**ŽILINSKÁ UNIVERZITA V ŽILINE**
FAKULTA RIADENIA A INFORMATIKY


# Prenos dát

## Semestrálna práca

Katedra technickej kybernetiky

Bc. Jaroslav Bútora
Bc. Lukáš Malatinský

Žilina, 2017

# 1. Špecifikácia zadania, konkretizácia problému

Zadanie semestrálnej práce spočívalo vo vytvorení webovej aplikácie, ktorá by spracovávala IOT dáta. Preferované nástroje pre vytvorenie tejto semestrálnej práce bolo využitie jazyka Java, webová platforma DropWizard, databáza MySQL a použitie ORM Hibernate. Na tejto semestrálnej práci môže pracovať tím maximálne troch študentov.

Semestrálna práca má pozostávať z troch komponentov:

1. REST Web servis
2. Prezentačná časť
3. Koncové zariadenie

## 1.1. REST Web servis

Tento servis bude prijímať a poskytovať dáta. Daný web servis bude pozostávať z minimálne troch koncových bodov. (užívatelia, zariadenia a dáta pre zariadenia). Nad jednotlivými koncovými bodmi má byť možné vykonať CRUD operácie.

## 1.2. Prezentačná časť

Webové rozhranie, bude zobrazovať namerané údaje, zoznam zariadení atď. Pre lepšie hodnotenie je potrebné implementovať prihlásenie sa do systému pomocou užívateľského mena a hesla.

## 1.3. Koncové zariadenie

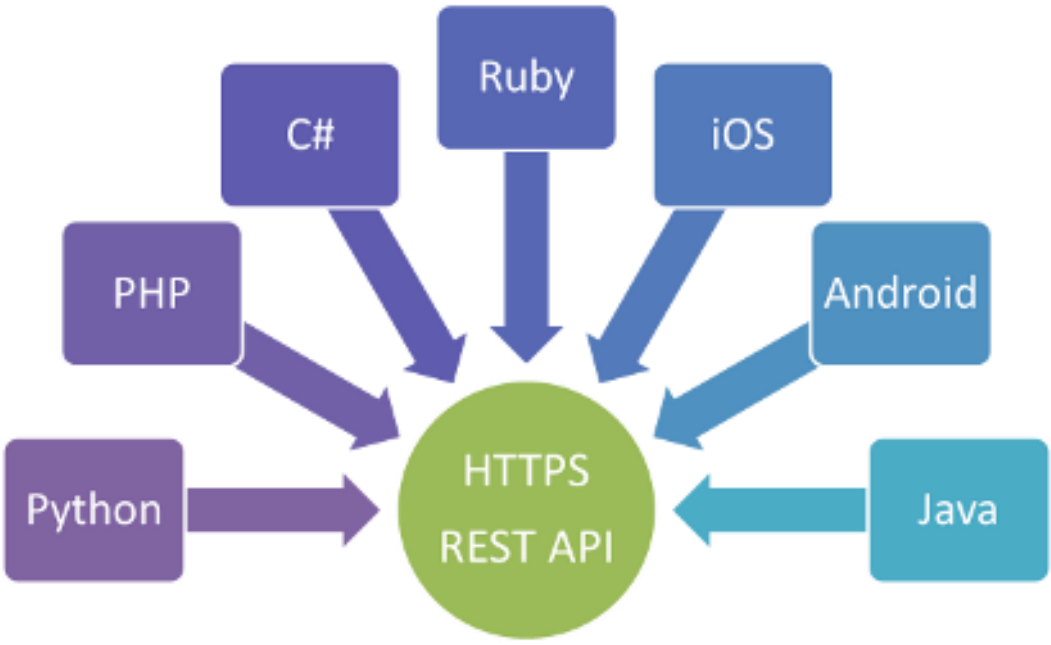
Buď reálne zariadenie postavené na ľubovoľnom MCU, Raspberry Pi alebo emulované zariadenie, ktoré bude spúšťané na PC. Toto koncové zariadenie bude predstavovať senzor(napr. teplota, tlak, svetelnosť, rýchlosť vetra atď. ), ktorý bude odosielať dáta cez REST rozhranie na server.

Jedným z najpopulárnejších typov API je REST, niekedy nazývané aj RESTful API. REST alebo RESTful API boli navrhnuté tak, aby výhodne stavali na existujúcich protokoloch. Zatiaľ čo REST alebo Representational State Transfer môže byť použitý skoro s každým

protokolom, v spojení s webovými API sa zvyčajne využívajú výhody HTTP. To znamená, že vývojári pri vytváraní REST API nemusia inštalovať prídavný software alebo knižnice.

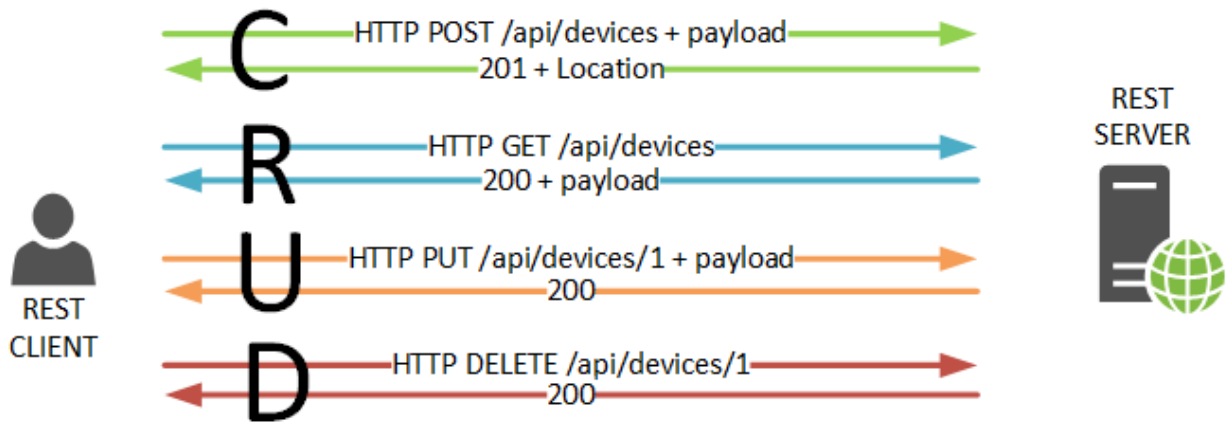
Jednou z hlavných výhod REST API je, že poskytuje vynikajúcu flexibilitu. Dáta nie sú viazané na zdroje alebo metódy, takže REST dokáže spracovať rôzne typy volaní a vrátiť rôzne dátové formáty. Táto flexibilita dovoľuje vývojárom vytvoriť API podľa potreby v kombinácii so spĺňaním požiadaviek veľmi odlišných zákazníkov.

Na rozdiel od SOAP REST nie je viazaný na XML, ale dokáže vrátiť XML, JSON, YAML alebo ktorýkoľvek iný formát v závislosti na tom, ktorý klient vyžaduje. A na rozdiel od RPC nie sú užívatelia nútení vedieť názvy procedúr alebo špecifické parametre v danom poradí.



# 2. Krátka analýza

Ako už bolo spomenuté, jednou z hlavných výhod **REST API** je, že poskytuje vynikajúcu flexibilitu. Dáta nie sú viazané na zdroje alebo metódy, takže REST dokáže spracovať rôzne typy volaní a vrátiť rôzne dátové formáty. Vďaka tejto flexibilite je REST API celkom populárne. Toto REST rozhranie má poskytovať CRUD operácie. Tieto operácie sú skratkou C- Create, R-Read, U-Update a D-Delete.




## 2.1. Dropwizard

je Java framework určený pre vysoko výkonné RESTful webové servisy. Balancuje na hranici medzi knižnicou a frameworkom. Jeho cieľom je poskytovať výkonné, spoľahlivé implementácie všetkého, čo potrebuje webová aplikácia. Obsahuje v sebe niekoľko ďalších známych knižníc ako napríklad:

### 2.1.1. Jetty pre HTTP

Pretože web aplikácia bez HTTP by nemala zmysel, Dropwizard používa Jetty HTTP knižnici aby ju zakomponoval do projektu. Namiesto predávaniu aplikácie komplikovanému aplikačnému serveru má Dropwizard hlavnú metódu main, v ktorej spúšťa HTTP server.

Spustenie aplikácie ako jednoduchý proces eliminuje početné množstvo problémových prvkov a dovoľuje použiť existujúce Unixové nástroje pre správu procesov.

#### 2.1.2. Jersey pre REST

Umožňuje napísať prehľadné, testovateľné triedy, ktoré mapujú HTTP requesty na jednoduché Java obejkty. Podporuje Streamovací výstup, matičné URI parametre, GET requesty s podmienkou a omnoho viac.

#### 2.1.3. Jackson

Jackson je populárna Java knižnica ktorá umožnuje mapovať JSON objekty na Java objekty.

#### 2.1.4. Metrics

Knižnica Metrics všetko zaobaľuje a poskytuje jedinečný náhľad do správania sa kódu vo vývoji.

#### 2.1.5. Iné knižnice

Ako Guava, Logback, Freemarker, Joda Time

### 2.2. Analýza semestrálnej práce

Aplikácia bude bežať na localhoste teda budeme využívať program WampServer a bude bežať na adrese http://localhost:8080/. Koncové zariadenie budeme mať emluovaná senzory pre meranie teploty vzduchu, tlaku vzduchu a výšky hladiny rieky. Prezentačná časť bude webové rozhranie ktoré poskytuje REST služby. Tu sa bude môcť užívateľ zaregistrovať, prihlásiť sa. Ďalej si bude môcť pozrieť všetky namerané hodnoty od daného senzora, jednu konkrétnu hodnotu. Takisto bude možné vymazať jednu hodnotu alebo všetky hodnoty. Užívateľ bude môcť taktiež pridať hodnotu alebo upraviť už prijatú hodnotu.

# 3. Implementácia

Semestrálne práca nadväzuje na prácu na cvičeniach, teda je vyvíjaná v jazyku JAVA a v prostredí NetBeans 8.2. Tu sme už len projekt upravili a doplnili o požadovanú funkčnosť.

V súbore hibernate.cfg.xml sme si nastavili vlastnú databázu a prístup na ňu:

```
<!-- Database connection settings -->
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="connection.url">jdbc:mysql://localhost:3306/prenosdat</property>
<property name="connection.username">root</property>
<property name="connection.password">root</property>
```

Ďalej sme vytvorili triedy jednotlivých zariadení a užívateľa. Pre každé zariadenie sme taktiež vytvorili tabuľku v databáze: Temp, Pressure, Level, USER. Ďalej sme tie triedy doplnili o požadované parametre. Pri zariadení ide len o ID a hodnotu, pri užívateľovi to je ID, meno a heslo.

V triedach Resource sme zadefinovali cesty ku daným CRUD funkciam. Pre každé zariadenie sme najskôr určili hlavnú cestu pre prístup ku celej triede:

```
@Path("/temp")
@Produces(MediaType.TEXT_HTML)
public class TempResource {
```

A ďalej cesty ku jednotlivým funkciám:

```
@GET
public TempView getTemp(@QueryParam("name") Optional<String> name) {
```
- získanie všetkých hodnôt

```
@GET
@Path("/{id}")
public TempView getTemp(@PathParam("id") String id) {
```
- získanie hodnoty podľa ID

```
@GET
@Path("/delete/{id}")
public Response deleteTemp(@PathParam("id") String id) {
```
- vymazanie hodnoty podľa ID

```
@GET
@Path("/delete")
public Response deleteTemp(@QueryParam("name") Optional<String> name) {
```
- vymazanie všetkých hodnôt

```
@POST
@Path("/new")
public Response newTemp(@FormParam("temp") String temp) {
```
- pridanie novej hodnoty

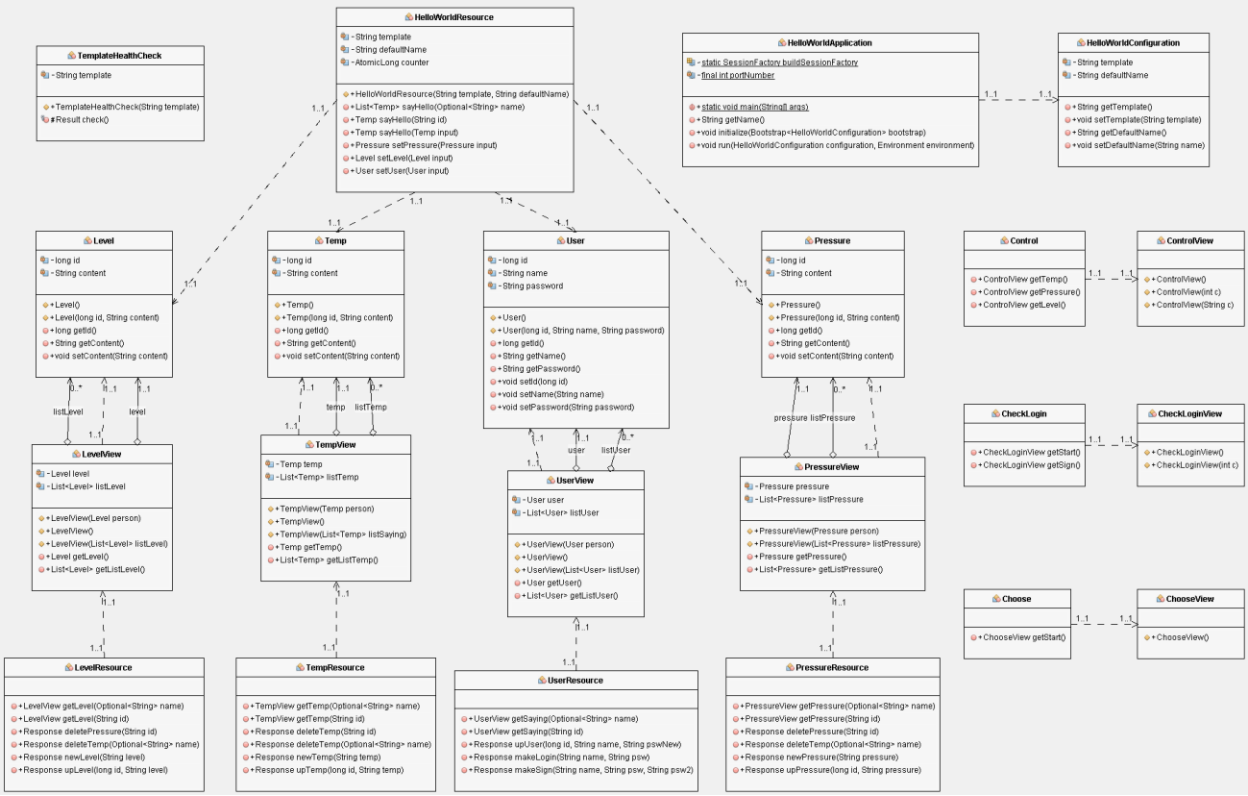
```
@POST
@Path("/up")
public Response upTemp(@FormParam("id") long id, @FormParam("temp") String temp) {
```
- aktualizácie hodnoty

Podobne je to pri užívateľovi, avšak tu je ešte funkcia pre overenie či zadaný užívateľ je už zaregistrovaný, ak áno tak ho prihlási.

```
@POST
@Path("/login")
public Response makeLogin(@FormParam("name") String name, @FormParam("psw") String psw) {
```

A funkciu pre registráciu, ktorá overuje či náhodou užívateľ s daným menom už náhodou nie je zaregistrovaný.

```
@POST
@Path("/sign")
public Response makeSign(@FormParam("name") String name, @FormParam("psw") String psw, @FormParam("psw2") String psw2) {
```



Ďalej pokračujeme vytvorením emulovaných senzorov. Je to nový Java projekt v ktorom je len jedna trieda. Tá obsahuje funkciu postTemperature(), ktorá zabezpečuje nadviazanie spojenia a odoslanie Json-u.

```
// HTTP Post request
private void postTemperature() throws Exception {

    String url = "http://localhost:8080/add/temp";
    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();

    // Setting basic post request
    con.setRequestMethod("POST");
    con.setRequestProperty("User-Agent", USER_AGENT);
    con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
    con.setRequestProperty("Content-Type", "application/json");

    String postJsonData = "{\"id\": 1,\"content\": "+temp+"}";

    // Send post request
    con.setDoOutput(true);
    DataOutputStream wr = new DataOutputStream(con.getOutputStream());
    wr.writeBytes(postJsonData);
    wr.flush();
    wr.close();

    int responseCode = con.getResponseCode();
    System.out.println("nSending 'POST' request to URL : " + url);
    System.out.println("Post Data : " + postJsonData);
    System.out.println("Response Code : " + responseCode);

    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String output;
    StringBuffer response = new StringBuffer();

    while ((output = in.readLine()) != null) {
        response.append(output);
    }
    in.close();

    //printing result from response
    System.out.println(response.toString());
}
```

Main metóda obsahuje nekonečný ciklus v ktorom sa generujú náhodne hodnoty, ktoré sa potom odosielajú v určenom intervale.

Ďalej nasledoval návrh prezentačnej časti a teda Web rozhrania. Pre zjednodušenie práce sme pri návrhy používali nástroj PSPad, v ktorom bolo možne jednoducho zobraziť návrh stránky.

Stránka sa skladá zo štýlu, v ktorom sú pripísané objekty na stránke, ich rozmer tvar, farba, typ a veľkosť písma, rozloženie a podobne. A z formulárov ktoré určujú metódu, teda buď POST alebo GET a akciu ktorá sa má vykonať.

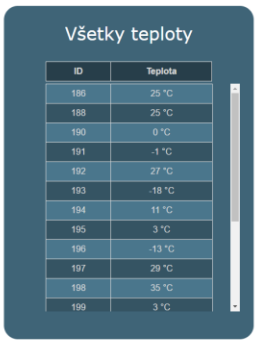
```
<form id="div_centrovany" method="post" action="http://localhost:8080/level/up" style="margin-top: 25px; line-height: 150%">
    <div  style="margin-top: 2px">
        Uprav ID: <br>
        <input type="text" name="id" required  style="width: 60%">
        <br>
        Nová hladina rieky: <br>
        <input type="text" name="level" required  style="width: 60%">
        <br>
        <input type="submit" value="Upraviť" style="margin-top:5px">
    </div>
</form>
```

Každá stránka má takto vytvorený svoj *.ftl súbor, ktoré sa volajú v jednotlivých triedach View.

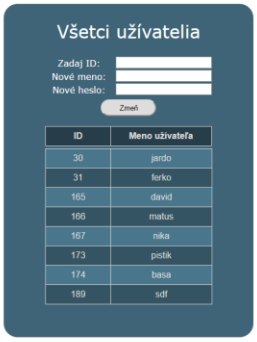

Registrácia
Meno:
Heslo:
Heslo znova:
Registrovať
SMART HOUSE


Teploty
Tlak
Výška hladiny
Všetci užívatelia


Teplota
Zadaj id teploty, ktorú chceš vypísať
Zobraz
Zobraz všetky teploty
Zobraz
Zadaj id teploty, ktorú chcete vymazať
Vymaž
Vymazať vsetky teploty
Vymaž
Pridajte novú teplotu
Pridaj
Uprav ID:
Nová teplota:
Upraviť


Všetky teploty
ID
Teplota
186 25 °C
188 25 °C
190 0 °C
191 -1 °C
192 27 °C
193 -18 °C
194 11 °C
195 3 °C
196 -13 °C
197 29 °C
198 35 °C
199 3 °C


ID:188
Teplota: 25 °C


Všetci užívatelia
Zadaj ID:
Nové meno:
Nové heslo:
Zmeň
ID
Meno užívateľa
30 jardo
31 ferko
165 david
166 matus
167 nika
173 pistik
174 basa
189 sdf

# Záver

Cieľom semestrálnej práce bolo overenie nadobudnutých vedomostí z predmetu Prenos dát. Vytvorili sme semestrálnu prácu, ktorá vie zobrazovať na Web aplikácie pomocou REST služieb namerané údaje z IOT zaradení.