FAKULTA MATEMATIKY FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO

Návrh pre projekt Vizualizácia údajov z meracieho zariadenia

Projekt na predmet Tvorba Informačných Systémov

Špecifikácia požiadaviek

zimný semester 2016/2017

Vedúci projektu: Andrej Jursa

Členovia: Jana Harvanová

Samuel Wendl

Michal Pandula

Sabína Fačkovcová

Obsah

1 Úvod

Tento dokument obsahuje celkový návrh pre náš projekt Meracie Zariadenia.

V prvej kapitole sa nachádza analýza technológií, nad ktorými sme pri implementácií lokálnej aplikácie a webu rozmýšľali

Druhá kapitola obsahuje diagramy, ktoré sme navrhli k nášmu projektu. Konkrétne to je Entitno-relačný, Use-case, stavový a sekvenčný diagram.

Následne v tretej kapitole je grafické rozhranie pre lokálnu aplikáciu a dizajn našej webovej stránky.

V štvrtej kapitole je návrh dátového modelu, dekompozície a triedneho diagramu.

A ako posledné v piatej kapitole sa nachádzajú testovacie scenáre, ktoré by bolo dobré otestovať.

2 Analýza technológií

2.1 Lokálna aplikácia

Voľba vhodného jazyka, ktorý by nám čo najefektívnejšie a najlepšie umožňoval implementáciu lokálnej časti aplikácie, nebola úplne triviálna.
Zvoliť C/C++ ? Javu ? C# ?

- Java je v dnešnom svete jeden z najrozšírenejších programovacích jazykov. Rozmýšľali sme, že to spravíme v nej, avšak po dlhšom zisťovaní informácii sme zistili, že v Jave sa veľmi zle robí práve so sériovými portmi - čo je jedna z kľúčových častí implementácie tejto časti aplikácie. Veľká väčšina frameworkov a knižníc, ktoré dokážu pracovať so sériovým portov, sú platené. Našli sme aj nejaké zadarmo, avšak u každej sa vyskytol nejaký problém. Jedna z nich bola napríklad Java Communication API (JavaComm). Tá však už oficiálne nie je ďalej podporovaná a vyvíjaná pre Windows, iba pre Solaris SPARC, Solaris x86 a Linux x86. To bol problém, keďže sme potrebovali, aby softvér bežal na Windowse. Našli sme inú knižnicu - RXTX, ktorá sa dá tiež zadarmo použiť, avšak tá zase nie je úplne vhodná pre niektoré Baud rate hodnoty. Našli sme veľa ľudí na mnohých fórach, ktorý mali problém s touto knižnicou práve kvôli tomu, program im pri jednoduchej komunikácii buď komunikoval zlé hodnoty alebo hádzal chyby. Nechceli sme sa zaoberať podobnými problémami, keďže očividne by to nešlo priveľmi hladko. Java teda v tomto prípade vyšla ako nevýhodná.

- Čo C/C++ ? To by mohla byť skvelá možnosť. Mohli by sme využiť všetky sily C++ od jeho efektívnosti cez veľkú kontrolu nad pamäťou. Existovali aj knižnice so sériovými portmi, ktoré nevyzerali na pohľad tak komplikovane a "pokazene" ako (zadarmo) knižnice predošlej javy. Nastal však iný problém - bolo by potrebné hľadať ďalšie riešenie iného problému - dynamické listovanie portov na zariadení, čo je nevyhnutná požiadavka. Na to by bolo potrebné doinštalovať balíky, pridať ďalšie knižnice. Dúfať, že to prejde v poriadku a nenarazíme na zbytočné, zdržujúce problémy počas tohto procesu - ešte pred tým než, vlastne začneme niečo relevantné robiť a čím dosiahneme výsledky. Musí existovať niečo menej komplikované...
- Bližšie sme sa pozreli na C# a .NET framework. Zistili sme, že oproti Jave sa v ňom oveľa lepšie robí práve so sériovými portmi a pristupuje k ďalším iným systémovým vlastnostiam daného počítača, kde bol zase problém s C++. Súčasťou .NET je totiž namespace System.Management, ktorý poskytuje prístup k množstvu informácii, správam, udalostiam systému, zariadeniam a aplikáciám registrovaných v infraštruktúre Windows Management Instrumentation (WMI). Aplikácie a servisy sa vedia systému pýtať na zaujímavé informácie (množstvo voľného miesta na disku, aktuálne využitie procesoru, ku ktorej databáze je pripojená určitá aplikácia a pod), pomocou triedy oddedenej z ManagementObjectSearcher a ManagementQuery, alebo zachytávať udalosti správy pomocou ManagementEventWatcher triedy. Okrem toho má .NET samozrejme zabudovanú prácu so sériovým portom, ktorá je na prvý pohľad intuitívna, podľa diskusných fór vcelku dobre funguje a samozrejme, keďže C# je od Microsoftu, pod Windowsom funguje presne ako má, čo potrebujeme. Keďže zo zadania projektu nepotrebujeme softvér vyvíjať aj na Linuxy, požiadavka bola len pre Windows, prakticky neexistuje dôvod, prečo po tomto všetkom nevybrať C#.

2.2 Komunikácia

2.2.1 Dynamické selektovanie sériového portu

Aplikácia vie dynamicky selektovať sériové porty na danom zariadení. Využíva na to už hore spomínaný namespace System.Management, resp. WMI triedu zvanú Win32_PnPEntinty, ktorá v sebe uchováva vlastnosti všetkých prítomných Plug and Play zariadení. Pomocou tejto triedy aplikácia pri spustení

robí query a vyberá všetky sériové porty, ktoré sú v zariadení zaevidované. Z tých si používateľ vie v grafickom rozhraní vybrať a pripájať sa na ne.

2.2.2 Šifrovanie

Prenos komunikácie je šifrovaný pomocou SSL na sprostredkovanie bezpečnej komunikácie ako s aplikáciou a databázou, tak aj aplikáciou a webom.

2.3 Web

Na realizáciu webovej aplikácie sme vyberali medzi viacerými jazykmi, medzi nimi Python, PHP, Ruby a Javascript.

- Python sme zvažovali kvôli obrovskému množstvu knižníc či už na vývoj backendu (napr. framework Django), ale aj na vykresľovanie dát (Plotly library).
- Ruby (konkrétne framework Ruby on Rails) pripadal do úvahy kvôli rovnakému dôvodu a okrem toho kvôli jeho rýchlosti a prehľadnosti a hlavne kvôli multiplatformovosti.
- Po dlhšej úvahe sme prišli na to, že najvhodnejší by bol jazyk PHP kvôli jeho kompatibilite s väčšinou serverov, natívnej podpore databázových systémov a veľkej používateľskej komunite čo ma za následok jednoduchšie riešenie prípadných problémov. De facto je to jeden z najpoužívanejších jazykov na tvorbu dynamických webov. Doplnený javascriptovou knižnicou na zobrazovanie grafov FusionCharts. FusionCharts bol praktickou voľbou kvôli množstvu šablón grafov, ktoré sú na výber a aj kvôli kompatibilite - funguje aj v IE6.

Webová aplikácia bude teda naprogramovaná v jazyku PHP 7.0 s použitím Bootstrapu a JavaScriptu (FusionCharts) na zobrazovanie grafov s RealTime zobrazovaním dát pretože:

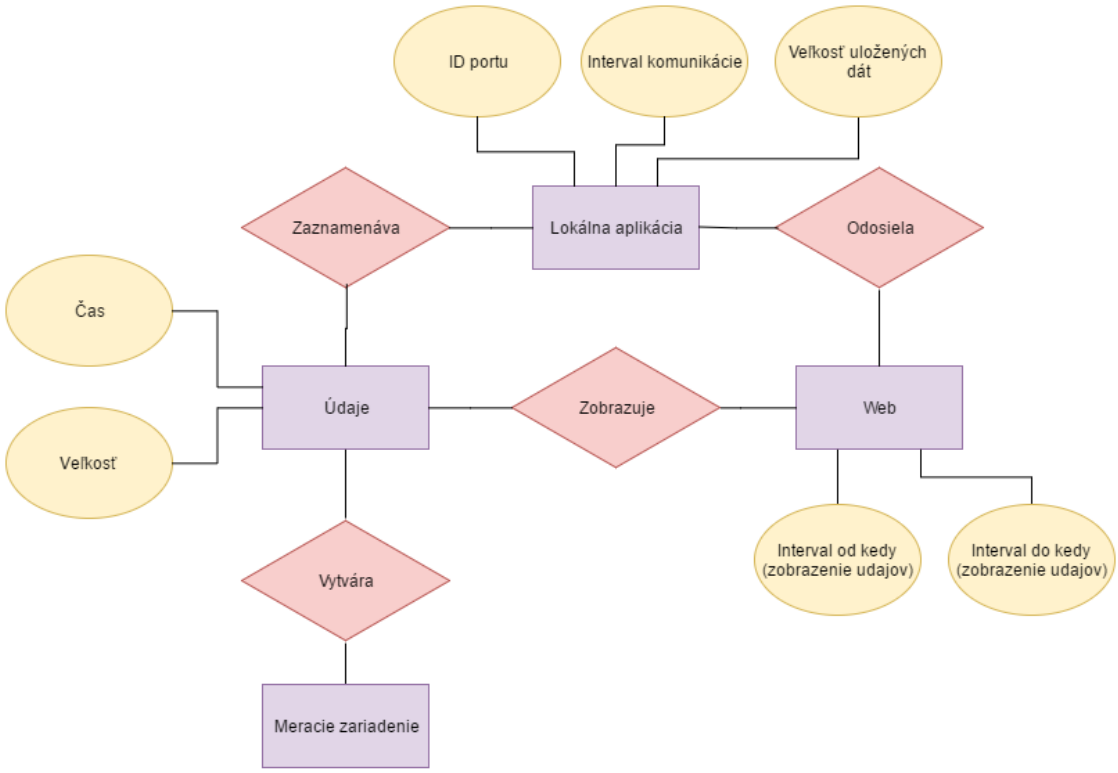
a) Stránka by mala byť dynamická a teda nestačí len čisté HTML a CSS.
b) Stránka by mala byť responzívna a teda mala by sa vhodne zobrazovať na ľubovoľnom zariadení s ľubovoľným rozlíšením.

Stránka by sa mala aj bez opätovného načítania vedieť aktualizovať.

3 Diagramy

3.1 Entitno relačný diagram

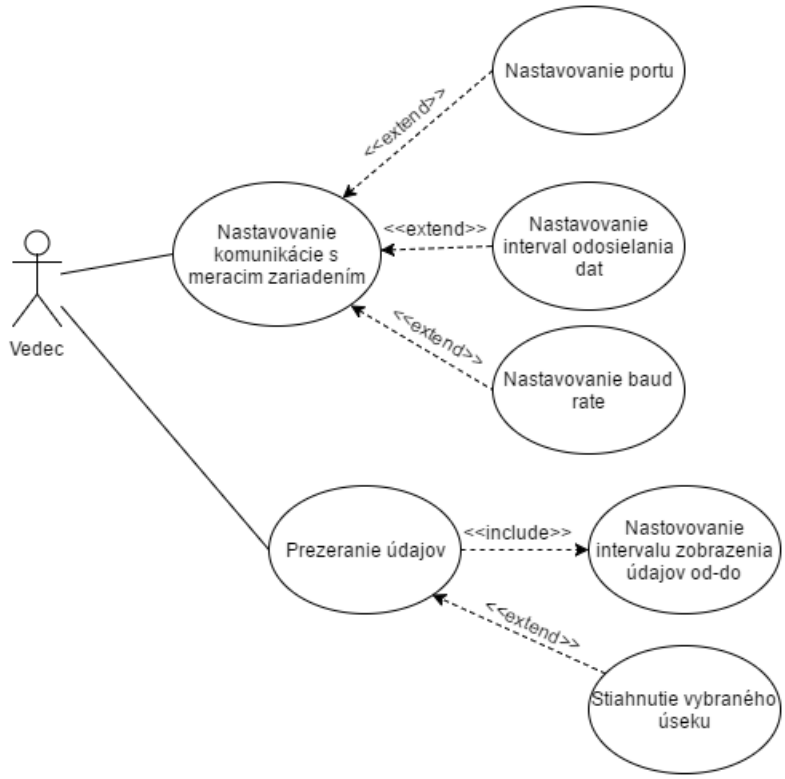
Obrázok 3.1.1 je diagram, kde máme zaznačené základné entity. Z meracieho zariadenia získavame údaje, ktoré majú svoje hodnoty (čas, veľkosť, číslo sondy). Údaje následne zaznamenáva lokálna aplikácia, ktorá je nastaviteľná. Následne sú dáta odosielane na webovú stránku, ktorá ich zobrazuje.



Obrázok 3.1.1

3.2 Use-case diagram

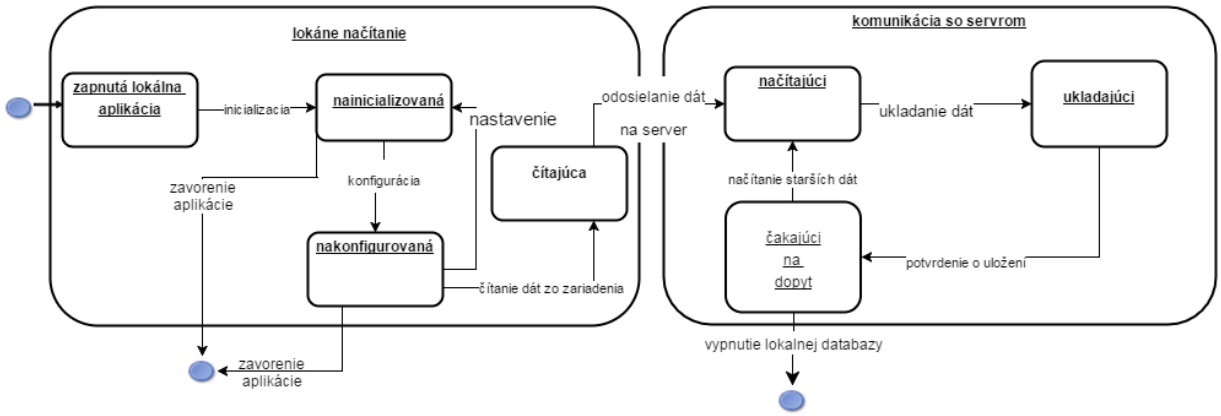
Na obrázku 3.2.1 je diagram, kde máme ukázané aké možnosti interakcie má používateľ aplikácie. Možné bude nastavovanie lokálnej aplikácie kde môžeme meniť parametre merania. Na webovej stránke bude mať možnosť prezerať si namerane údaje, pričom je nutné zadať v akom intervale budú vyžiadané. Tieto údaje sa dajú meniť aj stiahnuť.



Obrázok 3.2.1

3.3 Stavový diagram

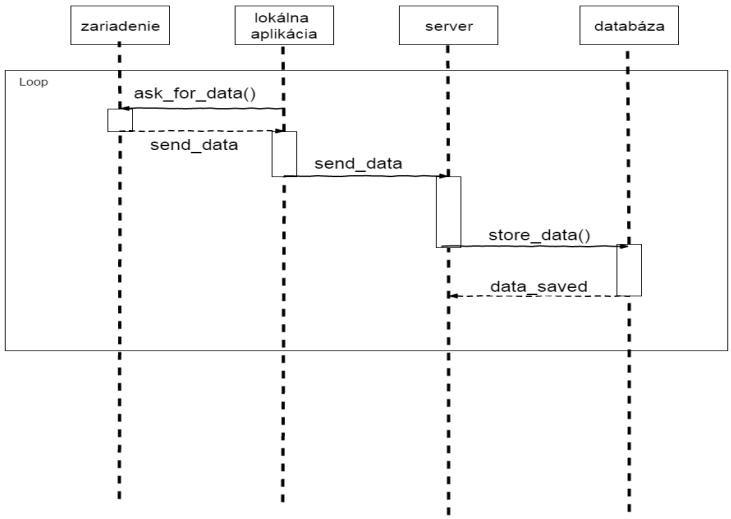
Obrázok 3.3.1 zobrazuje stavový diagram, kde je vidno, že po zapnutí lokálnej aplikácie sa začne automaticky inicializovať a po skončení inicializácie zobrazí okno s možnosťou nakonfigurovať jej správanie. Po skončení konfigurácie začne čítať dáta zo zariadenia, konvertovať ich a následne ich ukladať do lokálnej databázy. Odtiaľ sa webová aplikácia v určenom časovom intervale na dáta dopytuje a následne ich vykresľuje na web.



Obrázok 3.3.1

3.4 Sekvenčný diagram

Na obrázku 3.4.1 vidno sekvenčný diagram, kde na začiatku sa aplikácia cez sériový port pýta na dáta a tá jej odpovie odoslaním dát vo svojom formáte. Lokálna aplikácia dáta spracuje a pošle na lokálny server, ktorý dáta uloží v databáze a tá v prípade úspešného uloženia posiela serveru potvrdzovaciu správu



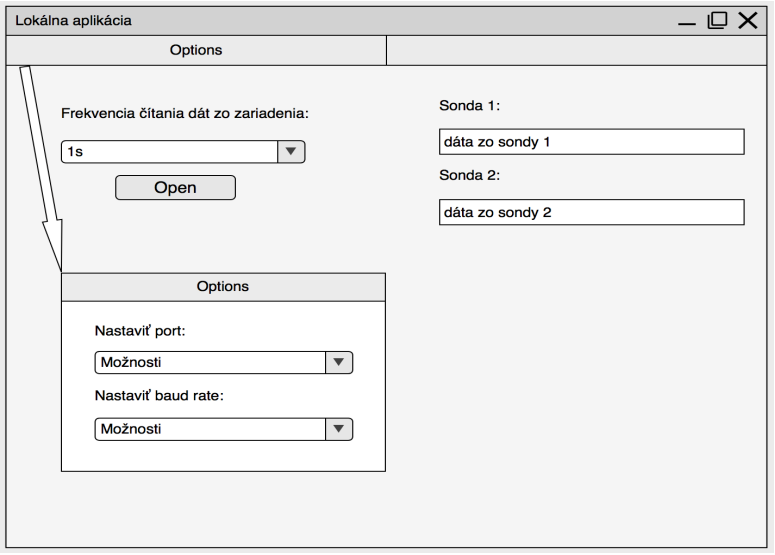
Obrázok 3.4.1

4 Používateľské rozhranie

4.1 Lokálna aplikácia

Ako vidno na obrázku 4.1.1, tak po spustení našej lokálnej aplikácie si sa zobrazí okno, kde si užívateľ bude môcť nastaviť frekvenciu čítania dát zo zariadenia a bude môcť vidieť osobitne dáta zo sondy 1 a zo sondy 2.

Možnosť Options umožní užívateľovi nastaviť ďalšie parametre ako je možnosť nastavenia portu a baud rate.



Obrázok 4.1.1

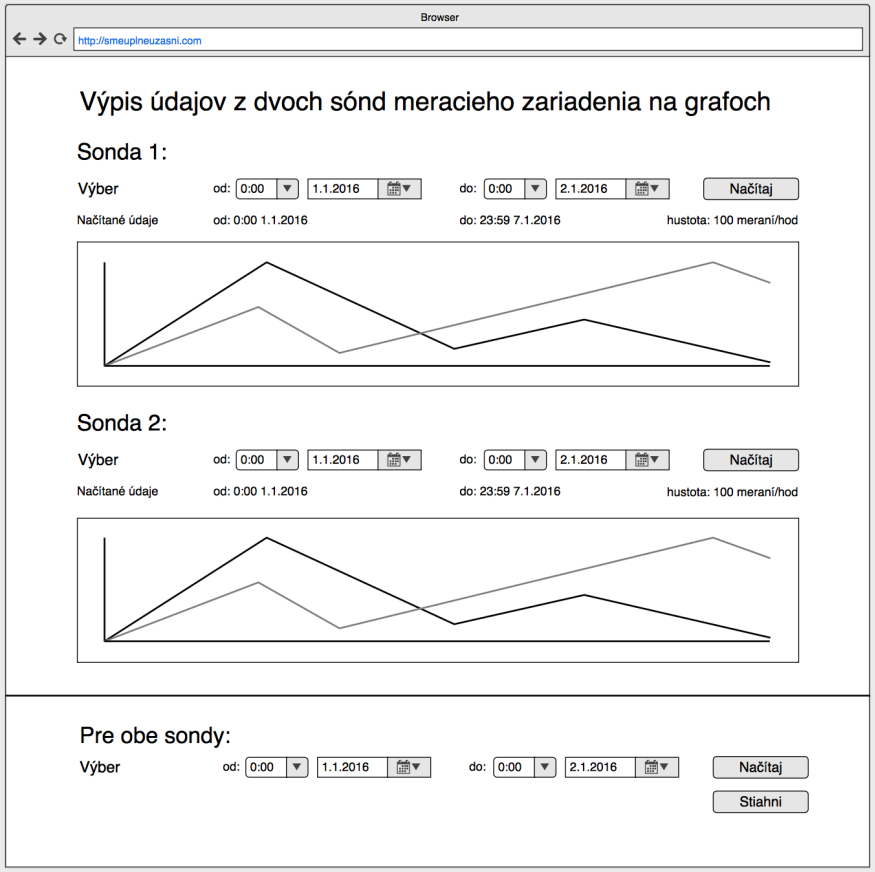
4.2 Web

Obrázok 4.2.1 zobrazuje dizajn našej stránky. Táto webová stránka bude bez prihlásenia zobrazovať údaje na grafoch.

V prvej časti stránky budú zobrazené 2 grafy – jeden pre sondu 1 a jeden pre sondu 2. Nad každým grafom bude vypísaný dátum od akého času a dátumu do akého času a dátumu sú práve teraz zobrazené údaje na grafe.

Dôležitou súčasťou je možnosť vybrať si časové pásmo zobrazovaných údajov pre jednu aj druhú sondu osobitne.

Druhá časť stránky obsahuje možnosť výberu časového pásma pre obe sondy na načítanie a taktiež výber časového pásma údajov, ktoré si chceme stiahnuť.



Obrázok 4.2.1

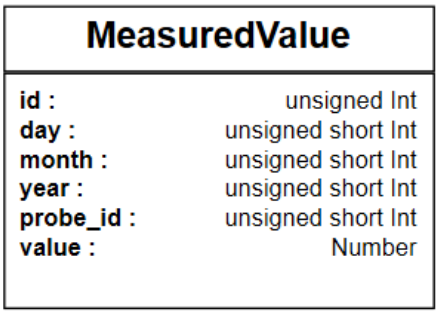
5 Dátový model, dekompozícia a triedny diagram

5.1 Dátový model

Obrázok 5.1.1 je dátový model, ktorý zobrazuje všetky parametre, ktoré bude obsahovať výsledná tabuľka MeasuredValue.

Každý údaj bude mať priradené unikátne ID, dátum a čas, kedy bola daná hodnota nameraná.

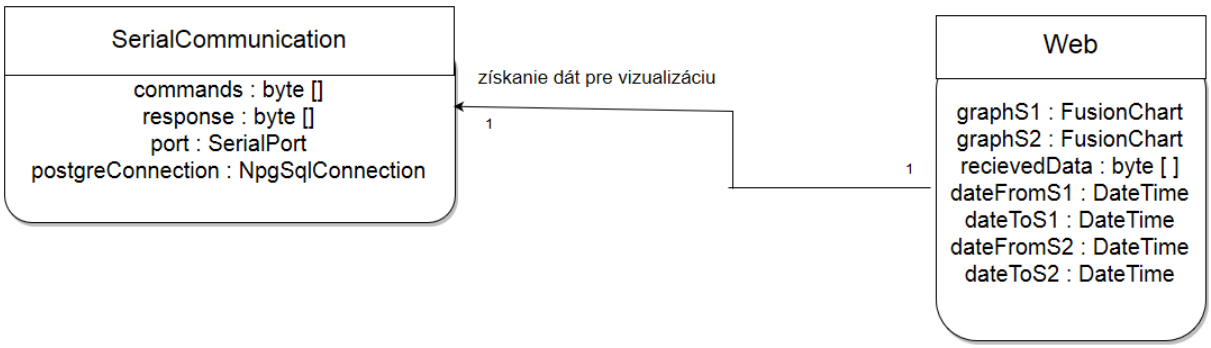
probe_id je hodnota 1, ak bol údaj nameraný zo sondy1, a hodnota 2, ak bol nameraný zo sondy2.



Obrázok 5.1.1

5.2 Dekompozícia

V diagrame dekompozície na obrázku 5.2.1 je vidieť dve hlavné časti programu, t.j 1) Sériová komunikácia so zariadením plus s pripojením na databázu, a 2) Webová, ktorá pre nastavené dva intervaly pre dva grafy vie získavať údaje a zobrazovať ich.

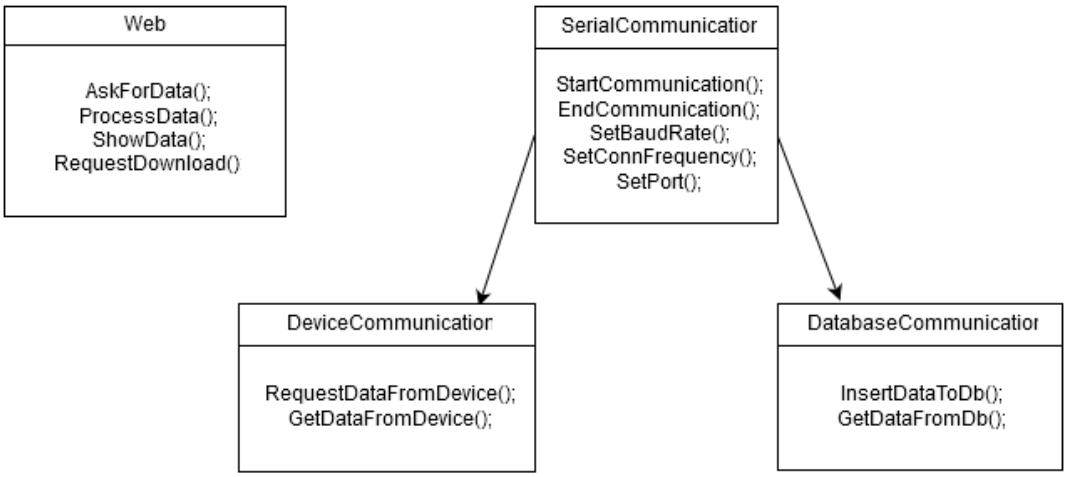


Obrázok 5.2.1

5.3 Triedny diagram

Triedny diagram na obrázku 5.3.1 popisuje všetky hlavné triedy našej aplikácie, prepojenia medzi nimi a náhľad, ako fungujú. Funkčnosť lokálnej aplikácie (sériovej komunikácie) sa dá reprezentovať ako časť 1) - komunikácia so zariadením s danými funkciami, a 2) - komunikácia s databázou, vkladanie údajov.

Web je samotný celok, ktorý sa vie pýtať na dáta, spracovať ich a zobraziť na grafe.



Obrázok 5.3.1

6 Testovacie scenáre

6.1 Komunikácia so zariadením

1) Vstup: Volanie metódy na konkrétnom zariadení, či existujú voľné sériové porty.
Výstup: Ak sa našiel aspoň jeden voľný port, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

2) Vstup: Názov sériového portu, baud rate. Pokus o otvorenie sériového portu.
Výstup: Ak sa podarilo otvorenie portu, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

3) Vstup: Pokyn na dopyt dát zo zariadenia, s otvoreným portom z predošlého testu.
Výstup: Ak sa podaril dopyt a nejaké dáta boli prijaté, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

4) Vstup: Kontrola dát prijatých zo zariadenia z predošlého testu.
Výstup: Ak sú dáta v správnom formáte, tj. dostaneme double hodnotu pre sondu 1, double hodnotu pre sondu 2 a dátum s timestampom, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

6.2 Komunikácia s lokálnym serverom

1) Vstup: Pokus o spojenie s databázou.
Výstup: Ak sa podarilo naviazať spojenie, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

2) Vstup: Pokus o vloženie dát do databázy.
Výstup: Ak sa vloženie podarilo bez chyby, test prešiel úspešne, inak nie.
Stav testovania: Treba otestovať.

6.3 Web

1) Vstup: Získanie dát z databázy cez SELECT dopyt.
Výstup: JavaScript graf s načítanými hodnotami, konkrétne maximami v danom intervale
Stav testovania: Treba otestovať.

2) Vstup: Snaha o SQL Injection.
Výstup: SQL dopyt neprebehne a graf sa znova nenačíta.
Stav testovania: Treba otestovať.

3) Vstup: Použitie tlačidla Stiahni.
Výstup: CSV súbor so správnymi údajmi.
Stav testovania: Treba otestovať.

4) Vstup: Zmena časového intervalu dát vykreslených na grafe.
Výstup: Graf sa prekreslí a budú v ňom správne zmenené údaje.
Stav testovania: Treba otestovať.