Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Sběr dat ze senzorů čistoty oleje a jejich vizualizace

Bakalářská práce

Autor: Lukáš Trýzna

Studijní obor: Aplikovaná Informatika

Vedoucí práce doc. Ing. Filip Malý, Ph.D.

Hradec Králové srpen 2017

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 18. 8. 2017 Lukáš Trýzna

Poděkování:

Chtěl bych poděkovat Ing. Filipu Malému za vedení mé bakalářské práce, cenné rady a odborný dohled. Děkuji také Ing. Miloši Vejdělkovi a Ing. Liborovi Hájkovi za umožnění vytvoření této práce pro podnik ARGO-HYTOS s.r.o. Vrchlabí.

Anotace

Tato bakalářská práce má za úkol vytvořit pro firmu ARGO-HYTOS s.r.o. Vrchlabí nový systém pro sledování čistoty hydraulického oleje na zkušebních stavech. Výsledný systém bude sloužit jako náhrada stávajícího řešení, které bylo vytvořeno v programu Control Web 6.1. V práci je popsáno, jakým způsobem byl pro komunikaci využit model klient-server a jaké programovací nástroje byly použity pro tvorbu jednotlivých částí projektu. Klientská část projektu zahrnuje tvorbu několika aplikací, které budou moct běžet na různých platformách. V dnešní době je vhodné mít možnost zobrazovat informace na mobilních zařízeních, a proto je součástí práce vytvoření aplikace pro operační systém Android.

Klíčová slova:
Control Web, čistota oleje, Java, Jetty, UDP protokol, Android, PHP, Nette

Annotation

Title: Collecting data from the sensors of oil cleanliness and their visualization

This bachelor thesis has the task to create a new system for monitorig hydraulic fluid purity in the testing stations in company ARGO-HYTOS s.r.o. Vrchlabí. Final product is going to serve as replacement of the old solution, which was created in Control Web 6.1. This text describes how was used client-server model and which development tools was used for creating each parts of project. The client part of the project includes the creation of several application that can run on different platforms. Nowadays , it's good idea to be able to view information at mobile devices, so it's a part of the project to create an application for the Android operating system.

Keywords:
Control Web, oil cleanliness, Java, Jetty, UDP protocol, Android, PHP, Nette

Obsah

1 Úvod ... 1
1.1 Cíl práce ... 2
2 Současné řešení sledování čistoty olejů ... 3
2.1 Aplikace Control Web ... 3
2.2 Převodník BF-440/480 ... 6
2.2.1 Sériová linka RS-232 ... 7
2.3 Stacionární snímače znečistění OPCom II ... 8
2.3.1 Stanovení úrovně čistoty kapaliny ... 10
3 Návrh nového řešení ... 11
3.1 Architektura klient-server ... 11
3.1.1 Charakteristiky klienta ... 12
3.1.2 Charakteristiky serveru ... 12
3.2 Desktopová aplikace ... 13
3.2.1 Serverová část ... 13
3.2.2 Klientská část desktopové aplikace ... 16
3.3 Mobilní aplikace ... 16
3.3.1 Operační systém Android ... 17
3.3.2 Aplikace v systému Android ... 17
3.4 Webová aplikace ... 17
3.4.1 PHP ... 18
3.4.2 Framework Nette ... 18
3.4.3 Model MVC ... 19
4 Analýza a Implementace ... 21
4.1 Uživatelské cíle ... 21
4.1.1 Zobrazení dat čistoty ze senzorů OPCom II ... 21
4.1.2 Úprava seznamu zobrazovaných OPComů ... 21

4.2 Použitá vývojová prostředí ... 22
4.2.1 NetBeans IDE ... 22
4.2.2 Android Studio ... 23
4.3 Implementace serverové části ... 26
4.3.1 Struktura aplikace ... 26
4.3.2 Uživatelské rozhraní ... 27
4.3.3 Komunikace s převodníky ... 29
4.3.4 Vícevláknové programování ... 30
4.3.5 HTTP server ... 32
4.4 Implementace klientské části ... 33
4.4.1 Desktop ... 34
4.4.2 Webová prezentace/aplikace ... 35
4.4.3 Mobilní zařízení ... 38
5 Shrnutí výsledků ... 43
6 Závěr ... 44
7 Citovaná literatura ... 45
8 Seznam obrázků ... 48
9 Seznam tabulek ... 49
10 Seznam ukázek kódu ... 50
11 Seznam příloh ... 51

1 Úvod

Společnost ARGO HYTOS a.s. se zabývá výrobou filtrační, řídící a regulační techniky v oblasti mobilní a průmyslové hydrauliky. Při výrobním procesu vznikají různé nečistoty, které se dostávají do provozních kapalin zkušebních stavů. Tyto nečistoty mohou mít významný vliv na funkci zkušebního stavu. Funkce zkušebního stavu může ovlivnit kvalitu výrobku. Hydraulické kapaliny vždy obsahují nějaké nečistoty. Kapaliny kontaminované pevnými částicemi můžou způsobit poškození stroje. Zdrojem kontaminace jsou většinou pevné částice nebo voda. Provozovatelům finálních strojů následné prostoje způsobují ztráty spojené s náklady na opětovný rozběh stroje, který vyžaduje opětovné naplnění kapalinou, seřízení, diagnostiku, likvidaci znehodnocených kapalin a další úkony spojené s údržbou. Z těchto negativních vlivů nefiltrované hydraulické kapaliny vyplývá, že je pro bezchybný chod přístroje nutná kvalitní filtrace. Čistotu hydraulických kapalin je nutné neustále kontrolovat za pomoci průtokových snímačů čistoty oleje. Pro kontinuální snímání počtu pevných částic firma ARGO-HYTOS vyrábí čidlo OPCOM II, které je umístěno pevně na přístroji, kde má být sledován stav kapaliny. Firma také vyrábí přenosné snímače čistoty kapalin. Společnost ARGO-HYTOS Vrchlabí využívá vlastních snímačů OPCom II pro sledování kvality hydraulických kapalin na svých zkušebních stavech. Získaná data slouží pro analýzu stavu oleje na zkušebním stavu a pro vyhodnocení, zda je ještě olej způsobilý pro provoz. (1) (2)

1.1 Cíl práce

Tato práce má za cíl zajistit pro společnost ARGO-HYTOS takové softwarové řešení, které spolehlivě nahradí dosud používané. Nyní používaný software nesplňuje požadavky na to, aby se daly za chodu programu přidávat další senzory čistoty oleje, ze kterých jsou získávány data. Pro každou změnu v programu je nutný zásah do zdrojového kódu. Součástí řešení je také návrh a implementace nového systému, který bude schopen komunikovat také s chytrými telefony. Veškerá získaná data se budou archivovat do databáze, tak aby se mohla provádět jejich analýza a vyhodnocení. Stěžejním bodem je analýza požadavků, ze které se bude vycházet pro návrh systému. Analýza bude definovat, jakým způsobem budou jednotlivé části systému mezi sebou komunikovat. Takové požadavky by měla splňovat architektura klient-server. Mobilní aplikace bude vyvinuta pro operační systém Android. Projekt má také za cíl vytvořit zcela novou webovou aplikaci, která bude vytvořená za pomoci frameworku v programovacím jazyce PHP. Aplikace pro desktopové systémy a aplikace serveru bude naprogramována v jazyce Java.

2 Současné řešení sledování čistoty olejů

Řešení, které se tato práce snaží nahradit, je v podniku ARGO-HYTOS Vrchlabí provozováno za pomoci aplikace vyvinuté v programovacím prostředí Control Web 6.1. Prostředí Control Web je určeno především pro průmyslovou automatizaci. Aplikace je spuštěna na PC, které je zapojeno do podnikové sítě, a může tak komunikovat s převodníky, které převádějí sériovou komunikaci na komunikaci za použití protokolu TCP/IP nebo UDP. Jako čidla čistoty oleje jsou použity stacionární snímače znečištění kapalin OPCom II. Čidla jsou zapojena do převodníku BF-440, který umožňuje převádět sériovou komunikaci na ethernetovou a naopak. Aplikace získává data z čidel čistoty oleje na zkušebních stavech každých 15 minut a poté je archivuje do databáze MySQL. Aktuální získaná data aplikace zobrazuje. Na PC je také spuštěn web server Apache, na kterém je spuštěn PHP skript, který jednoduchým způsobe čte data uložená v MySQL databázi a zobrazuje je ve webovém prohlížeči.

2.1 Aplikace Control Web

Aplikace vyvíjené v tomto prostředí jsou většinou určené pro průmyslovou automatizaci. Za tímto softwarem je česká firma Moravské přístroje a.s., která působí na trhu elektroniky a programového vybavení od roku 1991. Firma poskytuje vývojovou a runtime verzi aplikace Control Web. Vyvíjenou aplikaci lze spouštět a testovat ve vývojovém prostředí, ale pro chod aplikace stačí pouze runtime verze, která spouští vygenerovaný soubor vývojovou verzí. Ze zkompilovaného souboru nelze již nijak dostat zdrojové kódy aplikace. Tímto se zabraňuje kopírování kódů. Při každé úpravě aplikace je nutné vygenerovat nový spustitelný soubor.

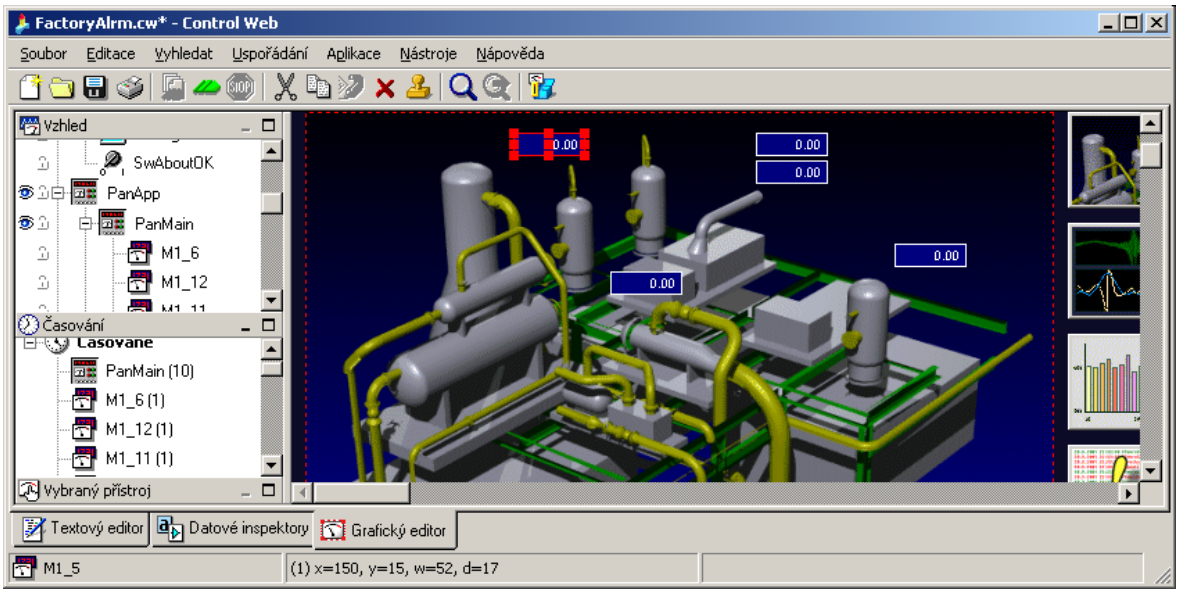
Co je Control Web obecně:

- programový systém pro průmysl, laboratoře, školy,
- vizualizace a řízení technologických procesů v reálném čase,
- most mezi technologií a informačním systémem podniku,
- přímé řízení strojů a technologií.

Control Web dokáže využívat velké množství otevřených protokolů jako je například ASCII komunikace po sériové lince, OPC Data Access, DDE/NetDDE, FastDDE, GSM, SMS, http přístup k web serverům, MODBUS TCP/IP a další. Důležitou vlastností je také

schopnost spolupracovat s databázemi typu Microsoft Access nebo přes příslušné ODBC spojení s databázemi MySQL nebo jinými SQL databázemi. (3)

Pro aplikace komunikující v síti je potřeba mít nainstalovaný driver pro program Control Web, který umožní vytvářet sokety pro UDP nebo TCP komunikaci. Protože je nutná souběžná komunikace s více zařízeními najednou, tak je nutné, aby v programu byl přidán driver pro každý snímač čistoty hydraulických kapalin.

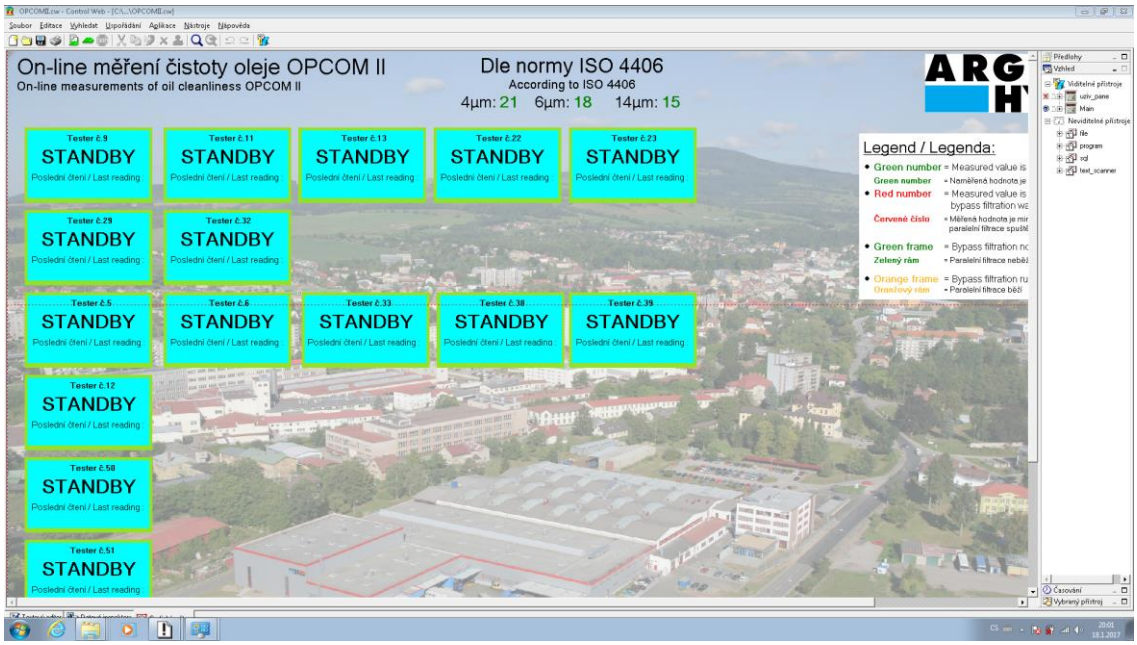


Obrázek 1: Control Web 6 (29)

V tomto systému lze vytvářet aplikace reálného času nebo aplikace řízené změnou dat. Aplikace v programu Control web se skládají z datových elementů a přístrojů. Přístroje jsou rozdělené do kategorií podle toho, jakou úlohu mají provádět. Aby přístroj mohl měnit svůj obsah, musí být časován, nebo je volán jinými přístroji a tímto je dosaženo jeho obnovení. Časování přístrojů znamená, jak rychle bude přístroj obnovovat to, co zobrazuje nebo jak rychle bude provádět svůj kód. Například pokud by časování přístroje bylo nastaveno na hodnotu 1, tak by každou sekundu prováděl svůj kód nebo obnovoval svůj obsah. Důležitým přístrojem je Panel, který umožní aplikaci umístit do okna tak, aby se s aplikací dalo na ploše Windows pohybovat. Aplikace nemusí mít žádný Panel, přístroje pak běží na ploše a lze je překrývat jinými programy. Prostředí také dělí přístroje na viditelné a neviditelné. Přístroje, které vykonávají pouze nějaký algoritmus, nemusí být vidět. Některé přístroje ani do viditelné části umístit nelze.

Pro sledování čistoty olejů byl vytvořen program právě v této aplikaci. Pro každé čidlo, které je jednoznačně určené IP adresou a číslem portu, je do aplikace přidán driver,

který komunikuje právě s jedním senzorem čistoty oleje. Každých 15 minut se odešle na všechny čidla dotaz na aktuální hodnoty. Odpověď je rozebrána a každý údaj je uložený do příslušného datového elementu, v tomto případě do indexovaného pole. V aplikaci jsou pro všechny senzory umístěny přístroje, které zobrazují hlavní tři ukazatele míry znečištění kapaliny. Pokud nějaký ukazatel překračuje stanovenou míru normy ISO 4406, tak se číslo zbarví do červena, jinak je černé. Zároveň se změní barva orámování, které je okolo přístrojů zobrazující, zda je v chodu paralelní filtrace.



Obrázek 2:Vývojové prostředí se současnou aplikací pro sledování čistoty oleje (Zdroj: vlastní)

Toto vývojové prostředí bylo ve firmě zvoleno proto, že tento systém se osvědčil na velkém množství zkušebních stavů jako nástroj automatizace výrobního procesu a je poměrně snadné tyto aplikace upravit. Aplikace vyvinuté v tomto softwaru jsou schopné nahradit programovatelné automaty PLC (Programmable Logic Controller). Nicméně snímače čistoty oleje se začínají používat čím dál tím na větším počtu zkušebních stavů a proto je nutné aplikaci neustále upravovat tak, aby zobrazovala data ze všech čidel. Takto jednoduchá úprava, jako je přidání dalšího snímače, by měla být implementována v samotném programu, což aktuální prostředí neumožnuje.

2.2 Převodník BF-440/480

Pro obsluhu a sledování snímače OPCom II je možné používat sériový protokol RS-232. Aby byla možná komunikace po ethernetové síti, je nutné sériovou komunikaci převádět na komunikaci ethernetovou (a obráceně). Na to slouží převodník BF-440 (se čtyřmi porty) nebo BF-480 (s osmi porty), který je vyráběn firmou CHIYU TECHNOLOGY z Tchaj-wanu. Převodník dokáže obsluhovat až čtyři zařízení nejednou (v případě modelu BF-480 až osm zařízení). Takové zařízení může pomoct připravit senzory, ovladače nebo jiná průmyslová zařízení komunikující po sériovém rozhraní na komunikaci přes síť. Takto zapojené zařízení se může rychle stát součástí internetu věcí.

Obrázek 3: Převodník BF-440 (30)

Převodník poskytuje tři typy sériového připojení:

- RS-232,
- RS-422,
- RS-485.

U každého portu lze nastavit: rychlost, počet datových bitů, kontrolu parity, počet stop bitů, flow control, dobu po jakou se bude přenášet, délku přenášeného paketu v bytech a oddělovače. V případě připojení RS-485 zařízení lze také nastavit zpoždění přenosu.

Zařízení může pracovat ve třech módech, podle toho, jak se chová a co vyžaduje připojené zařízení komunikující po sériové lince. Prvním módem je TCP Server. Tento mód slouží pro zařízení, která se chovají pasivně, to znamená, že pouze naslouchají. TCP protokol zajistí to, aby komunikovaná data byla přenesena v pořádku. Další mód je TCP Client, kdy zařízení odesílá data v reálném čase. Tyto data pak zachycuje aplikace na druhé straně spojení. Třetím módem je UDP Client. Pokud není nutné, aby byla data spolehlivě komunikována a nebylo potvrzováno spojení, tak lze použít tento mód.

Pro nastavování slouží webové konfigurační rozhraní, kde lze všechny parametry upravit a sledovat stav převodníku. Použití webového rozhraní pro nastavení činí práci s převodníkem přívětivější pro uživatele. V případě, že chce uživatel komunikovat se svým zařízením skrz internet a nemá veřejnou IP adresu, tak může používat službu Dynamic Domain Name System (DDNS), která umožní toto spojení. Tato služba je poskytována webem dyn.com za roční poplatek. Přístup k administračnímu rozhraní převodníku lze ochránit uživatelským jménem a heslem. (4)

2.2.1 Sériová linka RS-232

Pro komunikaci mezi snímačem čistoty oleje a převodníkem je použita komunikace po sériové lince typu RS-232. Jedná se o komunikaci mezi dvěma zařízeními, kdy jsou jednotlivé informace zasílány za sebou. Tento standard se hojně používá v průmyslových zařízeních. Toto rozhraní se na PC již delší dobu nepoužívá (bylo nahrazeno rozhraním USB). Nejčastěji se jím k PC připojovala myš. K propojení se používá konektor D-SUB typu DE-9, který má 9 pinů. Rychlost komunikace je závislá připojených na zařízeních. Rychlost může začínat od 110 baudů až po 115200 baudů. Při zvyšování rychlosti se zvyšuje riziko přeslechnutí nebo ztráta dát na delší vzdálenosti. (5)

Komunikace na tomto standardu je simplexní, polo duplexní, anebo plně duplexní. Přenos začíná start bitem, pak následují datové bity, kterých může být 7, 8 nebo i 9. Nejčastěji se používá 8 bitů pro data. Datové bity jsou seřazeny od Least Significant Bit (LSB) po Most Significant bit (MSB). Na konci datové sekce je bit, který signalizuje paritu. Tento bit může, anebo nemusí být komunikován. Za paritou je umístěn stop bit, který signalizuje konec přenosu bitů reprezentující data. Logický stav (0 nebo 1) je reprezentován pomocí napětí na vodičích, které může nabývat hodnot od ±3 V, ±10V , ±12 V nebo ±15 V. V největším množstvím případů se můžeme setkat s tím, že -12 V reprezentuje logickou 0 a +12 V reprezentuje logickou 1. (6)

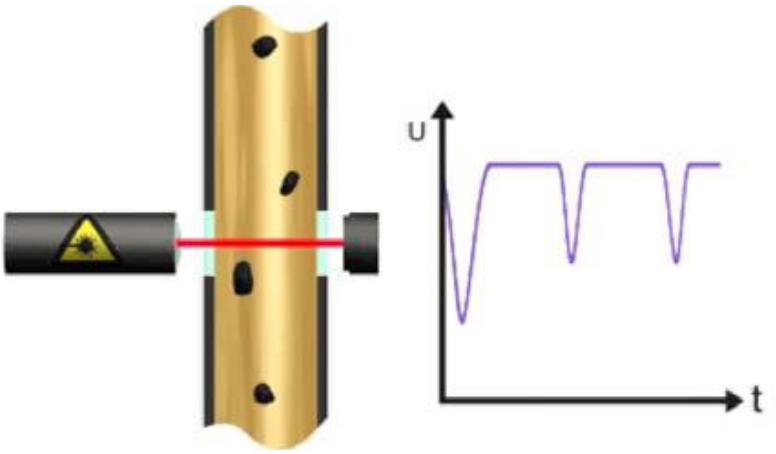
2.3 Stacionární snímače znečistění OPCom II

Firma ARGO-HYTOS vyrábí také techniku pro sledování čistoty hydraulické kapaliny. Toto řešení používá i ve svých provozech, tak aby se pracovalo s kvalitním hydraulickým olejem a nedocházelo velkému opotřebení zařízení na zkušebních a montážních stavech. Jedním z faktorů, které ovlivňují kvalitu, je počet pevných částic obsažených v 1 ml kapaliny.

Obrázek 4:Stacionární snímač znečištění OPCom II (7)

Pro průběžné sledování počtu pevných částic se používá stacionární snímač znečistění OPCom II. Snímač pracuje na principu zaclonění světelného paprsku. Měřící jednotka se skládá se tří částí:

- průtočná buňka,
- laser,
- fotodioda.



Obrázek 5: Způsob měření počtu pevných částic za pomoci laseru (8)

Laser prosvěcuje průtočnou buňku a fotodioda světelný paprsek zachycuje. Pokud dojde k poklesu intenzity světla, znamená to, že laser narazil na pevnou částici. V závislosti na míře poklesu světla se určuje velikost částice, která zastínila paprsek. Pro správnou funkci musí být zařízení zapojeno v hydraulickém obvodu tak, aby nedocházelo k výkyvům v tlaku nebo průtoku. Zařízení také dokáže měřit teplotu kapaliny, neměří ji přímo v průtočné buňce ale na elektronice měřícího zařízení. Měřící jednotka se připojuje na paralelní hydraulický obvod zařízení, kde je sledování potřeba. Kalibrace jednotky je provedena dle standardu ISO 11171:99. Sledují se tři hlavní velikosti částic, které se vyskytují v hydraulické kapalině. Při monitorování mohou být nastaveny až čtyři hranice míry znečistění, pokud by byla jedna ze čtyř hranic překročena, zařízení zobrazí na displeji varování a sepne digitální výstup. Tímto výstupem může být řízen obvod pro paralelní filtraci.

Pro komunikaci slouží sériová linka nebo propojení na sběrnici typu CAN bus (Controller Area Network). Tato sběrnice se používá u systému v automobilovém průmyslu pro komunikaci různých jednotek a senzorů. Při sériové komunikaci zařízení reaguje na příkazy, které jsou mu zasílány. Například příkaz „RVal“ slouží pro přečtení aktuálních hodnot. Zařízení vrátí jednotlivá data oddělená středníky. Oddělování středníky je výhodné pro programové zpracování získaných dat. Snímač má také paměť, do které ukládá naměřené hodnoty. Za každým příkazem musí být také odeslán symbol carriage return. V šestnáctkové soustavě tomuto symbolu odpovídá hodnota 0D. Pokud by tento speciální znak nebyl odeslán, tak by jednotka stále čekala na ukončení sekvence touto kombinací. (8)

2.3.1 Stanovení úrovně čistoty kapaliny

Snímač znečistění OPCom II může stanovovat úroveň znečistění podle dvou norem. První norma, zde použitá, je norma ISO 4406, která je používána v České republice. Tato norma stanovuje kódové označení pro jednotlivé počty částic. Velikosti pevných částic jsou rozděleny na tři kategorie. Jsou to, částice ≥4µm, částice ≥6µm a částice ≥14µm. Čísla kódu se od sebe oddělují lomítky, např. 21/18/14. Ke každému kódovému číslu je přiřazený rozsah počtu částic, kterému odpovídá. Druhou normou, kterou zařízení podporuje je norma SAE AS4059E. Tato norma se v závodě ARGO-HYTOS nepoužívá. Různé druhy zařízení, ve kterých je hydraulický olej, potřebují jinak stanovenou normu čistoty kapaliny. (2)

Kódové č.	Počty částic v 1 ml	
0	0	0,01
1	0,01	0,02
2	0,02	0,04
3	0,04	0,08
4	0,08	0,16
5	0,16	0,32
6	0,32	0,64
7	0,64	1,3
8	1,3	2,5
9	2,5	5
10	5	10
11	10	20
12	20	40
13	40	80
14	80	160
15	160	320
16	320	640
17	640	1
18	1	3
19	3	5
20	5	10
21	10	20
22	20	40
23	40	80
24	80	160
25	160	320
26	320	640
27	640	1 300 000
28	1 300 000	2 500 000
x28	2 500 000	

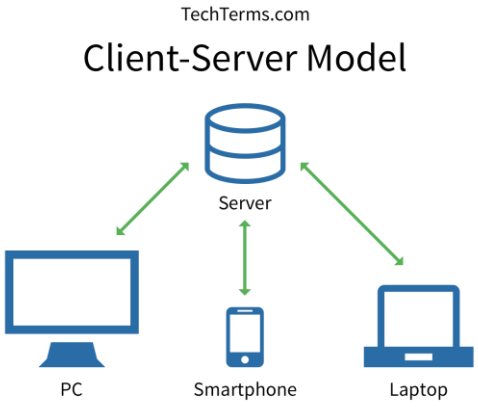
Tabulka 1:Příslušné počty částic ke kódům normy ISO 4406 (9)

3 Návrh nového řešení

Nové řešení pro systém sledování počtu pevných částic zahrnuje některé komponenty již použité ve stávajícím řešení. Hlavní komponenty, které budou zachovány, jsou: stacionární snímač čistoty oleje OPCom II a převodník BF-440. Zcela nově bude vytvořena aplikace sledující stavy jednotlivých snímačů. Zde bude využita architektura klient-server, která nám zajistí to, že aplikace běžící u klienta nebudou přímo komunikovat se snímači počtu částic v kapalině, ale budou komunikovat se serverovou aplikací, která tyto data bude získávat a poskytovat v takovém formátu aby data mohly být jednoduše zobrazeny na jakémkoliv klientovi. V tomto systému klienti budou představovat mobilní aplikace, aplikace na počítače a webová aplikace. Webová aplikace zajistí dostupnost dat nezávisle na tom, na jaké platformě je spuštěna. Všechny části budou propojeny za pomocí ethernetové sítě v podniku ARGO-HYTOS Vrchlabí.

3.1 Architektura klient-server

Nejprve je nutné vysvětlit, v čem spočívá síťová architektura klient-server. Jedná se o takovou architekturu, kde jeden program žádá o služby jiný program, kterých potom využívá. Program poskytující služby se nazývá server a program či zařízení, které o tyto služby žádá, se nazývá klient. Příkladem serverů můžou být webové servery, které poskytují webové stránky nebo emailové servery poskytující služby pro přijímání či zasílání emailů. Architekturu klient-server využívají také online hry, kde se hráči připojují na herní server a můžou tak spolu svádět různé souboje. Server své služby často poskytuje více než jednomu klientovi.



Obrázek 6: Architektura klient-server (31)

3.1.1 Charakteristiky klienta

Klient může být aplikace běžící na jakékoliv platformě a zprostředkovává (využívá) služby poskytované serverem. Klient zasílá požadavek na server, a poté obdrží požadovanou odpověď. Klienti se dělí na dvě kategorie podle toho, jaké množství lokálních prostředků používají a v jaké míře jsou závislé na serveru.

3.1.1.1 Tlustý klient

Obvykle se jedná o počítač, na kterém běží klientská aplikace, která využívá služeb serveru (databáze, služby apod.). Takto běžící aplikace využívá výpočetní výkon lokálního počítače a data ze serveru může ukládat na svá úložiště.

3.1.1.2 Tenký klient

Obecně se takto dá nazvat počítač nebo program silně závisející na serveru. Server mu poskytuje výpočetní výkon a informace, se kterými klient pracuje. Takové řešení se uplatňuje například v průmyslové výrobě, kde je zapotřebí pouze zobrazovat pracovní postup nebo jiné informace. Jeden server může svoje prostředky poskytovat velkému množství klientů. (10)

Obrázek 7: Tenký klient od společnosti HP (11)

3.1.2 Charakteristiky serveru

V počítačové technice pojmem server rozumíme program nebo počítač, který poskytuje data nebo služby připojeným klientům. Server přijímá požadavky od klientů a poté na tyto požadavky odpovídá. Počítače sloužící na tento účel jsou konstruovány

tak, aby dokázaly udržovat nepřetržitý provoz. Příklady serverů: Webový server, file server, poštovní server, proxy server, tiskový server a jiné další.

3.2 Desktopová aplikace

Aplikace pro sledování čistoty oleje na zkušebních stavech bude vytvořena za pomoci objektově orientovaného programovacího jazyku Java. Na této platformě budou vytvořeny dvě části projektu. První částí bude server zpracovávající požadavky na data od OPComů a druhá část je klientská aplikace, která data zobrazuje, popřípadě tyto data ukládá na databázový server. Jako databázový server zde slouží MariaDB, který je odnoží MySQL.

Programovací jazyk JAVA je podle TIOBE indexu na serveru tiobe.com (12) nejpoužívanější jazyk v komunitě programátorů.

3.2.1 Serverová část

V tomto systému bude server přijímat a odpovídat na požadavky od klientů (aplikace na PC, webová aplikace nebo mobilní aplikace), kteří si budou žádat o data z OPComů. Každá jednotlivá aplikace může přijímat data od jiného počtu zařízení. Komunikace mezi klientem a serverem bude probíhat za pomoci UDP protokolu a HTTP protokolu. Server přijme sekvenci dat, která obsahuje IP adresu, port a id OPComu. V sekvenci může být uvedeno více údajů pro více OPComů, každý oddělný středníkem. Server tuto sekvenci rozebere a vytvoří si seznam objektů obsahující údaje pro komunikaci s OPComem. Na serveru bude také možné spustit HTTP server, který bude na požadavky vracet odpovědi ve formě CSV souboru.

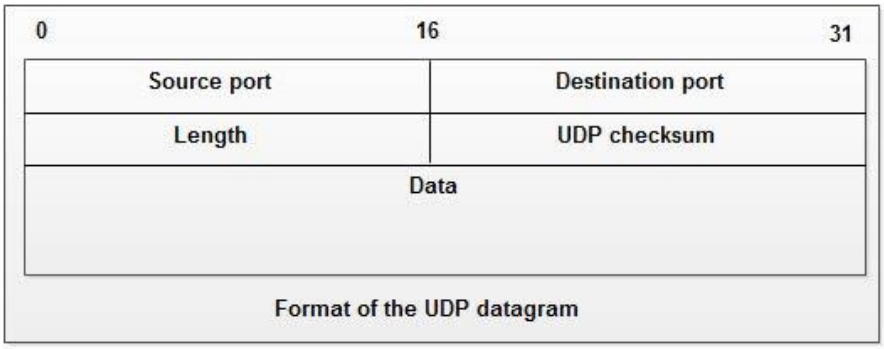
3.2.1.1 UDP komunikace

Pro komunikaci mezi serverem a převodníky sériového signálu, ke kterým jsou připojeny senzory čistoty hydraulického oleje, byla zvolena komunikace přes UDP (User Datagram Protocol) protokol. Byl zvolen proto, že není nutné navazovat žádné stále spojení a je nižší zatížení sítě. Na převodnících je neustále otevřen port pro naslouchání. Převodník také disponuje TCP/IP modem, ale tento projekt implementuje komunikaci mezi serverem a převodníkem pouze přes UDP protokol.

UDP je protokol pro komunikaci na ethernetové síti a internetu. Pro komunikaci je potřeba IP adresa a UDP číslo portu cíle. Tato kombinace se nazývá soket. Například soketem může být kombinace 127.0.0.1:80. IP adresa jednoznačně určí cílový počítač

nebo zařízení, se kterým se bude komunikovat. Pokud není nutná odpověď od cíle, tak zdrojový port nemusí být uveden, ale když bude potřeba, je nutné i toto číslo uvést. UDP datagram v sobě neobsahuje žádné potvrzení ani sekvenci, proto se tento protokol označuje jako nespolehlivý. Nijak není zaručeno to, že jsou datagramy (pakety) spolehlivě doručeny, to znamená, zda se cestou k cíli neztratili nebo nedorazily ve špatném pořadí. Špatné pořadí paketů může být způsobeno rozdílnou cestou nebo zahlcením některého ze síťových prvků. Aplikace využívající UDP protokol můžou disponovat opravným mechanismem. Pokud by data v požadovaný čas nepřišla, aplikace by si o data mohla žádat znova. Často se používá to, že pokud selže nespolehlivý UDP protokol, tak je použit TCP, který je již proti ztrátám a poškozením paketů ochráněn. (13)

UDP datagram se skládá z čísla zdrojového portu (16 bitů), čísla cílového portu (16 bitů), délkou paketu včetně dat (16 bitů), kontrolním součtem (16 bitů) a daty, pokud jsou nějaká zasílána. Na rozdíl od TCP/IP paketu zde chybí části pro potvrzovací a sekvenční byty.



Obrázek 8: Formát UDP datagramu (14)

Port je šestnáctibitové číslo sloužící pro jednoznačnou identifikaci komunikujících zařízení nebo programů. Čísla portů můžou být od 0 až do 65535. Tyto porty se rozlišují na tři kategorie: 1 – 1023 dobře známé (well known), 1024 – 49151 registrované a 49152 – 655365 dynamické porty pro nejrůznější použití. Každý port by měl být na počítači použit pouze jednou pro konkrétní aplikaci nebo službu. Příklady dobře známých portů jsou: 25 - SMTP, 80 - HTTP, 110 - POP3, 443 - HTTPS, 23 - Telnet. Čísla portů spravuje organizace IANA (Internet Assigned Numbers Authority), která se také stará o jiné záležitosti v internetu, jako je například přidělování kořenových domén nebo správa IP adres. (13)

Komunikace přes UDP má své výhody i nevýhody:

Výhody UDP:

- není zde potvrzování -> menší zatížení sítě,
- lze rozlišit jednotlivé datagramy při přijímání.

Nevýhody UDP:

- nezabezpečený přenos,
- možnost přehození pořadí odeslaných datagramů,
- nespojový protokol,
- nejistota doručení dat.

3.2.1.2 HTTP komunikace

HTTP (Hyper Text Transfer Protocol) patří do rodiny protokolů zajištující komunikaci mezi webovým serverem a klientem. Nejčastějším klientem (programem) využívající tento protokol, je internetový prohlížeč, který prostřednictvím tohoto protokolu zasílá požadavky na server a ten mu odpovídá požadovaným dokumentem, který webový prohlížeč správně interpretuje. Tento protokol nemusí být využíván pouze prohlížečem. Při komunikaci s HTTP serverem je zapotřebí znát jeho adresu nebo jeho jméno. Adresa serveru je reprezentována za pomoci URL (Uniform Resource Locator), která je pomocí DNS (Domain Name System) přeložena na IP adresu. Základní tvar URL adresy je například: „http://www.seznam.cz“, kde se na začátku stanovuje, jaký protokol používáme a poté určíme adresu serveru, se kterým hodláme komunikovat. Adresa serveru nemusí být zapsána jménem, ale může být nahrazen IP adresou serveru.

Obrázek 9: URL adresa ve webovém prohlížeči (Zdroj: vlastní)

Metody, které HTTP protokol používá, jsou především GET a POST. Protokol je založen na systém požadavek a odpověď. Metodou GET si klient žádá o obsah z webového serveru. Metoda POST je určena na pro odesílání dat ke zpracování na server. Za

pomoci metody GET lze na webový server odeslat uživatelské vstupy, které se připojí za adresu URL v podobě: „webovaStranka.cz/hledat?dotaz=auto“. V tomto případě se do proměnné „dotaz“ uložil řetězec „auto“ a byl by odeslán na webový server ke zpracování. Přenos uživatelských vstupů přes tuto metodu není nijak zabezpečen. Veškeré hodnoty proměnných lze vyčíst z URL adresy odeslané na server. Taková metoda by se nikdy neměla používat pro přenos citlivých dat, jako jsou hesla nebo jiné choulostivé osobní údaje. Pokud bude stránka obnovena, a bude pro předávání údajů použita tato metoda, tak se hodnoty odešlou na server znovu a může například dojít k duplicitnímu zápisu do databáze. Při zpracování formulářů na webové stránce se nejčastěji používá metoda POST. Tato metoda hodnoty proměnných nepřipojí za adresu URL, a proto nelze údaje přečíst pouhým pohledem na adresní řádek, nicméně pokud se používá nezabezpečený protokol HTTP, tak je možné tyto data odposlechnout na sítové komunikaci. Údaje posílané přes tuto metodu jsou uloženy do požadavku na odesílaném server. Pokud se bude uživatel snažit obnovit stránku, tak internetový prohlížeč zobrazí varování o tom, že dojde k opětovnému odeslání vyplněných údajů. Uživatel může odeslání zrušit nebo tyto údaje skutečně odeslat na server znovu. (15)

3.2.2 Klientská část desktopové aplikace

Uživatel (klient) si bude moct zobrazit pohodlně data ze všech OPComů, ze kterých chce data získávat. Program bude vyvinut také v programovacím jazyce JAVA. U tohoto programu je vyžadováno, aby běžel nepřetržitě a data ze senzorů čistoty oleje byly obnovována za předem definovanou dobu, kterou bude moct uživatel nastavit.

3.3 Mobilní aplikace

V dnešní době chytrých telefonů je dobré, pokud je možné jakákoliv data zobrazovat i na těchto zařízeních. I tento projekt se zaměřuje na to, jak tyto data pohodlně zobrazit na chytrém mobilním telefonu nebo tabletu.

Mobilní aplikací rozumíme program, který je spuštěn na chytrém telefonu, tabletu, chytrých hodinkách nebo na terminálu s operačním systémem. Tyto aplikace se dají získat z obchodů s aplikacemi pro každý systém (Apple store, Google play). Aplikace mohou sloužit pro prohlížení emailů, mapy a navigace, zapisování různých poznámek, pro zábavu a hry nebo jiné různé využití.

Jako cílový systém byl vybrán operační systém Android, protože je přítomen na velkém množství chytrých zařízení a vývoj aplikací na tuto platformu je relativně snadný. Podle serveru netmarketshare.com (16) má Android v roce 2017 64.43% podíl na trhu. Nejčastějším jazykem pro vývoj aplikací na Android je jazyk Java. Ostatními jazyky jsou například: Kotlin, Corona, Phonegap, C# anebo jiné další. Vývojář není nijak omezen platformou, na které vyvíjí. Aplikace může být vyvíjena na Windows, Linux nebo i Mac OS za pomoci různých vývojových prostředí.

3.3.1 Operační systém Android

Jedná se o operační systém pro různá zařízení, určený především pro chytré telefony (smartphone) nebo tablety. Momentálně se systém Android dostává do zařízení, jako jsou například televize, domácí systémy, autorádia, hodinky a další. Systém je šířen pod licencí open-source. Poprvé byl uveden a trh v roce 2007 jako svobodná platforma, kterou si mohl každý výrobce mobilních zařízení stáhnout a upravit podle svých představ. Momentálně je Android na trhu s verzí 7.0 Nougat a brzy se očekává uvedení následující verze, která má zatím označení Android O.

Obrázek 10: Logo Android (32)

3.3.2 Aplikace v systému Android

Architektura Androidu je založena na systému Linux. Jednotlivé spuštěné aplikace se chovají jako separátní uživatel. Každé aplikaci je přiděleno unikátní číslo (Linux user ID). Toto číslo zná pouze systém, aplikace o něm neví. Toto unikátní ID slouží k tomu, aby aplikace měla přístup jen ke svým složkám a souborům, pokud to jinak uživatel nepovolí. Android vyznává politiku přidělování co nejmenšího počtu oprávnění. Ve výchozím stavu aplikace dostane přístup pouze k tomu co je potřeba k jejímu běhu. Aplikace si může v průběhu požádat o různá oprávnění k přístupu. Tento požadavek uživatel buď potvrdí, nebo zamítne. (17)

3.4 Webová aplikace

Zobrazovat údaje pouze na desktopových nebo mobilních aplikacích by v některých případech nemuselo stačit. Pro tento projekt bude vytvořena webová aplikace, která bude v roli klienta stahovat data čistoty ze serveru a zobrazovat je. Webovou aplikaci není potřeba instalovat na klientské zařízení a není závislá na platformě. Do aplikace bude implementováno administrační rozhraní chráněné heslem, ve kterém bude moct

uživatel nastavit, ze kterých OPComů se mají stahovat data a jakým protokolem se bude komunikovat mezi klientem (webovou aplikací) a serverem. Pro tvorbu tohoto systému bude použit programovací jazyk PHP a framework Nette.

Web Adobe.com (18) pojem webová aplikace definuje takto:

> *„Webová aplikace je webové místo, které obsahuje stránky s částečně nebo úplně neurčeným obsahem. Konečný obsah stránky se určí až tehdy, když návštěvník požádá o stránku z webového serveru. Protože konečný obsah stránky se pro různé požadavky liší a závisí na akcích návštěvníka, říká se takové stránce dynamická stránka.“*

Dynamická webová stránka se může v čase měnit, podle toho jaké zadávají její uživatelé vstupu. Takové aplikace mohou sloužit jako internetový obchod, redakční systém, CRM systém nebo jiné systémy pro podporu rozhodování. Webová aplikace se dá podle zákazníka naprogramovat prakticky na jakékoliv použití. Princip takových stránek je, že uživatel, který si chce zobrazit obsah, pošle požadavek na webový (aplikační) server, který požadavek vyhodnotí a zpět vrátí už jen čistý HTML kód stránky. Internetový prohlížeč tento kód už jen správně interpretuje a zobrazí uživateli.

3.4.1 PHP

Pro vývoj dynamických webových stránek je zapotřebí zvolit vhodný nástroj, zde nám poslouží programovací jazyk PHP (PHP: Hypertext preprocesor, původně Personal Home Page). Jazyk PHP je šířen pod licencí open-source, každý tak může nahlédnout do jeho zdrojového kódu. Především je určen na to, aby pracoval na straně serveru a zpracovával požadavky od klienta. Programátor v něm může programovat procedurálně, objektově nebo i kombinací obojího. Není závislý na platformě, na které bude spuštěn. Může běžet na systémech: založených na Unixu, Windows, Mac OS X, RISC OS a dalších. Jediné co je potřeba, je mít na operačním systému nainstalován webový server. Takovým webovým serverem může být Apache, IIS nebo jiný. (19)

3.4.2 Framework Nette

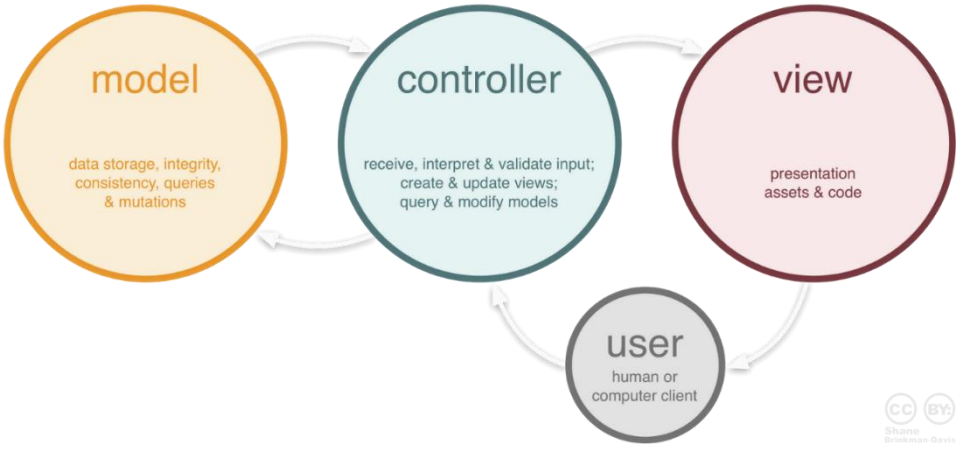
Při programování v jakémkoliv programovacím jazyce může být složité soustředit se na všechny aspekty, jako jsou: jak budou mezi sebou komunikovat jednotlivé vrstvy modelu nebo jak bude přistupováno k databázi. Tyto problémy řeší frameworky.

Dovolují programátorovi, aby se mohl co nejvíce zaměřit na zadání projektu a využívat všech výhod použití frameworků.

Pro jazyk PHP byl vyvinut Framework Nette, který má spoustu výhod pro vývojáře webových aplikací. Nativně v sobě obsahuje funkce a nástroje, které pomáhají při přístupu k databázi, zabezpečení proti útokům a při ladění aplikace. Je založený na modelu MVC (Model View Controller). Při zabezpečování webové aplikace je tento Framework nadmíru nápomocen. Ochrana proti XSS (Cross-Site Scripting) je řešena za pomoci technologie Context-Aware Escaping. Tato technologie vývojáře zbaví problému řešit striktní ošetřování výstupů, protože CAE to dělá všechno automaticky. Dalším útokem, před kterým tento Framework chrání, je Cross-Site Request Forgery (CSRF). Útočník využívající tuto zranitelnost se snaží uživatele přimět, aby navštívil stránku, která uskuteční útok na stránku, na které je uživatel právě přihlášen. Tuto ochranu přidáme jednoduchým zapsáním „*$form->addProtection()*“ u formuláře, který před tímto útokem chceme ochránit. Tímto zápisem se bude společně s formulářem vytvářet autorizační token, který se bude odesílat a kontrolovat. Před tímto útokem by měly být chráněny vstupy měnící důležitá aplikační data, respektive veškeré formuláře v administrační části aplikace. Nette automaticky ochraňuje proti dalším typům útoků, jako jsou URL attack, control codes, invalid UTF-8, session hijacking, session stealing a session fixation. (20) (21)

3.4.3 Model MVC

Při vývoji aplikací je potřeba rozdělit kód podle toho co dělá a k čemu slouží. Toto pomáhá řešit návrhový vzor Model View Controller. Hlavní myšlenkou tohoto návrhu je rozdělní částí, které provádí aplikační logiku od částí, které se starají o vykreslování



Obrázek 11: MVC architektura (27)

pro uživatele. Opakem toho je tzv. špagetový kód, kdy se mezi sebou aplikační logika a logika pro vykreslování promíchá a jsou pak tyto logiky na sebe silně vázané. Takový kód je velmi těžké udržovat, protože pokud by mělo dojít k nějakým změnám, vývojář musí dělat úpravy na mnoha místech.

3.4.3.1 Model

Část starající se o vše co je spojeno s aplikační logikou. Má na starosti různé výpočty, manipulaci s daty, rozhodování nebo připojení do databáze. Model vůbec neví o existenci pohledu (view) ani controlleru.

3.4.3.2 View

Touto součástí je zajišťováno zobrazování výstupu uživateli. Ve frameworku Nette se jedná o soubor s příponou latte, ve kterém jsou HTML značky a značky šablonovacího systému. Zde se používá šablonovací systém Latte. Systém za pomoci maker usnadňuje zápis HTML kódu stránky tak, že se složitá kombinace PHP a HTML omezí pouze na několik řádků zapsaných pomocí systému Latte.

Příklad Latte:

```
{* Latte template *}
{block title}
Trendy
{/block}
{block content}

{foreach $opcoms as $opcom}

        <div class="col-lg-3 col-md-12 col-sm-12">
            <div class="panel panel-primary">
             <div class="panel-heading ">
                 <span class="huge ">
                     <a class="text-white nounderline" n:href="showOpcomTrend,$opcom->id">
                         <i class="fa fa-line-chart"></i>  OPCOM {$opcom->id}</a></span>
                </div>
             </div>
        </div>

{/foreach}
```

Obrázek 12:Příklad šablonovacího systému Latte (Zdroj: vlastní)

V příkladu je uvedeno makro pro cyklus foreach, ve kterém se vypisují panely obsahující odkazy na presenter „showOpcomTrend“ s paremetrem id, který zde představuje hodnota id objektu „opcom“. Pomocí zápisu „n:href“ se vygeneruje odkaz, který je na výsledné HTML stránce zobrazen jako klasický URL.

3.4.3.3 Controller

Controler je mezičlánek mezi modelem a pohledem. Ve frameworku Nette se tato část nazývá presenter. Předává informace získané od uživatele a zpracovává je do podoby, ve které je potřebuje model, tak aby je mohl vyhodnotit, a vrátit zpět výsledek který je odeslán do pohledu na vykreslení. (22)

4 Analýza a Implementace

V projektu, kde se jedná o sledování čistoty hydraulického oleje je hlavním požadavkem to, aby se data srozumitelně zobrazovala na klientském zařízení. Dalším požadavkem je aby se tato data ukládala do databáze pro pozdější analýzu a vyhodnocení. Data se budou zobrazovat na několika typech zařízení. Těmito typy jsou: PC, mobilní telefon se systémem Android a webový prohlížeč. Klientské aplikace budou moct definovat od kterých OPComů požadují aktuální data.

Od serverové části se očekává to, že data poskytne rychle a spolehlivě. Komunikace mezi serverem a převodníky je realizována za pomoci protokolu UDP, a proto je nutné pro každý OPCom vytvořit soket, přes který se budou zasílat dotazy na aktuální data. Často se stává, že je potřeba zobrazit data z více OPComů najednou, proto je nutné provádět tyto operace zároveň a nezávisle na sobě. Na takové řešení je potřeba nasadit vícevláknové programování, kde se jednotlivé procesy provádějí paralelně. Tímto způsobem se operace stahování dat oproti původní verzi mnohonásobně zrychlí. Celá operace bude (v případě, že nebude žádný problém s komunikací) trvat maximálně několik sekund.

4.1 Uživatelské cíle

4.1.1 Zobrazení dat čistoty ze senzorů OPCom II

Data stahovaná z čidel čistoty hydraulického oleje za pomoci serveru se budou přenášet jednotlivým klientům, kteří o tyto data požádají. Uživatel klientského programu chce tyto data zobrazit srozumitelně a v přehledné formě. Vzhled aplikace běžící na desktopu se bude velmi podobat vzhledu aplikace naprogramované v prostředí Control Web 6.1.

4.1.2 Úprava seznamu zobrazovaných OPComů

Každá klientská aplikace bude moct žádat o data z jiných čidel čistoty oleje, proto musí být v aplikacích implementován mechanismus na spravování seznamu požadovaných

cílů (OPComů). Záznam v tomto seznamu bude obsahovat vše co je potřebné ke komunikaci s převodníkem z ethernetové komunikace na sériovou. Těmi údaji jsou: IP adresa, port, id opcomu a středisko ve výrobě na kterém je stav s čidlem čistoty. Údaj o středisku je nutný pro to, aby se údaje z OPComu zobrazovali na správném místě aplikace. Pro vytváření nového záznamu je zapotřebí vytvořit formulář, který tyto údaje do seznamu uloží. Každý záznam bude moct uživatel upravit nebo smazat.

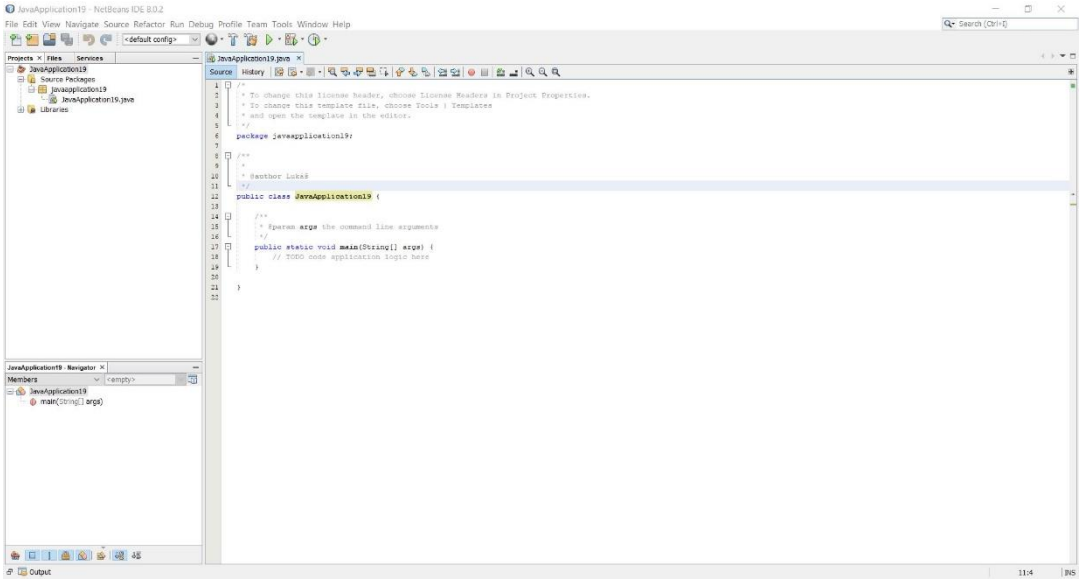
4.2 Použitá vývojová prostředí

Vývojová prostředí (dále jen IDE z anglického Integrated Development Enviroment) pro tvorbu aplikací se většinou specializují na jeden konkrétní programovací jazyk. Jedná se o aplikace vybavené editorem, kam vývojáři zapisují kód programované aplikace. Takové prostředí velmi usnadňuje práci programátorům, a to tím že barevně odlišují kód, radí se syntaxí kódu nebo upozorňuje na chyby v syntaxi. Tím se jejich čas strávený na vývoji aplikace značně zkracuje. IDE dokáží také generovat celé bloky kódů, například konstruktory, getry nebo setry tříd. Často obsahují možnost návrhu uživatelského prostředí, za pomoci palety komponent a systému drag and drop, kdy vývojář jednoduše přesouvá komponenty z palety na požadované místo v aplikaci. Nedílnou součástí programovacího prostředí bývá také debugger, pomocí kterého programátor může krokovat aplikaci a sledovat stavy jednotlivých proměnných. Debugger také může zastavit vykonávání programu, pokud zachytí nějakou výjimku. Prostředí také obsahují kompilátor daného programovacího jazyka. Nejnovější verze IDE obsahují například integraci verzovacích software, které pomáhají při práci v týmu. (23)

4.2.1 NetBeans IDE

Vývojové prostředí NetBeans je oficiální IDE pro programovací jazyk Java 8. Dokáže také efektivně pomáhat i při vývoji v jiných programovacích jazycích jako jsou například C/C++, PHP nebo Groovy. Uživatel si může za pomoci pluginů doinstalovat podporu pro jiné programovací jazyky. Je velmi vhodný také pro vývoj moderních webových stránek a webových aplikací, protože podporuje vývoj v HTML5, CSS3 a JavaScriptu. Při vývoji pomáhá také to, že obsahuje nápovědu pro různé jazykové syntaxe, a tím usnadňuje práci při vývoji. Je multiplatformní, to znamená, že může být nainstalován na jakémkoliv systému podporující jazyk Java. Mezi podporovanými systémy jsou: Windows, systémy založené na Linuxu a Mac OS X. Má pod sebou velkou

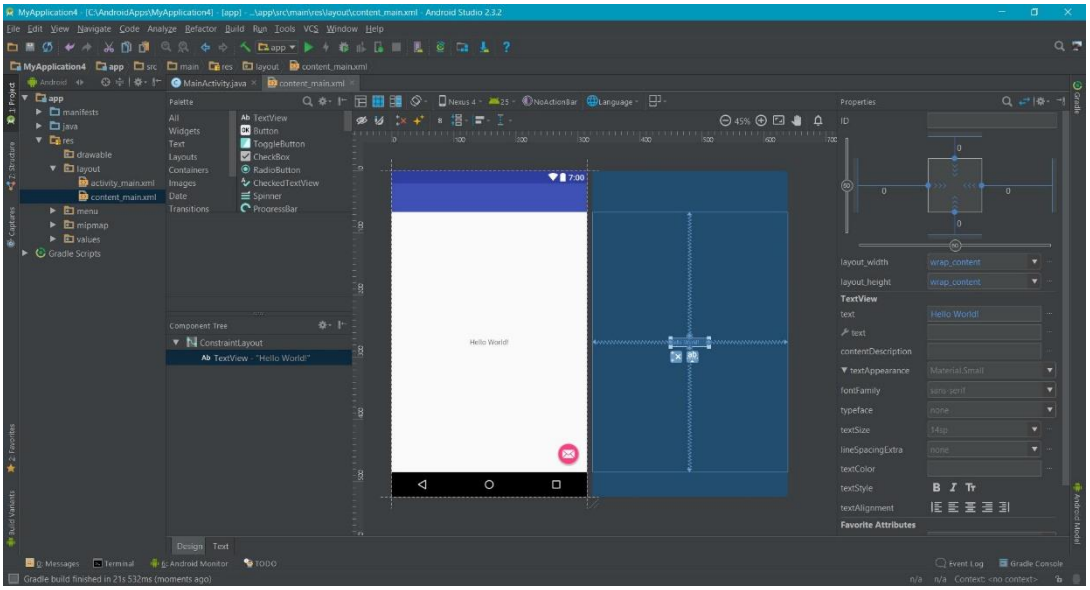
komunitu uživatelů a vývojářů, protože je šířen pod licencí open source, a proto je také zdarma ke stažení. (24)



Obrázek 13: Prostředí NetBens IDE (Zdroj: vlastní)

4.2.2 Android Studio

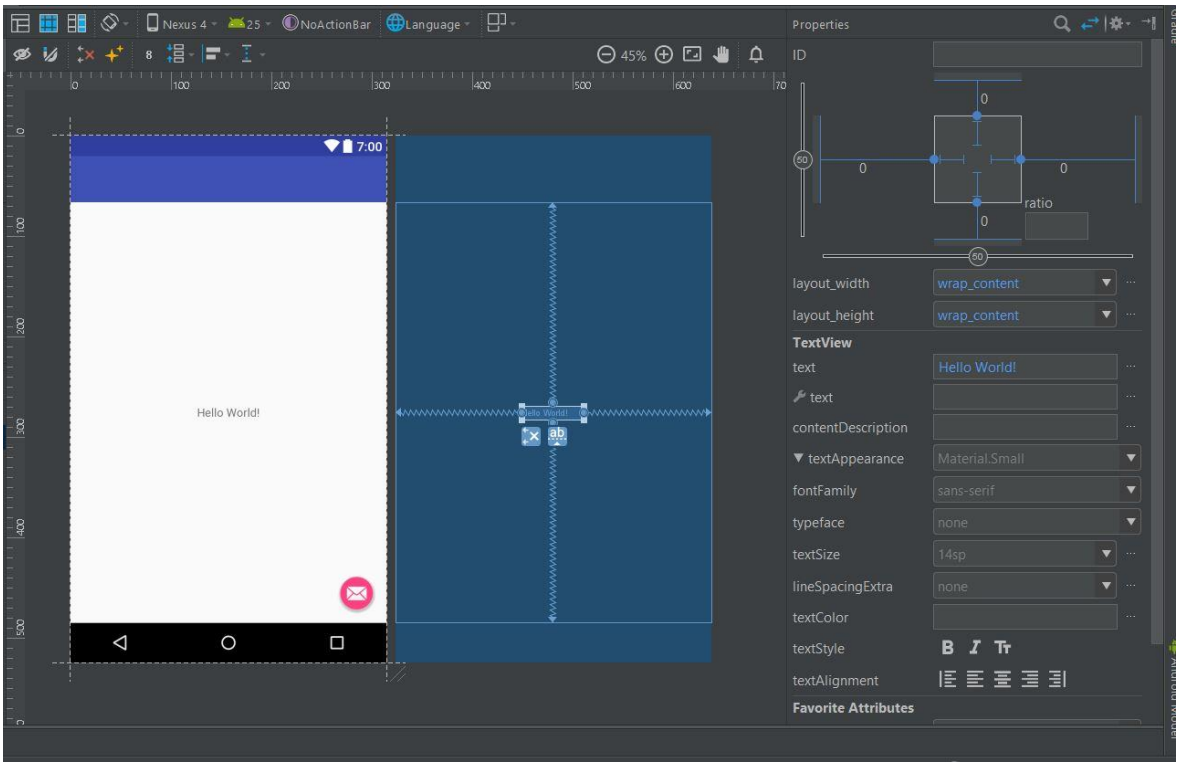
Pro vývoj aplikací na operační systém Android bylo speciálně vyvinuto IDE s názvem Android Studio, které je založeno na prostředí IntelliJ IDEA. Poprvé byl představen na konferenci Google I/O v roce 2013 a od června 2013 je zdarma k dispozici ke stažení. Jako výše zmíněné NetBeans IDE je také multiplatformní a proto jej lze nainstalovat na systémy Windows, Linux nebo Mac OS X. Toto IDE je určeno pouze na vývoj aplikací pro Android.



Obrázek 14:IDE Android Studio (Zdroj: vlastní)

Dokáže se propojit s verzovacími systémy jako je GitHub nebo Subversion. Verzovací systémy slouží pro sledování a udržování změn v kódu a pro práci v týmu. Toto se vyplatí při provedení změn, které způsobní nežádoucí chování, protože za pomoci verzovacího systému se lze pohodlně vrátit k plně fungující verzi.

Prostředí také velmi dobře pomáhá při grafickém návrhu aplikace. Obsahuje řadu předloh a návrhových vzoru uživatelského prostředí. Vývojář si může zobrazit návrh designu se všemi barvami tak, jak bude ve skutečnosti vypadat na spuštěné aplikaci, nebo si k tomuto návrhu zobrazit plán aplikace (tzv. blueprint), na kterém je znázorněno, jak jsou jednotlivé komponenty mezi sebou propojeny, nebo na sobě závislé. Layout aplikace nemusí být tvořen pouze v editoru, lze jej také zapsat do XML souboru, ale při použití grafického editoru je tento soubor generován automaticky.



Obrázek 15:Android Studio s grafickým návrhem aplikace (Zdroj: vlastní)

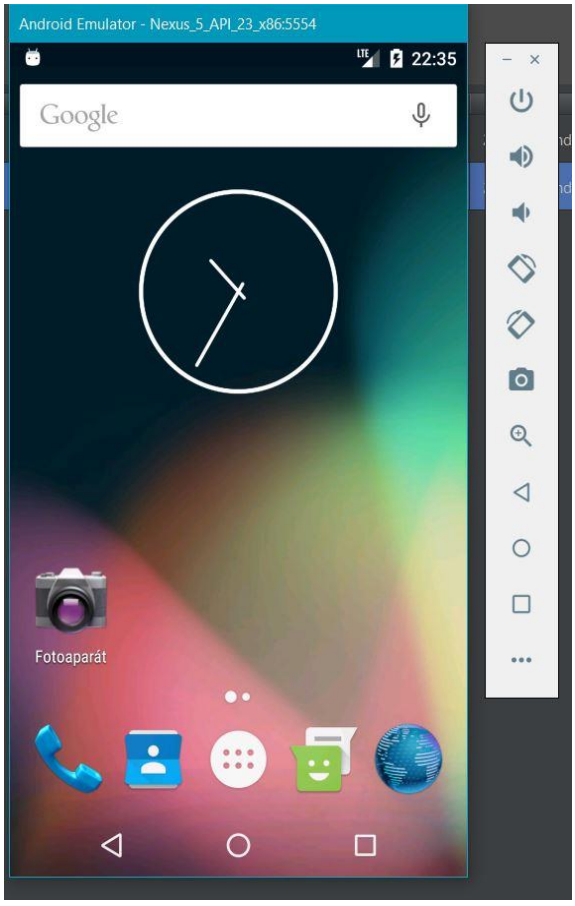
Součásti instalace prostředí jsou různé komponenty, které jsou zapotřebí pro jeho běh a vývoj aplikací. Při instalaci IDE se společně s programem do počítače nainstaluje také Android SDK Tools, kompilátor Android, emulátor a popřípadě Java Development Kit.

Důležitou funkcí je technologie Instant Run. Pokud vývojář provede nějaké změny v programu aplikace, tak se za pomoci Instant Run nahraje pouze změněná část aplikace, takže není nutné generovat nový apk soubor. Tuto technologii, která velmi

zrychlí testování aplikace, podporuje Android Studio verze 2.3 a vyšší. Také je zapotřebí vyvíjet pro Android 5.0 (resp. API level 21) nebo vyšší, pro správný chod této funkce.

Prostředí v sobě také obsahuje další editory pro zjednodušení práce. Například editor překladů pomůže s tvorbou vícejazyčné verze aplikace. Editor pro úpravu témat vývojáři dovolí pohodlně měnit barevné schéma vytvářené aplikace.

Jak je výše zmíněno, tak Android studio obsahuje emulátor. Emulátor slouží pro spuštění plnohodnotného operačního systému, v tomto případě systému Android, na hostitelském počítači. Na emulovaném systému lze pracovat naprosto stejně jako by byl systém naistalován na mobilním zařízení. Pomocí emulátoru lze simulovat různé stavy telefonu nebo jeho senzorů. Takto lze simulovat polohu telefonu v prostoru, stav baterie, lokaci za pomoci GPS souřadnic, stav telefonní sítě, stav barometru, teplotu CPU, magnetické pole, hodnoty světla nebo stav senzoru přiblížení. Lze také uskutečnit simulovaný hovor anebo zaslání sms zprávy. Emulátor slouží pro testování vyvíjených aplikací. (25)



Obrázek 16: Systém Android spuštěný v emulátoru (Zdroj: vlastní)

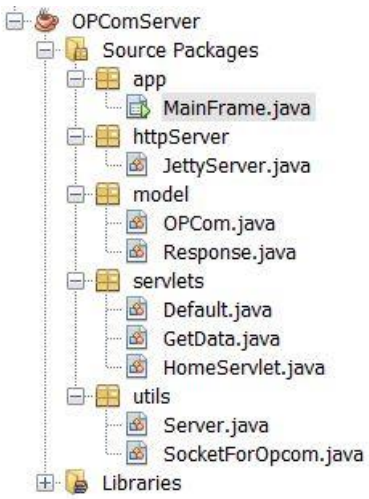
4.3 Implementace serverové části

Server je stěžejní částí projektu, protože obstarává veškerá data ze snímačů čistoty. Obsahuje v sobě dvě služby pro poskytování těchto dat. Jednou možností jak získat data ze serveru je odeslat dotaz na otevřený port (uživatel číslo portu může zvolit). Server dotaz vyhodnotí a odešle požadavky na převodníky. Druhou možností jak stáhnout data je komunikace s HTTP web serverem. Web server je implementací serveru Jetty od společnosti Eclipse, který umožnuje vytvářet servlety v jazyce JAVA. Servlet je aplikace naprogramovaná v jazyce Java, která zpracovává HTTP požadavky, resp. každý potomek třídy javax.servlet.http.HttpServlet. Dotaz klient zašle na web server jako parametr v URL adrese, které pak server vyhodnotí a rozešle jednotlivé požadavky na požadované snímače.

Aplikaci serveru lze také využít pro odeslání příkazu na jakoukoliv adresu a port. Netisknutelné znaky, které je potřeba připojit za řetězec aby byl příkaz kompletní a považován snímačem za potvrzený, jsou za odesílaná data připojeny automaticky. Tato funkce by mohla sloužit pro testování správného nastavení převodníků, nebo lze tímto způsobem manuálně získávat data ze senzorů čistoty oleje, nicméně tyto data budou v surovém formátu, tak jak je poskytuje senzor samotný. Provádět komunikaci tímto způsobem je vhodné, pouze tehdy když jsou obě služby, jak web server tak UDP server, zastaveny. Pokud by byl soket vytvořen a tím obsazen UDP port, tak by server nemohl vytvářet další sokety se stejným portem a IP adresou, a tím by byla znemožněna komunikace a následná odpověď klientovi, který si o data požádal.

4.3.1 Struktura aplikace

Při vývoji aplikace v jazyce Java je dobré logicky oddělovat do balíčků objekty podle toho, k čemu budou sloužit. Balíčky také napomáhají předcházet konfliktům se jmény tříd. Třídy, které slouží pro zobrazování uživatelského rozhraní, je vhodné shromažďovat v jednom jediném balíčku.



Obrázek 17: Struktura aplikace (Zdroj: vlastní)serveru

V aplikaci sloužící jako server je vytvořeno několik balíčků pro snadnou orientaci při vývoji a pro oddělení účelu jednotlivých objektů (viz obrázek 17). V balíčku „app“ je hlavní spustitelná třída „MainFrame“, která se

také stará o zobrazování uživatelského rozhraní. Třída je potomkem javax.swing.JFrame. V této třídě je i několik metod sloužících pro hlavní chod aplikace. Jsou to metody na spuštění serverů a jejich vykonávání. Spuštění těchto metod vyvolá zavedení nových vláken pro tyto procesy. Kdyby nebyly spuštěny nové vlákna, tak by bylo blokovalo grafické rozhraní a nebylo by tak možné aplikaci dále ovládat dokud by prováděná metoda neskončila.

Spustit nové vlákno lze dvěma způsoby. Jedním způsobem je vytvoření třídy, které implementuje rozhraní Runnable, anebo třídy která je potomkem třídy Thread. V oddělném vlákně běží proces UDP serveru naslouchající na zvoleném portu a sleduje příchozí požadavku. V dalším separátním vlákně může být spuštěn HTTP web server.

Balíček „model“ slouží pro třídy, které shromažďují data pro následní zpracování. Jsou to třídy „Opcom“ a „Response“. První ze jmenovaných tříd je určena pro data, které jsou přijata od klientů a za pomoci konstruktoru je vytvořen objekt obsahující IP adresu, port a id OPComu, ze kterého jsou požadovány data. Třída „Response“ je struktura sloužící pro uchování odpovědi z OPComu. Obsahuje v sobě text odpovědi a id zařízení, ze kterého byla přijata.

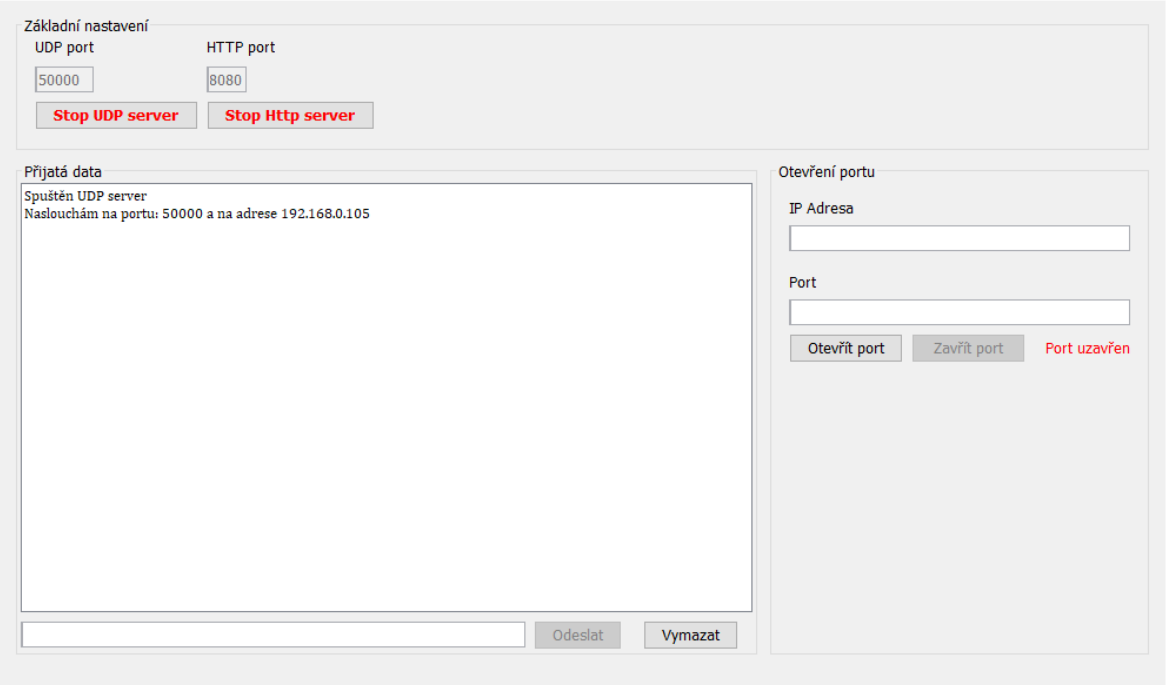
Pomocné třídy sloužící pro provádění výpočtů nebo různých akcí, ale nejsou v nich obsaženy žádná data, se v tomto případě uchovávají v balíčku „utils“. Třída SocketForOpcom je implementací rozhraní Callable, které dokáže z prováděného vlákna vrátit hodnoty.

4.3.2 Uživatelské rozhraní

Grafické uživatelské rozhraní (GUI) je nedílnou součástí každé aplikace, které má být ovládaná uživatelem. Před používáním GUI se programy ovládaly za pomoci příkazové řádky, tomuto prostředí se říkalo Command-line interface nebo jen CLI. S příchodem moderních systému se tento systém stal zastaralým a pro běžného uživatele nepoužitelným, protože pro ovládání je nutné si pamatovat různé příkazy a jejich parametry, nicméně některé systémy, jako jsou různé servery, nebo síťové prvky, se obsluhují právě přes příkazovou řádku. V grafickém uživatelském rozhraní najdeme různé prvky napomáhající při ovládání aplikace. Hlavním takovým prvkem jsou okna,

ve kterých jsou rozmístěny další komponenty. Některé komponenty slouží pro ovládání a vkládání dat, a jiné zase pro zobrazování výstupu pro uživatele.

Pro snadné ovládání uživatelem je vytvořeno jednoduché grafické uživatelské rozhraní, kde může uživatel služby zastavovat nebo spouštět. Před spuštěním UDP serveru může uživatel nastavit, na jakém portu bude aplikace naslouchat, ze stejného portu bude aplikace zasílat odpovědi. Web server komunikuje přes port 8080 a tento port je pevně daný a nelze jej uživatelsky měnit. Uživatel má zde možnost vyplnit IP adresu a port a spojit se přímo s daným převodníkem a získávat tak data manuálně, ale tuto komunikaci je dobré provádět jen tehdy, když nejsou spuštěny služby pro UDP server nebo HTTP server. Pokud jsou dotazy na data z OPComů přijata přes UDP server, tak jsou zobrazena v prvku textArea, do kterého se zapíše nový řádek s datem a časem přijetí a IP adresa od koho byl příkaz přijat a text dotazu.



Obrázek 18: Uživatelské prostředí serveru (Zdroj: vlastní)

Protože, aplikace serveru musí běžet neustále, tak aby bylo možné komunikovat s převodníky, je nutné zajistit to, aby nebylo možné aplikaci jednoduše náhodně ukončit. Ukončení aplikace se provádí za pomoci klávesové zkratky Ctrl + Q. Po stisknutí kombinace se zobrazí dialog pro potvrzení ukončení programu.

4.3.3 Komunikace s převodníky

Pro komunikaci je zapotřebí vytvořit soket, přes který budou data zasílána a přijímána. Jak je již zmíněno výše, soketem nazýváme jedinečnou kombinaci IP adresy a čísla portu. Reprezentací UDP soketu v jazyce Java je objekt DatagramSocket, který při konstrukci otevře port pro naslouchání a odesílání na lokálním zařízení. Pro zasílání dat existuje objekt DatagramPacket. Do konstruktoru tohoto objektu se vloží zasílaná data v bytech, jejich délka, adresa vzdáleného zařízení a port cíle. Takovýto paket je nutné vytvořit i pro přijímání dat, v tomto případě se v konstruktoru definuje pouze buffer a jeho délka.

Pro odesílání paketu slouží metoda send(DatagramPacket p), kterou nabízí třída pro soket. Přijímání dat probíhá obdobným způsobem. Do parametru metody receive(DatagramPacket p) se doplní paket pro příjem dat, který je předem definovaný. Tato metoda blokuje další provádění procedury, dokud nejsou data přijata. Pokud nechceme, aby se na data čekalo nekonečně dlouho dobu, jen nutné pro soket nastavit metodou setSoTimeout(int timeout) dobu čekání. Doba čekání se nastavuje v milisekundách. Pokud data do nastavené doby nedorazí, tak socket vyhodí výjimku SocketTimeoutException.

Při ukončení komunikace nebo při jakékoliv chyby je nutné soket uzavřít, tak aby jej bylo možné znovu použít. V případě kdyby nebyl soket uzavřen, tak při jeho tvorbě a pokusu otevřít již zabraný port vznikne výjimka SocketException s informací o tom že port je již použit.

Třída SocketForOpcom vytvořená právě pro komunikaci s převodníky je naprogramována tak, aby bylo možné zasílat i více příkazů. V konstruktoru je parametr typu pole řetězců, do kterého se příkazy vloží. Tato třída implementuje rozhraní Callable, proto disponuje metodou call, která má návratovou proměnnou, v tomto případě je to proměnná třídy Response.

Při komunikaci se senzory čistoty se používají tři příkazy, které se postupně odešlou. Pokud již při prvním příkazu nedojde k odpovědi, tak se další dva už nezasílají a do proměnné pro odpověď se uloží řetězec „Chyba“, signalizující problém s komunikací mezi serverem a převodníkem. Tento problém může být způsobem špatnou kombinací

portu a adresy zasílané od klienta, špatným nastavením převodníku nebo jenom tím, že je zkušební stav, na kterém je senzor umístěn, vypnutý.

Příkazy zasílané na senzory jsou:

- RID – přečte identifikaci senzoru (typ, výrobní číslo, verzi firmware),
- RCon – přečte aktuální konfiguraci senzoru (alarmy, dobu měření, dobu čekání mezi měřeními)
- RVal – přečte aktuální měřené hodnoty.

Příkladem odpovědi na příkaz RID je:

„$ARGO-HYTOS;OPCom II;SN:000949;SW:2.02.15;CRC: „.

Jak je z odpovědi na příkaz vidět, tak je každý údaj oddělen středníkem, a tak je snadné tyto údaje rozebrat. Za každou odpovědí jsou také netisknutelné znaky CR (carriage return) a LF (line feed). Všechny odpovědi na zaslané příkazy jsou poskládány za sebe a zbaveny netisknutelných znaků, tyto znaky jsou připojeny až na úplný konec řetězce sjednocených odpovědí. Odpovědi ze všech požadovaných senzorů, o které klient požádal se tak spojí do jednoho řetězce a ten je klientovi zaslán jako odpověď, kterou poté klientská aplikace zpracuje.

4.3.4 Vícevláknové programování

Při chodu aplikace, která běží pouze na jednom vláknu, se může stát, že toto vlákno skončí chybou nebo bude něčím blokováno a tím bude přerušen chod celého programu. Toto pomáhá řešit vícevláknové programování neboli multithreading.

Spuštění nového vlákna lze docílit několika způsoby. Můžeme dědit z třídy Thread nebo implementovat rozhraní Runnable popřípadě Callable.

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }

}
```

Ukázka kódu 1: Příklad implementace rozhraní Runnable (28)

Pokud je zapotřebí, aby z provádění vlákna byl navrácen nějaký výsledek je nutné místo Runnable implementovat rozhraní Callable. Tyto rozhraní jsou si velmi podobné, s tím rozdílem, že nemá metodu run(), ale metodu call(), která dokáže vrátit výsledek zpracování kódu. Výsledek této metody nám pomáhá získat další objekt, který je implementací třídy Future.

```
public class HelloThread extends Thread {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new HelloThread()).start();
    }

}
```

Ukázka kódu 2: Příklad dědění z třídy Thread (28)

Třída Future reprezentuje výsledek z asynchronní operace. Poskytuje informaci, zda je vyhodnocení funkce kompletní nebo se má ještě na výsledek čekat. Výsledek z třídy implementující rozhraní Callable je možné dostat pouze přes metodu get() až tehdy, když je operace dokončená. (26)

V projektu sledování čistoty oleje je zapotřebí rychle dostat data od všech snímačů, to vyžaduje současné do zaslání dotazů, proto je nutné spustit všechna vlákna najednou. Se spouštěním většího množství vláken pomáhá rozhraní ExecutorService. Toto rozhraní dokáže spravovat různý počet vláken, které můžou generovat objekty typu Future. ExucutorService má metodu submit(Callable<T> task>), která spustí vlákno a vrátí objekt typu Future.

```
        .
        .
        .
ExecutorService executor = Executors.newFixedThreadPool(opcoms.size());
        String[] q = new String[]{"RID", "RCon", "RVal"};
        ArrayList<Response> responses = new ArrayList<>();
        responses.clear();
        opcoms.stream().forEach((opcom) -> {
            try {
               Callable<Response> sockForOpcom = new SocketForOpcom(q,
opcom.getIpadresa(), opcom.getPort(), opcom.getId());
               Future<Response> future = executor.submit(sockForOpcom);
               list.add(future);
            } catch (SocketException ex) {
               executor.shutdown();
               Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }});
        list.stream().forEach((listItem) -> {
            try {
               responses.add(listItem.get());
            } catch (InterruptedException | ExecutionException ex) {
               Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        });
        .
        .
        .
```

Ukázka kódu 3: Použití ExecutorService v projektu (Zdroj: vlastní)

V aplikaci serveru je pro uchování Future objektů použita kolekce ArrayList. Z kolekce jsou poté postupně získávány odpovědi od snímačů čistoty.

Jednotlivé odpovědi na každý zaslaný příkaz ze snímače jsou poté seřazeny za sebe do jednoho řetězce.

4.3.5 HTTP server

V aplikaci je implementován webový server Jetty od společnosti Eclipse. Webové aplikace lze již nasadit na instanci serveru anebo lze server spustit jako komponentu Java aplikace.

Na serveru jsou definovány dva servlety. Jeden, defaultní, slouží pro zobrazení úvodní stránky s informací o tom jak se dostat k datům ze snímačů a druhý servlet už je pro přijímání požadavků od klientů a je jich následné zpracování.

Servlet GetData má metodu doGet (HttpServletRequest req, HttpServletResponse resp), která je spuštěna v případě, že je na URL adresu serveru (za adresu je nutné zapsat i číslo portu, na kterém je server naslouchá) zapsáno „/getData?opcoms= …“. Tato metoda rozebere řetězec který je obsažen v proměnné „opcoms“. Po rozebrání řetězce jsou jednotlivé údaje uloženy do objektů třídy Opcom a tyto objekty jsou

vloženy do seznamu, který je poté zaslán jako parametr do statické metody startComm pomocné třídy SocketForOpcom.

```
public JettyServer(Integer port) {
      server = new Server(port);
      ServletContextHandler cHandler = new
ServletContextHandler(server, "/");
      cHandler.addServlet(GetData.class, "/getData");
      cHandler.addServlet(Default.class, "/");
    }

    public void start() throws Exception {
      server.start();
      server.join();
    }

    public void stop() throws Exception {
      server.stop();
      server.join();
    }
```

Ukázka kódu 4: Konstruktor a metody HTTP serveru (Zdroj: vlastní)

Po zpracování pomocnou třídou, která zaslala dotazy na všechny požadované snímače, je navrácen zpět seznam všech odpovědí. Seznam je poté za pomoci objektu třídy OutputStream zapsán do odpovědi ze serveru. Prohlížeč odpověď interpretuje jako soubor hodnot oddělených čárkami, protože je metodou.setContentType("text/csv") nastaven typ přijímaného obsahu.

Odpověď od serveru trvá maximálně do 3 sekund v závislosti na tom, jak rychle jednotlivé snímače čistoty oleje zasílají odpovědi nebo zda vůbec odpoví.

4.4 Implementace klientské části

Klientská část projektu zahrnuje návrh a implementaci programu pro sledování počtu částic v hydraulickém oleji do tří prostředí. Tyto prostředí jsou desktop, mobilní zařízení a webová aplikace. Do jednotlivých aplikací je možné uložit seznam snímačů, od kterých má data získávat. Tento seznam se ukládá do souboru s hodnotami oddělenými středníky (soubory csv), proto aby jej bylo možné číst i po opětovném spuštění programu. Aplikace pro desktopy a chytré mobilní zařízení využívají stejnou strukturu dat ukládaných do souboru. Webová aplikace pro tento seznam používá SQL databázi typu MariaDB.

4.4.1 Desktop

Program pro desktopy je naprogramován v jazyce Java. Jeho vzhled je co nejvíce podobný předcházející aplikaci v programu Control Web 6. Pro klientské aplikace je od firmy ARGO-HYTOS požadavek, aby byly data z čidel čistoty aktuální každých 15 minut.

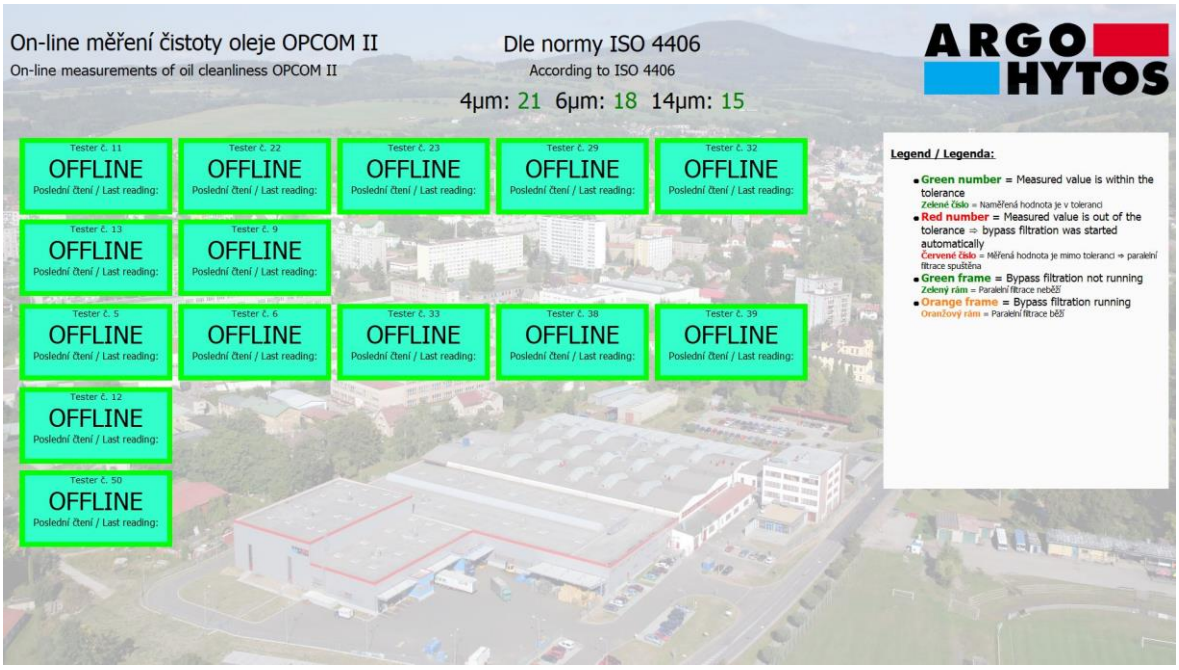
Tato desktopová aplikace je vytvořena tak, aby běžela přes celou obrazovku a dala se zobrazovat také na jiném než primárním zobrazovacím zařízení počítače. Aby bylo možné zobrazit aplikaci přes celou obrazovku, bylo nutné aplikaci zbavit rámečku s titulní lištou a ovládacími prvky. Protože aplikace nemá titulní lištu, tak se nedá přesouvat jednoduchým tažením myší po obrazovce. Přesouvání aplikace muselo být vyřešeno jiným způsobem. V dialogovém okně pro nastavení aplikace jsou tedy umístěny textová pole, do kterých lze zadat umístění okna na obrazovce. Pro tento případ jsou uvažovány souřadnice v rozmezí od -1920 bodů do +1920 bodů, a to proto, aby mohlo být okno posouváno ve všech čtyřech směrech. Absence ovládacích prvků zajišťuje to, aby aplikace nemohla být minimalizována na lištu nebo ukončena klikem myši. Jediná možnost ukončení aplikace je za pomoci klávesové zkratky Alt + F4. Po stisku se zobrazí výzva o potvrzení ukončení programu.

Jednotlivé údaje ze senzorů čistoty se zobrazují v malých rámečcích s barevným okrajem, který mění barvu podle toho, zda je překročena mez čistoty oleje a je spuštěna paralelní filtrace na zkušebním stavu. Zobrazují se pouze tři hlavní údaje, které jsou signifikantní pro rozhodování, kdy má být spuštěna filtrace. Těmi údaji jsou kódová čísla normy ISO 4406 pro velikosti částic 4μm, 6μm a 14μm. Pod těmito údaji je zobrazen čas poslední aktualizace údajů. V případě že zkušební stav není v chodu, tak není v hydraulické soustavě žádný průtok, a proto snímače vracejí nulové hodnoty míry znečištění. Tento stav je zobrazen tak, že se přes hodnoty v rámečku pro snímač zobrazí nápis „STANDBY“ a v případě, že se nelze se snímačem spojit, respektive je v řetězci přijatém od serveru obsaženo „Chyba“, tak se zobrazí nápis „OFFLINE“. Při kliknutím prvým tlačítkem myši se ukáže dialogové okno ve, kterém jsou vypsány další údaje ze snímače (například počty částic v 1ml, motohodiny, verze firmwaru nebo výrobní číslo).

Hodnoty stahované ze snímačů lze při každé aktualizaci uložit do databáze, kterou potom využívá webová aplikace pro zobrazení trendů. Pokud ještě není pro daný snímač vytvořená tabulka, tak se pomocí SQL příkaz CREATE TABLE IF NOT EXIST

vytvoří. Údaje se do databáze ukládají pouze tehdy, když je snímač ve stavu ONLINE, to znamená, že žádná z hodnot není nulová.

V dialogovém okně pro nastavení aplikace, do kterého se lze dostat pře pop-up menu po pravém kliku myší, lze nastavit parametry pro komunikaci se serverem, typ komunikačního protokolu, údaje pro připojení do databáze a pozice okna. Veškeré údaje nastavení jsou ukládány do registru systému Windows, aby je bylo možné používat i po opětovném spuštění aplikace. Přes pop-up menu se lze také dostat do formuláře pro správu OPComů, který zde bude reprezentován tabulkou s údaji o každém připojeném OPComu. Při vytvoření tohoto formuláře se načtou data z csv souboru a zobrazí se v tabulce. Při označení řádku tabulky, bude moct uživatel vybraný záznam smazat nebo upravit. Před provedením operace smazání se zobrazí formulář pro potvrzení.



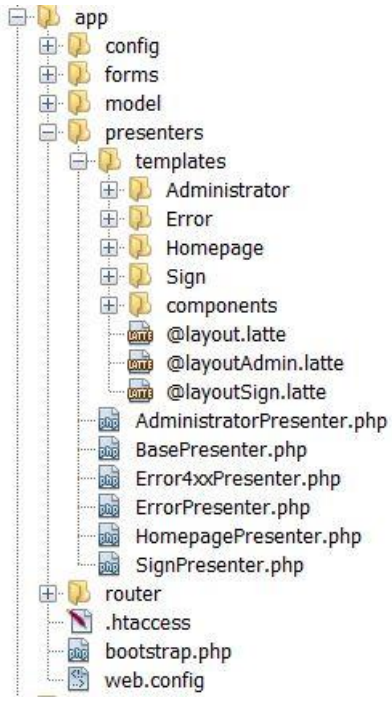
Obrázek 19: Uživatelské prostředí klientské desktopové aplikace (Zdroj: vlastní)

4.4.2 Webová prezentace/aplikace

Jak je již výše zmíněno, tak pro webovou aplikaci byl použit jazyk PHP a framework Nette. Na stránky bude moct uživatel přidávat a měnit údaje, ze kterých snímačů se budou stahovat. Přístup k seznamu OPComů bude umožněn přes administrační rozhraní, které bude ochráněno přihlašovacím jménem a heslem. V tomto rozhraní bude také možné zobrazit trendy hodnot z každého snímače. Možnost zobrazit

historické údaje bude závislá na tom, zda i požadovaný snímač bude sledován na desktopové aplikaci, která bude spuštěna nepřetržitě.

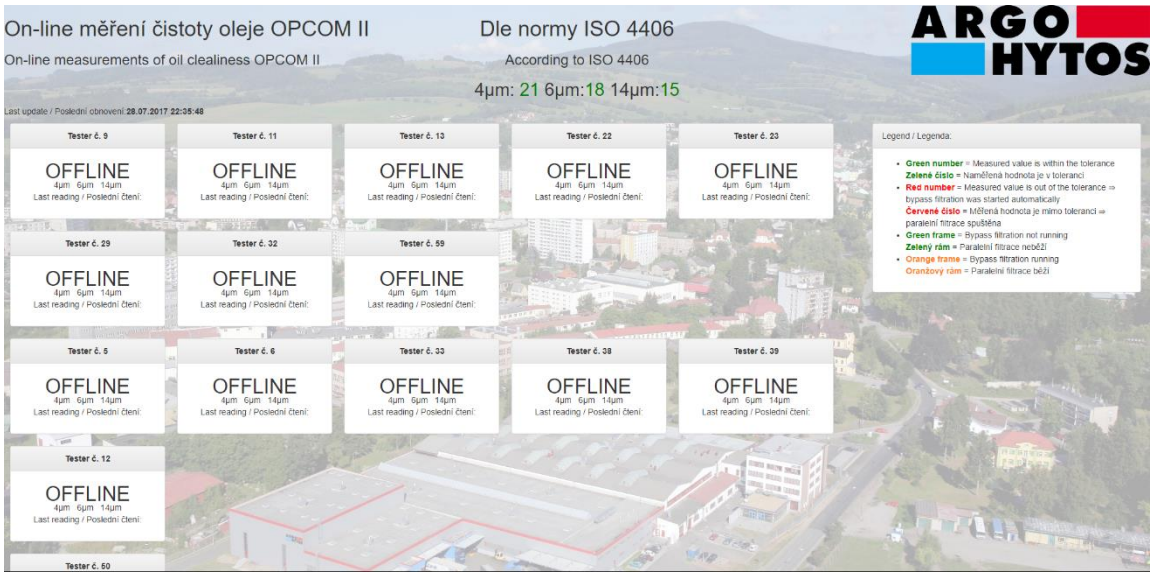
Jak je již výše zmíněno, je Framework Nette založen na modelu MVP. Pro každou vrstvu modelu je ve struktuře aplikace určena složka, do které se soubory vkládají a jsou určeny pravidla pro jejich pojmenování. Do presenterů se umísťují různé akce a metody na které jsou pak navázány pohledy definované ve složkách se stejným jménem jako má presenter. Například pro HomepagePresenter.php je ve složce Homepage pohled se jménem default.latte. Tento pohled je zobrazen jako hlavní stránka aplikace. V presenteru je metoda renderDefault(), která vyvolá právě zobrazení pohledu default.



Obrázek 20: Struktura webové aplikace (Zdroj: vlastní)

Metoda renderDefault přistupuje do databáze a z tabulky settings zjišťuje údaje pro komunikaci se serverem (IP adresa, port pro UDP, port pro HTTP, typ prokolu). Po přijmutí požadavku na zobrazení stránky od internetového prohlížeče je zjištěn seznam všech OPComů, ze kterých mají být získány údaje, je zaslán požadavek na server. Po přijetí odpovědi a jejím rozebráním na jednotlivé údaje je vytvořeno pole objektů OPCom, kterým jsou definovány atributy podle přijatých údajů. Pole je poté zasláno do view a tam jsou jednotlivé údaje ze snímačů zobrazeny v panelech. V případě že hodnoty jsou mimo toleranci, tak se barva panelu změní na oranžovou, jinak je panel zelený. Pokud nebude možné se snímačem možné komunikovat, tak je panel šedivý. V případě, že dojde při komunikaci se serverem k potížím nebo se nebude moc webová aplikace se serverem spojit, tak se zobrazí chybové hlášení s textem: „Chyba spojení se serverem“.

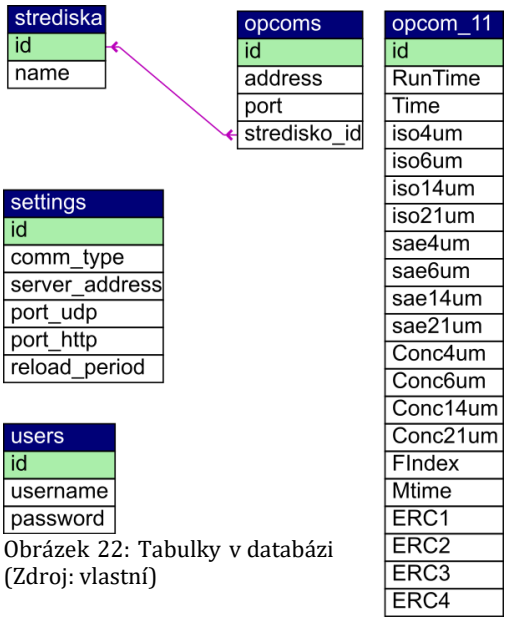
Samozřejmostí je tvorba kaskádových stylů, protože bez nich by aplikace vypadala stroze a obyčejný uživatel by se v ní těžko orientoval. Při tvorbě moderní webové aplikace se již prakticky nepíší kaskádové styly ručně. Při jejich tvorbě programátorovi pomáhají různé preprocesory nebo frameworky. Pro stylování aplikace bylo použit css framework Bootstrap 3, který pomáhá rychle vytvářet vzhled webových stránek. S jeho pomocí lze snadno tvořit přívětivé uživatelské prostředí, které je i responzivní, to znamená, že se styly prvků můžou měnit v závislosti na tom, na jakém zařízení a rozlišení je webová aplikace zobrazena.



Obrázek 21: Hlavní stránka webové aplikace (Zdroj: vlastní)

4.4.2.1 Databáze pro webovou aplikaci

Veškeré údaje pro webovou aplikaci se musí někde ukládat, tak aby je bylo možné znovu používat. Pro tento účel je zvolena SQL databáze MariaDB. V databázi je několik tabulek, které jsou důležité pro chod aplikace. Do tabulky opcoms se ukládají informace o snímačích ze kterých je potřeba stahovat data čistoty. Tato tabulka má také cizí klíč odkazující na záznam v tabulce strediska, ve které jsou uloženy jména středisek v závodu ARGO-HYTOS Vrchlabí. V tabulce users jsou záznamy jednotlivých uživatelů systému. Je zde zapsáno přihlašovací jméno, otisk hesla a role. Do databáze se přidávají i další tabulky, podle toho zda desktopový klient má povolené ukládání do



Obrázek 22: Tabulky v databázi (Zdroj: vlastní)

databáze. Do těchto tabulek se zapisují údaje čistoty z každého OPComu. Příkladem je tabulka opcom_11, která byla vytvořena desktopovým klientem.

4.4.2.2 Bezpečnost

Jak je již zmíněno v popisu frameworku Nette, je zabezpečen proti různým typům útokům mířící na webové aplikace. Formuláře v sekci pro nastavení aplikace jsou ochráněny proti útoku typu CSRF. Ochrana je přidána za pomoci metody addProtection() na požadovaném formuláři. V tomto případě jsou takto ochráněny formuláře pro nastavení údajů a změnu hesla. Veškeré výstupy vypisující se v pohledu jsou automaticky zbaveny jakýchkoliv HTML značek, aby nedošlo k útoku typu XSS.

Úpravu dat pro aplikaci můžou provádět jen oprávnění uživatelé, kteří znají uživatelské jméno a heslo. Přihlašovací údaje se zadávají, pokaždé když uživatel hodlá přistoupit do administračního rozhraní. V AdministratorPresenteru je metoda startup(), která se zavolá hned po vytvoření presenteru a kontroluje, zda je uživatel přihlášený, pokud není, tak je přesměrován na přihlašovací stránku, kde zadá přihlašovací jméno a heslo. Po přihlášení je přesměrován požadovanou stránku. V přihlašovacím formuláři je možnost zaškrtnout volbu „Pamatuj si mě“ a poté je přihlášení z daného počítače zapamatováno na 14 dní, nebo dokud se uživatel neodhlásí.

Nette framework do databázových tabulek pro uživatelské údaje neukládá surová hesla, ale pouze jejich otisk. Pro tvorbu otisku hesla se používá hašovací funkce bcrypt, která v sobě obsahuje kryptografickou sůl, což pomáhá ochránit proti útokům hrubou silou.

V administračním rozhraní aplikace bude v sekci pro nastavení umístěn formulář pro změnu přístupového hesla. Konktrétně dvě textové pole, kdy se po jejich vyplnění musí obě hesla shodovat a poté lze heslo administrátora změnit.

4.4.3 Mobilní zařízení

Jako poslední typ klientského zařízení budou chytré telefony a tablety. Aplikace je vytvořena pro operační systém Android. Podpora aplikace bude od API 22 (Android 5.1 Lollipop). S touto verzí systému je mezi uživateli 22.3 %[1] mobilních zařízení, a řadí

[1] Zdroj: https://developer.android.com/about/dashboards/index.html, k datu 28. 7. 2017

se tak jako druhá nejpoužívanější verze Androidu. Verzí s momentálně největší podílem na trhu je verze s označením Marshmallow (verze 6.0), která využívá API 23.

Pro vývoj aplikace bylo zvoleno vývojové prostředí Android Studio. Při vytvoření projektu, aplikace pomůže vývojářovi rozhodnout jaké nejnižší API má zvolit. Vývojové prostředí vytvoří souborovou strukturu, pro uchovávání různých částí projektu. Velmi důležitým souborem je AndroidManifest.xml, ve kterém jsou definovány důležité údaje pro aplikaci, jako jsou například oprávnění, která bude aplikace po systému požadovat. Zápis se provádí za pomoci párových tagů a jejich vlastností. V souboru jsou také zapsány údaje pro aktivity. Pojmem aktivita se v systému Android rozumí práce, která směřuje ke splnění uživatelského cíle. Každá aktivita má svůj název, který se může

```
<activity
            android:name=".OPComConnections"
            android:label="@string/title_activity_opcom_connections"
            android:parentActivityName=".MainActivity">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>
```

Ukázka kódu 5: Popis aktivity v souboru AndroidManifest.xml (Zdroj: vlastní)

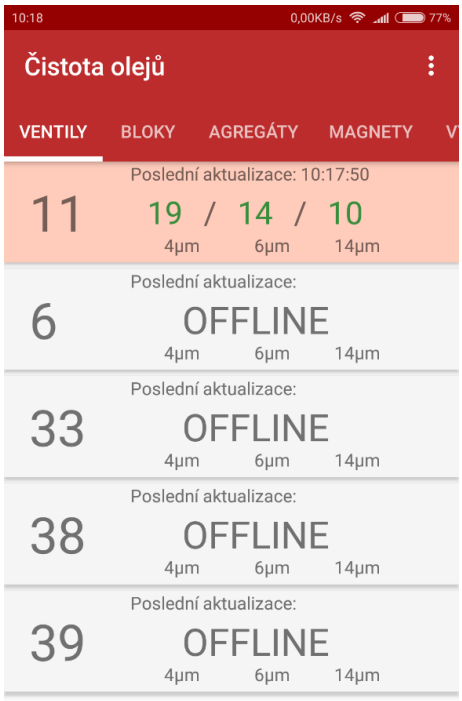
zobrazovat v horní části obrazovky. Lze také nastavit rodičovskou aktivitu, ke které se lze vrátit za pomoci stisku šipky vedle názvu aktivity. Ke každé aktivitě náleží soubor s příponou java, do kterého se zapisují třídy a procedury pro danou aktivitu. Název tohoto souboru musí být zapsán v definici aktivity v souboru AndroidManifest.xml.

Aplikace bude fungovat podobně jako ostatní klientské programy, s tím rozdílem že zde se na jedné obrazovce budou ukazovat všechny snímače z jednoho střediska, mezi středisky se bude moct uživatel pohybovat za pomocí gest (přesouváním prstu po obrazovce), nebo kliknutím na název střediska. Přes kontextové menu se bude moct uživatel dostat na obrazovku pro nastavení aplikace nebo na obrazovku, kde bude moc spravovat seznam sledovaných snímačů. Pro každý OPCom bude, po kliknutí na položku seznamu, k dispozici obrazovka s dalšími detaily a údaji. Aktualizaci údajů bude možné uskutečnit pomocí gesta tažení prstem dolů přes obrazovku. Procedura aktualizace musí běžet v odděleném vlákně, tak aby neblokovala hlavní vlákno obsluhující uživatelské prostředí.

Mobilní aplikaci budou moc uživatelé, zde to budou firemní zaměstnanci, získat přes odkaz umístěný na webové aplikaci. Tento způsob distribuce byl zvolen proto, že se jedná o aplikaci využívající interní data firmy, nicméně data budou dostupná pouze ve firemní síti.

4.4.3.1 Vzhled aplikace

Na hlavní obrazovce se budou zobrazovat údaje z jednotlivých snímačů. Snímače budou v seznamu, kterým bude možné rolovat za pomocí tažení prstem. Pokud bude seznam posunut na svůj začátek a uživatel provede gesto tažení dolů, tak se vyvolá procedura aktualizace dat. Pokud seznam bude prázdný, tak se na jeho místě zobrazí informace o tom, že nejsou nahrána žádná data a pod tímto textem bude tlačítko pro aktualizaci. Pro každé středisko bude samostatná záložka, mezi kterými bude moc uživatel přecházet pomocí gest. Pro tento způsob zobrazení je zvolena komponenta TabLayout.



Obrázek 23: Hlavní obrazovka mobilní aplikace (Zdroj: vlastní)

Jednotlivé seznamy budou vloženy do fragmentů. Právě mezi těmito fragmenty bude komponenta TabLayout přepínat. Každý fragment bude mít svůj view definovaný xml souborem. Ve fragmentech jsou definovány dva pohledy, jeden pro zobrazení seznamu s daty a druhý pohled pro situaci kdy bude seznam prázdný. Pro každý seznam musí být definována třída která je potomkem třídy BaseAdapter. Tato třída zásobuje komponentu ListView daty. BaseAdapter má metodu getView, která dokáže položce seznamu nastavit pohled definovaný v souboru. Ve fragmentu je použita komponenta SwipeRefreshLayout, která zachycuje právě gesto tažení ,sloužící pro aktualizaci dat. Pokud by jakákoliv ze tří sledovaných hodnot překročila stanovenou mez nastavenou ve snímači, tak se danému číslu změní barva na červenou a pozadí položky se také změní na světle červenou.

Pro správu OPComů je určena speciální obrazovka do které se bude uživatel moct dostat přes kontextové menu na hlavní obrazovce. Při otevření se načte z csv souboru seznam uložených údajů pro každý snímač. Budou zobrazeny jako řádky seznamu (komponenta ListView), na které bude moct uživatel kliknout a popřípadě údaje měnit, nebo celý záznam smazat. Soubor pro ukládání seznamu údajů pro komunikaci s OPComy se bude ukládat do vnitřního uložiště telefonu, kde bude je vyhrazené místo právě pro data této aplikace. Při smazání záznamu se nalezne, na jakém řádku v souboru byl, a tento řádek se smaže.

Obrázek 24: Obrazovka pro správu OPComů (Zdroj: vlastní)

I tato aplikace musí mít nějaké parametry, podle kterých se bude rozhodovat. Z těchto parametrů zjistí, jaký protokol bude použit nebo na jakou adresu a port budou zasílány požadavky. V systému Android je pro nastavení aplikací určená speciální obrazovka, do které se vkládají položky nastavení. Pro nastavení je definován speciální xml soubor, do kterého je vložen objekt PreferenceScreen, který je vlastně kontejnerem pro další objekty typu Preference. Například objekt nastavení pro textový řetězec se nazývá EditTextPreference. Každá položka nastavení má svůj klíč, pod kterým se uloží do systému. Android poskytuje rozhraní SharedPreference sloužící pro přístup k hodnotám těchto nastavení za pomoci názvu jeho klíče. Každá položka nastavení musí mít název a stručný popis k čemu toho nastavení slouží. Může také být nastavená výchozí hodnota, která se vyplní předem, v případě že předtím nebude uložené žádné nastavení. V aplikaci pro sledování čistoty oleje bude na této obrazovce nastavení pro IP adresu serveru, port serveru a typ komunikačního protokolu.

Veškeré textové prvky v mobilní aplikaci mohou být vícejazyčné. Místní jazyk se získává z jazyku nastaveného v systému. Ve vývojovém prostředí se na každém prvku se nastaví jméno zdroje textů, ten se nachází v souboru values.xml ve složce values, kde ke každého názvu zdroje je příslušná hodnota. Překlady se získávají tak, že se vytvoří

nová složka pro soubor s názvem „values“ a za pomlčku se napíše kód jazyku. Například pro češtinu je to složka s názvem: „values-cs“. Ve vývojovém prostředí je editor překladů, který vedle sebe zobrazuje hodnoty ve všech jazycích pro který je aplikace vytvořena. Pro každý text se musí nastavit výchozí text, ten se použije, pokud není k dispozici překlad.

5 Shrnutí výsledků

Při implementaci projektu, zejména jeho serverové časti, bylo zapotřebí zvolit správné komunikační protokoly tak, aby při přenášení dat neblokoval komunikaci firemní firewall. Byly tedy zvoleny dva protokoly a to: HTTP a UDP. Oba tyto protokoly mají své výhody i nevýhody. Proto si může každý uživatel klientské aplikace zvolit, na jakém protokolu bude se serverem komunikovat.

Největší nevýhodou nasazení protokolu UDP je to, že pokud se serverem již komunikuje jeden klient, tak musí další klient čekat, než bude server schopen odpovědět. Nicméně komunikace nikdy netrvá tak dlouho, aby byl port serveru pro naslouchání zbytečně dlouho blokovaný. Za nevýhodu komunikace přes UDP protokol by se dalo považovat to, že je nutné mít program, který dokáže přes specifický port odesílat příkazy a následně přijímat odpovědi.

Pro HTPP protokol musel být do aplikace nasazen také webový server komunikující právě přes tento protokol. Na rozdíl od UDP protokolu, zde není nutné mít speciální program pro komunikaci se serverem. O data ze serveru si lze požádat například z jakéhokoliv internetového prohlížeče, a to tak že do pole pro URL adresu se zadá adresa serveru a vyplní parametr, který poté webový server vyhodnotí a osloví požadované snímače. Webový server odpovídá na požadavek, tak že klientovi zašle csv soubor, ve kterém jsou na každém řádku odpovědi ze snímačů.

Klientská část projektu zahrnovala desktopovou, mobilní a webovou aplikaci. Při implementaci desktopové aplikace byl použit programovací jazyk Java, tím pádem se stal program nezávislý na platformě, na které je spuštěný. Jedinou podmínkou pro běh programu je nainstalované prostředí pro běh aplikací napsaných v Javě. Podobně je na tom webová aplikace, ke které uživatelé přistupují přes internetový prohlížeč, a tím pádem také není důležité, jaký má uživatel operační systém. Mobilní aplikace byla vyvinuta pouze pro jeden typ operačního systému, a to pro operační systém Android. Klientské aplikace představují jednouchý způsob jak se dostat k datům ze snímačů čistoty hydraulických kapalin.

6 Závěr

Hlavním cílem projektu bylo zajistit firmě ARGO-HYTOS Vrchlabí takový systém pro sledování a zobrazování čistoty hydraulických kapalin na zkušebních stavech, který by dokázal plnohodnotně nahradit stávající řešení a vylepšit ho o různé funkce. Předchozí řešení s použitím vývojového prostředí Control Web bylo velmi nepraktické z několika pohledů. Aplikace naprogramované v tomto prostředí nepodporovaly vícevláknové zpracování, nebo nebylo možné jednoduše přidávat další snímače, které by chtěla firma nasadit. Proto byl vybrán pro vývoj nové aplikace programovací jazyk Java, který dokáže tyto problémy spolehlivě řešit.

V projektu byl implementován model klient-server, kdy serverem byla aplikace napsaná v programovacím jazyce Java, na které byly spuštěny dvě služby. První službou byl UDP server, to znamená, že aplikace naslouchala na specifickém portu pro požadavky a na ty poté náležitě odpovídala. Druhá služba, která byla na serveru spuštěna, byl webový HTTP server. Webový server naslouchal požadavkům na specifické URL adrese. Serverová aplikace také odesílala požadavky na jednotlivé snímače a poté od nich přijímala odpovědi. Všechny odpovědi od snímačů poté server odesílal klientským aplikacím.

Jako klientské aplikace byly naprogramovány tři programy, každý program mohl běžet na jiné platformě (desktop, web nebo mobilní zařízení). Největší výhodou klientských aplikací je to, že nejsou vázány (krom mobilní části) na hostitelský operační systém. Klientské aplikace splnily hlavní požadavek zadání, a to že na nich lze libovolně přidávat další sledované snímače. Jako další rozvoj projektu by se dalo uvažovat vytvoření aplikace pro jiné operační systémy běžící na mobilních zařízeních (např. Windows nebo iOS).

7 Citovaná literatura

[1] **ARGO-HYTOS.** O nás. *ARGO-HYTOS.* [Online] http://www.argo-hytos.com/cz/o-spolecnosti/o-nas.html.

[2] **Kolektiv techniků ARGO-HYTOS Vrchlabí.** *Čistota hydralických kapalin.* 2007.

[3] **Moravské přístroje, a.s.** Co je Control Web? *Moravské přístroje.* [Online] http://www.mii.cz/art?id=380&cat=146&lang=405.

[4] **CHIYU TECHNOLOGY CO., LTD.** *BF-440/480, User's guide.* 2005.

[5] **Wikipedia.** RS-232. *Wikipedia.* [Online] https://cs.wikipedia.org/w/index.php?title=RS-232&oldid=13132133.

[6] **Pavel, Tišnovský.** Komunikace pomocí sériového portu RS-232C. *Root.cz.* [Online] https://www.root.cz/clanky/komunikace-pomoci-serioveho-portu-rs-232c/.

[7] **ARGO-HYTOS.** OPCom II. *ARGO-HYTOS.* [Online] http://www.argo-hytos.com/cz/vyrobky/snimace-a-merici-technika/stacionarni-snimace-znecisteni/opcom-ii.html.

[8] —. *Příručka OPCom II.*

[9] —. *Technical Handbook, Solutions for clean oil.*

[10] **GIGA PC.** Tenký klient . *GIGA PC.* [Online]

[11] **HP.** t620 HP Thin Clients . *HP® Official Site.* [Online] http://www8.hp.com/us/en/thin-clients/t620.html.

[12] **TIOBE.** TIOBE Index. *TIOBE - The Software Quality Company.* [Online] [Citace: 13. 07 2017.] https://tiobe.com/tiobe-index/.

[13] **Spurná, Ivona.** *Počítačové sítě, Praktická příručka správce sítě.* Kralice na Hané : Computer Media s.r.o., 2010. 978-80-7402-036-0.

[14] **Thakur, Dines.** User Datagram Protocol (UDP). *ComputerNotes.* [Online] 06. 08 2017. http://ecomputernotes.com/computernetworkingnotes/routing/user-datagram-protocol.

[15] **W3schools.com.** HTTP Methods GET vs POST. *W3schools.com.* [Online] [Citace: 12 . 07 2017.] https://www.w3schools.com/tags/ref_httpmethods.asp .

[16] **Netmarketshare.com.** Operating system market share. *Netmarketshare.com.* [Online] https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qpsp=2017&qpnp=1&qptimeframe=Y.

[17] **Android Developers.** Application Fundamentals. *Android Developers.* [Online] [Citace: 15. 07 2017.] https://developer.android.com/guide/components/fundamentals.html.

[18] **Adobe.** Co jsou to webové aplikace a dynamické webové stránky? *Adobe.com.* [Online] [Citace: 7. 15 2017.] https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html.

[19] **The PHP Group.** PHP: What can PHP do? *php.net.* [Online] [Citace: 13. 07 2017.] http://php.net/manual/en/intro-whatcando.php.

[20] **Nette Foundation.** Rychlý a pohodlný vývoj webových aplikací v PHP. *Nette Framework.* [Online] [Citace: 15. 07 2017.] https://nette.org/cs/.

[21] —. Zabezpečení před zranitelnostmi . *Nette Framework.* [Online] [Citace: 15. 07 2017.] https://doc.nette.org/cs/2.4/vulnerability-protection.

[22] **Jak psát PHP?** Architektura MVC. *Jak psát PHP?* [Online] [Citace: 15. 07 2017.] http://jakpsatphp.cz/MVC/.

[23] **Wikipedia.** Vývojové prostředí. *Wikipedia.* [Online] 01. 05 2017. [Citace: 17. 07 2017.] https://cs.wikipedia.org/w/index.php?title=V%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD&oldid=14948570.

[24] **NetBeans.** NetBeans IDE - Overview. *NetBeans IDE.* [Online] [Citace: 17. 07 2017.] https://netbeans.org/features/index.html.

[25] **Android Studio.** Android Studio Features . *Android Studio.* [Online] [Citace: 17. 07 2017.] https://developer.android.com/studio/features.html.

[26] **Oracle .** Future. *Java Platform SE 7 .* [Online] [Citace: 07. 24 2017.] https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Future.html.

[27] **Zeeshan, Afzaal Ahmad.** Programming in Java using the MVC architecture. *CodeProject.* [Online] 24. 02 2015. [Citace: 16. 07 2017.] https://www.codeproject.com/Articles/879896/Programming-in-Java-using-the-MVC-architecture.

[28] **ORACLE.** Defining and Starting a Thread . *The Java Tutorials.* [Online] [Citace: 24. 07 2017.] https://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html.

[29] **Moravské přístroje, a.s.** Control Web vývojová verze. *Moravské Přístroje.* [Online] [Citace: 06. 08 2017.] http://www.mii.cz/art?id=71&lang=405.

[30] **Grid Connect.** Ethernet TCP / IP controller with 4 Serial Ports - RS232, RS422, RS485. *Grid Connect.* [Online] 06. 08 2017. https://gridconnect.com/ethernet-tcp-ip-controller-with-4-serial-ports-rs232-rs422-rs485.html#.

[31] **Sharpened Productions .** Client-Server Model Definition. *TechTerms.* [Online] 06. 08 2017. https://techterms.com/definition/client-server_model.

[32] **Google.** Android Developers. *Brand guidelines .* [Online] [Citace: 06. 08 2017.] https://developer.android.com/distribute/marketing-tools/brand-guidelines.html.

8 Seznam obrázků

Obrázek 1: Control Web 6 (29) 4
Obrázek 2:Vývojové prostředí se současnou aplikací pro sledování čistoty oleje (Zdroj: vlastní) 5
Obrázek 3: Převodník BF-440 (30) 6
Obrázek 4:Stacionární snímač znečištění OPCom II (7) 8
Obrázek 5: Způsob měření počtu pevných částic za pomoci laseru (8) 8
Obrázek 6: Architektura klient-server (31) 11
Obrázek 7: Tenký klient od společnosti HP (11) 12
Obrázek 8: Formát UDP datagramu (14) 14
Obrázek 9: URL adresa ve webovém prohlížeči (Zdroj: vlastní) 15
Obrázek 10: Logo Android (32) 17
Obrázek 11: MVC architektura (27) 19
Obrázek 12:Příklad šablonovacího systému Latte (Zdroj: vlastní) 20
Obrázek 13: Prostředí NetBens IDE (Zdroj: vlastní) 23
Obrázek 14:IDE Android Studio (Zdroj: vlastní) 23
Obrázek 15:Android Studio s grafickým návrhem aplikace (Zdroj: vlastní) 24
Obrázek 16: Systém Android spuštěný v emulátoru (Zdroj: vlastní) 25
Obrázek 17: Struktura aplikace (Zdroj: vlastní)serveru 26
Obrázek 18: Uživatelské prostředí serveru (Zdroj: vlastní) 28
Obrázek 19: Uživatelské prostředí klientské desktopové aplikace (Zdroj: vlastní) 35
Obrázek 20: Struktura webové aplikace (Zdroj: vlastní) 36
Obrázek 21: Hlavní stránka webové aplikace (Zdroj: vlastní) 37
Obrázek 22: Tabulky v databázi (Zdroj: vlastní) 37
Obrázek 23: Hlavní obrazovka mobilní aplikace (Zdroj: vlastní) 40
Obrázek 24: Obrazovka pro správu OPComů (Zdroj: vlastní) 41

9 Seznam tabulek

Tabulka 1:Příslušné počty částic ke kódům normy ISO 4406 (9) 10

10 Seznam ukázek kódu

Ukázka kódu 1: Příklad implementace rozhraní Runnable (28) 30
Ukázka kódu 2: Příklad dědění z třídy Thread (28) 31
Ukázka kódu 3: Použití ExecutorService v projektu (Zdroj: vlastní) 32
Ukázka kódu 4: Konstruktor a metody HTTP serveru (Zdroj: vlastní) 33
Ukázka kódu 5: Popis aktivity v souboru AndroidManifest.xml (Zdroj: vlastní) 39

11 Seznam příloh

1) Zdrojové kódy aplikací na CD

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Trýzna Lukáš	Českých bratří 974, Vrchlabí	I14153

TÉMA ČESKY:

Sběr dat ze senzorů čistoty oleje a jejich vizualizace

TÉMA ANGLICKY:

Collecting data from the sensors of oil cleanliness and their visualization

VEDOUCÍ PRÁCE:

doc. Ing. Filip Malý, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem je navrhnout softwarové řešení pro monitoring čistoty hydraulického oleje na zkušebních stavech, a to včetně implementace klienta pro PC a chytrý telefon.

Osnova:
1. Úvod
2. Současné řešení
3. Návrh nového řešení
4. Analýza a implementace
5. Závěr
6. Literatura

SEZNAM DOPORUČENÉ LITERATURY:

Google Scholar

Podpis studenta: Trýzna ..... **Datum:** 17.10.2016

Podpis vedoucího práce: ..... **Datum:** 17.10.2016