**Rapport**

1. Introduktion
   1. Beskrivelse af problemet som systemet skal løse og det mål der er forsøgt opnået
2. Design
   1. Arkitektur (hvilke komponenter systemet består af)
   2. Beskrivelse af funtionaliteten af de enkelte komponenter
   3. Gerne en illustration der viser komponenterne og kommunikationen mellem dem
3. Implementation
   1. Valg af teknologier og argumentation for disse valg
   2. Kort beskrivelse af implementationen af hver komponent
4. Aftestning af systemet
   1. Beskriv kort hvad I har gjort for at teste

I alt forventer jeg at rapporten har et omfang på 10-20 sider. Der gives ikke point per skreven linje - en god rapport er en kort, præcis og informativ tekst, der beskriver netop hvad der er relevant for at besvare ovenstående.  
Derudover afleveres en zip-fil med kode og en beskrivelse af hvordan systemet bringes til at køre.

# PA International Web Presence

## Introduction

**Design**  
In the design of a RESTful web-service we need to identify the resources. In terms of physical resources we have devices. The zones are considered to be a conceptual resource that will lend itself to the URI structure. The devices have a configuration, which is considered a resource within the device. The device readings, status and locations have resulted in some discussion, but as there are a potentially large number of readings, and the user might request any combination/number of readings by specification of a time interval and/or a status flag, we’ve decided to access readings through query parameters. The final URI structure is presented in the table below. The table also names the HTTP requests that will go along with the URIs and the system interpretation of such requests. The use of the HTTP requests follow the W3C guidelines as specified in <http://www.w3.org/Protocols/rfc2616>.   
  
Web-service API

|  |  |  |
| --- | --- | --- |
| URL (www.pa.com/geolog/) | Request | Description |
| devices | GET | Returns a list of all devices registered with the service. This list contain device ID links only. These links must be queried for the details about the device. If authentication is employed only the devices accessible to the user are returned. |
| devices?owner=peter | GET | Used to indicate that only a subset of the devices should be returned, namely the ones owned by peter. |
| devices?status=ERROR | GET | Used to indicate that only a subset of the devices should be returned, namely the ones that are currently in an error state. |
| devices?... | GET | Multiple other filter possibilities exist, including limiting the amount of information returned, e.g. returning only the location and not the last readings (?noReadings=on), etc. |
| zones | GET | Returns a list of all zones created with the service. This list contains zone ID links only. The links must be queried for the details about the zone. If authentication is employed only the zones accessible to the user are returned. |
| devices/<ID> | GET | Return details about this device including information about owner and its past and current readings, status and location. |
| devices/<ID>?from=<unix time>&to=<unix time> | GET | Return the readings, status and location information for the specified device and the indicated time interval. |
| devices/<ID>?noLocaton=on&noStatus=on | GET | Return the readings for the specified device (no status or location information). |
| devices/<ID>?... | GET | Multiple other filtration possibilities exist, including limiting the amount of information returned, e.g. from=NOW, indicating that only the current status, location and readings should be returned. |
| zones/<ID> | GET | Return the details about the zone, including the information required to draw the zone in Google Maps. |
| devices | PUT | Creates a new device. This is not often used, as a device register itself on the system when it POST its first reading/status/location, however it is possible to create a device before it registers itself. |
| devices/<ID> | POST | Add one or more new readings/locations/status-info for the given device to the device with the given ID. If the device is new to the system it is registered with the system. |
| devices/<ID>/configuration | GET | Return the current configuration for the device |
| devices/<ID>/configuration | POST | Update the current configuration for the device. This is a POST as a device always has a configuration, and a PUT (add a new) therefore do not make sense. |
| zones | PUT | Create a new zone. |
| zones/<ID> | POST | Update an exiting zone. |
| zones/<ID>/devices | GET | Utility method. Return links to the devices that ar at present in this zone. This list can be calculated by comparing the polygon of the zone and the device location. |
| devices/<ID>/zone | GET | Utility method. Return the zone that the device is in, if any. |
| devices/<ID>/zone | POST | Manual override method for directly assigning a zone to a device, effectively disabling the automatic zone assignment for this device. |
| devices/<ID>/owner | GET | Return the owner of the device, if any |
| devices/<ID>/owner | POST | Update the owner of the device |

For details about the data format used for exchanging the information mentioned above please refer to the XML schema found in geolog.xsd.

## Implementation

The implementation is described on the components web-service, web-server ad client in turn.  
**The web-service**  
We’ve chosen to implement the web-service as Servlets on an Apache tomcat server. In our context the advantages of this choice are:

1. multi-threading - we expect several simultaneous requests (devices and web page)
2. the declarative configuration lends itself to the mapping from request URIs to resources on the server.

The web-server

### The client

The client is implemented as a stub for testing purposes. It is a command-line based application taking relevant parameters. On application launch the client will start to transmit readings and status information to the web-service based on the start-up parameters. The construct enables us to launch several clients with different signatures for testing the web-service and the web-server.

## Testing

## References

1. C. Pautasso, O. Zimmermann, F. Leymann. RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision.WWW 2008 / Refereed Track: Web Engineering - Web Service Deployment.
2. Roger L. Costello. <http://www.xfront.com/REST-Web_Services.html>