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

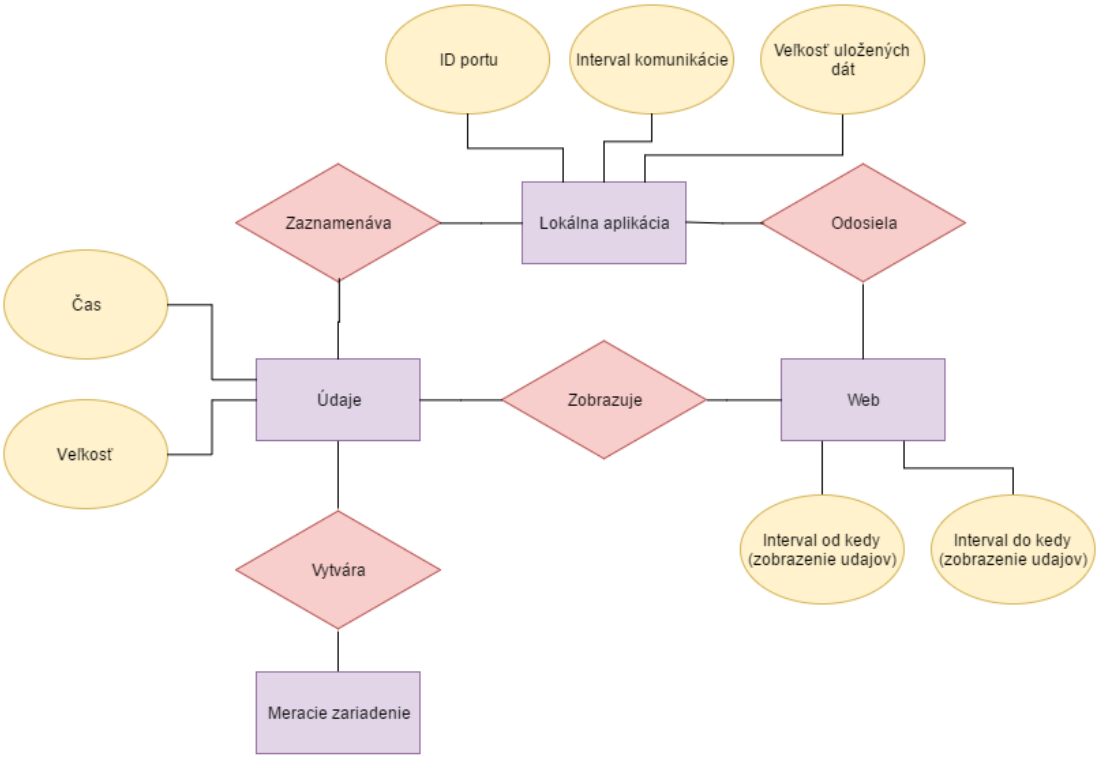
Michal Pandula

Sabína Fačkovcová

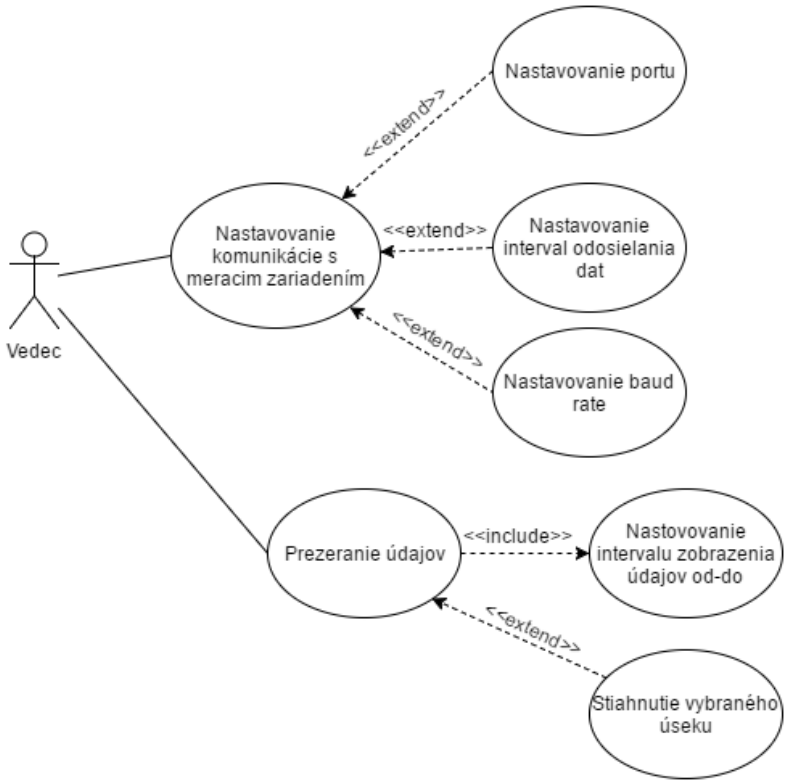
Obsah

1 Diagramy

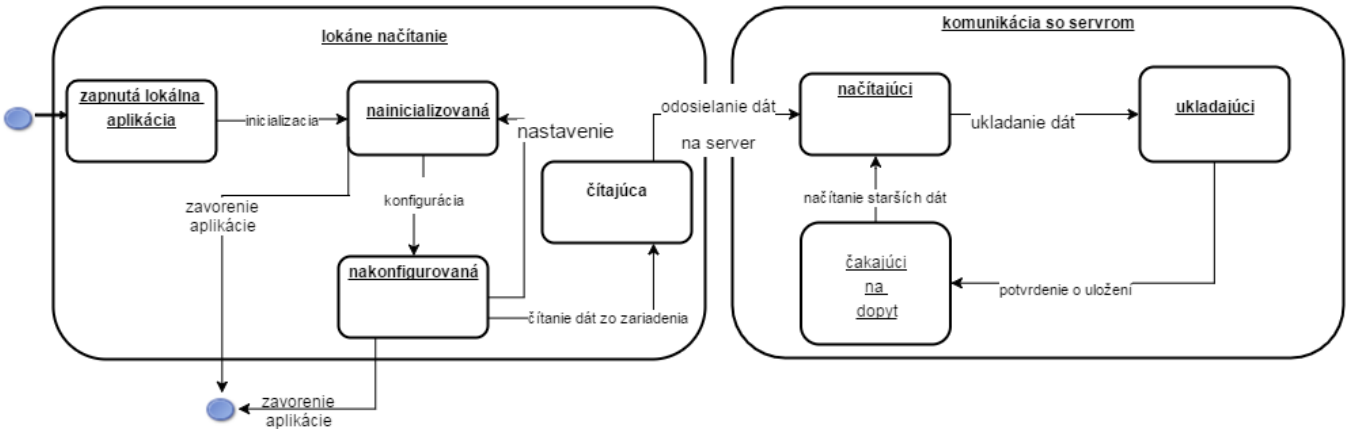
1.1 Entitno relačný diagram



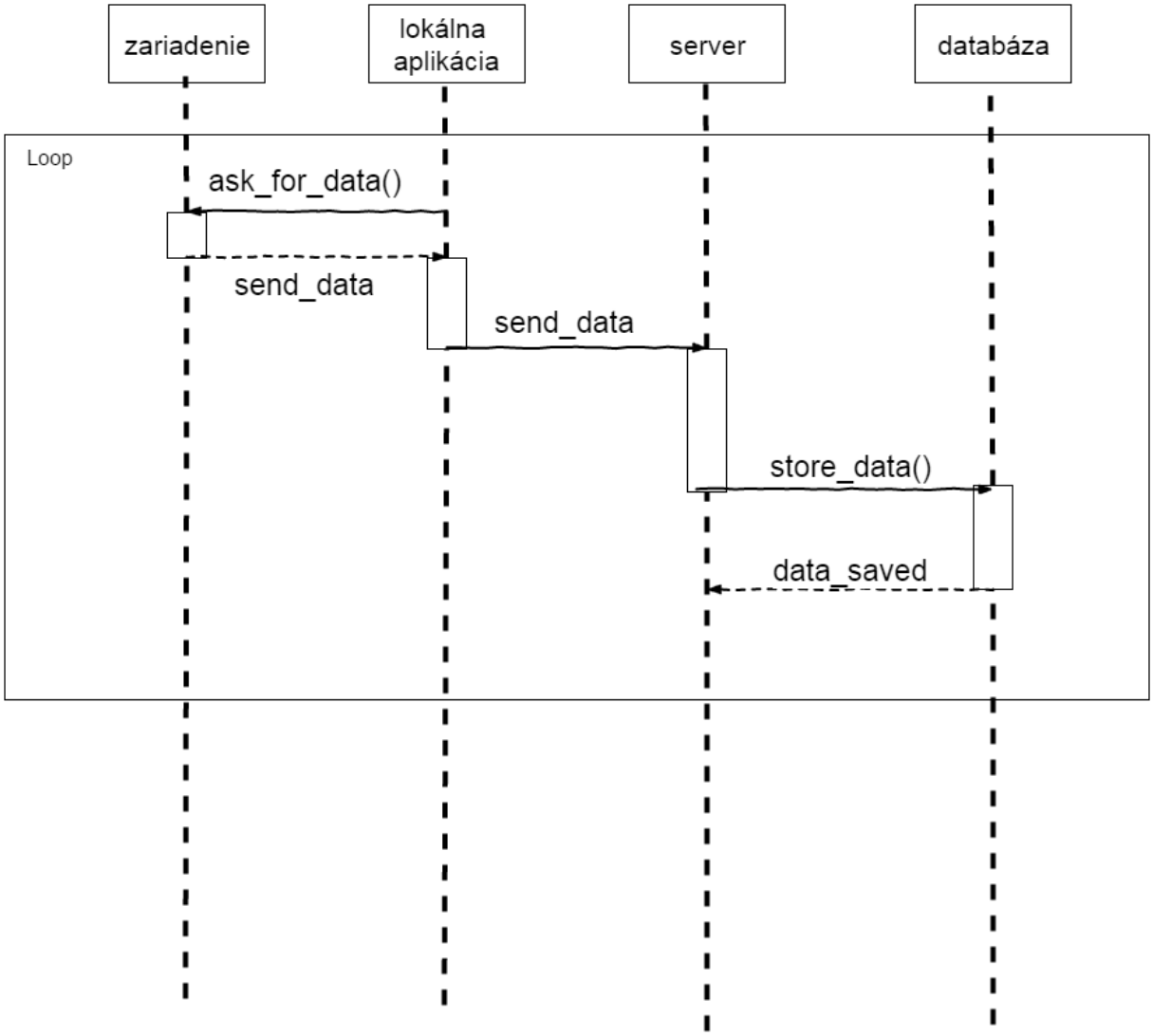
1.2 Use-case diagram



1.3 Stavový diagram



1.4 Sekvenčný diagram



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:

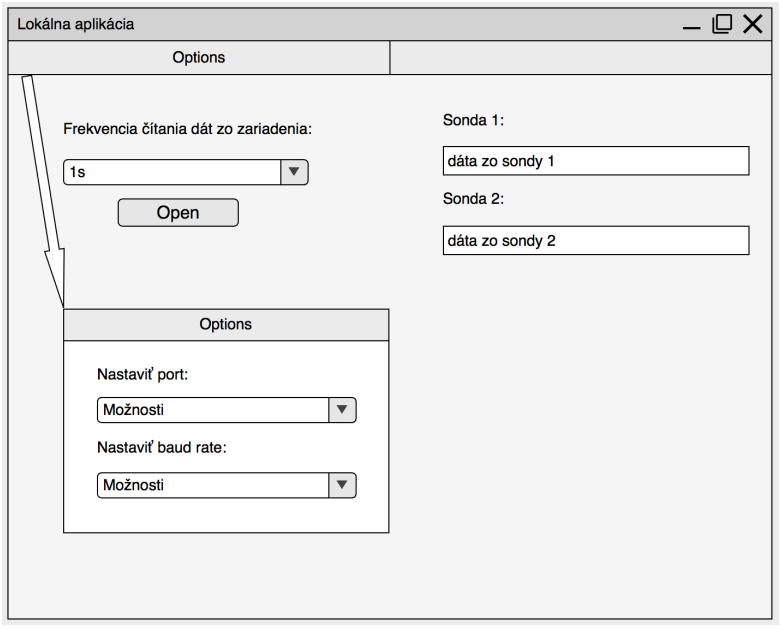
a) Stránka by mala byť dynamická a teda nestačí len čisté HTML a CSS.
b) Stránka by mala byť responzívna a teda mala by sa vhodne zobrazovať na ľubovoľnom zariadení s ľubovoľným rozlíšením.
c) Stránka by sa mala aj bez opätovného načítania vedieť aktualizovať.

3 Používateľské rozhranie

3.1 Lokálna aplikácia

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.

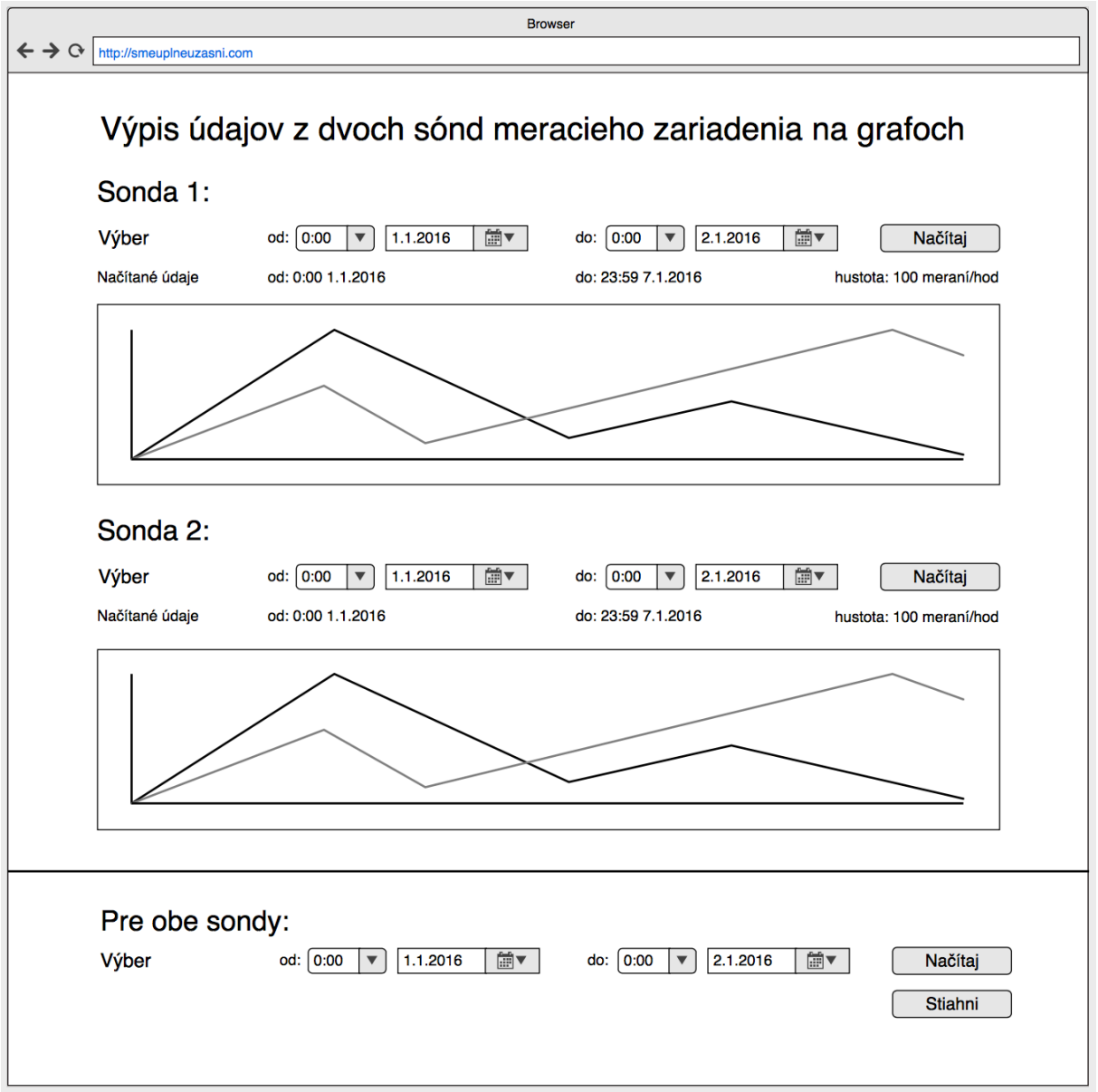


3.2 Web

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.

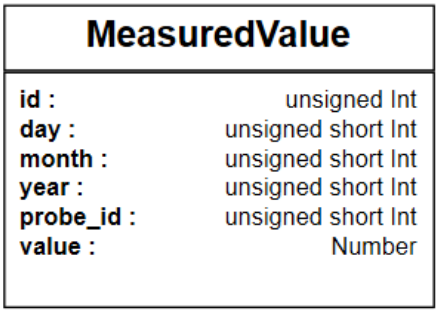
Dôležitou súčasťou je možnosť vybrať si časové pásmo zobrazovaných údajov pre jednu aj druhú sondu osobitne.

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

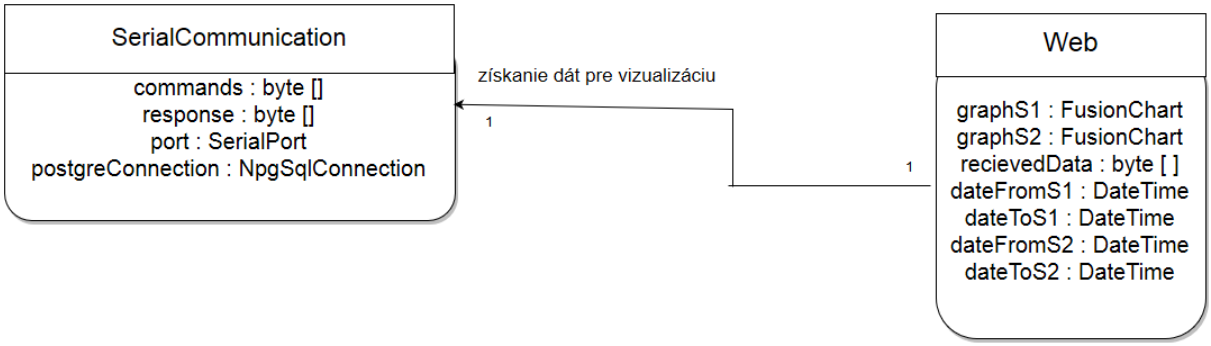


4 Dátový model, dekompozícia a triedny diagram

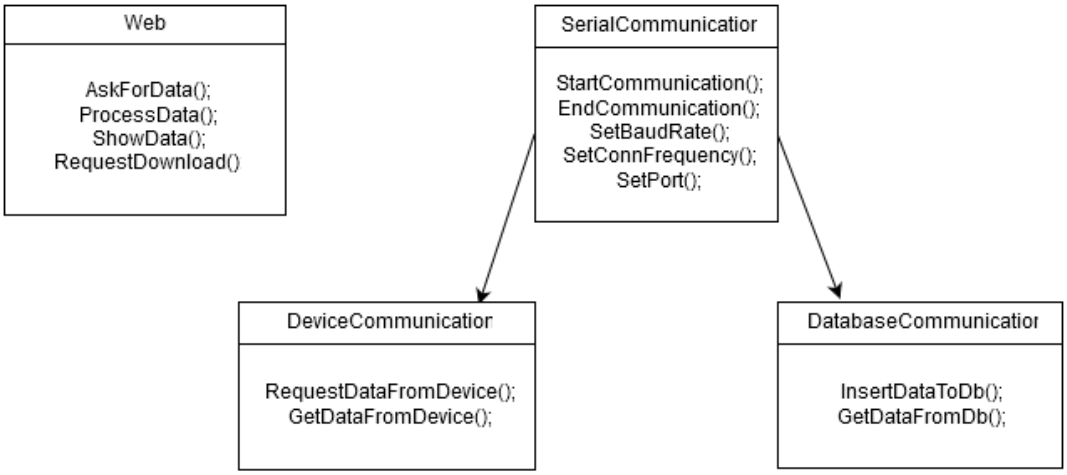
4.1 Dátový model



4.2 Dekompozícia



4.3 Triedny diagram



5 Testovacie scenáre

5.1 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.

5.2 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.

5.3 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.