**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA

PRÁCA

DANIEL JEŽÍK

**Webová aplikácia pre vizualizáciu nameraných dát v IoT prostredí**

Vedúci práce: Ing. Martin Húdik, PhD.

Registračné číslo: 299/2016

Žilina, 2017

**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA

PRÁCA

POČÍTAČOVÉ INŽINIERSTVO

DANIEL JEŽÍK

**Webová aplikácia pre vizualizáciu nameraných dát v IoT prostredí**

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Školiace pracovisko..............

Žilina, 2017

#### Úvodné pokyny pre použitie šablóny

Väčšina nadpisov šablóny je previazaná na dokument Pokyny\_pre\_vypracovanie\_ZP.docx. Sú v ňom podrobnejšie informácie o vypracovaní záverečnej práce. Linky budú správne fungovať vtedy a len vtedy keď si uložíte dokument so záverečnou prácou do toho istého priečinka, v ktorom sa nachádza uvedený dokument. Priamo z tejto šablóny nefungujú.

#### (Pri vytváraní svojho dokumentu pokyny pre použitie šablóny vymažte!)

#### [Čestné vyhlásenie](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Čestné_vyhlásenie)

Podľa národnej legislatívy nie je v záverečnej práci povinné.

#### [Poďakovanie](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Poďakovanie)

Nie je povinné, vysvetlenie a príklad poďakovania

[ABSTRAKT V ŠTÁTNOM JAZYKU](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt)

PRIEZVISKO, Meno: *Názov témy záverečnej práce*. [Bakalárska/ Diplomová/ Dizertačná práca]. – Názov univerzity. Názov fakulty; Názov školiaceho pracoviska. – Školiteľ/Vedúci: (tituly pred menom) Meno a priezvisko (tituly za menom) – Stupeň odbornej kvalifikácie: bakalár/magister/iný. – Mesto: skratka fakulty a univerzity, rok predloženia. Počet strán (napr. 35 s.)

[Vysvetlenie a príklad.](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Príklad_abstrakt)

[**Kľúčové slová**](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Kľúčové_slová)**:**  (jednoslovné alebo viacslovné termíny, ktoré charakterizujú vecný obsah práce)

[ABSTRAKT V CUDZOM JAZYKU](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt_CJ)

[Vysvetlenie.](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Abstrakt_CJ)

**Key words:**

[Obsah](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Obsah)

Zoznam obrázkov 4

Zoznam tabuliek 4

Zoznam skratiek 4

Úvod 4

1 Súčasný stav riešenej problematiky doma a v zahraničí 4

2 Ciele práce 4

3 Metodika práce a metódy skúmania 4

4 Výsledky práce a diskusia 4

4.1 Výsledky práce 4

4.2 Diskusia 4

Záver 4

Zoznam použitej literatúry 4

Zoznam príloh 4

Prílohy 4

Príloha A: Názov prílohy 4

Príloha B: Obsah DVD 4

# [Zoznam obrázkov](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

**Nenašli sa žiadne položky zoznamu obrázkov.**

# [Zoznam tabuliek](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

**Nenašli sa žiadne položky zoznamu obrázkov.**

# [Zoznam skratiek](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_skratiek)

ANSA Automated Network Simulation and Analysis

BFD Automated Network Simulation and Analysis

Cisco IOS Cisco Internetwork Operating System

# [Úvod](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Úvod)

Táto technológia sa do nášho života dostáva čoraz viac aj keď o tom vôbec nevieme či si to len neuvedomujeme . Ťažko by sme hľadali odpoveď na význam skraty IoT či vysvetlenia pojmu Internet vecí u človeka, ktorý sa nevenuje informačným technológiám hoci i u mladšej generácie aj napriek tomu, že sa s touto technológiu môžu stretávať roky či dokonca nevedomky ju aj využívať. Možno si teraz kladiete otázku kde ? kedy ?

Medzi jednoduchý prvok IoT zariadenia môžeme zaradiť aj bezpečnostnú kameru pripojenú na internet, jej obraz tak môžete sledovať prostredníctvom webovej stránky kdekoľvek. Takáto technológia pred pár rokmi nezažívala až taký boom, dovolil by som si tvrdiť, že príchod smartfónov vo veľkom ovplyvnil vývoj tejto technológie. Keďže dnešný trend telefónov s veľkými displejmi, rýchlym pripojením na internet či veľkým výpočtovým výkonom sa stali dokonalým útočiskom pre jej využitie. Užívateľ tak nemusí využívať notebook či domáce PC na prácu s IoT.

Svoje prvé oboznámenie s touto technológiu si pamätám ešte z roku 2014 kedy americká spoločnosť Amazon uviedla na trh svoj produkt s názvom Amazon Dash. Ide o malé zariadenie s jedným gombíkom, ktoré je pomocou Wi-Fi pripojené na internet. Toto zariadenie v prípade napr. pracieho prášku sa umiestni do blízkosti práčky. Ak sa daný prací prášok minie, jediným stlačením gombíku sa vykoná objednávka a jeho následné dodanie. Takéto zariadenie nepotrebuje údržbu okrem výmeny batérie. Dá sa konfigurovať pomocou mobilného telefónu. Aktuálna cena je cca 5$. Samozrejme je vyrábané v rôznych variantoch či už pre objednávku jedla či služieb.



Možnosť mať nejaké takéto zariadenie napr. doma ma nadchla, predstavil som si čo všetko by sa stým dalo robiť a aké to má potenciálne využitie, vtedy som sa viac zameral na oblasť senzorov, ktoré by mohli byť v domácom prostredí a teda nejaké merače teploty, vlhkosti, dymu a pod. Vzhľadom na to, že v roku 2014 som mal len malé skúsenosti v oblasti programovania tak som túto oblasť „smart“ zariadení bližšie neštudoval, no vždy ma dokázala nadchnúť myšlienka inteligentného domu. Na konci prvého ročníku na vysokej škole som objavil vývojovú dosku Arduina a tak som začal skúšať prvé pokusy od jednoduchého rozsvecovania LED diód až po zložitejšiu simuláciu parkoviska, ktorá obsahovala senzory vzdialenosti, pohybu, IR senzory či servomotory na ovládanie rampy.

Môj posledný projekt s týmto zariadením sa najviac priblížil k momentálnej bakalárskej práci a tiež projektu, ktorý by som chcel do budúcna ďalej rozvíjať. Pomocou pár senzorov vlhkosti, teploty som skúšal zaznamenávať hodnoty týchto veličín, ktoré som zobrazoval na webovej stránke len s malým oneskorením. Pri tejto príležitosti som dostal nápad navrhnúť toto riešenie trochu komplexnejšie a tak na ňom postaviť bakalársku prácu. A tak vytvoriť webovú aplikáciu, pomocou, ktorej by som mohol ovládať či využívať prvky takýchto senzorov či zariadení disponujúcimi senzormi napríklad z mobilného telefónu. Aj keď z počiatku som mal trochu inú predstavu ako by mohla práca vyzerať no po konzultácii s mojím vedúcim práce som sa rozhodol práve pre riešenie tejto práce.

V dnešnej dobe sa teda stretávam stále častejšie so zariadeniami, ktoré sa dokážu pripojiť na internet a tak nás formou notifikácií dokážu informovať o udalostiach, ktoré sa dejú okolo nás dejú. Takýto systém je už štandardný prvok v modernom inteligentnom dome, kde si môžete rozsvecovať svetlo či odostrieť žalúzie pomocou mobilného telefónu či keď, vám dokonca práčka zašle správu o úspešnom dokončení procesu prania.

No trend vývoja tejto technológie rokmi postúpil do dnešného dňa tak, že za pár eur si ju môže dopriať ktokoľvek. Priekopníkom tejto technológie je firma Philips, ktorá uviedla systém domáceho inteligentného osvetlenia HUE pre domácnosti a užívateľ ho tak môže pomocou smartfónu intuitívne ovládať . ( ALZA ) V tomto trende nezaostali ani internetový giganti ako spoločnosť Google či Apple, ktorá predstavila svoju aplikáciu HomeKit, ktorá dokonca umožňuje využívať jej funkcie pomocou inteligentného osobného asistenta Siri, ktorá je súčasťou mobilných telef. ( APPLE )

Na Slovensku na tento trend upozorňuje aj akcia s názvom IoT Expo Bratislava či stále častejšie prednášky konajúce sa na vysokých školách. Spoločnosť Slovnaft taktiež ako aj O2 spustila svoj IoT pod názvom Smart Connect.

# [Súčasný stav riešenej problematiky doma a v zahraničí](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Súčasný_stav_riešenia)

Pri danej problematike sa nám naskytne možnosť vyberať si z veľkého množstva platforiem, ktoré internet ponúka zadarmo či riešenia od firiem, ktoré sa naplno venujú tejto problematike. Väčšinou sa teda takéto služby rozdeľujú do dvoch kategórií a to na služby, ktoré sú otvorené alebo na služby, ktoré sú vytvorené pre určitý segment produktov a teda zväčša je to riešenie konkrétnej značky.

Vzhľadom na to, že takýchto platforiem je skutočné kvantum opíšeme si len pár riešení, ktoré nás bližšie uvedú do danej problematiky.

Prvé riešenie ponúka spomenutá spoločnosť Amazon, ich platforma sa na nazýva AWS ( amazon web services ). Táto platforma je cloudová a umožňuje pripojeným zariadeniam jednoduchú a bezpečnú interakciu s cloudovými aplikáciami a ostatnými zariadeniami. Samozrejme disponuje dátovým úložiskom, štatistikami atď. O týchto funkciách si povieme neskôr.



Druhé riešenie ponúka na Slovensku spoločnosť Telekom s ich službou SMARTHOME. Toto riešenie je práve z kategórie, ktorú sme spomenuli vyššie a teda táto služba využíva určitý sortiment zariadení a teda pokrýva len službu inteligentného domu. Užívateľ tak pomocou mobilnej aplikácie a po zakúpení zariadení podľa jeho výberu môže vybaviť svoj dom inteligentnými prvkami a tak ho premeniť na inteligentný dom.



Vzhľadom na to, že slovenský trh nedisponuje riešením pre IoT, ktoré by nás dostatočne zaujalo zamerali sme sa na zahraničnú službu, ktorá ponúka spoločnosť Ubidots a ich služba sa volá tak isto Ubidots. Túto platformu sme si vybrali kvôli tomu, že je inovatívna, prehľadná a efektívna. Naša práca sa teda v štádiu vývoja zameriava práve na vývoj podobnej aplikácie ako ponúka toto riešenie akurát v odľahčenej forme a teda bude sa vyznačovať značne menšou komplexnosťou. Toto riešenie ponúka pokročilé užívateľské funkcie či multi-protokolové API rozhranie. Ďalšie funkcie si opíšeme nižšie.

**Live Dashboards** je užívateľský panel nástrojov, je to akoby princíp pracovnej plochy, kde vidíme prehľad zo zvolených senzorov v reálnom čase. Tieto dáta môžeme zdieľať cez verejné odkazy či zobrazovať ich na mobilných zariadeniach.

Služba ponúka viac ako 50 knižníc pre zariadenia pre rýchle pripojenie týchto zariadení do ich služby. Tieto knižnice sú stále testované a vylepšované ich komunitou. Vzhľadom na to, že takéto zariadenia môžu podľa konfigurácie generovať veľké množstvo záznamov disponuje táto služba inteligentným správcom dát, ktorý ich vie vhodným spôsobom zhromažďovať a kategorizovať a pri prípade nehody ich obnoviť. Takéto záznamy z rôznych zariadení môžu prichádzať v rôznom formátovaní, kvôli tomu je takáto služba vybavená API rozhraním, ktoré je viac protokolové a teda dokáže vyhovieť rôznym formátom.

## IoT - Internet vecí

### Čo je internet vecí?

„Internet vecí“ (IoT - skratka z aglického Internet of Things) je názov pre technológie umožňujúce prepojenie komunikáciu zariadení spravidla s nízkou spotrebou a nárokmi na objem prenesených dát. Typickým použitím budú monitorovacie a meracie senzory (priemysel, poľnohospodárstvo, životné prostredie, domácnosti) a sledovanie pohybu a polohy osôb alebo zvierat, dopravných prostriedkov či tovaru. V týchto oblastiach v blízkej dobe nahradia aj mnohé doteraz používané zariadenia a systémy na báze mobilných sietí. Prinesú najmä radikálne nižšiu spotrebu (životnosť batérie aj mnoho mesiacov), ale aj nízku cenu a prevádzkové náklady.“  (SLOVNAFT)

### Čo internet vecí prinesie?

Sieť a zariadenia IoT samozrejme umožnia obojsmernú komunikáciu, takže aj diaľkové ovládanie takýchto zariadení, ako aj vlastnú technológiu lokalizácie bez nutnosti GPS prijímača. Podľa niektorých predpovedí bude v roku 2020 celosvetovo pripojených do sietí „internetu vecí“ desiatky miliárd zariadení, čo predstavuje minimálne niekoľko zariadení na jedného obyvateľa planéty.

### Ako veľký je rozsah IoT ?

Vzhľadom na to, že éra konektivity notebookov a smartfónov je už dávno za nami nasledujeme do doby konektivity áut, inteligentných domov, nositeľného príslušenstva či inteligentného mesta alebo zdravotníctva. Zjednodušenie tento trend smeruje do doby kedy bude prepojené prakticky všetko zo všetkým. Na základe správy spoločnosti Gartner, lídra v oblasti prieskumov informačných technológií, ktorá hovorí, že do roku 2020 bude prepojených naprieč všetkými technológiami 20,6 miliardy zariadení. (GARNET)

|  |  |
| --- | --- |
| Rok | Počet pripojených zariadení |
| 1990 | 0.3 miliónu |
| 1999 | 90 miliónov |
| 2010 | 5 miliárd |
| 2013 | 9 miliárd |
| 2025 | 1 bilión |

HP tabulka

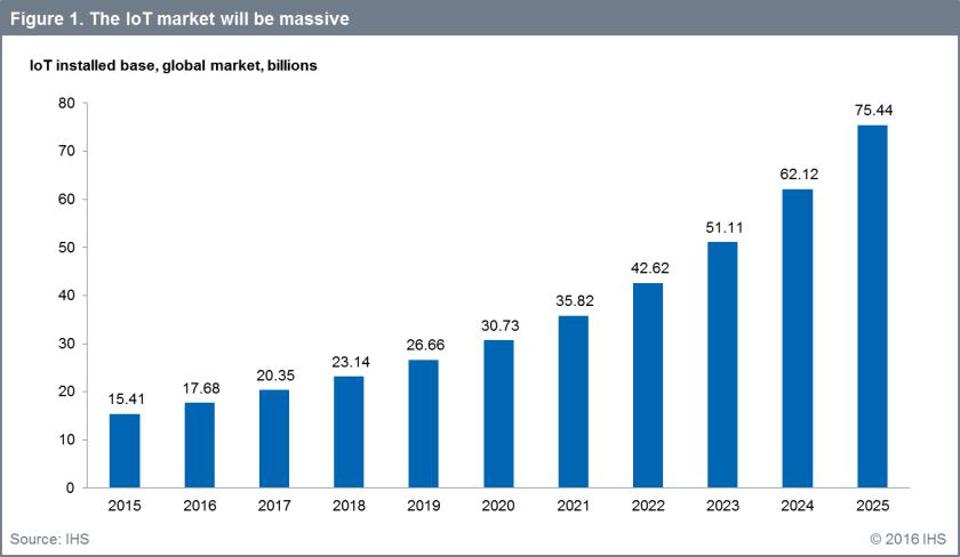
Na základe pireskumu spoločnosti HP , prieskum nárastu

### Ekonomický dopad

Ak vás zaujíma, aké dopady internetu vecí bude mať na ekonomiku potom pre Vašu informáciu, ako ich uvádza Cisco internetu vecí bude generovať hodnotu 14,4 $ bilióna vo všetkých odvetviach priemyslu v nasledujúcom desaťročí.

General Electric predpovedá počiatočné investície do priemyselu internetu vecí (IoT) vo výške $ 60 biliónov počas nasledujúcich 15 rokov.

IHS predpovedá, že trh internetu vecí porastie z počtu 15,4 miliardy zariadení v roku 2015 na 75,4 miliardy do roku 2025 (twitter oba)



<https://blogs-images.forbes.com/louiscolumbus/files/2016/11/IHS.jpg?width=960>

### Prínos / výhody

Najväčšou výhodou internetu vecí je úspora peňazí. V prípade, že cena za monitorovacie zariadenie je menšia ako množstvo ušetrených peňazí, potom internet vecí bude veľmi vítane prijatý. Takéto úspory sa jednak môžu odzrkadliť na priaznivom vývoji vo firme ale aj v cene produktov či služieb. IoT sa zásadne ukázalo byť veľmi užitočné pre ľudí v ich každodennej rutine tým, že zariadenia vzájomne komunikujú účinným spôsobom, čím šetria energiu a náklady s nimi spojenými. Tieto zariadenia umožňujú zdieľať dáta medzi sebou a ich transformáciu požadovaným spôsobom, to robí tieto systémy efektívne. Výhodou je predikcia, ktorá v konečnom dôsledku môže ušetriť množstvo času a finančných prostriedkov. Realtime monitoring takýchto prostriedkov teda dokáže efektívne pomáhať v rôznych infraštruktúrach pre jej riadenie či pomoc s riadením a tak ju zefektívniť, zrýchliť a ušetriť tým peniaze a čas.

* **Bezpečnosť, komfort, účinnosť**

Predstavte si, že môžete merať a kontrolovať nebezpečné prostredia bez toho aby ste vystavili ľudí nejakému riziku pričom takéto meranie môže byť nepretržité a strojovo presné. Obslužný pracovník môže takéto údaje komfortne z pracoviska sledovať. Napríklad inteligentné montážne linky môžu hlásiť chyby a varovania v reálnom čase, tým produkovať vyššie výnosy a menej prestojov.

* **Informácia**

Je zrejmé, že viac informácií pomáha robiť lepšie rozhodnutia. Či už ide o banálne rozhodnutie vedieť, čo kúpiť v obchode alebo či má vaša firma dostatok materiálov a zásob, vedomosť je veľmi nutná. Pri nepoznaní takýchto faktov môže dôjsť k pozdržaniu či zastaveniu výroby a následným stratám.

* **Spotrebiteľské preferencie**

IoT môže analyzovať spotrebiteľské preferencie takým spôsobom, aby mu na základe návykov, obsahu vyhľadávania, pohybu či aktivity vyhovovali reklamné produkty a tak boli viac založené na individuálnych potrebách a preferenciách. Napríklad - inteligentné chladničky by vám len nepovedali, kedy budete končiť trvanlivosť mlieka, ale môže tiež analyzovať koľko mlieka ostáva, kde bolo kúpené a aká je jeho aktuálna cena. Takýmto spôsobom vie informovať vopred o nutnosti dokúpenia potravín a taktiež povedať, ktoré potraviny je nutné kúpiť. Užívateľ tak má prehľad o potravinách, zmenšuje objem odpadu a šetrí peniaze.

### Zápory / nevýhody

* **Bezpečnosť**

Ak ste niekedy venovali pozornosť hlavným televíznym novinám určite vám neunikla správa o firme, ktorá bola hacknutá. Takéto útoky sú stále častejšie pričom sa útočník nevystavuje priamemu riziku a aj tak profituje zo svojho úkonu za účelom krádeže identít, hesiel, kreditných kariet dokonca odcudzenie kľúčov áut, ktoré komunikujú skrz mobilnú aplikáciu. Skutočnosť, že sú tieto zariadenia pripojené do sietí či na internet ich vystavuje bezpečnostnému riziku. Vzhľadom na tento fakt sa stále posilňuje bezpečnosť takýchto zariadení aby sa predišlo ich zneužití. Častým nedostatkom je pomalé vydávanie bezpečnostných aktualizácií či zdokonaľovanie samotného hardwaru v novších zariadeniach.

* **Súkromie**

Otázka tiež padá na tých kto sleduje tých čo sledujú ? Internet vecí totiž sprostredkováva obrovskú databázu citlivých dát. Pri inteligentnom dome je jednoduché zistiť vaše návyky, kedy idete spať, kde sa pohybujete, kedy ste v práci... Na jednej strane sa takýto systém javí ako dobrý sluha ale aj zlý pán, zvlášť v dnešnej dobe kedy zákony rôznych krajín umožňujú sledovanie ľudí len na základe podozrení zo strany vlády.

* **Spracovanie dát**

Internet vecí vytvára nespočetné množstvo dát, ale obchodná hodnota týchto dát sa nemeria v bajtoch ,ale v ich analýze. Samotné namerané dáta nemajú žiadnu trhovú hodnotu. Ak si tieto dáta chceme predstaviť väčšinou ide len o číslo 1 alebo 0. V praxi väčšia komplexnosť znamená väčšiu šancu pre vznik chyby. Pokiaľ takéto dáta nedokážeme efektívne spracovať a vyhodnotiť stane sa to, že nás monitorovacie zariadenia len zasypú údajmi, čo bude viac neefektívne ako efektívne využitie ich potenciálu.

Takéto obrovské množstvo dát, ktoré sa neskôr analyzuje sa nazýva BIG DATA. Vzhľadom na to, že takýto objem dát nie je možné len tak analyzovať, spracovanie týchto dát podlieha niekoľkým fázam spracovania, zhromažďovanie, filtrovanie, fúzia, spracovanie a ich archivovanie. Spravovanie takéhoto množstva dát často podlieha automatizácií či používaní modelov pre spracovanie.

* **Kompatibilita**

V internete vecí neexistujú žiadne medzinárodné štandardy pre kompatibilitu. Čo môže mať za dôsledok problém pri komunikácií so zariadeniami, keďže často pochádzajú od rôznych výrobcov.

* **Nahradenie človeka strojom**

Automatizácia môže byť dobrá vec, pokiaľ vás kvôli nej nenahradí v práci stroj. Na jednej strane spoločnosť investuje do modernizácie, skvalitnenia a zrýchlenia práce no na druhej strane na úkor ľudskej pracovnej sily. Takýto vývoj kedy zariadenia nadobúdajú stále viac a viac miesta v bežnom živote človeka. Človek tak stráca svoje základné interakcie s inými ľuďmi.

### Aplikácie

Pri praktických využitiach sme na základe internetového vyhľadávania vybrali 5 najčastejších využití:

1. **Inteligentný dom**

Inteligentný dom alebo Smart Home je položka IoT, ktorá vyčnieva najviac zo všetkých aplikácií a zažíva najväčší boom v danej oblasti. Viac ako 60.000 ľudí v súčasnosti vyhľadáva termín "Smart Home" každý mesiac prostredníctvom webových vyhľadávačov. Predpokladá sa, že inteligentné domy budú v budúcnosti tak bežné ako mobilný telefón. Prvky takéhoto inteligentného domu dokážu efektívne pracovať so svetlami, vykurovaním a pod. a teda sľúbia úsporu energií, času a peňazí.

1. **Nositeľné príslušenstvo**

Záujem o nositeľné príslušenstvo v posledných rokoch prudko zrástol po celom svete. Pod pojmom nositeľné príslušenstvo alebo z anglického slova „Wearables“ si môžeme predstaviť inteligentné náramky, hodinky, pásy na hruď a mnoho ďalších. Sú to zariadenia, ktoré disponujú senzormi, ktoré zbierajú dáta a taktiež nesú informáciu o ich užívateľovi ako sú napríklad výška, váha, vek, pohlavie. S týmito dátami neskôr pracujú aplikácie či ľudia, ktorí tieto dáta vyhodnocujú. Takéto zariadenia je najčastejšie využívaný v športovom odvetví či zdravotníctve.

1. **Automobilový priemysel**

Jeden zrejmý príklad je vývoj autonómnych ("bez vodičových") motorových vozidiel. Na základe prieskumu svetovej zdravotníckej organizácie uviedla, že ročný počet úmrtí spojených s autonehodou na celom svete je viac ako jeden milión. Väčšina týchto úmrtí je zapríčinená v dôsledku ľudskej chyby. IoT technológie, najmä vzostup bezpečnostne zameraných senzorov v automobiloch, má potenciál výrazne znížiť počet úrazov a úmrtí zapríčinených motorovým vozidlom.

1. **Inteligentné mestá**

Inteligentné mestá ponúkajú rôzne využitie IoT napríklad v riadení dopravy, distribúcie vody či elektriny, správu odpadu, monitorovanie životného prostredia či bezpečnosti. Faktom je, že dnes sú takéto mestá realitou. Systémy sú efektívne a nenáročné a riešia problémy mesta a obyvateľov.

1. **Poľnohospodárstvo**

Vzhľadom na stále stúpajúci dopyt po potravinách bolo nutné využiť pokročilé techniky a výskum na zvýšenie produkcie potravín. Poľnohospodári využívajú takto nahromadené dáta pre lepšie využitie znalostí za účelom zvýšenia efektívnosti a produktivity pri nižších nákladoch. IoT v takomto odvetví dokáže farmárovi dopomôcť pri zisťovaní faktorov ako je hodnota vlahy či živín v pôde, spotreba vody pre rastliny či ich automatické zavlažovanie alebo automatické kŕmenie dobytku.

### Protokoly pre IoT

**MQTT** (Message Queuing Telemetry Transport) je jednoduchý a nenáročný protokol pre predávanie správ medzi klientmi prostredníctvom centrálneho bodu tzv. brokeru. Pri protokole MQTT prebieha prenos pomocou TCP a používa návrhový vzor publisher – subscriber. Týmto spôsobom je možné v sieti na báze TCP/IP zverejňovať ( publish ) informácie, ktoré potom broker prepošle klientom alebo zariadeniam, ktoré túto správu odoberajú ( subscribe ).



**CoAP** (Constrained Application Protocol) je webový komunikačný protokol založený na RESTovom modely. Je hlavne určený pre odľahčenú M2M komunikáciu vzhľadom na použitú binárnu hlavičku a nie textovú ako používa protokol http, z ktorého tento protokol vznikol.

C:\Users\DanielJežík\AppData\Local\Microsoft\Windows\INetCache\Content.Word\coap.png

**XMPP** ( Extensible Messaging and Presence Protocol ) je otvorený protokol používaný na sieťovú komunikáciu, posielanie správ či zisťovanie stavu založený na XML.

C:\Users\DanielJežík\AppData\Local\Microsoft\Windows\INetCache\Content.Word\xmpp.png

## REST

**RE**presentational **S**tate **T**ransfer v skratke REST alebo RESTful. REST je webové rozhranie, ktoré je založené na architektúre, ktorá využíva HTTP protokol pre dátovú komunikáciu. Pojem REST bol definovaný v roku 2000 Royom Fieldingom v jeho dizertačnej práci. REST je orientovaný dátovo nie procedurálne, webové služby definujú vzdialené procedúry a protokol pre ich volanie, REST určuje ako sa bude pristupovať k dátam. Týmto sa líši od známejších technológií ako sú napr. RPC, WSDL či SOAP.

V RESTovej architektúre sú zdroje tzv. resources ľahko prístupné. Každý takýto zdroj má vlastný identifikátor URI. REST využíva viaceré reprezentácie zdrojov a to ako Text, JSON alebo XML, kde dnes najpoužívanejší formát pre webové služby je JSON a pre prístup k týmto dátam definuje 4 základné metódy a to GET, POST, PUT a DELETE. Webové služby založené na RESTovej architektúre sú známe pod pojmom RESTful Web Services. Takmer každý zásadný vývojový jazyk dnes obsahuje framework na tvorbu RESTful služieb a teda API.

* GET (Retrieve) – táto operácia sa využíva na získanie zdroja
* POST (Create) – táto operácia sa využíva na vytvorenie zdroja
* PUT (Update) – táto operácia sa využíva na zmenu hodnoty zdroja
* DELETE – táto operácia sa využíva na zmazanie zdroja

C:\Users\DanielJežík\AppData\Local\Microsoft\Windows\INetCache\Content.Word\rest.png

### HTTP stavové kódy ( status code )

Stavové kódy sú odpovede ( response ) zo strany servera, ktorý nám pomocou kódu hovorí aká akcia bola vykonaná na základe požiadavky ( requestu ). Napríklad si môžeme uviesť známy príklad stavového kódu, s ktorým sa užívateľ často stretáva a to 404 Not Found. Tento stavový kód nám hovorí, že sa pokúšame pristúpiť k zdroju, ktorý neexistuje. Vzhľadom na to, že HTTP je rozšíriteľný protokol je možné generovať vlastné stavové kódy. Z hľadiska vývoju by malo byť prioritou využívať všetky stavové kódy, no táto požiadavka sa nie vždy plní a preto sa zväčša používajú iba tieto stavové kódy :

Stavové kódy sa rozdeľujú do 5 kategórií :

* 1xx Informational – informačné
* 2xx Success – úspešné
* 3xx Redirection – presmerovanie
* 4xx Client Error – chyba klienta
* 5xx Server Error – chyba serveru

|  |  |
| --- | --- |
| Kód | Popis |
| 200 | OK – požiadavka prebehla v poriadku |
| 201 | Created – pri POST, pokiaľ bol vytvorený nový obsah |
| 304 | Not Modified – pokiaľ od poslednej požiadavky nebol zmenený obsah |
| 400 | Bad Request – zlá požiadavka na server |
| 401 | Unauthorized – klient je neoverený |
| 403 | Forbidden – klient nemá prístup k danému obsahu |
| 404 | Not Found – zdroj neexistuje |
| 422 | Unprocessable Entitiy – chyba validácie dát |
| 500 | Internal Server Error – na servery došlo k neočakávanej chybe |

### RESTFul Web Services / RESTové webové služby

„Webová služba je riešenie, ako spolu môžu aplikácie či systémy komunikovať a vymieňať si medzi sebou informácie prostredníctvom Internetu.“ **Wiki**.

V takomto prípade môžu aplikácie naprogramované v rôznych programovacích jazykoch bežiace na rôznych platformách využívať webové služby pre výmenu dát cez Internetovú sieť obdobným spôsobom akoby prebiehala komunikácia medzi procesmi na jednom počítači. Vďaka týmto otvoreným štandardom potom môžu medzi sebou jednoducho komunikovať systémy ako Windows a Linux či Java a Python atď.

### REST v IoT

REST a celkovo aj API hrajú dnes v oblasti IoT zásadnú rolu. Bez webového API by IoT platformy celkom stratili svoj význam. REST sa svojou jednoduchosťou a ľahkosťou stáva dokonalým riešením pre komunikáciu medzi zariadením a IoT platformou. REST tak pracuje ako aj vstupná brána pre prichádzajúce dáta zo zariadení tak aj spracúva požiadavky od iných klientov, ktorým svoje dáta pomocou webového API sprístupňuje.

### API

„**Application programming** **interface** alebo skratkou **API** (rozhranie pre programovanie aplikácií). Tento termín je používaný v programovaní. Ide o zbierku funkcií a tried (ale aj iných programov), ktoré určujú akým spôsobom sa majú funkcie knižníc volať zo zdrojového kódu programu. API funkcie sú programové celky, ktoré programátor volá namiesto vlastného naprogramovania.“ **Wiki**

### Vlastnosti RESTu

Každý systém využíva zdroje (resources). Tieto zdroje môžu byť obrázky, videá, webové stránky alebo hocijaká iná informácia, ktorá môže byť reprezentovaná počítačom. Účelom týchto služieb je poskytnúť užívateľovi prístup k týmto zdrojom. Architektúra RESTu ponúka vývojárom jednoduchú implementáciu, údržbu či rozšíriteľnosť. Vo všeobecnosti by RESTové služby mali disponovať týmito vlastnosťami:

* Representations ( reprezentácia )
* Messages ( správy )
* URIs
* Uniform interface ( jednotné rozhranie )
* Stateless ( bezstavovosť)
* Links between resources ( odkazy medzi zdrojmi )
* Caching

### Reprezentácia dát

Hlavnou úlohou RESTových služieb je správa zdrojov a prideľovanie prístupu k týmto zdrojom. Takýto zdroj si môžeme predstaviť ako objekt pri OOP a teda zdroje môžu obsahovať ďalšie zdroje. V takomto prípade je nutné najskôr identifikovať formát zdroju, s ktorým sa bude pracovať. Vzhľadom, na to, že REST nijako nelimituje formát pre prezentáciu zdrojov, je na vývojárovi aký si vyberie. Táto výhoda využitia viacerých formátov je vhodná vtedy, že takéto rozhranie vie komunikovať s rôznymi zariadeniami.

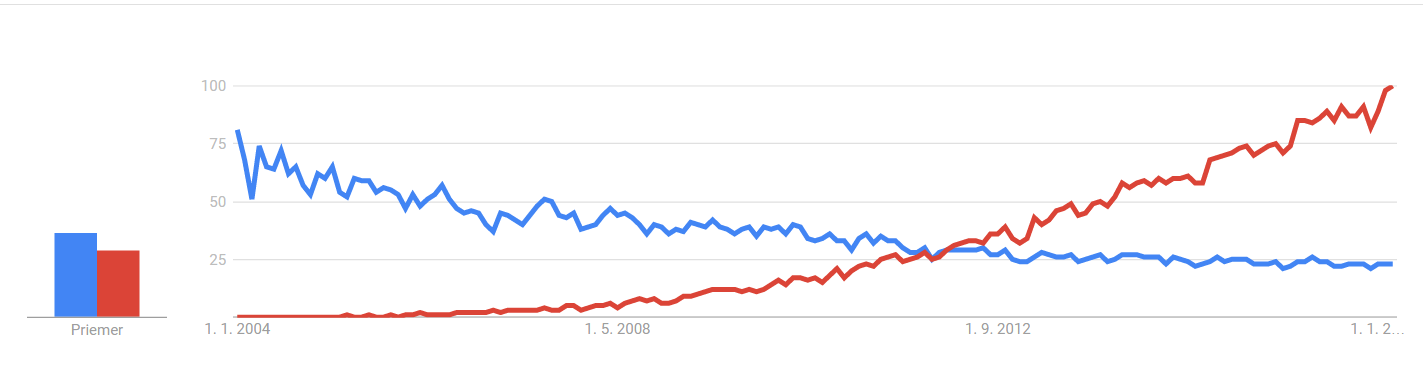
Napríklad ak požiadavka na server obsahuje parametre pre odpoveď v určitom formáte, ak server disponuje takýmto formátovaním dokáže jeho požiadavku spracovať. Odpoveď servera ( response ) by mala reprezentovať celý požadovaný obsah v čo najmenšom obsahu. Pokiaľ to tak nie je možné je nutné je nutné tieto zdroje rozdeliť na menšie a tým skrátiť dobu prenosu a tak zrýchliť služby.

**Príklad zobrazenia JSON:**

1. {
2. "ID": "1",
3. "Name": "Jozef",
4. "Email": "jozef@gmail.com",
5. "Country": "Slovakia"
6. }

|  |  |
| --- | --- |
|  |  |

V nasledujúcom obrázku bol použitý nástroj Google Trends pre porovnávanie vyhľadávaných výrazov pomocou Google search. Daný graf nám porovnáva hľadané výrazy pre „xml api“ a „json api“, z tohto grafu je jednoznačne vidieť, že termín „xml api“ ako hľadaný ale aj formát postupom času upadá pričom formát JSON naberá na popularite.



### Správy

**HTTP Request**

Klient a webová služba spolu komunikujú pomocou správ. Klient zašle požiadavku (request) na server a ten mu odpovie (resposne). Odhľadnúc od aktuálnych dát, tieto správy taktiež obsahujú metadáta o danej správe. Tieto dáta požiadaviek a odpovedí sú dôležité informácie pre navrhovanie RESTových služieb.

C:\Users\DanielJežík\Desktop\bakalarky\http request.png

<VERB> typ HTTP metódy napr. GET, PUT, POST, DELETE, OPTIONS atď.

<URI> Je URI adresa zdroja, na ktorej sa bude operácia vykonávať

<HTTP Version> je verzia HTTP, najčastejšie "HTTP v1.1"

<Request Header> Obsahuje metadáta tzv. kľúčové hodnoty (key-value). Tieto nastavenia obsahujú informáciu o správe a jej odosielateľovi, formáty, ktoré klient podporuje, formát tela správy, cache nastavenia odpovede a iné.

<Request Body> Je skutočný obsah správy.

**Príklad POST požiadavky:**

1. POST http://MyService/Person/ HTTP/1.1
2. Host: MyService
3. Content-Type: text/xml; charset=utf-8
4. Content-Length: 123
5. **<?xml** version="1.0" encoding="utf-8"**?>**
6. **<Person>**
7. **<ID>**1**</ID>**
8. **<Name>**Jozef**</Name>**
9. **<Email>**jozef@gmail.com**</Email>**
10. **<Country>**Slovakia**</Country>**
11. **</Person>**

Môžete vidieť príkaz POST, ktorý je nasledovaný adresou URI a verziou HTTP. Táto požiadavka taktiež obsahuje hlavičkové požiadavky. **HOST** je adresa serveru. **Content-Type** hovorí o type obsahu v tele správy, **Content-Length** je dĺžka dát v tele správy, taktiež môže byť využitá na overenie či bola doručená celá správa.

**GET**

V požiadavke GET sa nenachádza telo správy. **Accept** parameter v hlavičke odkazuje serveru o možnostiach formátu, ktoré klient podporuje. Vzhľadom na tento parameter ak server podporuje prezentáciu zdrojov v takomto formáte tak požiadavke vyhovie inak môže vrátiť požiadavku spracovanú vo formáte servera alebo vrátiť chybový hlášku. **User-Agent** obsahuje informácie o klientovi, ktorý vykonal požiadavku na server. **Accept-Encoding/Language** sú informácie o kódovaní a jazykovej podpore klienta.

**Príklad GET požiadavky:**

1. GET http://www.w3.org/Protocols/rfc2616/rfc2616.html HTTP/1.1
2. Host: www.w3.org
3. Accept: text/html,application/xhtml+xml,application/xml; …
4. User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 …
5. Accept-Encoding: gzip,deflate,sdch
6. Accept-Language: en-US,en;q=0.8,hi;q=0.6

|  |  |
| --- | --- |
|  |  |

**HTTP Response**

C:\Users\DanielJežík\Desktop\bakalarky\http response.png

<response code> vracia server, ktorý obsahuje stav požiadavky tj. stavový kód.

<Response Header> obsahuje metadáta a nastavenia správy odpovede

<Response Body> obsahuje reprezentovaný zdroj ak bola požiadavka úspešná

**Daná odpoveď požiadavky GET:**

1. HTTP/1.1 200 OK
2. Date: Sat, 23 Aug 2014 18:31:04 GMT
3. Server: Apache/2
4. Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
5. Accept-Ranges: bytes
6. Content-Length: 32859
7. Cache-Control: max-age=21600, must-revalidate
8. Expires: Sun, 24 Aug 2014 00:31:04 GMT
9. Content-Type: text/html; charset=iso-8859-1
10. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"**>**
11. **<html** xmlns='http://www.w3.org/1999/xhtml'**>**
12. **<head><title>**Hypertext Transfer Protocol -- HTTP/1.1**</title></head>**
13. **<body>**
14. ...

|  |  |
| --- | --- |
|  |  |

Stavový kód **200 OK** znamená, že všetko prebehlo v poriadku a požadovaný obsah tela správy obsahuje platnú reprezentáciu zdroja, ktorá bola požadovaná. V tomto prípade, bol požadovaný HTML dokument, ktorý je deklarovaný **Content-Type** v hlavičke.

### Adresovanie zdrojov

REST vyžaduje aby bol každý zdroj adresovaný aspoň jednou URI. Pri adresovaný, sa snaží využívať čo najjednoduchšie odkazy a to také aby aj klient na základe tohto odkazu vedel pochopiť hierarchiu odkazov RESTovej služby. Adresa URI by mala byť jednoznačná cesta ku zdroju, nemala pre prezrádzať operáciu, ktorá sa nad ňou bude vykonávať vzhľadom na to, že pri rozličnej HTTP požiadavke sa môže využívať rovnaká URI adresa.

### URI

**Jednotný identifikátor prostriedku** ( Uniform Resource Identifier ) je najobecnejší z niekoľko príbuzných typov identifikátorov. URI môže popisovať zdroj z hľadiska identity ( a neurčovať, kde je možné ho získať) a taktiež ako je zdroj možné nájsť ( ale nepopisovať jeho identitu) a zároveň aj obe možnosti naraz.

URL na rozdiel od URI primárne popisuje ako sa ku zdroju dostať, naopak URN špecifikuje zdroj ako taký. URI tvorí akoby nad množinu týchto identifikátorov.

C:\Users\DanielJežík\Desktop\b.png

**Príklad ako by URI adresa nemala vyzerať:**

1. http://MyService/FetcthPerson/1

alebo

1. http://MyService/DeletePerson?id=1

* URI by nemala obsahovať meno operácie, ktorá sa nad ňou bude vykonávať
* Adresa by mala byť čo najvšeobecnejšia pre prehľadnosť či pre prípad, že by sa zdroj presunul
* Nemala by obsahovať medzery namiesto toho by mali byť použité znaky ako
* \_ ( podtržník ) alebo – ( pomlčka )

**Zhrnutie**

REST naberá na význame spolu s formátom JSON sa pomaly stáva štandardom API webových služieb. Vzhľadom na rozšírenie techniky AJAX, ktorej REST vychádza v ústrety pomáha jeho rozšíreniu, taktiež to že sa prakticky zásadne nelíši od štandardného volania a získavania dát pomocou HTTP. REST sa svojou jednoduchosťou a rýchlosťou čoraz častejšie podporovanými modernými frameworkami pre jeho jednoduchšiu implementáciu. Stáva sa štandardom webových služieb a môžeme vidieť, že aj spoločnosti ako Google, Twitter či Facebook využívajú služby RESTu.

# [Ciele práce](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Ciele_práce)

## Návrh riešenia

Pri návrhu riešenia sme uvažovali nad vhodným výberom stavebných prvkov pre našu platformu a teda frameworkov, ktoré by tak aj uľahčili našu prácu pri vývoji. Naša webová aplikácia by sa mala držať princípu jednoduchosti a efektívnosti, mala by využívať všeobecne známe štandardy pre jej použiteľnosť v praxi, mala by byť orientovaná multi platformovo aby bola dostupná z čo najväčšieho počtu zariadení ale zároveň bola bezpečná.

Použitie frameworkov pri vývoji tejto aplikácie bol hlavný bod tohto návrhu, frameworky obsahujú už integrované metódy pre prácu s databázou, formulármi či API rozhraním a tak by nám ponúkli overené funkčné riešenia.

## Cieľ práce

Cieľom práce je vytvoriť webovú aplikáciu, ktorá bude vizualizovať dáta vo forme rôznych grafov alebo tabuliek. Aplikácia bude obsahovať hlavnú obrazovku na ktorej si bude užívateľ môcť navoliť aké namerané dáta sa budú zobrazovať a formu zobrazenia(typ grafu, tabuľka, mapa, atď.). Aplikácia bude mať RESTové rozhranie, cez ktoré bude prijímať nové dáta a taktiež aj poskytovať tieto dáta tretím stranám. Dáta budú ukladané do databázy. Autentifikácia jednej inštancie aplikácie bude založená na princípe API tokenov.

# [Metodika práce a metódy skúmania](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Metodika_práce)

## Framework

V preklade rámec či štruktúra je softwarová štruktúra, ktorá pomáha programátorovi pri vývoji. Obsahuje už naprogramované funkcie, knižnice, API rozhrania a iné. Cieľom frameworku je ponúknuť vytvorenú štruktúru funkcií a metód, ktoré programátor bude ďalej len implementovať a tak nebude nútený tieto funkcie vytvárať od základu. Takáto možnosť značne zrýchľuje prácu. Frameworky sú často aktualizované a preverované komunitou a teda užívateľ frameworku si môže byť vedomý, že pracuje na bezpečnej a chybovo overenej platforme.

Na druhej strane zlým výberom frameworku sa práca na projekte môže značne skomplikovať. Aby človek mohol s frameworkom pracovať musí si najskôr naštudovať jeho dokumentáciu aby pochopil ako funguje, čo môže zabrať množstvo hodín ale pri budúcej práci na inom projekte znalosť frameworku prácu značne zefektívni a zrýchli.

### Výber frameworku pre front-end

Front-end je opak back-endu a teda stará sa ako o zobrazovanie dát z back-endu užívateľovi tak aj o celkový design webstránky. Je to vlastne užívateľské rozhranie pomocou, ktorého sa ovláda logika back-endu.

Od daného frameworku bolo požadované aby bol schopný využívať najnovšiu verziu HTML 5 a pre zobrazovanie grafov podporu JavaScriptu. Vzhľadom na to, že chceme aby sa webová služba vedela prispôsobiť rôznym zobrazovacím zariadeniam dbali sme aby aj na podporu responzívneho dizajnu.

Responzívny dizajn je spôsob štýlovania HTML kódu tak aby sa zobrazovaná stránka dokázala prispôsobiť zariadeniu, na ktorej je stránka zobrazovaná. Stránka sa tak vie prispôsobiť displeju počítača či tabletu pri zobrazovaní rovnakého obsahu.

### Bootstrap

Dá sa povedať, že tento framework bol vybraný už vopred. S Bootstrapom sme sa stretli už dávnejšie pri skúšaní tvorby HTML webstránok a preto, keď došlo na výber frameworku pre front-endovú časť po inom frameworku sme ani nepátrali. Bootstrap bol vytvorený vývojármi Twitteru a vydaný v roku 2011 a momentálne predstavuje jednotku vo front-endových frameworkoch, má skvelú podporu čo sa týka užívateľského fóra, návodov, video návodov či spracovanej dokumentácie. Pomocou tohto frameworku môžete behom pár minút vytvoriť užívateľskú navigáciu, formuláre, panely a mnoho ďalších.

**Výhody:**

* Podpora responzívneho webdizajnu ( táto možnosť sa môže vypnúť )
* Prepracovaná dokumentácia
* Aktuálnosť a pravidelné aktualizovanie a zlepšovanie

**Nevýhody:**

* Pomerne veľká veľkosť balíčku vzhľadom na množstvo zahrnutých komponentov
* Veľké množstvo HTML tried, ktoré môžu prinášať do návrhu neprehľadnosť

### Podpora Bootstrapu

Mobilné zariadenia:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Chrome | Firefox | Safari | Android prehliadač | Microsoft Edge |
| Android | Podporované | Podporované | N/A | Android v.5+ | N/A |
| iOS | Podporované | Podporované | Podporované | N/A | N/A |
| Windows 10 Mobile | N/A | N/A | N/A | N/A | Podporované |

Počítačové prehliadače:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Chrome | Firefox | IE | Opera | Microsoft Edge | Safari |
| Mac | Podporované | Podporované | N/A | Podporované | N/A | Podporované |
| Windows | Podporované | Podporované | IE10+ | Podporované | Podporované | Nepodporované |

### Inštalácia Bootstrapu

Inštalácia Bootstrapu je veľmi jednoduchá, samotný návod na stránke frameworku ponúka viacero metód na inštaláciu my si spomenieme dve.

Najjednoduchšou metódou, ktorú sme využili aj my je stiahnutie kompletného balíčku Bootstrapu, ktorý sa rozbalí a umiestni sa do zložky s projektom. Potom stačí vo vašej webstránke odkazovať na tento balíček a využívať tak funkcie Bootstrapu. Rozbalený balíček má potom nasledovnú štruktúru:

**bootstrap**/

├── css/

├── js/

└── fonts/

Druhá možnosť inštalácia Bootstrapu je pomocou Composeru. O composeri si povieme neskôr, keď budeme rozoberať inštaláciu jednotlivých modulov PHP frameworku práve pomocou composeru.

### Použitie Bootstrapu

Použitie Bootstrapových knižníc sa nijako nelíši od napr. používania externých kaskádových štýlov CSS, JavaScriptu a pod.

Množstvo zahrnutých súborov pre kaskádové štýly, JavaScripty či iné komponenty závisí od užívateľských preferencií a teda základný balík Bootstrapu sa môže líšiť v závislosti od jeho požiadaviek.

* Načíta kaskádové štýly CSS Bootstrapu:

1. **<link** href="css/bootstrap.min.css" rel="stylesheet"**>**

* jQuery nutné pre Bootstrapové moduly:

1. **<script**
2. src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"**>**
3. **</script>**

* Načíta všetky Bootstrapové moduly:

1. **<script** src="js/bootstrap.min.js"**></script>**

### Praktický príklad

Práca s Bootstrapom je pomerne jednoduchá aj pre úplného začiatočníka, je veľmi intuitívna. Programátor vlastne nemusí tento framework vopred nejako hĺbkovo študovať aby ho mohol používať. Ako bolo už spomenuté Bootstrap disponuje vynikajúcou dokumentáciu, ktorá pri každom komponente obsahuje jeho stručný opis a následne praktickú ukážku použitia aj s kódom. Ak napr. teda pri vývoji programátor zistí, že potrebuje použiť tlačidlo pre odoslanie formulára, tak si v dokumentácii nájde sekciu Buttons ( tlačidla ), kde je daná problematika bližšie rozpísaná. Nižšie si uvedieme príklad na vytvorenie tlačidla klasickým spôsobom a za pomoci Bootstrapu. V príklade budeme uvažovať len nad samotným kódom pre vytvorenie tlačidla, dané zobrazenie tlačidla sa môže líšiť vzhľadom na použitý prehliadač.

* Základné zobrazenie tlačidla bez použitia kaskádových štýlov:



1. **<button** type="button"**>**Odoslať**</button>**

* Základné zobrazenie tlačidla pomocou Bootstrapu:



1. **<button** type="button" class="btn btn-success"**>**Odoslať**</button>**

Takéto tlačidlo nielenže vyzerá krajšie ale už svojou farbou napovedá, že vedie k operácii, ktorá by mala byť priaznivá / úspešná. Programový zápis samotného tlačidla sa od klasického zápisu líši len minimálne. Jeho parameter **class** vlastnosti tohto tlačidla a teda že využíva triedu **btn** z knižnice Bootstrapu a druhým parametrom **btn-success** hovorí, že pôjde o tlačidlo úspechu. Ako sme už spomínali používanie je naozaj intuitívne, jednoduchou náhradou slova **success** za výraz default, waring či danger dokážu výrazne ovplyvniť výzor tlačidla. Pri takomto programovaní niekedy môže prevažovať znalosť anglického jazyka nad znalosťou frameworku, programátor teda pri vývoji môže intuitívne skúsiť prepísať niektoré parametre komponentov. Ak napríklad používame navigáciu s parametrom **up** s vysokou pravdepodobnosťou môže použiť aj parameter **down.** No spravidla to nie vždy tak funguje. V nasledujúcom príklade obsahuje každé tlačidlo názov parametru pomocou, ktorého bolo vytvorené napr. tlačidlo Danger obsahuje parameter **btn btn-danger**.



### Využitie šablóny

Tak isto ako nám framework ponúka možnosť tvorby jednotlivých komponentov do stránky ponúka nám aj hotové riešenia vo forme templatov / šablón. Takáto šablóna najčastejšie zobrazuje už plne vytvorenú stránku so všetkými elementmi, ktoré podporuje. Ak napríklad podporuje grafy tak odkazuje na ukážkovú stránku, ktorá zobrazuje všetky možné zobrazenia grafov v danej šablóne, programátor si tak vie predstaviť ako v danej šablóne graf bude vyzerať, podobne to vyzerá aj pri ostatných komponentoch.

Šablóny slúžia na zrýchlenie tvorby webstránky tým, že odbremeňuje programátora od programovania často používaných komponentov. Tieto šablóny sa ďalej implementujú metódami z back-endu tak aby zobrazovala službou dané prvky namiesto predpripravených hodnôt.

V našom projekte sme sa rozhodli využiť voľne dostupnú šablónu zo stránky Bootstrapu s názvom **SB Admin.**

<https://startbootstrap.com/template-overviews/sb-admin/>

Spomenutá šablóna sa podobá spomínanému konceptu stránky Ubidots, je prehľadná a je orientovaná na administrátorskú tému tz. že obsahuje prvky, pre správu webstránky, grafy či administrátorské rozloženie ikon a teda sa dokonale hodí ako štartovací bod pre našu webovú aplikáciu.

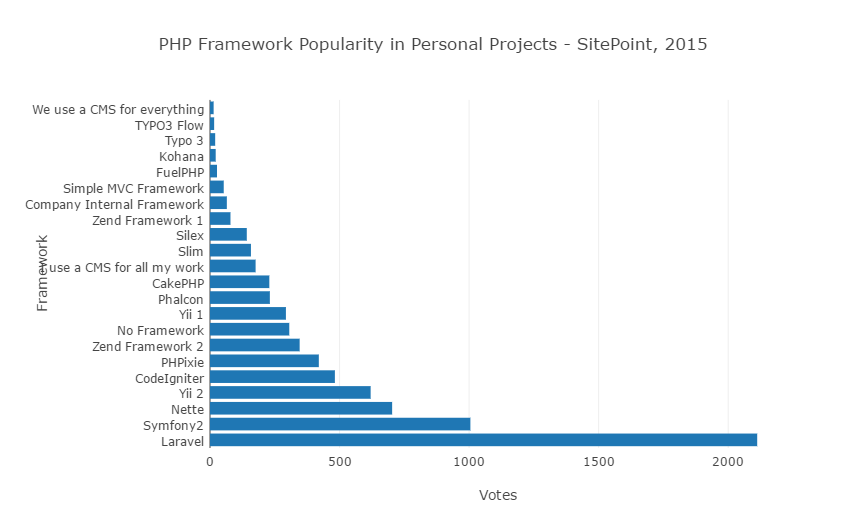
### Framework pre back-end

Back-end „je podporná aplikácia, alebo služba "v pozadí" s ktorou užívateľ nepríde priamo do styku, často sa neprekladá. „

<http://www.prehladsoftveru.webzdarma.cz/index.files/Page1986.htm>

Keďže na začiatku práce na bakalárskej práci sme nemali žiadne skúsenosti z oblasti frameworkov bolo nutné túto oblasť preštudovať a tak vybrať ten vhodný, ktorý sa na danú problematiku bude najviac hodiť, bude mať dobré užívateľské referencie či podporu.

Už na začiatku sme sa rozhodli, že back-end teda časť webovej aplikácie, ktorá sa stará o správu databázy, programovú logiku a celkovú správu stránky budeme vyvíjať v programovom jazyku PHP. Tento jazyk sme si vybrali kvôli tomu, že sme sa sním krátko stretli na školskom predmete Vývoj aplikácií pre internet a intranet a mali o ňom ako takú predstavu. Vzhľadom na použitie tohto jazyku bolo nutné vybrať aj framework, ktorý daný jazyk podporuje. Z internetového prieskumu sme sa presvedčili, že medzi najosvedčenejšie frameworky patria Laravel, Symfony či Nette.



### Nette

Framwerok Nette sme si na začiatku vybrali ani nie kvôli informáciam, ktoré sme našli na internete, ale na základe referencií od skúsenejších programátorov či ľudí, ktorý sa tejto problematike venujú. Mali sme tú česť dokonca navštíviť dve firmy v našom okolí, ktoré sa zaoberajú tvorbou webstránok a obe pracovali s týmto frameworkom a tak myšlienka jeho využitia v našom projekte bola na mieste.

Nette je stavaný tak aby bol čo najpoužiteľnejší a ústretový, čo naozaj je. O významnosti tohto frameworku hovoria aj jeho referencie. Konkrétne Nette si vybudovalo veľkú komunitu priaznivcov, nasvedčuje tomu aj ich webové fórum, kde sa rieši mnoho problémov súvisiacich s týmto frameworkom či popredné stránky, ktoré ho využívajú a to napr. CSFD, Bandzone či ESET.

Nette ako aj iné frameworky pracuje na základe návrhového vzoru MVC( Model-View-Presenter ). MVC je softwarová architektúra, ktorá rozdeľuje dátový model, užívateľské rozhranie a riadiacu logiku do troch nezávislých tried.

* **Model** – je vrstva pracujúca s dátami, je oddelená od aplikácie a komunikuje iba s Presenterom
* **View ( Pohľad )** – je front-endová vrstva reprezentovaná modelom, ktorá zobrazuje dáta užívateľovi
* **Controller ( Presneter )** – je prepojovacia vrstva Modelu a View. Spracováva požiadavky, požaduje dáta od Modelu a vracia ich do View.

C:\Users\DanielJežík\AppData\Local\Microsoft\Windows\INetCache\Content.Word\mvc.png

### Používanie Nette

Práca s Nette od začiatku vyzerá veľmi jednoducho, dostupný návod na vytvorenie prvej aplikácie sa nachádza na internetovej stránke tohto frameworku, je jednoduchý a prehľadný. Po pár prípravných krokoch, prechádza návod do budovania prvej mini aplikácie. Pri tomto vytváraní prvej aplikácie nám veľmi pomáhala knižnica Tracy čo je ladiaci ( debugovací ) nástroj pre Nette, vzhľadom na to, že v jazyku PHP sa ťažko odhaľujú chyby. Naša prvá aplikácia sa podarila úspešne vytvoriť, ale s ďalším vývojom na tejto platforme začali prichádzať aj prvé problémy.

Ako sme už vyššie spomenuli s jazykom PHP sme nemali nejaké väčšie skúsenosti a našu znalosť by sme mohli ohodnotiť statusom začiatočník či dokonca laik. Nette v tomto prípade pri našom vývoji začínalo byť viac prekážkou ako služobníkom, ktorý mal našu prácu uľahčovať. Pri širšom vyhľadávaní sme postupne narazili na návody ako vytvárať užívateľskú registráciu, ako pracovať s databázou či ako implementovať RESTové rozhranie pomocou dostupných balíčkov.

Na jednej strane sme mali teda výbornú platformu, na ktorej sme mohli rozvíjať našu webovú službu ale na druhej strane sme nemali dostatok znalostí na to aby sme dokázali tento potenciál využiť v náš prospech.

Od frameworku Nette sme teda časom upustili a začali sme pátrať po frameworku, ktorý by pre nás predstavoval lepšie útočisko, niečo „ľahšie“. Pri výbere iného frameworku sme sa viac zamerali na jednoduchší framework s menej funkciami ale taktiež s dobrou užívateľskou podporou či dobrou dokumentáciu .

## Slim

### Úvod

Počas pátraní po novom frameworku boli najčastejšie sa objavujúce kľúčové slová vo vyhľadávači „easy, framework, rest, restful, api“. Výsledkom tejto kombinácie vypadol v mnohých zdrojoch odkazujúcich sa na framework Slim.

Slim už na úvodnej stránke v svojom logu prezrádza svoj koncept „Slim a micro framework for PHP“ a teda Slim mikro framework pre PHP. Úvodné stránky opisujú Slim ako ideálny nástroj pre vytváranie API, ktoré konzumujú, opakovane využívajú či publikujú dáta. Podporuje rýchle vytváranie prototypov. Odkazuje sa na svoju jednoduchosť voči konkurenčným frameworkom ako Symfony alebo Laravel a užívateľa láka na svoj jednoduchý kód, ktorý ako opisujú vývojári je možné zvládnuť za jedno odpoludnie.

Namiesto balíka plných funkcií prichádza Slim s implementáciou základných HTTP metód s tým, že necháva užívateľa aby si do tohto frameworku implementoval jemu potrebné komponenty.

### XAMPP

Je nástroj, ktorý nám umožňuje vytvoriť Apache server na počítači bez nutnosti zakúpenia nejakej webovej služby a taktiež je nevyhnutným prvkom pre spustenie PHP kódu či databáz MySQL.

### Composer

Composer sme síce už spomenuli ale jeho význam a funkcie v tomto projekte sme si ešte nepopísali. Composer je tzv. dependenci manager čo v preklade znamená manažér závislostí ale tento termín sa v praxi nepoužíva. Pomocou tohto nástroja je možné sťahovať a inštalovať balíčky do projektov s tým, že sa tieto balíky inštalujú lokálne a nie globálne takže v rôznych projektoch môžete používať rovnaké balíky rôznych verzií. Composer sa tak stará o jednoduchú inštaláciu a manažment balíkov.

Pre inštaláciu Slimu pomocou Composera teda stačí zadať nasledujúci kód ( v závislosti od verzie):

1. composer require slim/slim "^2.\*"

## 

# [Výsledky práce a diskusia](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Výsledky_práce)

## Výsledky práce

### Nastavenie prostredia

Pred začiatkom na práci bolo nutné zhromaždiť a nainštalovať potrebné balíčky / moduly, ktoré neskôr v práci budeme využívať. Tieto balíčky sme do projektu nainštalovali pomocou spomenutého Composeru. Vzhľadom na použitú verziu Slimu 2 bolo nutné zvoliť staršie verzie balíčkov pre zaručenie ich kompatibility so Slimom verzie 2.

Obsah súboru composer.json :

1. {
2. "autoload":{
3. "psr-4":{
4. "Options\\": "app/Options"
5. }
6. },
7. "require": {
8. "slim/slim": "2.\*",
9. "ircmaxell/random-lib": "^1.2",
10. "twig/twig": "~1.0",
11. "alexgarrett/violin": "^2.2",
12. "illuminate/database": "^5.4",
13. "hassankhan/config": "^0.10.0",
14. "phpmailer/phpmailer": "^5.2",
15. "slim/views": "^0.1.3"
16. }
17. }

Pri tvorbe našej webovej služby sme začali najskôr pracovať so Slimom verzie 3, táto verzia sa podstatne líši od verzie 2 či už štruktúrou, obsiahlosťou, prehľadnosťou atď. no pri našom projekte sme sa rozhodli po určitom čase používať staršiu verziu vzhľadom na rozsiahlejšiu podporu balíkov, užívateľských návodov či pri riešení problémov.

### Návrh databázy

Pri návrhu a správe databázy sme využívali nástroj phpMyAdmin pomocou jazyka SQL, ktorý bol dostupný z prostredia XAMPP. Pred začatím programovania samotnej webovej služby bolo teda nutné najskôr navrhnúť databázu, do ktorej by sa ukladali informácie o užívateľoch, ich zariadeniach a taktiež hodnôt, ktoré tieto zariadenia obsahujú.

Keďže s návrhom databázy sme mali minimálne skúsenosti tak sme si najskôr preštudovali vytvorené databázy, na základe čoho sme potom dokázali vytvoriť našu vlastnú.

### Databáza

Naša databáza obsahuje 4 tabuľky, ktoré si ďalej bližšie opíšeme.

Tabuľka **user** predstavuje užívateľa.

|  |  |
| --- | --- |
| user | Popis |
| id | Automatické pridelenie ID každému užívateľovi |
| name | Prihlasovacie meno užívateľa |
| email | Prihlasovací email užívateľa |
| password | Heslo užívateľa |
| created\_at | Dátum a čas vytvorenia účtu |
| updated\_at | Dátum a čas úpravy účtu |
| api\_key | Pridelený API kľúč |

Užívateľ pri vytvorení účtu / registrácii automatický obdrží ID, ktoré sa spája s jeho zariadeniami. Do účtu sa môže prihlásiť voľbou prihlasovacieho mena alebo emailu a hesla. Automaticky sa mu pridelí čas vytvorenia, úpravy a  API kľúč.

Tabuľka **device** predstavuje zariadenie.

|  |  |
| --- | --- |
| device | Popis |
| id\_device | Automatické pridelenie ID každému zariadeniu |
| user\_id | ID vlastníka zariadenia |
| id\_type | ID typu zariadenia |
| device\_name | Meno zariadenia |
| created\_at | Dátum a čas vytvorenia zariadenia |
| updated\_at | Dátum a čas úpravy zariadenia |

Zariadenie vzniká jeho vytvorením cez užívateľské rozhranie vo webovej službe. Užívateľ môže vytvoriť viacero zariadení ale jedno zariadenie nemôže mať viacero vlastníkov. Každému zariadeniu je pridelený jeho typ, ID, ID vlastníka a meno zariadenia. Čas vytvorenia a úpravy sa mu automaticky pridelí.

Tabuľka **type** predstavuje typ zariadenia.

|  |  |
| --- | --- |
| type | Popis |
| id\_type | Automatické pridelenie ID každému typu |
| device\_name | Pomenovanie typu |

Každé zariadenie musí byť nejakého typu. A teda ak užívateľ napr. pridáva zariadenie teplomer, z tabuľky typov zvolí typ „thermometer“ a teda typ s ID č.1. Tieto typy sú v databáze už vopred pridané tvorcom aplikácie.

Tabuľka **device\_value** predstavuje hodnoty zariadenia.

|  |  |
| --- | --- |
| device\_value | Popis |
| id | Automatické pridelenie ID každej zaznamenanej hodnote |
| id\_device | ID zariadenia, ktorému patrí hodnota |
| created\_at | Dátum a čas vytvorenia hodnoty |
| updated\_at | Dátum a čas úpravy hodnoty |
| device\_val | Zaznamenaná hodnota |

Každé zariadenie obsahuje hodnoty, ktoré zaznamená a posiela ich webovej aplikácii. Opäť je jednotlivej hodnote priradený dátum a čas vytvorenia a jej úpravy ako aj ID. Jednotlivé hodnoty obsahujú ako aj samotnú hodnotu tak aj ID zariadenia, ktorej patria.

Pri práci s databázou sme využívali balíček **illuminate/database**, ktorý natívne využíva ako databázovú vrstvu framework Laravel. Pri získavaní údajov z databázy nám prácu uľahčoval query builder, pomocou ktorého sa práca s databázou stala veľmi jednoduchá a intuitívna, no používali sme aj klasický zápis SQL. Nevýhodou tohto balíčku bola nutnosť dodržiavania návrhu jeho modelu čo v našom prípade znamenalo, že tabuľky, ktoré aktívne využívali tento balíček museli implementovať tento model a tak zahŕňať aj polia ako updated\_at a created\_at či dodržiavať názvy polí napr. id v tabuľke user, ktoré bolo pôvodne navrhnuté ako user\_id.

Príklad použitia query builderu pre získanie maximálnej hodnoty z vybraného zariadenia **:**

1. Capsule::**table**('device\_value')->**where**('id\_device','=',$id)->**max**('device\_val');

### Autentifikácia a validácia

Vedieť rozlíšiť užívateľa a jeho bezpečnosť je v našej aplikácii nevyhnutné, aby tak nedošlo k neoprávnenému prístupu k dátam iných užívateľov bolo nutné vytvoriť užívateľskú autentifikáciu a taktiež validáciu pre korektné vytváranie užívateľov, zariadení a ostatných údajov. Autentifikácia užívateľa prebieha viacerými spôsobmi, ktoré zaručujú, že webová aplikácia pracuje práve s údajmi užívateľa a že užívateľ má právo k týmto dátam pristupovať.

#### API kľúč / token

Každému užívateľovi, ktorý si vytvorí účet pri registrácii je pridelený API kľúč. Tento API kľúč resp. API token je ďalej využívaný pri práci s naším RESTovým API rozhraním. Slúži ako parameter hlavičkového súboru pre autorizáciu užívateľa pred vykonaním určitej operácie cez API rozhranie, je vyžadovaný v závislosti od požadovanej operácie. Generovanie kľúču prebieha pomocou kryptovacieho algoritmu MD5 a generovanie kľúču vyzerá nasledovne:

Hashovanie:

1. 'api\_key' => md5(uniqid(rand()))

Vygenerovaný kľúč:

1. 051c97bd4289a9eef0b4f39f2a1231af

#### Prihlasovanie

Užívateľ svoju identitu preukáže webovej aplikácii pomocou prihlásenia zadaním mena / emailu a hesla, ktoré sa overia s databázou, pri úspešnom prihlásení sa priradí užívateľská session, na základe ktorej sa ďalej aplikácia riadi pri zobrazovaní užívateľských dát. Session je uložená na serveri, užívateľ k nej tak nemá prístup a nemôže ju modifikovať.

Validácia údajov:

1. $v->validate([
2. 'identifier' => [$identifier, 'required'],
3. 'password' => [$password, 'required']
4. ]);

Priradenie session:

1. $\_SESSION[$app->config->get('auth.session')] = $user->id;

#### Prístup

Keď naša aplikácia vedela rozoznať autentifikovaného a neautentifikovaného užívateľa, na základe tejto skutočnosti vedela povoľovať prístup k určitým funkciám stránky. Nezaregistrovaný užívateľ by tak nemal mať prístup k stránke zobrazujúcej zariadenia, užívateľský profil či možnosti zmeny hesla naopak zobrazovať stránku registrácie či prihlásenia pre už autentifikovaného užívateľa by bolo zbytočné. Tento problém sme vyriešili funkciou **authenticationCheck**, ktorá tento problém rieši.

1. $app->get('/devices/:id', $authenticated(), **function** ($id) **use**($app)

Pri požiadavke pre získanie stránky zobrazujúcej zariadenia parameter **$authenticated()** zavolá funkciu **authenticationCheck**, ktorá kontroluje autentifikáciu používateľa a na základe toho povoľuje prístup.

#### Validácia údajov

Pre validáciu v našej aplikácii sme použili balíček **violin**,jeho implementácia bola jednoduchá a ponúkala už širokú škálu preddefinovaných pravidiel pre validáciu. Jednotlivé pravidlá sa od seba oddeľujú symbolom „ | “.

1. $v->validate([
2. 'email' => [$email, 'required|email|uniqueEmail'],
3. 'name' => [$name, 'required|alnumDash|max(20)|uniqueUsername'],
4. 'password' => [$password, 'required|min(6)'],
5. 'password\_confirm' => [$passwordConfirm, 'required|matches(password)'],
6. ]);

|  |  |
| --- | --- |
| Názov | Popis |
| required | Vyžaduje zadanie hodnoty |
| min(6) | Zadaná hodnota musí mať minimálne 6 znakov |
| matches(password) | Kontroluje či sa zadaný reťazec zhoduje s parametrom |

Použitie balíčku **violin** značne urýchlilo kontrolu vstupných údajov svojim jednoduchým použitím a možnosťou vytvárania vlastných pravidiel, vybrané pravidlá pre kontrolu, ktoré boli použité v našej aplikácii sme si opísali v tabuľke vyššie.

### Zobrazovanie stránok

Zobrazovanie stránok v našej aplikácii prebieha dynamicky. Pomocou šablóny a fameworku Bootstrap bola vytvorená základný vzhľad stránky, ktorej jednotlivé časti ďalej implementujú funkcie našej aplikácie.Nato aby bolo zobrazovanie týchto stránok v prostredí frameworku Slim bolo nutné použiť balíčky **Twig-View** a **Slim-Views**, ktoré spolu navzájom spolupracujú. Tieto balíky umožňujú prácu s PHP v našich šablónach s minimálnym kódom, pôvodný PHP kód zjednodušujú vlastným zápisom. Samotný kód sa vkladá do tzv. blokov.

1. {% block content %}
3. ...  obsah ...
5. {% endblock %}

Základ zobrazovania stránok v našej aplikácii tvorí šablóna default.php, táto šablóna zobrazuje bočný panel pre odkazy ako aj navigačný panel s užívateľskou ponukou, načíta všetky potrebné komponenty Bootstrapu a definuje **block content** a teda priestor, kde sa bude zobrazovať obsah ostatných stránok. Tento obsah taktiež podlieha autorizácií, určité odkazy sú tak zobrazované len neautentifikovaným alebo autentifikovaným užívateľom.

1. {% if auth %}
3. ... obsah ...
5. {% endif %}

#### View a Presenter

Pre zobrazenie stránky potrebujeme vytvoriť jej šablónu ( view ) a back-end, ktorý ju bude definovať a bude sa starať o údaje, ktoré sa do nej pošlú ( presenter ). Štruktúra našej stránky tak obsahuje zložku **views**,ktorá obsahuje naformátované šablóny a zložku **routes**, ktorá obsahuje presentery, ktoré tieto šablóny spravujú.

View pre jednotlivé stránky vyzerá nasledovne, dedí prostredie základnej šablóny a tak získava navigačné menu ako aj postranné linky, formátovanie, kaskádové štýly a všetko čo táto šablóna definovala a dopĺňa ju o svoj obsah.

1. {% extends 'templates/default.php' %}
3. {% block tittle %} ... nadpis ... {% endblock %}
5. {% block content %}
7. ... obsah ...
9. {% endblock %}

Presentery šablóny definujú s akým obsahom bude šablóna pracovať, ako bude reagovať na požiadavky GET a POST.

1. $app->get('/devices\_edit/:dev\_name', $authenticated(), **function** ($dev\_name) **use**($app)
2. {
3. $session = $\_SESSION[$app->auth->user];
5. $device = Capsule::select('SELECT \* FROM device WHERE device\_name = "'.$dev\_name.'"');
6. $types = Capsule::table('type')->get();
7. var\_dump($device);
8. $app->render('user/devices\_edit.php', [
10. 'session' => $session,
11. 'dev' => $device,
12. 'types' => $types,
13. ]);
15. })->name('edit');

V danej ukážke skráteného kódu vidíme čo sa udeje pri požiadavke o získanie tejto stránky. V prvom riadku definujeme adresu, na ktorej bude šablóna pre editovanie zariadení dostupná, bude vytvorená za základe parametru dev\_name. Po vykonaní ostatných operácií sa premenné, s ktorými budeme v šablóne pracovať vkladajú ako pole do funkcie render. V šablóne bude potom premenná $device dostupná pod názvom dev. Danej ceste môžeme priradiť meno ako v riadku č.15 **edit**.

Z ostatných stránok aplikácie na ňu potom môžeme odkazovať nasledujúcim spôsobom:

1. **<a** href="{{ urlFor('edit') }}"**>**Edit**</a>**

## Diskusia

Výsledky, ktoré boli dosiahnuté riešením.

# [Zoznam použitej literatúry](file:///C:\Users\DanielJežík\AppData\Roaming\Microsoft\Word\Pokyny_pre_vypracovanie_ZP.docx#Zoznam_použitej_literatúry)

<http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069?pgno=1>

<https://sk.wikipedia.org/wiki/Jednotn%C3%BD_identifik%C3%A1tor_prostriedku>

<https://en.wikipedia.org/wiki/Uniform_Resource_Identifier>

<http://wiki.servicenow.com/index.php?title=REST_API#gsc.tab=0>

<http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>

<http://www.itnetwork.cz/nezarazene/stoparuv-pruvodce-rest-api>

<http://www.abclinuxu.cz/blog/backinabag/2013/10/co-to-je-rest>

<https://sk.wikipedia.org/wiki/Webov%C3%A1_slu%C5%BEba> (web sluzba )

<https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>

<https://en.wikipedia.org/wiki/Representational_state_transfer#Architectural_properties>

<https://www.tutorialspoint.com/restful/restful_introduction.htm>

<http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming>

Citaty ekonomika

<https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#5a5524eb292d>

[https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world- applications-of-internet-of-things-iot/](https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world-%20applications-of-internet-of-things-iot/)

<https://arstechnica.com/gadgets/2016/06/control-all-your-smart-home-accessories-from-a-single-app-called-home/>

<https://www.slovanet.net/sk/internet/iot/>

<https://www.alza.sk/inteligentne-osvetlenie-philips-hue/18860240.htm?kampan=adw2_osvetleni_philips-hue_c_9062580_b_1t1_150544901695&gclid=CjwKEAjwzKPGBRCS55Oe46q9hCkSJAAMvVuMZCef7Els4-LlhXjdBsY6HQKkzPrnqdlb2bSq2cTMiRoCgULw_wcB>

<http://mqtt.org/faq>

<http://www.o2bs.sk/smart-connect>

<https://trends.google.com/trends/explore?date=all&q=xml%20api,json%20api>

<https://www.aig.com/knowledge-and-insights/the-rise-ramifications-and-risks-of-the-internet-of-things> - - autonomne vozidla

<https://cs.wikipedia.org/wiki/Model-view-controller>

<https://cs.wikipedia.org/wiki/Model-view-controller>

<http://vyvoj.hw.cz/internet-veci-a-protokol-mqtt.html>

<https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>

mqtt

<http://automatizace.hw.cz/files/styles/full/public/story_automat/11853/mqtt_publisher_subscriber.png?itok=HU8x__X9>

<https://www.keycdn.com/blog/front-end-frameworks/>

<http://getbootstrap.com/getting-started/#download>

<http://getbootstrap.com/getting-started/#whats-included-source>

<http://automatizace.hw.cz/zakladni-uvod-do-oblasti-internetu-veci-iot.html>

<https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>

<https://en.wikipedia.org/wiki/Constrained_Application_Protocol>

<https://www.element14.com/community/groups/internet-of-things/blog/2017/02/05/iot-protocols-an-overview>

<https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php>

<https://en.wikipedia.org/wiki/XMPP>

<https://cs.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol>

<https://www.digikey.com/en/maker/blogs/how-important-are-apis-in-iot/1c3a3a9a3681458f9a4c019ffdc5f2c6>

<https://www.pluralsight.com/blog/tutorials/representational-state-transfer-tips>

# [Zoznam príloh](C:\\Users\\DanielJežík\\AppData\\Roaming\\Microsoft\\Word\\Pokyny_pre_vypracovanie_ZP.docx" \l "Prílohy)

**Príloha A** Názov

**Príloha B** Názov

# 

# Prílohy

## Príloha A: Názov prílohy

Každá ďalšia príloha začína na novej strane.

## Príloha B: Obsah DVD

Priložené DVD obsahuje:

* Práca v elektronickej podobe (formát PDF)