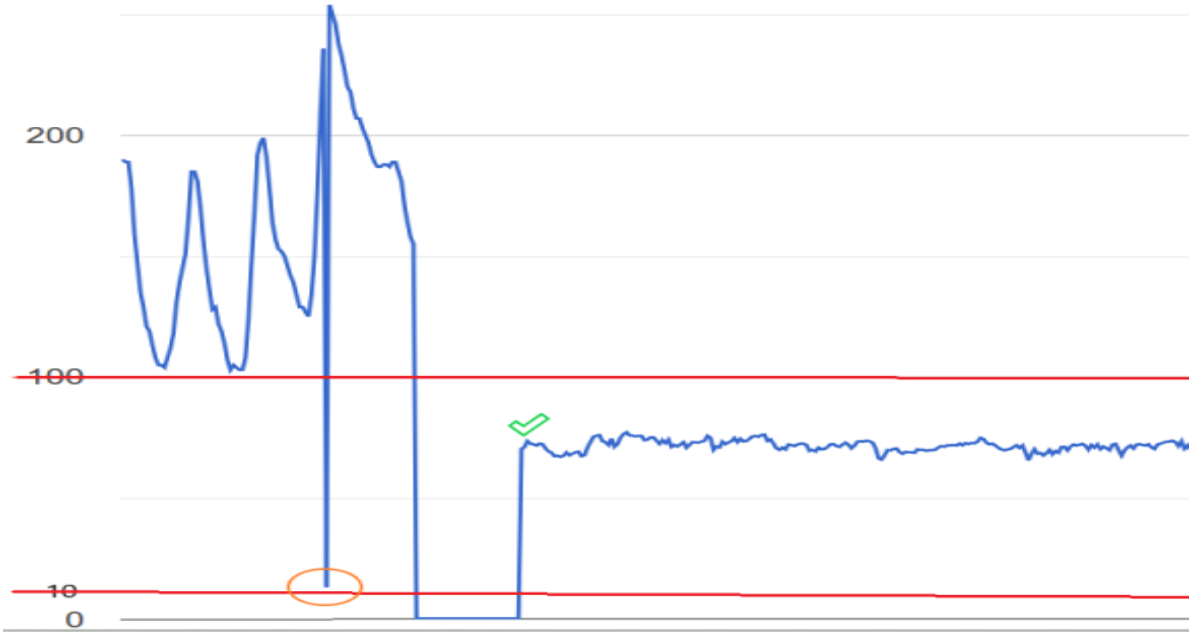
Nyní se podíváme na vybraný graf teploty. Modrá křivka opět představuje data z databáze. Červená křivka zobrazuje hodnoty ponechané knihovnou. Knihovna nejdříve odstraní v selectech všechny hodnoty menší jak -40 °C a větší jak 70°C. V dalším kroku jsou odstraněny zbývající hodnoty, které nemají sousední hodnoty metodou go_Thru_Remove_Invalid_Blocks popsanou v kapitole 4.4.4.2.




**Obrázek 13: Srovnání grafů z dat Databáze vs Knihovny**

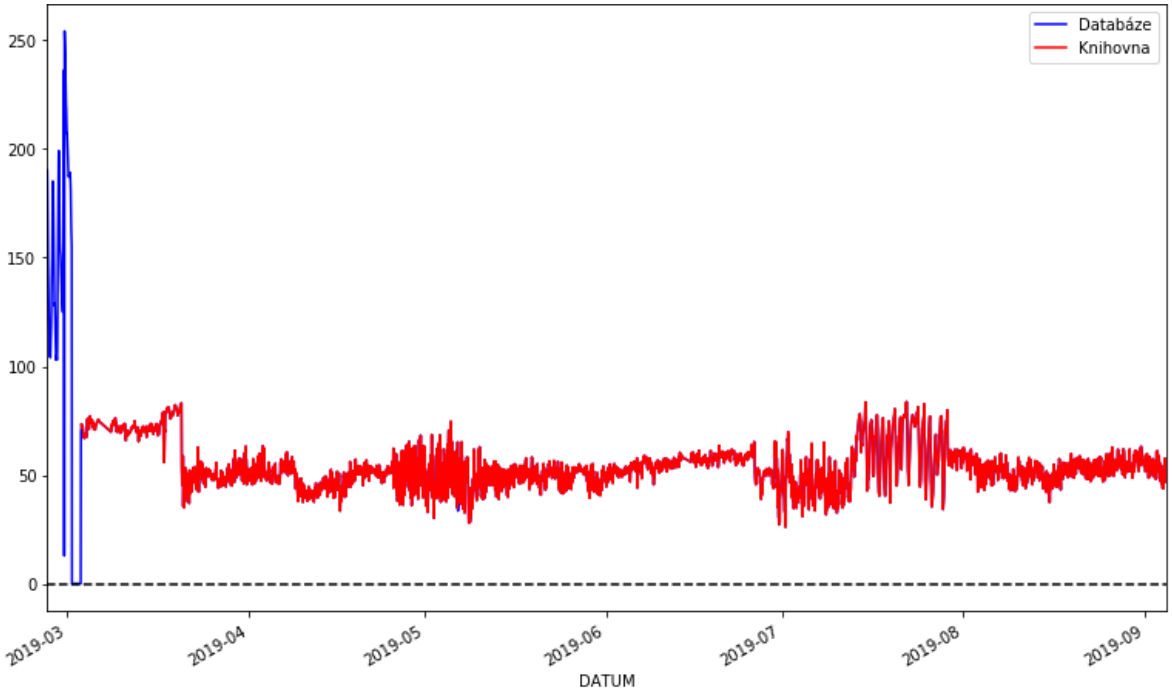
### 4.6.3 Vlhkost vzduchu uvnitř úlu (čas / %)

Následující graf zobrazuje procentuální vlhkost vzduchu v čase pro vybrané včelstvo. Jak si můžeme povšimnout začátek grafu jsou nesmyslné hodnoty. Ty jsou knihovnou opět vyřazeny jako první. Dalším krokem knihovna vyřadí zbytek hodnot do zelené značky. Jedna z takových zbylých hodnot je označena oranžovým kolečkem, kdy tato hodnota 12 projde selectem, ale je vyřazena pro chybějící sousední hodnoty.




**Obrázek 14: Graf včelstva (čas / %)**

Nyní předešlý graf porovnáme s výstupem knihovny. Pro lepší viditelnost dlouhodobého horizontu dat jsou do grafu přidány další navazující záznamy na pravou stranu grafu.




**Obrázek 15: Srovnání grafů z dat Databáze vs Knihovny**

## 4.7 Testování

Testování této knihovny probíhalo v rámci každého vývojové cyklu. Vzhledem k malé velikosti knihovny byly ověřeny jednotlivé metody v každém cyklu pouze ručně nikoliv napsanými unit testy.

Finální verze knihovny byla otestována metoda po metodě. U metod obsahující selecty se testoval počet načtených záznamů knihovny proti počtu vrácených řádků stejného selectu spuštěného nad ostrými daty v databázi. U metod vyřazující nevalidní hodnoty byla provedena kontrola vyřazených dat na vzorku 200 záznamů, kdy jednotlivá data byla vyřazena knihovnou a porovnána s daty vyřazenými člověkem. U metod počítající se směrodatnými odchylkami byla provedena kontrola vyřazených hodnot spočtenými knihovnou proti hodnotám označeným statistickým programem SAS. Knihovna splňuje, že množství získaných dat z databáze se rovná množství dat na výstupu knihovny plus data, která knihovna vyřadila v jednotlivých krocích.

Po otestování všech metod samostatně prošly testováním obslužné metody programu. Tyto metody byly otestovány průběžnými výpisy a porovnáváním vypsaných hodnot proti spočteným číslům početních metod.

Poslední testovanou částí byl výpis chybových hlášek, kdy knihovna byla používána několika dobrovolníky. V první části tohoto testu se dobrovolníci pokoušeli knihovnu rozbít. V druhé části se dobrovolníci pokoušeli o nalezení neošetřených výjimek a errorů knihovny.

Během testování nebyla zjištěna žádná zjevná závada.

# 5 Výsledky a diskuse

## 5.1 Výsledky

Výsledkem této práce je plně funkční knihovna po nastavení konfiguračního souboru Pump_config.ini, který slouží k poskytnutí přihlašovacích údajů k databázi. Tato knihovna načte strojově získaná data z databáze. Tato data byla zadána administrátorem hromadným pohledem(view) do databáze. Knihovna proto používá selecty na data získaná tímto pohledem nikoliv rychlejší selecty přes jednotlivé tabulky v rámci reálné struktury databáze. Knihovna použije zadané parametry uživatelem a vyfiltruje získaná data z databáze pomocí pravidel velikosti a neúměrných skoků hodnot. Knihovna dále nabízí eliminaci vzdálených pozorování pomocí relativní odchylky.

### 5.1.1 Nevyužité metody

V průběhu vývoje existovaly metody, které nebyly z různých důvodů implementovány v konečném řešení. Asi nejzajímavější metoda dokázala napárovat původně nevalidní hodnoty z databáze z důvodu chybějících hodnot ve sloupcích stanoviste_id a vcelstvo_id. Pro nezařazení této metody jsem se rozhodl z dvou důvodů. Prvním důvodem je neefektivita knihovny, kdy by se použitím této metody zvýšil požadavek na paměť a zpomalila by se celá knihovna, zvláště pokud bychom chtěli načíst všechny data včelstva. Druhým důvodem je jednoduché řešení v podobě SQL skriptu spuštěného jednorázově nad celou databází.

### 5.1.2 Možné rozšíření a vylepšení

Knihovna je navržena a implementována tak, aby byla jednoduše rozšiřitelná, proto základní filtrace je oddělena od pokročilých funkcí. Je velmi jednoduché vzít základní filtr dat a dodělat si k němu další funkcionalitu v třídě Solver. Dalším krokem by bylo přidání této metody do vlastní metody v rámci třídy Pump. Konkrétním příkladem je implementovaná filtrace vzdálených pozorování pomocí směrodatné odchylky.

V rámci zrychlení knihovny by bylo dobré vylepšit tyto místa knihovny. Prvním místem je předělání selectů po jednotlivých tabulkách dle reálné struktury databáze, nikoliv přes všechny data v databázi. Především dotaz na senzory by se tímto násobně

zrychlil. Druhým je napsání či použití rychlejší struktury pro přístup k datům. Dosavadní řešení je nicméně více než dostačující. Aktuálně včelstvo s nejvíce záznamy hodnot má 30 tisíc záznamů hodnot. Toto množství je za poslední tři roky. Limit znatelného zpomalení byl v rámci testování určen na 100 tisících záznamů. Toto číslo je samozřejmě také ovlivněno hardwarem serveru. V rámci výstupů metod omezených časem platí stejná hranice, vzhledem k využití stejných algoritmů.

### 5.1.3 **Zhodnocení přínosu knihovny**

Tato knihovna vznikla z důvodů nesprávných ochran vstupů strojově naměřených dat do databáze včelstva on-line. Tato knihovna zamezuje symptomům tohoto problému (příkladem je hodnota -1000 stupňů celsia), kdy vyřazuje nevalidní údaje a vrací údaje validní, takže díky této knihovně jsou data použitelná.

Knihovna nicméně neřeší problém ukládání nevalidních dat do databáze, takže musí tyto nevalidní data při každém požadavku znovu vyřadit, aby bylo možné použít složitější algoritmy knihovny.

# 6 Závěr

Cílem bakalářské práce bylo navrhnout a implementovat knihovnu, která načte data jednotlivého včelstva a odstraní nevalidní data. Nejprve byla prostudována teoretická východiska a proběhlo seznámení se všemi softwarovými komponenty, které jsou v reálném provozu Včelstva on-line používány nebo jsou potřeba k vývoji.

Praktická část obsahuje seznámení s problematikou a nahrání testovacích dat do nainstalované databáze. Na základě analýzy problematiky je navrženo základní řešení. V dalších kapitolách praktické části je popsán spirálový vývoj začínající implementací navrženého řešení.

Druhý a třetí cyklus je popsán pouze nejdůležitějšími změnami struktury a přidanými metodami. Změny jsou popsány komentářem v rámci vývoje, kdy je daná metoda otestována na testovacích datech, a plánování dalších cyklů.

Konečná implementace posledního cyklu je rozebrána po jednotlivých třídách. V rámci jednotlivých tříd jsou popsány jednotlivé metody. U výpočetních metod je nejdříve popsána myšlenka dané metody a až následně implementace. Ostatní metody jsou popsány v rámci souvislostí knihovny a jejich implementace.

Další kapitola zobrazuje grafy utvořené z testovacích dat. Tyto grafy jsou popsány slovním komentářem, kdy je u každého uveden počet hodnot zobrazených na grafu a v jakých místech do těchto hodnot knihovna zasáhne. Následně je graf porovnán s grafem složených z výstupu knihovny.

Výsledkem této práce je plně funkční knihovna pro filtraci dat. V rámci nastavení knihovny je potřeba vyplnit přihlašovací údaje k databázi.

# 7 Seznam použitých zdrojů

1. Abt Bill, Jouni Ahto, Alexander Aulbach a další. PHP: Preface - Manual. *www.php.net.* [Online] Tuesday, March 3, 2020, 6:05:23 AM. [cit. 2020-02-28]. Dostupné z: https://www.php.net/manual/en/preface.php.

2. Abt Bill, Jouni Ahto, Alexander Aulbach a další. PHP: History of PHP - Manual. *www.php.net.* [Online] Wednesday, March 4, 2020, 6:07:26 AM. [cit. 2020-02-28]. Dostupné z: https://www.php.net/manual/en/history.php.php.

3. Holec, Jan. Staré dobré CGI . *Computerworld.cz.* [Online] 01. 11 2000. [cit. 2020-02-28]. Dostupné z: https://computerworld.cz/archiv/stare-dobre-cgi-15768.

4. Hujer, Martin. Jaké novinky přinese PHP 7 - Zdroják. *https://www.zdrojak.cz/.* [Online] 2015-06-07T22:00:44+00:00. [cit. 2020-02-28]. Dostupné z: https://www.zdrojak.cz/clanky/jake-novinky-prinese-php-7/.

5. Bodár, Ján. Novinky jazyka PHP 7 - Root.cz. *https://www.root.cz/.* [Online] Wednesday, March 4, 2020, 11:18:16 AM. [cit. 2020-02-28]. Dostupné z: https://www.root.cz/clanky/novinky-jazyka-php-7/.

6. Bhusman, Chandar. What's New in PHP 7 - DZone Web Dev. *dzone.com.* [Online] Friday, January 17, 2020, 2:00:00 AM. [cit. 2020-02-28]. Dostupné z: https://dzone.com/articles/whats-new-in-php-7-and-php-7-new-features.

7. Monus, Anna. PHP 7: 10 Things You Need to Know . *Hongkiat.* [Online] Thursday, March 5, 2020, 3:25:07 PM. [cit. 2020-02-28]. Dostupné z: https://www.hongkiat.com/blog/php7/.

8. Abt Bill, Jouni Ahto, Alexander Aullbach a další. PHP: What can PHP do? - Manual. *www.php.net.* [Online] Thursday, March 5, 2020, 6:04:42 AM. [cit. 2020-02-28]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php.

9. Running PHP from Windows command line. *SunAnt Interactive.* [Online] Wednesday, March 4, 2020, 11:21:20 AM. [cit. 2020-02-28]. Dostupné z: https://www.sunant.com/running-php-from-windows-command-line/.

10. Fuecks, Harry. PHP on the Command Line - Part 1 Article — SitePoint. *SitePoint.* [Online] Wednesday, March 4, 2020, 11:24:59 AM. [cit. 2020-02-28]. Dostupné z: https://www.sitepoint.com/php-command-line-1/.

11. Team, PHP-GTK. PHP-GTK. [Online] Wednesday, March 4, 2020, 11:26:39 AM. [cit. 2020-02-28]. Dostupné z: http://gtk.php.net/.

12. Koster, Johnathon. How Does PHP Work With The Web Server And Browser? - Stillat. *Stillat.* [Online] Wednesday, March 4, 2020, 11:29:53 AM. [cit. 2020-02-28]. Dostupné z: https://stillat.com/blog/2014/04/02/how-does-php-work-with-the-web-server-and-browser.

13. Rouse, Margaret. What is MySQL? - Definition from WhatIs.com. *@searchoracle.* [Online] Wednesday, March 4, 2020, 11:36:53 AM. [cit. 2020-02-28]. Dostupné z: https://searchoracle.techtarget.com/definition/MySQL.

14. MySQL - Wikipedia. *https://en.wikipedia.org/.* [Online] Monday, March 2, 2020, 2:13:24 PM. [cit. 2020-02-28]. Dostupné z: https://en.wikipedia.org/wiki/MySQL.

15. Zawodny, Jeremy D. a Balling, Derek J. *High Performance MySQL Optimization, Backups, Replication and Load Balancing.* místo neznámé : O'Reilly Media, 2004 First edition. ISBN-10: 0-596-00306-4.

16. Tkachenko, Vadim, Zaitsev, Peter a Schwartz, Baron. 1. MySQL Architecture and History - High Performance MySQL, 3rd Edition [Book]. *O'Reilly Online Learning.* [Online] O'Reilly Media, Inc., Wednesday, March 4, 2020, 8:59:07 PM. [cit. 2020-02-28]. Dostupné z: https://www.oreilly.com/library/view/high-performance-mysql/9781449332471/ch01.html.

17. Sandhu, Mandeep K. MySQL – Architecture and Components. *MANDY SANDHU'S BLOG.* [Online] Wednesday, March 4, 2020, 9:02:08 PM. [cit. 2020-02-28]. Dostupné z: https://mandysandhu.com/2018/02/05/mysql-architecture-and-components/.

18. MySQL :: MySQL Workbench. [Online] Wednesday, March 4, 2020, 9:12:19 PM. [cit. 2020-02-24]. Dostupné z: https://www.mysql.com/products/workbench/.