Møde 1. 20100830  
Spørgsmål til afklaring.

1. Specifikation af REST i WSDL 2.0. Kan man det? Hvordan?
2. Metode til specifikation af REST Web-service?
3. HTTP POST vs. PUT. Hvad er forskellen? Giver det mening af sammenligne med SQL INSERT og UPDATE?
4. Skal vi bruge PUT og DELETE?
5. URL schema? Strategi for opbygning af vore URL’er, så vi anvender best practice. Brug query fields vs. ressourcen direkte i URL’en (e.g. /devices?ID=1 vs. /devices/1)
6. Anbefalinger for authentication i REST? Bør device owner stå som en del af URL?

Uddybelse og afklaring:  
AD. 1: Følgende citat *... the latest WSDL version 2.0 could be applied to describe RESTful Web services, thanks to its more ﬁne-grained control over the HTTP binding and the possibility of supporting non-SOAP message encodings.* i vores reference #1 (C. Pautasso, O. Zimmermann, F. Leymann). Besvarer første del af spørgsmålet. Mht. anden del har vi ikke fundet noget godt. Derfor vælger vi at beskrive vores løsning ved at lægge os op ad denne <http://www.openrdf.org/doc/sesame2/system/ch08.html#d0e168>  
  
AD. 2: <http://www.xfront.com/REST-Web-Services.html> indeholder en spiseseddel i afsnittet *REST Web Service Design*. Vi var på mødet i gang med at identificere *the conceptual entities* som vi ønsker at stille til rådighed på vores web-service - der er tilføjet lidt til rapporten efter denne strategi .  
  
AD. 3: Her er det vist bedst at gå til kilden.   
POST: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.5>   
PUT: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6>  
Følgende citat fra ovenstående tydeliggør forskellen: *The fundamental difference between the POST and PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request -- the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource.* Dermed må det være klart, at POST er den rette request at anvende ved device upload af nye målinger. Det kan evt. overvejes at anvende PUT når et device registrerer sig selv første gang. Idet dette medfører oprettelse af en ny ressource (URI) på web-servicen.  
Endvidere er der i <http://www.xml.com/pub/a/2004/12/01/restful-web.html> nævnt en mapping mellem CRUD (Create, Read, Update, Delete) fra databaser til HTTP POST, GET, PUT, DELETE - de er enige i at POST er til Create (INSERT), men de argumenterer for at anvende PUT til at opdatere. Hvilket også er i overensstemmelse med definitionen af PUT i HTTP/1.1 specifikationen.  
  
AD. 4: Se diskussion mht. PUT under Ad. 3. Mht. DELETE kan der igen henvises til HTTP 1.1 specifikationen <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.7>. Eneste kandidat til DELETE i vores projekt er zoner.  
  
AD. 5: I ref. #1 nævnes, at omkostningen ved REST ligger i en omhyggelig analyse og design af URL strukturen. Ved at anvende en query string kan angive hvad det er man oplyser, men under URL vil man skulle bruge et ekstra lags dybde for det, e.g. /devices?ID=1 ved man at det er ID der er 1, mens /devices/1 er det ukendt hvad 1 betyder. Er der en “standard” for hvordan man udtrykker disse URL’er?   
I opgaven om REST fra WEBTEK seminar 2 anvendes en opskrifts navn som /recipes/<name>, men der er ikke noget om hvordan det dokumenteres at hvis man skriver noget efter /recipes så er det et navn, eller endnu vigtigere, hvis man ønsker at kunne skrive noget der ikke er et navn, hvordan håndteres det så. Er det god kotume at undgå det, f.eks. kan spørgsmålet om devices i zone 1 udtrykkes /devices/zone/1, eller /zone/1, hvor det sidste ikke giver problemer med at zone kunne være et device ID. Vi styrer selvfølgelig selv vores ID’er, så der kan vælges et passende pattern for en ID. I dette dokument <http://www.xfront.com/REST-Web_Services.html> beskrives en løsning som vi kan være relevant for vores projekt.  
  
AD.6: Hvis man requester som /user/devices er det muligt at få en bestemt brugers devices uden authentication, men hvis man bruger authentication (e.g. HTTP Basic) så vil man skulle indtaste user mere end én gang (hvorfor det?) , og da man kender user via authentication kunne man bare requeste /devices og så lade serveren hente dem brugeren har rettigheder til. Hvad anbefaler REST? Se nærmere om HTTP authentication her <http://www.apps.ietf.org/rfc/rfc2617.html>  
  
Andre:

1. Vi diskuterede om devices skulle være servere eller om de skulle være klienter. Da en web-service er passiv af natur (server), vil det betyde at der skulle oprettes en polling server uafhængigt af denne, som kunne requeste data fra devices. Dette har forskellige fordele og ulemper. polling-serveren vil nemmere kunne fordele pollingen sådanne at load bliver jævn, hvor devices godt kan finde på at sende alle deres data samtidig (og måske overbelaste web-servicen). Omvendt er devices som klienter simplere, da der ikke behøves en polling server. Rent logisk giver begge mening, da sporadiske ændringer i konfigurationen peger i retning af et device som server, og sporadiske ændringer i device status peger i retning af device som klient. Rent læringsmæssigt er der ikke den store forskel, og vi vælger derfor den simpleste løsning - devices som klienter. Dette betyder at devices skal polle konfiguration på faste tidspunkter. Dette kan gøres ved at anvende GET og HTTP response code 304 Not Modified tli at indikere inden ændring (passer godt med standarden).

## Referencer uden hyperlinks

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.