TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Sběr a vizualizace dat z modulů ESP8266

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Jan Noll**
Vedoucí práce: Ing. Miloš Hernych

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Data collection and visualisation from ESP8266 modules

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Jan Noll**
Supervisor: Ing. Miloš Hernych

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Noll**
Osobní číslo:	**M15000108**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Sběr a vizualizace dat z modulů ESP8266**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Proveďte rešerši stávajících řešení serverů pro sběr dat z prvků IoT a jejich vizualizaci ve webových prohlížečích.
2. Navrhněte vlastní serverovou aplikaci pro sběr a vizualizaci dat s možností zpětné konfigurace prvků IoT, založených na modulu ESP8266.
3. Návrh realizujte a zhodnoťte provoz aplikace v delším časovém horizontu.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **TROELSEN, Andrew W. C# a .NET 2.0 profesionálně. Brno: Zoner Press, 2006. Encyklopedie Zoner Press. ISBN 80-86815-42-0.**

[2] **EVJEN, Bill, Scott HANSELMAN a Devin RADER. ASP.NET 3.5 v jazycích C# a Visual Basic. Brno: Computer Press, 2009. Programujeme profesionálně. ISBN 978-80-251-2069-9.**

Vedoucí bakalářské práce: **Ing. Miloš Hernych**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Přemysl Svoboda**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2017**

Termín odevzdání bakalářské práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 9. 5. 2018

Podpis: 

Poděkování

Rád bych poděkoval Ing. Miloši Hernychovi za cenné rady, věcné připomínky a vstřícnost při vedení bakalářské práce.

Abstrakt

Bakalářská práce je zaměřena na návrh a realizaci serverové aplikace, sloužící k archivaci a následné vizualizaci měřených dat moduly ESP8266. Zabývá se rešerší již existujících systémů a následným návrhem a implementací vlastní aplikace. Celý navrhovaný systém využívá produkty firmy Microsoft, především frameworku .NET. Samotná aplikace obsahuje dvě rozhraní. První rozhraní je pouze datové a slouží k šifrované komunikaci mezi aplikací a měřicími moduly. Druhé rozhraní je uživatelské, pomocí něj uživatel spravuje své měřicí moduly, a to včetně jejich zpětné konfigurace. Dále má možnost vizualizovat měřená data pomocí předpřipravených komponent. Pro archivaci dat je využívaná relační databáze Microsoft SQL Server, která slouží jak k uložení všech naměřených hodnot, tak i k uložení informací o uživateli.

Klíčová slova: Informační systém, IoT, ESP8266, C#, .NET, REST API, Razor MVC

Abstract

This bachelor thesis is focused on design and implementation of server application which is used to collect and visualize data measured via ESP8266 modules. The bachelor thesis also covers research of existing applications providing functions mentioned above. The designed system is using Microsoft products, especially .NET framework. The application has two interfaces. The first is used to encrypted communication with measurement modules. The second one is user's interface which produces measured data visualization via prepared components and module management ability (with backward configuration included). Measured data and every other information are stored in Microsoft SQL Server database.

Keywords: Information system, IoT, ESP8266, C#, .NET, REST API, Razor MVC

Obsah

Seznam zkratek

AES	Advanced Encryption Standard, algoritmus používaný k šifrování informací
AJAX	Asynchronous JavaScript and XML, technologie umožňující komunikaci prohlížeče se serverem po načtení stránky
API	Application Programming Interface, programové rozhraní pro komun. s aplikacemi
CRUD	Create, Read, Update, Delete, základní funkce prováděné nad určitým záznamem
DLL	Dynamic-link library, knihovní soubor obsahující připravené funkce
HTTP	Hypertext Transfer Protocol, protokol pro přenos hypertextových dokumentů
HW	Hardware
IoT	Internet fo Things, internet věcí
JSON	JavaScript Object Notation, způsob zápisu dat
MAC	Media Access Control, identifikátor síťového rozhraní
MVC	Model-View-Controller, návrhový vzor aplikací
MVVM	Model-View-ViewModel, návrhový vzor aplikací
MQTT	Message Queue Telemetry Transport, protokol pro přenos dat
ORM	Object-relational mapping, technika pro převoda dat mezi objekty a relační databází
REST	Representational State Transfer, architektura rozhraní API
SMTP	Simple Mail Transfer Protocol, protokol k přenosu elektronické pošty
SQL	Structured Query Language, dotazovací jazyk pro komunikaci s relační databází
SSL	Secure Sockets Layer, síťová vrstva sloužící k zabezpečení komunikace
TCP	Transmission Control Protocol, protokol transportní vrstvy (garance doručení)
UDP	User Datagram Protocol, protokol transportní vrstvy (bez potvrz. doručení)
UI	User interface, uživatelské rozhraní
URL	Uniform Resource Locator, identifikátor dokumentů na internetu
Wi-Fi	Wireless Fidelity, protokol užívaný k bezdrátové komunikaci

Seznam obrázků

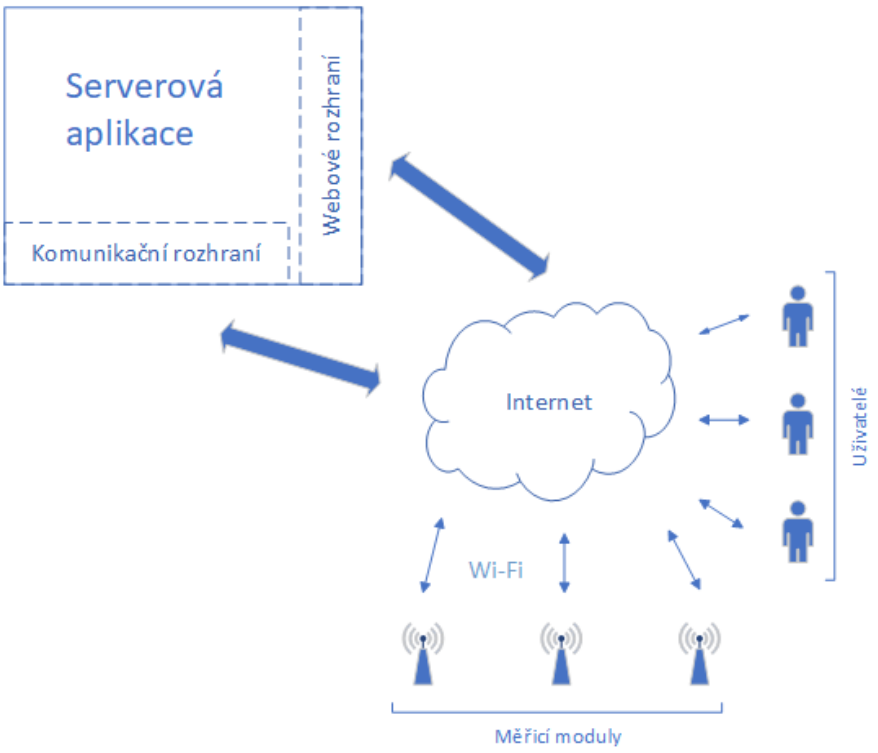
1 Úvod a cíle práce

Moduly založené na čipu ESP8266 využívají k přenosu dat technologii Wi-Fi. Moduly mohou měřit různorodá data. Od základních environmentálních veličin jako je např. teplota, tlak, vlhkost, až po více specifické veličiny jako je vodivost materiálů, odebíraný výkon, otáčky hřídele apod. (toto záleží pouze na použitých senzorech). Uživatel používající takovýchto modulů musí rozhodnout, jak s měřenými daty nakládat. V nejzákladnějším případě má uživatel dvě možnosti.

První možností je ukládat data pouze interně v rámci modulu a následně přímo v něm vystavit uživatelské rozhraní. Uživatel se pak připojuje přímo k danému modulu, kde je následně schopen pomocí uživatelského rozhraní modul ovládat a zobrazovat naměřené hodnoty. Uživatelské rozhraní bývá řešeno zpravidla webovou aplikací. Nevýhodou takovéhoto řešení je limitace pamětí modulu, což se projeví zejména na maximálním možném počtu uložených dat, ale i na rozsahu uživatelského rozhraní.

Druhou možností je oddělení ukládaných dat od měřicího modulu, a to včetně oddělení uživatelského rozhraní. V tomto případě modul komunikuje se serverovou aplikací, tj. odesílá jí měřené hodnoty a zpětně načítá konfigurační data. Komunikační protokol může být různorodý, nejčastěji se používá protokol HTTP a MQTT (z důvodu nativní podpory moduly). Také lze využít prostého spojení TCP a UDP. Uživatelské rozhraní je jako v předchozím případě také zpravidla řešeno webovou aplikací, ale oproti předchozímu řešení zde není aplikace nijak zásadně limitována. Nevýhodou tohoto řešením je nutnost stálé dostupnosti serverové aplikace, jelikož v případě výpadku spojení se serverem nelze data uložit do aplikace. Ale i toto lze řešit dočasným uložením do paměti modulu a odesláním na server po obnovení dostupnosti. Jednoduché blokové schéma takovéto aplikace z pohledu uživatele je možné vidět na obr. 1.

Cílem této práce je provést rešerši již existujících serverových aplikací, které umožní uložení a vizualizaci měřených dat. Následně navrhnout a implementovat obdobný způsob řešení, tj. navrhnout a implementovat serverovou aplikaci, která umožní registraci (připojení) měřicích modulů a následnou archivaci získaných dat pomocí databáze. Aplikace vystaví rozhraní ke komunikaci s měřicími moduly a uživatelské (webové) rozhraní, pomocí kterého bude uživatel spravovat měřicí moduly. Pomocí tohoto rozhraní budou také vizualizovány měřené hodnoty.



Obrázek 1: Blokové schéma serverové aplikace z pohledu uživatele

2 Moduly ESP8266

Jádrem měřicích modulů je obvod (čip) ESP8266 vyráběný čínskou firmou Espressif. Tyto obvody mají integrovaný modul Wi-Fi, díky němuž je lze připojit k běžným domácím sítím. Současně obsahují 32bitový procesor Tensilica pracující na frekvenci 80 MHz, který obsluhuje zmíněný modul Wi-Fi a současně vykonává uživatelský program. Samotný obvod bývá používán zřídka, častěji se lze setkat s obvodem integrovaným do modulu. Modulů existuje více druhů, liší se především různým počtem vývodů (tedy i velikostí) a anténou Wi-Fi (buď bývá integrována v modulu, nebo je pro ni připraven konektor).

Pro potřeby testování vyvíjené aplikace budou použité dva kompaktní měřicí moduly řízené zmiňovanými obvody, konkrétně s označením ESP-12F. Tyto moduly obsahují oproti předchozím verzím zejména vylepšenou integrovanou anténu s větším signálovým ziskem. Oba druhy měřicích modulů jsou vybaveny obvodem hodin reálného času, který je schopen iniciovat zahájení měření v nastaveném časovém intervalu. Dále obsahují senzory pro měření teploty, relativní vzdušné vlhkosti a atmosférického tlaku. Jeden druh měřicího modulu je napájen pomocí akumulátoru, a proto bude možné u něj snímat i napětí baterie pro zjištění přibližné zbývající kapacity. Druhý typ měřicího modulu byl navrhnut a vyroben v průběhu vypracování bakalářského projektu. Více detailních informací o tomto měřicím modulu se lze dočíst v [1]. Jeden z použitých modulů je možné vidět na obr. 2.

Obrázek 2: Ukázka použitého měřicího modulu (dostupné z [1])

3 Rešerše stávajících řešení

Vzhledem k růstu, který momentálně odvětví internetu věcí zažívá, je dostupné značné množství služeb poskytujících sběr a vizualizaci měřených dat. Jednotlivé služby se liší zejména podporou jednotlivých komunikačních protokolů, maximálním množstvím uložených dat, či podporou dalších analýz prováděných se získanými daty.

3.1 ThingSpeak

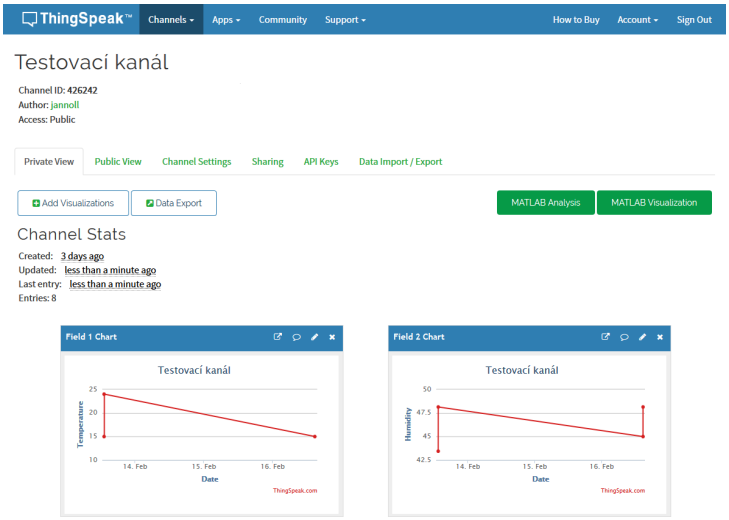
ThingSpeak je jednou z nejznámějších a pravděpodobně nejpoužívanějších služeb v odvětví IoT. Nabízí široké možnosti analýzy a zpracování příchozích dat. Toto je umožněno především nativní podporou algoritmů vytvořených v systému MATLAB®.

Všechna příchozí data jsou přiřazena do předem vytvořených kanálů. Každý kanál může obsahovat maximálně osm datových sérií. Jednotlivé datové série mohou být vizualizovány pomocí komponent v kanálech. Ve výchozím stavu je podporován graf měřených hodnot v závislosti na čase. Dále je možné využití vizualizací MATLAB – vytvoření vlastního specifického grafu pomocí skriptu z prostředí MATLAB. Ukázku jednoduché vizualizace kanálu je možné vidět na obr. 3.

Vytvořený kanál automaticky obsahuje dva páry API klíčů – jeden pro čtení druhý pro zápis. Tyto API klíče jsou zásadní pro zápis a čtení dat pomocí vystaveného API, které služba ThingSpeak nabízí. Uživatel má možnost tyto klíče měnit (systémem vygenerovat nové). Data je možné do kanálů odesílat pomocí protokolu HTTP, nebo MQTT. V obou případech je zapotřebí vždy uvést unikátní identifikátor kanálu, do kterého jsou data odesílána, a klíč API pro ověření autentičnosti dat. Vždy je možné vybrat do jaké datové série v patřičném kanálu hodláme publikovat. Samozřejmostí je možnost odeslání více rozdílných dat v jednom požadavku (limitováno konkrétním kanálem). [2]

- **Výhody:**
  - \+ podpora algoritmů MATLAB®
- **Nevýhody:**
  - – nízký počet vizualizačních komponent (nutnost použití externích pluginů)

– vizualizační komponenty nejsou plně responzivní pro mobilní zařízení



Obrázek 3: Ukázka vizualizace kanálu služby ThingSpeak

3.2 Beebotte

Práce se zdroji dat je u služby Beebotte řešena obdobně jako u služby ThingSpeak. Nejprve je nutné vytvořit kanál. Následně jsou v kanálu vytvořeny zdroje. Zdrojům je možné explicitně předdefinovat datový typ, který bude akceptován. K dispozici jsou základní typy jako *numeric*, *string*, *boolean*, ale i více specifické jako *gps*, *rate*, *conductivity* apod. Výhodou oproti službě ThingSpeak je neomezený počet zdrojů v jednom kanálu. Nevýhodou je nemožnost provádět výpočty libovolných matematických modelů nad příchozími daty.

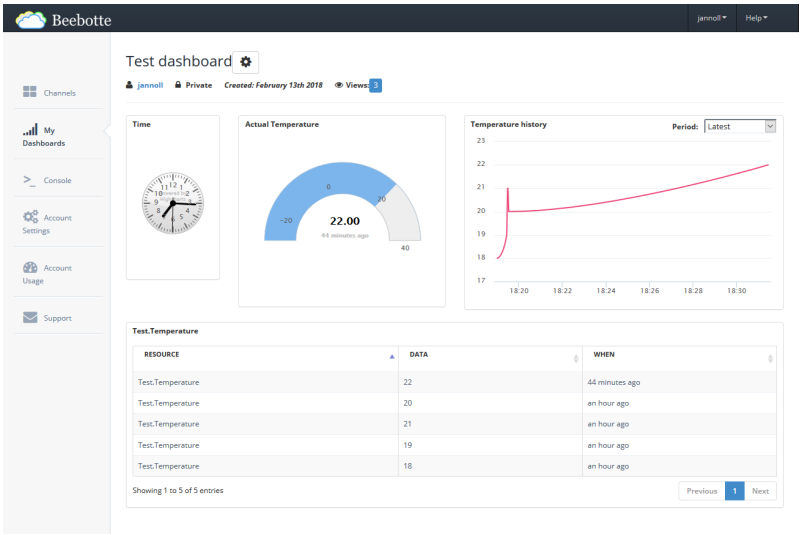
Vizualizace naměřených dat je prováděna na stránce nazvané Dashboard. Uživatel může vytvořit neomezený počet dashboardů s neomezeným množstvím widgetů různých typů. K dispozici je například graf s časovou závislostí (s možností prolnutí více zdrojů), měřidlo zobrazující aktuální hodnotu, tabulka přijatých hodnot, nebo teplotní mapa. Příklad dashboardu se základními widgety je na obr. 4.

Každému uživateli je asociován pár autentizačních klíčů. Prvním je API klíč, který slouží k rozpoznání uživatele příchozího požadavku. Druhým klíčem je tzv. „tajný klíč“ (z angl. *secret key*), ten je použit pro zašifrování přenášených dat. Služba Beebotte pro práci s daty podporuje protokoly MQTT a HTTP. Pro uložení dat je nutné v požadavku specifikovat jméno kanálu a jméno zdroje, do kterého se mají příchozí data zařadit. [3]

- **Výhody:**

  \+ neomezený počet zdrojů v kanálu

- **Nevýhody:**
  - nemožnost odeslání více dat do různých kanálů jedním požadavkem
  - absence nativní podpory aplikace mat. modelů na měřená data



Obrázek 4: Ukázka dashboardu služby Beebotte

3.3 Corlysis

Ve službě Corlysis jsou data sdružována v databázích. Databázi je nutné vytvořit před prvním přijetím dat. Do databáze mohou být následně zapisovány naměřené hodnoty. Při přijetí dat jsou jednotlivé veličiny rozřazeny dle názvů. Pokud daný název není v databázi zaveden, vytvoří se automaticky.

Pro vizualizaci naměřených dat služba Corlysis používá produkt Grafana. Platforma Grafana nabízí velké množství vizualizačních komponent s rozsáhlými možnostmi uživatelské konfigurace. Vizualizace jsou prováděny na stránce dashboard. Uživatel má možnost vytvoření neomezeného počtu dashboardů. K dispozici je například graf, konkrétní hodnota, tabulka hodnot, tepelná mapa atd. Příklad dashboardu vytvořeného pomocí produktu Grafana je možné vidět na obr. 5.

Corlysis využívá pro archivaci měřených dat databázi InfluxDB (databázový systém optimalizovaný pro *time series data*). Pro odeslání dat na server pak může být použita příkazová konzole dodávaná přímo společností influxdata (společnost produkující databázový systém influxDB). Jelikož tato konzole využívá protokol HTTP, uživatel může data odesílat pouze pomocí HTTP pořadavků typu POST. Z důvodu ověření autentičnosti dat má každá uživatelova databáze vygenerované jméno a heslo. Toto heslo je použito pro vložení dat do uživatelovy databáze. [4]

- **Výhody:**
  - \+ pro vizualizaci dat použit produkt Grafana
  - \+ automatická inicializace proměnných
- **Nevýhody:**
  - – omezení celkového počtu naměřených hodnot (není podmíněno časovým rámcem)
  - – bez možnosti vytváření upozornění (vzhledem k měřeným hodnotám)

Obrázek 5: Ukázka dashboardu produktu Grafana

3.4 Ubidots

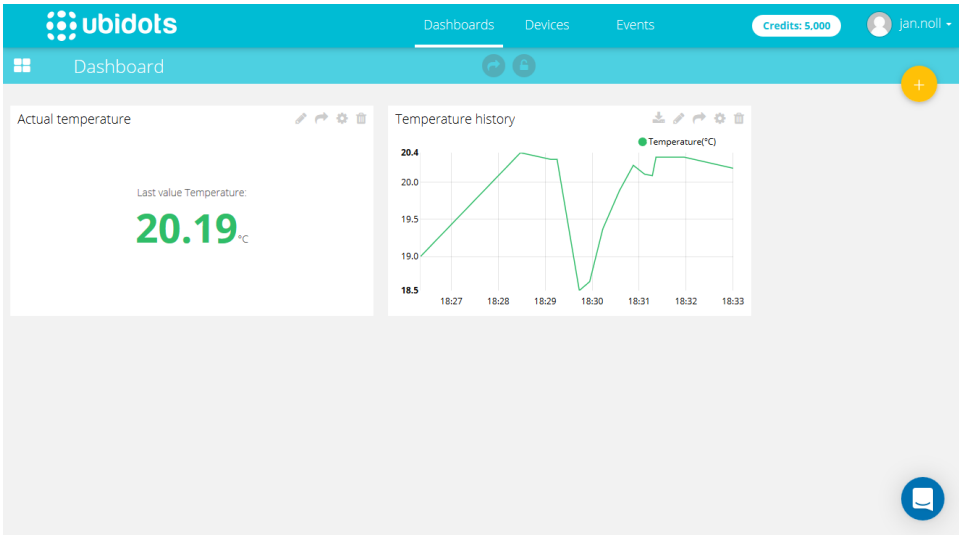
Služba Ubidots primárně cílí na senzorické prvky internetu věcí. Tomu odpovídá i organizace měřených dat. Do systému jsou registrována zařízení, která dále obsahují jednotlivé proměnné. V praxi se jedná například o jednotlivé senzory připojené k měřicímu modulu. Vždy se ale jedná o data závislá na čase. Inicializace proměnných se děje automaticky při obdržení prvních dat ze zařízení.

Proměnné mohou být i „virtuální“. Ty nepředstavují skutečný senzor, ale aplikují určitý matematický model na libovolná měřená data - např. průměrnou hodnotu, extrémy (minimum/maximum) atd.

Správa zařízení umožňuje pojmenování proměnných, přiřazení popisků, výpis historických dat s možností filtrace a mimo jiné export do souboru .csv. Dále je možné nastavit tzv. události vztahující se vždy ke konkrétní proměnné. Události mohou být spouštěny např. dosažením nastavené mezní hodnoty, nebo například neaktivitou (pokud data nepřijdou v předem stanoveném časovém úseku).

Samotná data jsou vizualizována na dashboardu do různých komponent. K dispozici je například graf, tabulka, nebo číselná hodnota. Náročnější uživatelé si mohou na dashboardu vytvořit vlastní komponenty nakódováním skriptu (k dispozici je jazyk HTML, JavaScript a kaskádové styly). Uživatelé mají možnost vytvoření více dashboardů s různými komponentami. Příklad vytvořeného dashboardu je možné vidět na obr. 6. [5]

- **Výhody:**
  - \+ možnost vlastních skriptů na dashboardu
  - \+ automatická inicializace proměnných
- **Nevýhody:**
  - – dlouhá odezva na REST API (řádově sekundy až desítky sekund)



Obrázek 6: Ukázka dashboardu služby Ubidots

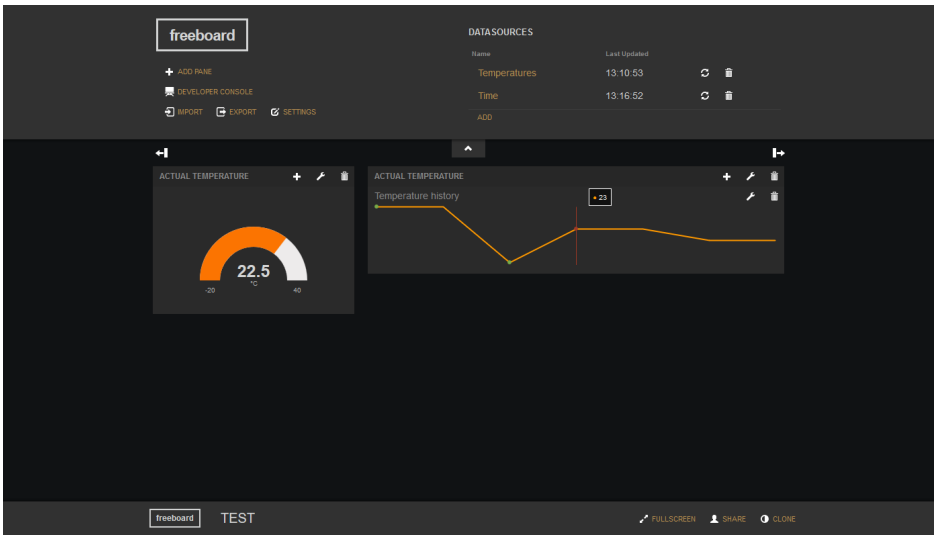
3.5 freeboard

Služba freeboard poskytuje jen vizualizační rozhraní, nemůže být použita samostatně pro archivaci měřených dat. Uživatel musí nejprve zaregistrovat datový zdroj. K dispozici jsou „standardní“ protokoly typu HTTP, či MQTT. Dále jsou v nabídce připraveny služby třetích stran, mezi ně patří například dweet.io, PubNub, nebo Xively. V tomto případě tyto služby vytváří mezičlánek mezi měřicím modulem a službou freeboard s vizualizačními komponentami.

Data jsou uživateli opět vizualizována na dashboardu. Počet dashboardů a komponent na nich není omezen. Pro každý dashboard je nutné nejprve nadefinovat již výše zmíněný datový zdroj. Následně je možné vytvořit panel. Do panelu je možné

přidat samotné komponenty. K dispozici je například poslední změřená hodnota, graf, mapa, text, nebo indikační kontrolka pro vyjádření dvoustavových veličin. [6]

- **Výhody:**
  - \+ možnost čerpat data z různorodých služeb třetích stran
- **Nevýhody:**
  - – nutnost využít další služby pro správu měřených dat



Obrázek 7: Ukázka dashboardu služby freeboard

3.6 Vyhodnocení předchozí analýzy

Všechny výše zmíněné služby umožňují vizualizovat data z prvků internetu věcí. Liší se dostupností různých vizualizačních komponent, případně nabízenými možnostmi jejich uživatelské konfigurace. V některých případech jsou nabízeny i komponenty podporující odesílání příkazů/dat do prvků internetu věcí (typicky formou přepínačů, nebo posuvníků).

Z hlediska možností ukládání dat se služby liší primárně v dostupnosti podporovaných protokolů. Téměř všechny zmíněné podporují protokol HTTP a MQTT. Některé služby nabízejí i další možnost. Například služba Ubidots umožňuje i použití základních protokolů TCP a UDP, což umožňuje minimalizovat objem přenášených dat. Každá služba má omezení vztažené k měřeným datům, ale často se liší způsobem limitace. Některé služby limitují celkovým počtem naměřených bodů, ale některé pouze omezují počet bodů za časový úsek. Tyto kvóty pak bývají ještě proměnlivé vzhledem k aktivovanému licenčnímu plánu (zvoleného uživatelem), které služby nabízejí.

Některé systémy nabízejí další doplňkové služby. Často bývá v nabídce například možnost vytvoření notifikační zprávy v případě dosažení nějaké prahové hodnoty apod. Způsob doručení bývá různý, ale za standard lze považovat odeslání emailu, zprávy SMS, či vytvoření HTTP požadavku na další libovolný systém. Často bývá k dispozici možnost zálohování/exportu uložených dat do souboru (nejčastěji .csv), případně pak i importu již existujících dat.

Porovnání jednotlivých služeb dle poskytovaných funkcionalit je uvedeno v tabulce 1. Vzhledem k velice rozlišným nabídkám napříč licencemi jednotlivých služeb, bude v následující části provedeno porovnání pro licence v přibližné cenové hladině $20 - $30/měsíc. Služba freeboard je v tabulce uvedena pro úplnost. V případě kdy je daná funkce závislá na použité službě třetí strany, je pole přeškrtnuto.

Funkce	ThingSpeak	Beebotte	Corlysis	Ubidots	freeboard
podpora komunikačního protokolu HTTP	x	x	x	x	-
podpora komunikačního protokolu MQTT	x	x		x	-
podpora komunikačních protokolů TCP/UDP				x	-
možnost hromadného importu dat	x				-
možnost hromadného exportu dat	x	x		x	-
podpora nastavení notifikačních emailů				x	-
podpora nastavení notifikačních SMS				x	-
možnost soukromé vizualizace dat	x	x	x	x	x
možnost veřejné vizualizace dat	x	x	x	x	x
podpora vlastních komponent	x				
neomezený počet prvků na dashboardu		x	x	x	x
podpora ovládání aktorů z dashoardu		x		x	

Tabulka 1: Srovnání existujících služeb dle dostupných funkcí

4 Návrh řešení

Nejprve je nutné rozhodnout jaké funkce má aplikace umožňovat. Z pohledu uživatele by měla aplikace nabízet možnost registrace neomezeného počtu měřicích modulů, a to včetně neomezeného množství ukládaných dat. Připojení měřicího modulu musí být co nejsnazší, tj. s minimální vyžadovanou konfigurací ze strany uživatele. Uživatel bude k aplikaci přistupovat pouze z vystaveného webového rozhraní. Uživatelské rozhraní by mělo splňovat následujících šest zásadních kritérií:

1. Autentizace uživatelů pomocí emailové adresy a hesla
2. Možnost uživatelem registrovat moduly
3. Možnost konfigurovat registrované moduly
4. Možnost nastavit notifikaci uživateli v případě naměření určité limitní hodnoty
5. Možnost uspořádání vizualizačních komponent v těle stránky k tomu určené
6. Automatická (*real-time*) obnova vizualizačních komponent dle přijatých dat

4.1 Analýza

Pro návrh serverové aplikace je zprvu nezbytné rozhodnout o architektuře aplikace. Lze se držet konvenčního přístupu, kdy je všechna logika obsažena v jedné aplikaci, nebo ji lze rozdělit na více takzvaných mikroslužeb (z angl. *microservices*). V takovém případě je spuštěno současně více menších aplikací, přičemž každá vykonává určitou část. Výhodou tohoto přístupu je možnost vysoké škálovatelnosti. V případě, že je nezbytné zvýšit výkon jedné části aplikace, lze použít výkon jiné, která není plně vytížena apod. Zásadní nevýhodu představuje nutnost komunikace mezi jednotlivými službami, nutnost synchronizace dat atd.

Konkrétně v našem případě by bylo možné rozdělit aplikace na dvě části, na uživatelem používající webové rozhraní a na rozhraní komunikující s měřicími moduly. Ovšem pokud bude uvažována situace, ve které je třeba příchozí data okamžitě *odesílat* uživateli do vizualizačních komponent, nebylo by dosaženo zamýšlené oddělitelnosti a byly by stále vytěžovány obě aplikace. Navíc by bylo nutné vyřešit výše zmíněnou komunikaci mezi rozhraními. Není ani momentálně předpokládaný extrémně vysoký nápor, který by podporoval rozdělení aplikace do více samostatných služeb. Z těchto důvodů bude vhodnější postupovat s návrhem *konvenčním*

způsobem, tj. ponechat všechnu logiku a všechna vystavená rozhraní v jedné aplikaci.

4.1.1 Komunikace s měřicími moduly

Další důležitou částí návrhu aplikace je výběr podporovaného komunikačního protokolu, který bude použit pro komunikaci s měřicími moduly. Jak bylo zmíněno dříve – nejpoužívanějšími protokoly pro komunikaci s moduly ESP8266 je protokol HTTP a protokol MQTT, přičemž každý vyžaduje naprosto odlišný návrhový přístup, a tedy i následný implementační postup.

V případě komunikace pomocí protokolu HTTP je implementace velmi snadná. V aplikaci postačí vystavit REST API. V praxi pak modul pošle v těle HTTP požadavku naměřená data. Server v odpovědi na požadavek vrátí příslušný stavový kód, který slouží pro modul jako hlášení o úspěšné archivaci dat.

Pokud by byl použit ke komunikaci protokol MQTT, byla by implementace složitější a to zejména proto, že protokol MQTT vyžaduje spuštění separátní služby třetí strany zvané MQTT broker. Tato služba se stará o přeposílání zpráv mezi klienty. V praxi by se pak měřicí moduly připojovaly k MQTT brokeru a publikovaly zprávy s naměřenými daty. Serverová aplikace by musela být také připojena formou klienta k MQTT brokeru a naslouchat příchozím zprávám. Další nevýhodou je nutnost ručně publikovat stavovou zprávu zpět do modulu, což představuje i další nutnou režii v měřicím modulu. Naopak výhodou protokolu MQTT je jeho nenáročnost z hlediska datových přenosů.

Pro komunikaci s měřicími moduly bude použit protokol HTTP a to zejména z důvodu snadné implementace. Od měřicích modulů mohou přicházet pouze dva požadavky. Prvním z nich je žádost o registraci do systému, druhým je požadavek o uložení naměřených dat.

Registrace modulu

Proces registrace modulu je prvním krokem ke spárování skutečného modulu s uživatelovým, předem vytvořeným modulem v administrační části aplikace. K tomuto je třeba autorizační klíč. Ten je uživateli vygenerován právě při vytvoření nového modulu v administračním rozhraní. Současně při registraci modul odešle do systému základní informace o hardwaru (MAC adresu, unikátní sériové číslo apod.). Dále odešle i informace o připojených senzorech, které obsahuje. Tyto informace jsou následně převedeny do podoby přijatelné pro uživatele. Ten tedy pro registraci vidí všechny připojené senzory, a to bez nutnosti další konfigurace ze strany uživatele.

Archivace naměřených dat

Druhou podporovanou akcí v komunikaci je samotný příjem naměřených dat z modulů. Příchozí data jsou identifikována pomocí unikátního sériového čísla modulu (získaného při registraci). Každé měření je opatřeno časovou známkou, což umožní zpřesnit udávaný čas měření a zároveň přidá možnost vkládat do systému hodnoty starších měření. Jelikož celá komunikace je jednosměrná (požadavek je směřován

pouze z modulu na server), tak v okamžiku přijetí dat serverem jsou do modulu odeslána konfigurační data, jsou-li nějaká nová k dispozici.

Zabezpečení přenášených dat

Jelikož některé firmwary pro moduly ESP8266 nejsou schopné kooperovat s API zabezpečené SSL certifikátem, přenos takovouto zabezpečenou komunikací není vyžadován, a proto jsou všechny datové přenosy šifrovány algoritmem AES-128. Každý modul má vygenerovaný vlastní šifrovací klíč. Tento klíč je vygenerován při registraci. Registrace probíhá s tzv. *sdíleným* klíčem. Tento klíč musí být nahrán ve všech modulech a je použit pouze pro registraci modulu do systému. Jelikož registrační požadavky obsahují unikátní autorizační token a požadavky na archivaci dat obsahují časové známky, nemůže dojít ani při měření stálých hodnot k neunikátnosti přenášených dat, což zabraňuje případnému odposlechnutí a znovu-odeslání požadavku případným narušitelem.

4.1.2 Uživatelské rozhraní

Dalším rozhraním aplikace bude rozhraní webové. To bude sloužit k interakci s uživatelem. Jak bylo zmíněno výše, přístup uživatele bude autorizován, tedy bez přihlášení nebudou dostupné žádné jiné akce. Aby byl systém samostatný, bude dostupný registrační formulář. Tímto formulářem si bude moci nový uživatel bez zásahu administrátora vytvořit nový účet. Unikátním rozlišovacím klíčem uživatele bude jeho emailová adresa, proto po odeslání registračního formuláře bude uživatel nucen potvrdit vlastnictví zadané emailové adresy. A to navštívením webové stránky, jejíž URL adresa bude uživateli sdělena automaticky vygenerovaným emailem. Pro případy kdy uživatel ztratí heslo ke svému účtu, bude vytvořen speciální formulář, pomocí kterého bude uživatel moci zažádat o změnu hesla. Pro tento krok bude nezbytné zadat pouze emailovou adresu. Pokud tato adresa bude vedena v systému, systém opět automaticky odešle na tuto adresu email s odkazem na webovou stránku s formulářem, pomocí kterého uživatel dokončí změnu hesla, a to nastavením hesla nového. Použitím emailové komunikace je ověřena autentičnost uživatele vznášející požadavek na změnu hesla.

Po úspěšné autentizaci bude uživateli pohyb po webu umožněn pomocí menu umístěném v horní části. Webové rozhraní se bude skládat ze dvou základní stránek, a to stránek *DASHBOARD* a stránky *SPRÁVA*.

Stránka *DASHBOARD*

Stránka DASHBOARD bude sloužit pouze k vizualizaci měřených dat. Vizualizace bude možná pomocí předem definovaných komponent. Uživatel bude moci konfigurovat stránku dashboard samostatně pomocí těchto komponent. Komponenty budou předpřipravené ve třech různých velikostech a na stránce budou umisťovány v tzv. gridu (jakási šablona, která předem vymezuje prostor pro umístění komponent). Komponenty budou dvojího typu. První typ bude zobrazovat pouze jednu hodnotu. Druhý typ pak znázorní přijatá data v grafu v závislosti na čase měření. U grafického

znázornění bude také vhodné umožnit uživateli zvolit rozsah zobrazovaných hodnot (časový úsek – 1 den, 1 měsíc, 1 rok atp.). Každá komponenta bude mít přiřazen datový zdroj (konkrétní senzor z konkrétního měřicího modulu). Dále bude obsahovat titulek, který bude vypsán nad samotnou komponentou, aby bylo možné rozlišit zobrazené údaje. Uživatel bude moci komponenty přemisťovat pomocí tažením myši.

Stránka *SPRÁVA*

Druhou stránkou v menu je stránka *SPRÁVA*. Pomocí této stránky bude mít uživatel možnost spravovat své měřicí moduly a editovat svůj uživatelský profil. Vše bude jako na dashboardu rozděleno do bloků. Oproti stránce *DASHBOARD* rozložení této stránky nebude pro uživatele konfigurovatelné, ale bude pevně dané. A to zejména z důvodu nízkého počtu dostupných bloků, a tedy i příliš malého počtu možných kombinací, což by pro uživatele nebylo atraktivní. Mimo jiné i z důvodu zjednodušení následné implementace.

Prvním blokem bude správa uživatelova profilu. Defaultně bude zobrazeno celé jméno, emailová adresa a avatar profilu. Pomocí tohoto bloku se uživatel dostane ke dvěma formulářům. První bude pouze pro změnu hesla, druhý bude pro změnu ostatních údajů, a to včetně možnosti nahrát vlastní obrázek/avatar. Nebude možné pouze měnit emailovou adresu, jelikož jak bylo zmíněno výše, je plně provázána s vlastníkem profilu a slouží např. k ověření autentičnosti uživatele. Proto by nebyly vhodné její následné změny.

Dalším blokem bude tabulka posledních přijatých dat. Ta je vhodná zejména pro informační účely – na první pohled uživatel vidí, kdy bylo provedeno poslední měření, což je možné použít i pro ověření funkčnosti modulů. V této tabulce se bude zobrazovat vždy poslední zaznamenaná hodnota každého senzoru. Dále bude možné prokliknout se na seznam všech naměřených hodnot. Tedy bude možné v tabulce přehledně procházet všechna historická data.

Posledním dostupným blokem bude seznam aktivních měřicích modulů. V tomto seznamu budou pouze základní informace jako název modulu, MAC adresa, nebo datum registrace. Pomocí tohoto bloku se bude moci uživatel dostat na stránku se všemi svými moduly, uvidí zde tedy i ty, které nejsou ve stavu *aktivní* (to mohou být stavy nepřipojen a vyřazen). Na této stránce bude mít uživatel možnost zaregistrovat nový modul. Při registraci bude třeba zadat pouze název modulu, jako doplňující parametr pak bude uživatel moci zadat například krátký popis, nebo umístění. Naopak při registraci dostane autorizační token, který použije pro spárování modulu se systémem. V tuto chvíli se objeví modul v seznamu, a to s informací, že dosud nebyl připojen a s návodem, jak modul připojit (předpokládá se použití jednotných měřicích modulů s jednotným uživatelským rozhraním). Po prvotním spojení modulu se systémem se ve výše zmíněném seznamu zobrazí modul s jeho MAC adresou a s hodnotou úrovně signálu Wi-Fi.

Jakmile je modul připojen, uživatel má možnost se skrze položku v seznamu modulů dostat na stránku detailu modulu. Na této stránce jsou zobrazeny všechny dostupné informace o modulu (jeho název, MAC adresa, interní sériové číslo, či případně popis a umístění). Dále je dostupná tabulka konfiguračních dat modulu.

Po připojení modulu je načtena stávající konfigurace (v seznamu musí být alespoň jedna položka – interval měření). V dalším bloku má uživatel možnost konfigurovat notifikace související s modulem. Ve výchozím stavu není žádná aktivní. Uživatel má možnost přidat neomezené množství notifikací (aktuálně pouze pomocí emailu). Každá notifikace se musí vztahovat ke konkrétnímu senzoru. Uživatel může odeslání notifikace podmínit dvěma možnostmi. První možností je, pokud měřená hodnota senzoru stoupne nad nastavenou hranici. Druhá, pokud klesne pod tuto hranici. Posledním parametrem pak je text notifikace, který bude uživateli doručen. Posledním blokem v detailu modulu, je tabulka posledních přijatých hodnot. Její funkce je stejná jako na stránce *SPRÁVA*, jen její obsah závisí pouze na naměřených hodnotách z konkrétního modulu. To může být výhodné, pokud má uživatel registrovaný větší počet modulů.

Ostatní stránky

Ačkoliv na tyto stránky nevede přímý odkaz, jejich vytvoření je nezbytné – jedná se především o stránky chybové. Ačkoliv bude implementace probíhat tak, aby se minimalizovala pravděpodobnost výskytu jakýchkoliv chybových stavů, existují případy které nelze vyřešit jinak (např. neexistující stránky oproti uživatelem zadané URL adrese). V tomto případě bude uživatel přesměrován na chybovou stránku, která ho informuje o neexistující akci. Dále musí být ještě vytvořena chybová stránka, která bude použita při jakékoliv jiné neočekávané chybě. Touto chybou může být například výpadek databázového serveru. V tu chvíli nejsou dostupná žádná data a aplikace nemůže nabídnout jiné řešení.

4.2 Návrh informačního systému

Po předchozí úvaze o předpokládaných funkcionalitách je následně nutné provést konkrétní návrh informačního systému. Nejprve bude nutné rozhodnout o jednotlivých technologiích s přihlédnutím k možnostem hostování výsledné aplikace. Následně bude proveden návrh databázové struktury dle předchozí analýzy a bude proveden návrh architektury samotné aplikace.

4.2.1 Výběr serverových technologií

Celý informační systém je navrhován pro platformu Microsoft Windows a to zejména z důvodu dostupného serveru, který bude moci být použit, jak pro testování aplikace, tak pro její spuštění v produkčním režimu. Tento server využívá operační systém Microwoft Windows Server 2012 R2 a již obsahuje nezbytný software k hostování webových aplikací vytvořených pomocí frameworku ASP.NET. Další možností by bylo vytvoření aplikace pomocí platformy .NET Core. Platforma .NET Core je relativně novinkou (verze 1.0 byla oficiálně vydána v roce 2017), z pohledu vývoje je platforma .NET Core velmi podobná .NET Frameworku. Výhodou je například multiplatformnost, tedy možnost takovéto aplikace hostovat mimo operační systém Windows. To je způsobeno implementací vlastního webového serveru

do balíčku .NET Core, nicméně je nutné ručně nainstalovat jádro .NET Core na hostující server. To je i případ zde dostupného serveru. Vzhledem k těmto nutným administrátorkým úkonům a vzhledem k faktu, že se jedná stále o nový a velmi rychle rostoucí projekt, pro který nejsou dostupné všechny doplňky, které platforma .NET nabízí, bude vhodné vyvíjet aplikaci pomocí stále více rozšířenější platformy .NET Framework.

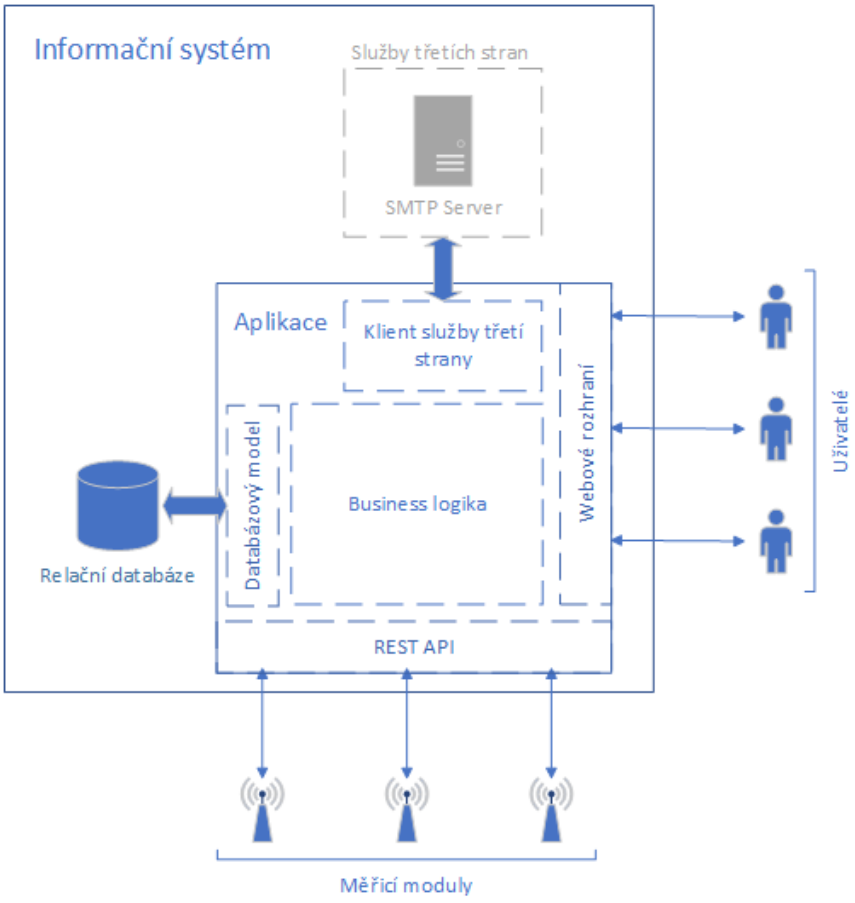
Druhou a neméně důležitou volbou je volba databáze, do které budou ukládána data, a to jak naměřené hodnoty přijaté z měřicích modulů, tak i všechny ostatní informace nutné k provozu služby (např. informace o uživatelích, nastavené notifikace, konfigurace modulů apod.). Při výběru je nutné rozlišit dva druhy databází. Buď lze použít tzv. relační databázi, nebo databáze využívající tzv. NoSQL koncept. Relační databáze bývá zpravidla rozdělena do několika tabulek a záznamy v tabulkách jsou pak provázány tzv. relacemi. Tyto relace umožňují vytvoření rozsáhlejších struktur a následné zachování integrity dat. Databáze NoSQL se vyznačují naprosto odlišným způsobem práce s daty. Nejčastěji bývají používány v režimu *key-value*, to znamená, že pod jeden klíč je uložena jedna hodnota (hodnotou může být celá datová struktura). To má za následek velmi vysokou rychlost, a to i při práci s velkým množstvím dat. Nevýhodou je velmi obtížná správa integrity uložených dat (obvykle je tato správa řešena samostatnou aplikací). V případě této práce by bylo vhodné zkombinovat oba výše zmíněné způsoby – relační databázi použít k uložení konfiguračních dat, uživatelských údajů apod. a všechna měřená data archivovat pomocí NoSQL databáze. Ovšem při takovémto řešení by se projevoval problém se zachováním integrity dat, a to zejména napříč oběma databázovými poskytovateli. I v průběhu implementace by bylo obtížné pracovat se dvěma zdroji dat. Proto bude nejoptimálnější použít jednu relační databázi, ve které budou všechna data. Konkrétně bude použita relační databáze MS-SQL, pro níž jsou v prostředí .NET připravené konektory. Navíc je již patřičný databázový server připravený na dostupném serveru, tudíž opět nebude vyžadován další administrátorský zásah.

Ačkoliv žádné další serverové technologie nebudou přímo provozovány, informační systém bude využívat i služeb třetích stran. Typickým příkladem této služby je SMTP server. Navrhovaný systém se bude pouze připojovat ke vzdálenému SMTP serveru a jeho prostřednictvím odesílat emaily uživatelům. Nebude tedy nutné vynakládat další úsilí ke spravování těchto služeb.

4.2.2 Návrh architektury informačního systému

Jak již bylo zmíněno v analýze, všechna logika systému bude implementovaná do jediné aplikace. Tudíž tato aplikace bude muset obsahovat všechny dílčí části. Celou aplikaci lze rozdělit do několika vrstev. První a nejzásadnější vrstvou bude vrstva datová. V té bude přesně nadefinovaná struktura databáze a bude zajišťovat výměnu dat mezi databází a aplikací. Další vrstva bude tzv. prezenční, pod ní budou zařazena obě rozhraní, jak uživatelské, tak i REST API pro komunikaci s měřicími moduly. Mezi těmito vrstvami bude tzv. business vrstva, která bude představovat celou aplikační logiku a bude zprostředkovávat výměnu dat mezi těmito vrstvami. Poslední vrstvou bude pomocná komunikační, která bude zajišťovat komunikaci

s externími službami (např. již zmíněný SMTP server). Blokové schéma navrženého informačního systému lze vidět na obr. 8.



Obrázek 8: Blokové schéma navrženého informačního systému

4.2.3 Návrh struktury databáze

Jelikož bude použita relační databáze, je nutné před zahájením implementace navrhnout její strukturu tak, aby umožňovala všechny předpokládané funkce. Při návrhu databáze je třeba dbát několika základních pravidel (dostupné např. z [7]). Prvním zásadním pravidlem je nutnost normalizovat navrhované tabulky. To znamená vyvarovat se vzniku duplicitních dat. Tento stav nastává zejména v případech, kdy se velké množství dat ukládá pouze do jedné tabulky. Pokud bude aplikované pravidlo normalizace, bude tato jediná tabulka rozdělena do více samostatných a jednotlivé záznamy budou provázány pomocí tzv. klíčů. Existují dva základní klíče. Prvním je tzv. primární klíč, ten musí být v celé tabulce unikátní. S ohledem na další návrhové pravidlo je nutné vhodně zvolit primární klíč a to tak, aby byl samostatný bez jiného významu. Pokud by se toto pravidlo nedodrželo a v budoucnu by se objevila nutnost změnit hodnotu, která je použitá jako primární klíč, bylo by to velmi obtížné. Druhým dostupným klíčem je tzv. cizí klíč, ten představuje samotné vazby (relace) mezi

tabulkami. Pokud má tabulka pole cizího klíče, do tohoto pole se ukládá právě primární klíč řádku z tabulky na kterou vytváříme vazbu. Přímo na tuto problematiku navazuje pravidlo referenční integrity. Problémy referenční integrity (nekonzistence dat) se projeví zejména ve chvíli, kdy je třeba měnit data, nebo mazat celé záznamy. Pro dodržení tohoto pravidla se nejčastěji využívá kaskádovitých operací, ty nám umožní např. při odstranění záznamu automaticky kaskádovitě odstranit všechny ostatní navázané prvky.

Dále je nutné rozdělit vazby, které mohou v relační databázi nastat. Prvním případem je vazba *one-to-one*, to znamená, že jednomu záznamu odpovídá pouze jeden další záznam. Tento typ vazby není často používaný, jelikož tato data lze obvykle uložit do jediné tabulky. Ani v této práci nebude použit. Druhou vazbou je *one-to-many*. Tato vazba je velmi často používaná, pomocí ní můžeme jeden záznam přidělit libovolnému množství dalších záznamů. V tomto konkrétním případě bude tento typ vazby použit např. pro spárování měřicího modulu a uživatele – jeden uživatel může mít libovolné množství měřicích modulů. Poslední vazbou je *many-to-many*, tato vazba dovoluje propojit libovolný počet záznamů jednoho typu s libovolným počtem záznamů jiného typu. Pro realizaci této vazby je nutné vytvořit novou tabulku, pomocí které budou spojovány primární klíče jednotlivých záznamů. Tato vazba nebude opět v tomto případě použita, jelikož není nadefinovaná funkce, která by použití vyžadovala.

S ohledem na zmíněná návrhová pravidla a dostupné vazby byla navržena struktura databáze s deseti následujícími tabulkami. Výslednou strukturu navržené databáze lze vidět na obr. 10.

Tabulka *User*

Tabulka *User* bude obsahovat všechny údaje o uživateli, které zadává při registraci, tj. křestní jméno, příjmení, emailovou adresu, tituly a heslo. Heslo nebude z bezpečnostních důvodů ukládáno v čitelné podobě, ale bude jednosměrně převedeno do nečitelné podoby tak, aby nemohlo být převedeno zpět, jedná se o tzv. hashování. Dále bude v databázi uložen obrazový soubor (avatar uživatele) v binární podobě.

Ostatní údaje nebudou pro uživatele viditelné, ale jsou nezbytné pro provoz aplikace. Mezi tyto hodnoty patří unikátní identifikátor uživatele (sloužící k vytvoření relací – viz. výše), datum registrace a stav uživatele. Tato stavová informace je důležitá pro rozlišení etapy uživatelovi registrace – pro úspěšnou registraci je nutné ověřit emailovou adresu, pokud uživatel ověření neprovedl, systém musí na tuto skutečnost reagovat, proto tato informace musí být obsažena ve stavu uživatele.

Tabulka *UserLink*

Tabulka *UserLink* bude sloužit pro uložení generovaných stránek, jejichž odkaz se posílá uživateli prostřednictvím emailu. Mezi tyto stránky patří žádost o změnu hesla v případě jeho ztráty a stránka pro ověření autentičnosti emailové adresy.

Tabulka musí obsahovat cizí klíč vázaný na konkrétního uživatele, datum vytvoření odkazu (z důvodu omezené platnosti), stav odkazu (připravený, použitý,

exspirovaný apod.) a druh odkazu, který rozliší o jaký druh stránky se jedná. Posledním záznamem je opět primární klíč, ten je přímo obsažen v generované URL adrese a pomocí něj je uživatelův požadavek převeden na konkrétní záznam v tabulce a je vykonána patřičná akce.

Tabulka *Module*

V tabulce *Module* budou uloženy všechny uživatelem zaregistrované moduly, tedy i všechny uživatelem zadané parametry (název, popis, umístění). Jak již bylo zmíněno výše, při registraci je vygenerován tzv. autorizační token, který je použit pro spárování modulu se systémem. Tento autorizační klíč je představen unikátním identifikátorem, který je současně primárním klíčem v tabulce.

Dále jsou zde uloženy informace týkající se modulu – konkrétně jeho MAC adresa, unikátní HW ID. Dále je to i datum vytvoření záznamu, datum registrace, AES klíč (ten je použit pro šifrování/dešifrování komunikace s modulem), cizí klíč uživatele (pro rozlišení vlastníka) a opět stavová informace, která informuje o stavu modulu – dostupné jsou stavy: zatím nepřipojen (myšleno po vytvoření registrace), aktivní, deaktivovaný.

Tabulka *Sensor*

Do tabulky *Sensor* budou ukládány dostupné senzory připojených modulů. Záznamy budou vkládány pouze dle příchozích dat z měřicích modulů. Tato tabulka bude pouze propojovat konkrétní modul s daným druhem senzoru, navíc zde bude pouze pořadové číslo senzoru (v rámci jednoho modulu), a to pro případy kdy by jeden měřicí modul obsahoval více stejných senzorů.

Tabulka *SensorType*

Do tabulky *SensorType* nebude mít uživatel možnost zapisovat. Zde budou zaznamenány všechny podporované druhy senzorů. Pomocí hodnot v této tabulce budou příchozí data rozpoznána a tím bude moci systém uživateli konkrétně pojmenovat senzory, přiřadit fyzikální jednotky atd.

Bude zde uloženo interní označení snímané veličiny (tím jsou označena příchozí data). Dále bude uložen název senzoru (zobrazovaný uživateli), fyzikální jednotka, značka a informace týkající se následné vizualizace dat z těchto senzorů. Mezi tyto informace patří počet desetinných míst na které se zaokrouhlí zobrazovaná hodnota a limitní hodnoty zobrazovacích komponent.

Tabulka *ModuleData*

Zde budou ukládány všechny naměřené hodnoty. Jednotlivé hodnoty budou spojeny pomocí cizího klíče s konkrétním senzorem. Dále bude v tabulce zaznamenán čas vložení, čas měření (nemusí být totožný s časem vložení vzhledem k době přenosu dat apod.) a samozřejmě naměřená hodnota.

Tabulka *ModuleNotification*

Tabulka *ModuleNotification* bude obsahovat uživatelem nadefinované notifikace. Bude tedy obsahovat všechny informace, které uživatel vyplnil při tvorbě notifikace - jmenovitě je to klíč senzoru, jehož příchozí data se kontrolují, text notifikace a parametry hodnot, které vyvolají samotnou notifikaci. Poslední hodnotou je příznak, zda je notifikace aktivní.

Tabulka *ModuleNotificationHistory*

Tabulka *ModuleNotificationHistory* je nutná pro správnou funkci odesílání notifikací. Zde budou automaticky ukládány záznamy o odeslaných notifikacích, a to zejména z důvodu vytvoření prodlevy v odesílání totožných notifikací. Díky tomuto může být uživatel notifikován např. pouze jednou za hodinu, i když data přesahují limit v každém např. minutovém měření.

Tabulka *ModuleConfig*

V tabulce *ModuleConfig* budou uloženy konfigurační data měřicích modulů. Pokud uživatel změní konfiguraci modulu, nová konfigurace se uloží do tabulky. Bude nutné uložit klíč modulu, který je konfigurován, stav konfigurace (nekonfigurováno, konfigurováno a odvoláno), typ konfiguračních dat (změna intervalu měření, kalibrační ofset senzoru) a nově nastavená hodnota. V případě, že se konfigurace týká konkrétního senzoru, bude uložen i záznam, o jaký senzor se jedná. Na závěr pak bude automaticky doplňován čas konfigurace (pouze z informativních důvodů).

Tabulka *DashboardItem*

Tabulka *DashboardItem* bude obsahovat uživatelovu konfiguraci stránky Dashboard. Bude zde obsažen klíč uživatele, aby mohl být záznam spojen s konkrétním uživatelem. Také bude nutné spojit komponentu s konkrétním senzorem, rovněž pomocí cizího klíče. Dále bude definován typ vizualizační komponenty (případně její velikost, pokud bude více připravených velikostí) a uživatelem nastavený titulek komponenty. V případě že se bude jednat o komponentu typu Graf, bude nastaven rozsah zobrazovaných hodnot. Posledním parametrem pak bude pořadí jednotlivých komponent, to bude přepočítáváno automaticky, dle uživatelova aktuálního rozvržení.

5 Implementace vlastního řešení

Po navržení aplikace lze postoupit k implementaci. V této části bude ještě nutné upřesnit konkrétní postupy týkající se samotné implementace. Mezi tyto kroky spadá zejména výběr technologií třetích stran, které budou použity, či navržení konkrétní komunikace s měřicími moduly.

5.1 Použité technologie

Jelikož by implementace některých funkcionalit (např. vizualizace naměřených hodnot do grafů, obsluha databáze apod.) byla příliš náročná, byly použity technologie, které minimalizují nutné vynaložené úsilí k implementaci daných funkcionalit. Tyto technologie byly použity, jak na front-endové části (logika implementovaná do prohlížeče uživatele) a to zejména pro práci s vizualizačními komponentami a další úlohy týkající se elementů zobrazovaných na stránce, tak i na straně back-endové. Konkrétně pro práci s databází pomocí databázového modelu, tvorbu logů aplikace a mimo jiné i zprostředkování real-time komunikace s uživatelem.

Při výběru jednotlivých technologií bylo také hleděno na licenci, pod kterou jsou technologie nabízeny. Buď byly technologie zcela uvolněny k volnému použití, případně byly použity v souladu s licenčními podmínkami pro neziskové společnosti/osobní použití, což tato práce splňuje.

5.1.1 Front-end

Front-endovou částí aplikace je rozuměna ta část, se kterou uživatel přichází přímo do styku. Uživatel tedy bude v přímém kontaktu s výstupy následujících technologií.

Bootstrap

Knihovna Bootstrap nabízí velké množství komponent pro webové stránky. Tyto komponenty lze brát za jakýsi základ, který ulehčí např. tvorbu layoutu stránky, vytvoření základních kontrolek jakou jsou tlačítka, navigační menu apod. Obvykle je třeba tento základ dále rozvíjet a přizpůsobovat konkrétním požadavkům aplikace.

Mezi zásadní výhody použití frameworku Bootstrap patří plná responzivita všech komponent. Připravený layout obsahuje čtyři různá velikostní zobrazení, která se přizpůsobují šířce zobrazovacího zařízení. Další výhodou je připravený tzv. grid. Pomocí něj lze obsah stránky (jednotlivé komponenty) orientovat do různě širokých sloupců, což umožňuje zobrazit více informací vedle sebe. Toho bylo použito například na

Dashboardu k rozmístění vizualizačních komponent. Dalším důležitým prvkem jsou modální okna, které nevyžadují další tvorbu skriptů a lze je otevírat/zavírat pouze pomocí tlačítek s patřičně nadefinovanými atributy.

Pro potřeby navržené aplikace byla knihovna Bootstrap aplikována zejména na layout stránky, grid, navigační menu, modální okna a základní komponenty (tlačítka, tabulky). Ovšem téměř vše zmíněné bylo dále upravováno tak, aby vyhovovalo konceptu navržené aplikace.

jQuery

jQuery je jednou z nejpoužívanějších javascriptových knihoven ve webových aplikacích (dostupné z [8]). Pomocí jQuery je možné velmi usnadnit vývoj skriptů pracujících v prohlížeči uživatele. Pomocí této knihovny je možné velmi snadno maninpulovat s prvky umístěnými na stránce (je možné měnit jejich atributy, tedy např. měnit jejich polohu na stránce, funkci, barvu atd.). Také je možné tyto prvky ze stránky dynamicky odebírat, či přidávat nové.

Velmi důležitou součástí knihovny jQuery je technika AJAX. Což je „zapouzdření“ XMLHttpRequestu, čímž je docíleno možnosti komunikovat se serverem po načtení stránky (na pozadí). Díky tomu je možné např. na pozadí odeslat formulář a ihned zobrazit případné validační zprávy – bez nutnosti znovu-načtení celé stránky.

jQuery UI

jQuery UI rozšiřuje knihovnu jQuery. Samostatné použití této technologie tedy není možné. Knihovna se zaměřuje na uživatelské rozhraní/interakci s uživatelem. Nabízí různé widgety (např. dialogová okna, záložky, kalendář, indikátor průběhu apod.). Dále jsou dostupné efekty, které umožňují aplikovat různé animace.

V případě této aplikace byla z knihovny jQuery UI použita pouze část zaměřená na interakci s uživatelem. V této části je např. funkce uživatelské změny elementu na stránce, nebo možnost přesouvání libovolných elementů po stránce tažením myši. Ale tato aplikace používá pouze funkcionality, jenž umožňuje řadit vybrané elementy na stránce. Toho bylo použito na stránce Dashboard, kde si uživatel může sám jednoduše měnit pořadí (tedy i polohu) jím přidaných vizualizačních komponent.

Highcharts

Knihovna Highcharts nabízí mnoho interaktivních vizualizačních komponent (včetně grafů). K dispozici jsou základní liniové grafy, sloupcové, koláčové, ale i specifičtější jako bublinové, či např. tepelné mapy. Jednotlivé druhy grafů lze i kombinovat, tedy vyjádřit více datových sérií na jednom plátně. K dispozici jsou i jednoduchá měřidla, která zobrazují pouze jednu hodnotu.

Pro vizualizaci měřených hodnot v čase byl použit liniový graf a pro zobrazení aktuální hodnoty jednoduché měřidlo. Grafy obsahují možnost přiblížení v časové ose, což je vhodné při zobrazení velkého počtu dat. A zobrazují informace (přesná

hodnota, konkrétní čas měření, případně název měřené veličiny) o konkrétním bodu formou tooltip zprávy, zobrazující se po najetí myši na daný bod.

5.1.2 Back-end

Back-end je zde chápán jako část aplikace, která je uživateli zcela neznámá. A proto i technologie použité v této části aplikace jsou uživateli zcela neznámé a jejich použití nemá přímý vliv na uživatelův prožitek z používání aplikace. Byly použity pouze pro zjednodušení implementační náročnosti, tedy i zvýšení efektivity vývoje, případně pro optimalizaci zdrojového kódu aplikace.

Všechny následující technologie jsou volně dostupné prostřednictvím tzv. NuGetů. NuGety jsou jakési balíčky obsahující zkompilovaný kód (obdobné knihovnám DLL). Tyto balíčky je pak možné velmi snadno instalovat do projektu pomocí vývojového prostředí. Pomocí něj lze i balíčky aktualizovat.

NLog

Platforma NLog umožňuje produkci a správu „logů“ v aplikaci. Je možné vytvářet zprávy různých úrovní. Úrovní je k dispozici celkem šest, ovšem mezi ty základní patří pravděpodobně zpráva informační, varovná, chybová a fatální. Pro použití této technologie je nutné provést konfiguraci. Ta se skládá zejména z nastavení výstupů. Mezi podporované výstupy patří např. konzole (v případě, že aplikace je spuštěna v konzoli), textový soubor, nebo email. Různé výstupy lze i přiřadit pro zprávy různých úrovní. Dále je možné konfigurovat formát ukládané zprávy (např. vložení aktuálního času, výpis místa vzniku zprávy apod.)

Entity Framework

Entity Framework je představitelem objektově relačního mapování (zkr. ORM). Tato technologie zajišťuje konverzi mezi relační databází a objektově orientovaným zdrojovým kódem. Díky tomu lze při vývoji pracovat pouze s objekty představující skutečné entity v databázi. Entity Framework následně tuto práci s objekty převádí automatizovaně do jazyka SQL, který je využívaný pro komunikaci s relační databází.

Existují dva základní přístupy k takovéto práci s databází. První přístup je vhodné zvolit, pokud je třeba začít pracovat s již existující databází (nejčastěji naplněné daty). Tímto přístupem je Entity Framework schopen vygenerovat všechny potřebné datové modely do zdrojového kódu aplikace dle dané databáze. Druhým přístupem je tzv. *Code First*, ten byl mimo jiné použit i pro implementaci navrhované aplikace. Jedná se o přístup, kdy vývojář nejprve vytvoří datový model a následně jej pomocí Entity Frameworku migruje do databáze. Tedy změnou datového modelu v kódu se změní i struktura databáze.

SignalR

Knihovna SignalR umožňuje „real-time“ komunikaci mezi back-endem a front-endem. Vzhledem k tomuto faktu její zařazení do back-endových technologií není příliš jednoznačné, ale tato technologie je primárně instalována na back-end a s front-enovou částí (prohlížečem uživatele) komunikuje pomocí websocketů, které jsou ve webových prohlížečích standardizovány. Ale i tak je pro front-endovou část aplikace vhodné použít knihovnu, která umožní připojení k tzv. *Hubu* běžícím na back-endu. Tento *Hub* umožňuje předem nadefinovat metody, které mohou následně klienti „volat“ a tím např. zahajovat předem definovanou akci. Oproti AJAXU má tento druh komunikace výhodu v tom, že je možné klienta zpětně informovat např. o průběhu dané operace apod.

Pro potřeby této práce byla technologie SignalR použita pouze pro komunikaci směrem z back-endu na prohlížeč uživatele. Díky tomu je možné na Dashboardu uživatele „real-time“ aktualizovat vizualizační komponenty, v okamžiku přijmutí dat z měřicího modulu.

5.2 Implementace šifrované komunikace s měřicími moduly

Veškerá komunikace je jednosměrná, konkrétně směrem z měřicích modulů na server. Komunikace tedy musí být vždy podnícena měřicím modulem vytvořením požadavku (HTTP Requestu) na API serveru. Veškerá data, která je nutno přenést do měřicího modulu, jsou odesílána jako odpověď (HTTP Response) právě na zmíněný požadavek daného modulu.

Všechna data jsou přenášena výhradně ve formátu JSON. Formát přenášených dat je přesně nadefinovaný a nelze jej měnit. Jelikož jsou konkrétní informace šifrovány pomocí algoritmu AES-128, lze je přenést pomocí jednoho textového řetězce. Proto server očekává příchozí data pouze formou jednoho textového řetězce a až následně je prováděno převedení do konkrétní podoby.

5.2.1 Registrace modulu

Registrace modulu je prvním krokem k připojení nového zařízení do systému. Proto musí odeslaná data obsahovat autorizační token, který byl uživateli vygenerován při vytvoření nového modulu v systému. Data dále obsahují seznamy všech připojených senzorů k modulu a seznam uložených konfiguračních dat. Poslední zásadní přenášenou informací je unikátní sériové číslo modulu, které je používáno pro veškerou následující komunikaci ke spárování požadavku s konkrétním modulem (místo autorizačního tokenu).

Pro šifrování přenášených informací je použit společný šifrovací klíč a společný inicializační vektor. Tento šifrovací pár je společný pro celý systém. Tedy všechny registrační požadavky lze rozšifrovat pouze pomocí tohoto jediného páru. Součástí registrace měřicího modulu je mimo jiné i vygenerování unikátního šifrovacího klíče.

Tím je zajištěno, že následná komunikace není mezi moduly zaměnitelná. Šifrovací klíč je vygenerován na straně serveru a měřicímu modulu je předán formou odpovědi na jeho registrační požadavek. Další informace o registračním požadavku, včetně výpisu datové struktury a návratových hodnot jsou uvedeny zde:

- **URL**
  /api/ModuleRegister
- **Metoda**
  POST
- **Struktura odesílaných dat**

  ```
  {
      "data":  "string"
  }
  ```

  Přičemž hodnotou parametru *data* je následující struktura zašifrovaná algoritmem AES-128 společným šifrovacím klíčem a společným inicializačním vektorem

  ```
  {
      "chipId":  "number",
      "mac":  "string",
      "voltage":  "number",
      "rssi":  "number",
      "authorizationToken":  "uuid",
      "availableSensors":  ["string"],
      "configurations":  "object",
      "registrationTime":  "dateTime",
  }
  ```

- **Potvrzující návratové hodnoty**
  - **Kód:** 200 OK
    Obsah: [EncryptedAesKey]
    Význam: Unikátní šifrovací klíč, který je použit pro veškerou následující komunikaci
- **Chybové návratové hodnoty**
  - **Kód:** 400 Bad Request
    Obsah: ERR1
    Význam: Neplatná příchozí data
  - **Kód:** 400 Bad Request
    Obsah: ERR2
    Význam: Neznámý modul

- **Kód:** 400 Bad Request
  Obsah: ERR3
  Význam: Neplatný formát zašifrovaných dat
- **Kód:** 400 Bad Request
  Obsah: ERR101
  Význam: Chybějící příchozí data
- **Kód:** 400 Bad Request
  Obsah: ERR102
  Význam: Modul v databázi není v očekávaném stavu
- **Kód:** 400 Bad Request
  Obsah: ERR103
  Význam: Neplatné unikátní identifikátory modulu (ID čipu, nebo MAC adresa)
- **Kód:** 400 Bad Request
  Obsah: ERR200
  Význam: Neidentifikovatelná chyba

5.2.2 Archivace dat/zpětná konfigurace

Požadavek o archivaci dat je na rozdíl od registrace modulu prováděn periodicky. Každou sérii naměřených hodnot je třeba odeslat na server. Data mohou obsahovat neomezené množství různorodých naměřených hodnot. Jedinou podmínkou je jejich jednotný čas měření. A to zejména z důvodu, že příchozí data mimo naměřených hodnot obsahují i konkrétní čas měření, čímž je možné provádět vizualizace dat s přesnějším časem měření. Odpovědí na archivaci hodnot jsou případná konfigurační data, která byla uživatelem vytvořena po poslední úspěšné archivaci. Do modulu jsou tedy přenášena pouze nová a neaplikovaná konfigurační data.

Všechna zmíněná komunikace je šifrována opět pomocí algoritmu AES-128, ale již s unikátním šifrovacím klíčem, inicializační vektor je opět sdílený. Z tohoto důvodu musí být příchozí data rozdělena do dvou částí. V první je unikátní HW číslo modulu, pomocí kterého je v systému vyhledán potřebný dešifrovací klíč. Zbylá data (včetně měřených hodnot) jsou již přenášena v zašifrované podobě. Další informace jsou opět uvedeny zde:

- **URL**
  /api/DataSave
- **Metoda**
  POST
- **Struktura odesílaných dat**

```
{
    "data":  "string string"
}
```

Přičemž hodnotou parametru *data* jsou dva textové řetězce oddělené mezerou. Prvním je unikátní ID čipu (nutné k nalezení patřičného unikátního dešifrovacího klíče). Druhým řetězcem je následující struktura zašifrovaná algoritmem AES-128 unikátním šifrovacím klíčem získaným při registraci a společným inicializačním vektorem

```
{
    "chipId":   "number",
    "sensors": [{
          "type'':  "string",
          "sn'':  "number",
          "value'':  "number"
    }, ...],
    "voltage":   "number",
    "rssi":   "number",
    "loggedTime":   "dateTime",
}
```

- **Možné potvrzující návratové hodnoty**
  - **Kód:** 200 OK
    Obsah: [ConfigurationData]
    Význam: Seznam všech neaplikovaných konfiguračních dat ve formátu

    ```
    [{
        "type": "number",
        "entity": "string",
        "sn": "number",
        "value'': "string"
    }, ...]
    ```

- **Možné chybové návratové hodnoty**
  - **Kód:** 400 Bad Request
    Obsah: ERR1
    Význam: Neplatná příchozí data
  - **Kód:** 400 Bad Request
    Obsah: ERR2
    Význam: Neznámý modul
  - **Kód:** 400 Bad Request
    Obsah: ERR3
    Význam: Neplatný formát zašifrovaných dat
  - **Kód:** 400 Bad Request
    Obsah: ERR4
    Význam: Nepodporovaný druh senzoru

– **Kód:** 400 Bad Request
Obsah: ERR101
Význam: Chybějící příchozí data
– **Kód:** 400 Bad Request
Obsah: ERR102
Význam: Modul v databázi není v očekávaném stavu
– **Kód:** 400 Bad Request
Obsah: ERR200
Význam: Neidentifikovatelná chyba

5.3 Implementace uživatelského rozhraní

Uživatelské rozhraní bylo implementováno do několika stránek. K této implementaci byly použity dva různé přístupy. První byl prostý, spočíval v pouhém vykreslení kompletní stránky v jeden okamžik. Tento způsob byl použit na jednoduchých stránkách, které obsahovali jen přihlašovací, či registrační formulář apod. Pro případy, kdy je třeba uživateli poskytnout více informací (typicky pokud jejich získání trvá delší časový úsek) byla stránka rozdělena do dílčích bloků. Toho bylo použito např. na stránce Dashboard, kde např. načítání naměřených hodnot v delším časovém horizontu může trvat několik set milisekund. Celý princip spočívá v prvotním vykreslení stránky s připravenými bloky a následném asynchronním načtení obsahu všech dílčích bloků. Tím je uživateli kontinuálně zajišťován přísun informací bez zdlouhavého čekání na načtení všech dat.

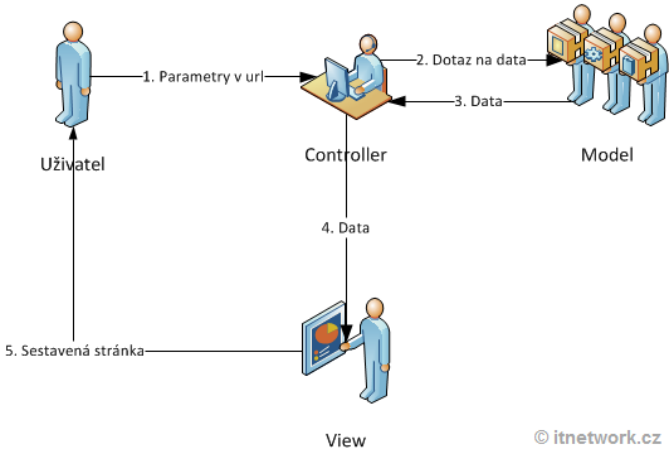
5.3.1 Architektura MVC

Pro implementaci uživatelského rozhraní byla použita architektura MVC, která je přímo podporovaná technologií ASP.NET pod označením Razor MVC. Jak je možné dočíst se přímo v dokumentaci dostupné z [9], tato architektura rozděluje aplikaci na tři základní celky – na Model (Model), View (Pohled) a Controller (Kontrolér). V tomto rozdělení uživatel přímo interaguje s pohledem, v něm může například odeslat formulář, stisknout tlačítko apod. Tato informace je následně odeslána do controlleru, kde je akce rozpoznána a zpravidla je prováděna akce s daty. Tato akce probíhá pomocí modelu. V případě této aplikace se jedná o databázový model. Přijatá data z modelu jsou v controlleru následně předána opět do pohledu, kde je již zajištěné jen jejich vykreslení uživateli. Diagramové znázornění architektury MVC lze vidět na obr. 9.

5.3.2 Další postupy

Dále byl pro implementaci v několika případech použit návrhový vzor MVVM (Model-View-ViewModel), který lze zařadit mezi výše zmíněný View (pohled) a Model. Tím bylo docíleno možnosti upravit uživatelské kontrolky bez přímé závislosti na datovém modelu. Toho bylo použito zejména na formulářích. Většina CRUD

formulářů (Create, Read, Update, Delete – formuláře umožňující základní operace s vybraným objektem) byla implementována do modálních oken, čímž byl snížen počet potřebných stránek a současně tím byl uživateli zjednodušen pohyb po celém rozhraní. Výslednou podobu uživatelského rozhraní lze vidět na obr. 11, 12, 13.



Obrázek 9: Diagram MVC architektury (dostupný z [10])

5.4 Nasazení aplikace

Nasazení aplikace na produkční server bylo posledním krokem implementace navrženého systému. Tuto operaci lze rozdělit do několika kroků. Nejprve bylo nutné na produkční prostředí připravit databázi. Jelikož server již obsahuje Microsoft SQL Server Express, stačilo vytvořit produkční databázi a aplikovat připravené migrace vytvořené při implementaci. Jelikož aplikace vyžaduje v databázi definice podporovaných druhů senzorů, bylo nutné je do databáze doplnit.

Následně bylo nutné aplikaci tzv. vypublikovat. Tím získáme ze zdrojového kódu přeloženou aplikaci, kterou lze spustit na produkčním serveru. Touto publikací se zároveň aplikuje konfigurace platná pro produkční prostředí (nastavení logování, údaje o databázi apod.). Následně je třeba na produkčním serveru pomocí správce IIS (internetová informační služba), která je dostupná na serveru, vytvořit nový web, nastavit vazby a zvolit cestu k vypublikované aplikaci. Mezi nastavení vazeb spadá konfigurace cílové URL adresy, cílového portu, příp. i výběr SSL certifikátu.

Následně již stačilo vypublikovanou aplikaci umístit do připravené složky, této složce nastavit patřičná oprávnění pro nově vytvořený web a samotnou aplikaci spustit opět pomocí správce IIS. Tím se dokončila konfigurace serveru a vytvořená aplikace byla spuštěna a uveřejněna.

5.5 Testování

Testování aplikace lze rozdělit do dvou částí. V první části bylo nutné otestovat vytvořené API, zda plní všechny předpokládané funkce apod. Dále bylo nutné provést testování uživatelského rozhraní. Testování probíhalo vždy ve dvou fázích. Jednak bylo prováděno testování v průběhu vývoje (kontinuálně) a následně po dokončení implementace na dočasném testovacím prostředí.

5.5.1 Testování API

Vývoj API byl rozčleněn do dvou kroků. Nejprve byly implementovány akce a jejich procedury pro nešifrovaná data a až následně po ověření funkčnosti byla implementována podpora šifrovaného spojení. Jelikož první etapa byla pouze dočasná, a cílem bylo v krátkém časovém úseku připravit API, které bude schopné archivovat měřené hodnoty, nebyla v průběhu vývoje vyvořena separátní aplikace pro testování, ale byla použita, již existující, volně dostupná. Konkrétně bylo použito doplňku „RESTClient“, který je volně instalovatelným doplňkem pro webový prohlížeč Mozilla Firefox. Tento doplněk umožnil odesílat testovací data a pohodlně ověřit všechny návratové hodnoty. V druhé etapě pokračovalo testování již s reálnými moduly ESP8266. Toto testování bylo nutné zejména pro odladění šifrování, resp. dešifrování příchozích dat. Další dlouhodobé testy pobíhaly rovněž s těmito měřicími moduly.

5.5.2 Testování uživatelského rozhraní

Testování uživatelského rozhraní probíhalo současně s vývojem a bylo prováděno přímo na vývojovém prostředí pomocí různých webových prohlížečů (konkr. pomocí Mozilla Firefox, Internet Explorer, Chrome a Microsoft Edge - všechny nejnovější verze). Testování bylo soustředěno zejména na scénářové testy, při kterých bylo cílem ověřit každou nově implementovanou funkcionalitu. Jelikož je aplikace navržena jako plně responzivní, bylo prováděno i testování na mobilních zařízeních (zejm. na zařízení iPhone 7) a to primárně z důvodu ověření ovladatelnosti aplikace při malých rozlišeních a ověření čitelnosti vytvořených zobrazovacích komponent. Po dokončení implementačních prací byl proveden akceptační test, při kterém se ověřovala funkčnost vytvořené aplikace oproti zadání a vytvořenému návrhu. Dále byla testováním ověřena schopnost aplikace vypořádat se s neočekávanými podněty/daty ze strany uživatele a na tyto podněty zareagovat, tím byl otestovaný tzv. Error Handling.

5.5.3 Výsledky testování

Testování je nepostradatelnou součástí vývoje, což se potvrdilo i v případě této aplikace. Prováděním testů bylo odhaleno velké množství chyb, které byly následně opraveny. Mimo jiné bylo průběžné testování API nápomocno při implementaci šifrované komunikace. To zejména z důvodu neznalosti kompletních informací o příchozích šifrovaných datech z modulů ESP8266, jejichž dokumentace v tomto směru není kompletní.

6 Závěr

Práce byla zaměřena na návrh a implementaci serverové aplikace umožňující sběr a vizualizaci měřených dat pomocí modulů ESP8266. Nejprve však bylo nutné provést rešerši již existujících řešení poskytujících podobnou funkcionalitu. Součástí této rešerše bylo porovnání pěti různých služeb, jejich uživatelských rozhraní, jejich možností komunikace s měřicími moduly a příp. další více specifické funkcionality. Po dokončení rešerše byla provedena analýza požadavků a vytvořen návrh vlastní serverové aplikace umožňující vše výše zmíněné. Následně byla samotná aplikace implementována dle vytvořeného návrhu. Dokončená aplikace byla publikována na produkční server, kde probíhalo dlouhodobé testování s měřicími moduly.

Výsledkem práce je vytvořený nezávislý systém, který je schopný archivovat a vizualizovat měřená data uživatelům. Vizualizace dat je dostupná pomocí liniového grafu pro znázornění časového průběhu a měřidlové komponenty pro zobrazení jedné (poslední) hodnoty. Mimo jiné nabízí možnost „vzdálené“ konfigurace měřicích modulů, pomocí níž je možno prostřednictvím webového rozhraní např. měnit interval měření, kalibrovat jednotlivé senzory apod., což neumožňuje žádná ze služeb, se kterou jsem se v průběhu rešerše setkal. Dále vytvořený systém uživateli umožňuje automaticky spárovat podporované senzory připojené k měřicímu modulu s konkrétními daty ve webovém rozhraní, což také není obvyklé pro rešeršované služby. Poslední implementovanou funkcí je funkce notifikací, pro které si může uživatel nakonfigurovat vlastní podmínky spuštění.

Možností k pokračování této práce může být více, například rozšíření o podporu dalších protokolů pro komunikaci s měřicími moduly. Tím by bylo možné zajistit větší rozšíření služby, příp. i vytvoření plně oboustranné komunikace. Rozšířit by bylo možné i uživatelské rozhraní a to např. o více vizualizačních komponent, které by umožňovali prolnutí více datových sérií atd. Avšak zřejmě nejpodstatnějším rozšířením by byla integrace NoSQL databáze do informačního systému. Tato integrace v současné době nemá opodstatnění, ale v budoucnu při dlouhodobém periodickém sběru dat je pravděpodobný nárůst dob odezev systému, což by rozšíření o NoSQL databázi pravděpodobně zoptimalizovalo.

Použitá literatura

[1] NOLL, Jan. *IoT — modul pro sběr environmentálních dat ve vnějším prostředí*. Liberec, 2017. Bakalářský projekt. Technická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií.

[2] *IoT Analytics - ThingSpeak Internet of Things* [online]. Natic (Massachuttes): The MathWorks, ©2018 [cit. 2018-03-18]. Dostupné z: https://thingspeak.com/

[3] *Beebotte* [online]. Paris: Beebotte, ©2012-2017 [cit. 2018-03-18]. Dostupné z: https://beebotte.com/

[4] *Corlysis: the platform for storin...* [online]. Corlysis, ©2017 [cit. 2018-03-18]. Dostupné z: https://corlysis.com/

[5] *IoT platform | Internet of Things | Ubidots* [online]. Doral (Florida): Ubidots, ©2018 [cit. 2018-03-18]. Dostupné z: https://ubidots.com/

[6] *Freeboard - Dashboards For the Internet Of Things* [online]. Bug Labs, ©2018 [cit. 2018-03-18]. Dostupné z: https://freeboard.io/

[7] 7 smrtelných hříchů návrhu databáze. *Embarcadero CZ+SK* [online]. Embt.biz, c2008-2012 [cit. 2018-03-31]. Dostupné z: Embarcadero CZ+SK

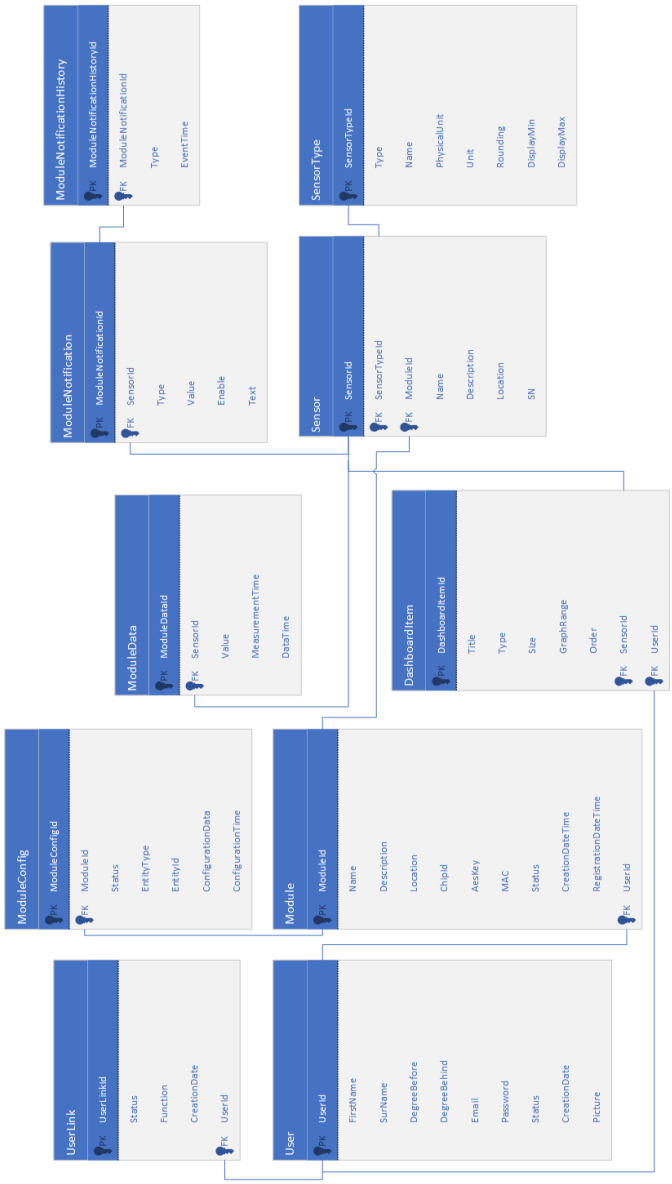
[8] Best JavaScript Frameworks, Libraries and Tools to use in 2017. *SitePoint* [online]. Collingwood (Australia): SitePoint Pty., c2000-2018 [cit. 2018-04-14]. Dostupné z: https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/

[9] ASP.NET MVC Overview. In: *Microsoft Developer Network | MSDN* [online]. Dublin: Microsoft, c2018 [cit. 2018-04-22]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[10] MVC architektura. In: *Itnetwork.cz - Ajťácká sociální síť...* [online]. Praha: Čápka, c2018 [cit. 2018-04-22]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

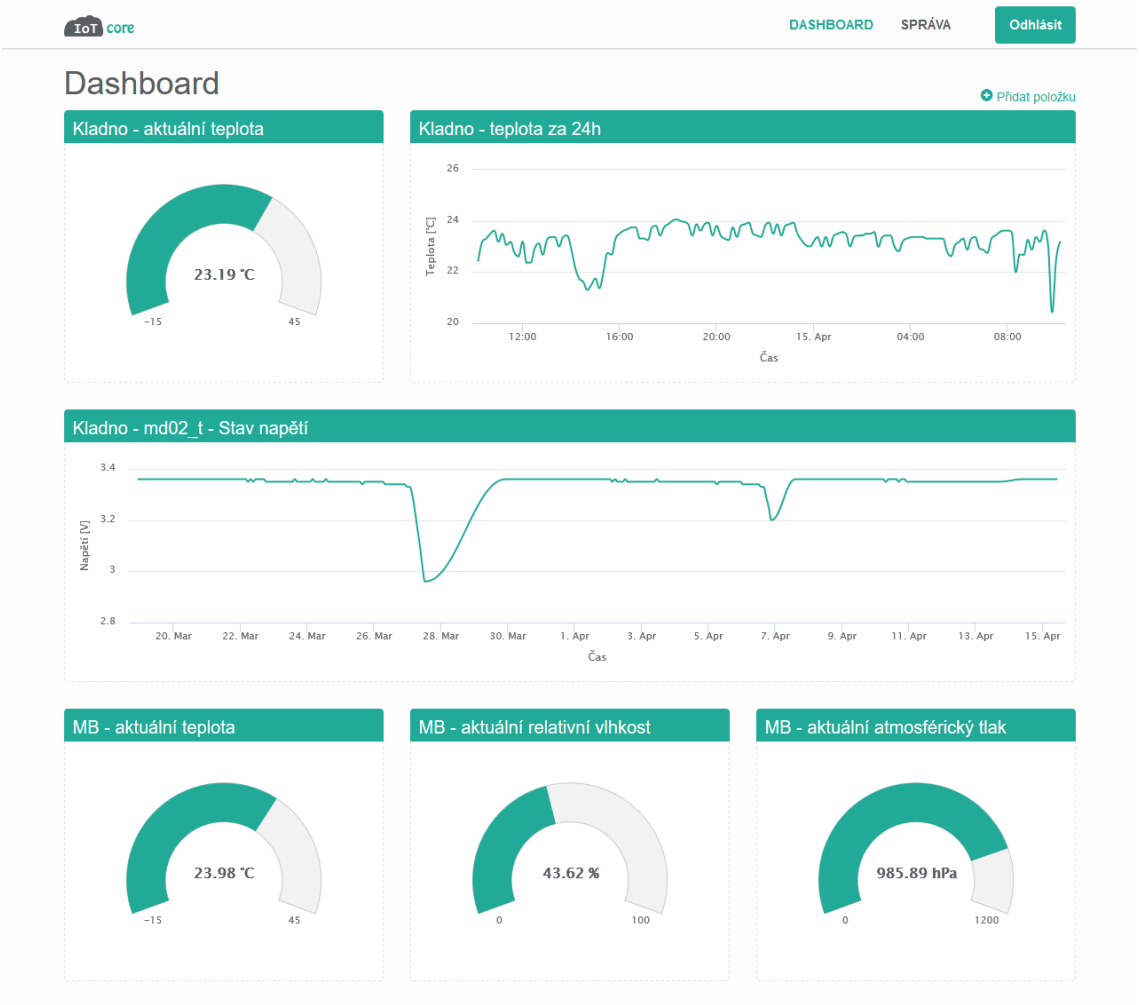
Přílohy

A Struktura databáze

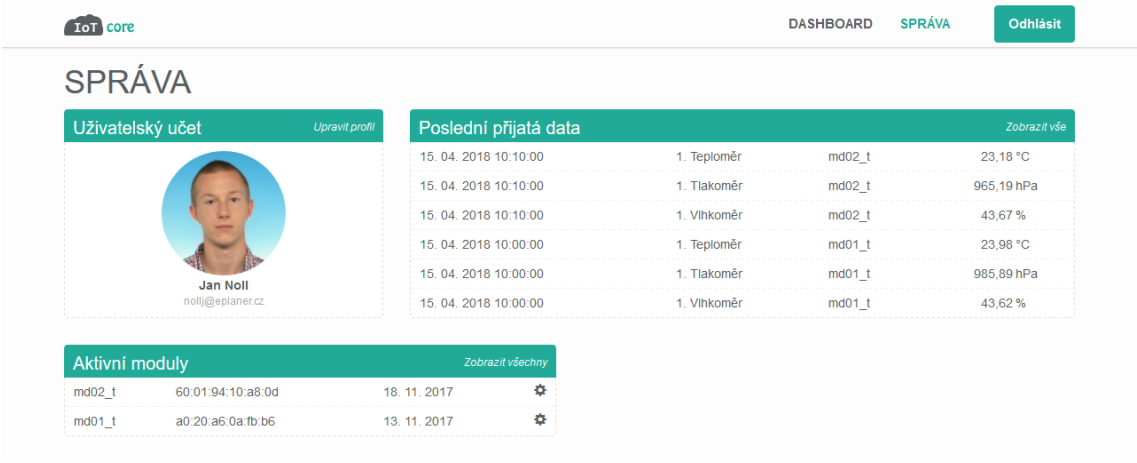


Obrázek 10: Struktura navržené databáze

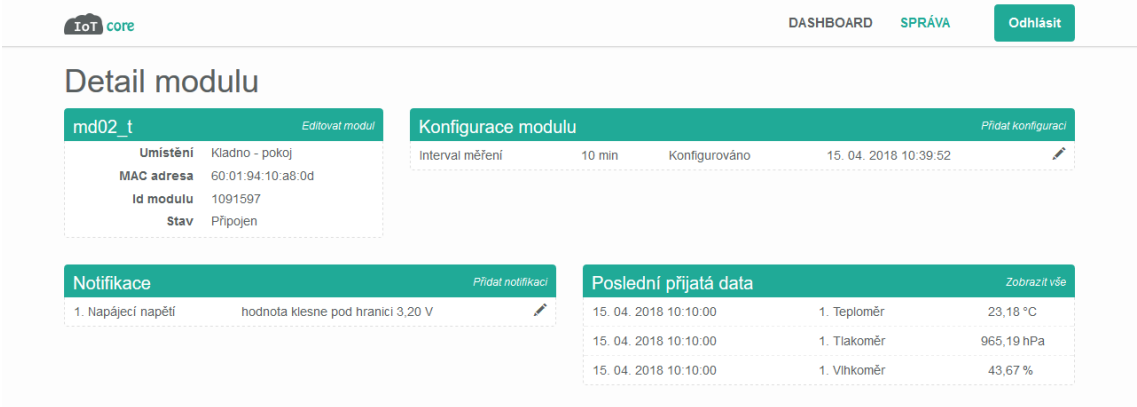
B Výsledná podoba uživatelského rozhraní



Obrázek 11: Výsledná stránka *DASHBOARD*



Obrázek 12: Výsledná stránka *SPRÁVA*



Obrázek 13: Výsledná stránka detailu modulu

C Přiložené CD

Přiložené CD obsahuje:

- Zdrojové kódy aplikace (komprimované do jednoho souboru zip)
- Aplikaci ve spustitelném formátu pro produkční server (komprimované do jednoho souboru zip)
- Grafiku vytvořenou pro UI aplikace (komprimované do jednoho souboru zip)
- Práci v elektronické podobě (ve formátu pdf)