Hurtigt

Nyuddannet datalog og passioneret kitesurfer Jakob Aarøe Dam har lavet et mashup, der viser, hvor vindforholdene er gunstige for kitesurfing. Google App Engine gjorde vejen fra idéen til skyen kort.

Af Stig Andersen, prosabladet@prosa.dk
Foto: Søren Holm, Chili

Det er sjældent, at man kan kombinere specialeskrivning og sin yndlingshobby. Men for Jakob Aarøe Dam, 27 år og nyuddannet datalog fra Datalogisk Institut på Aarhus Universitet, lå det ligefor. Som ivrig kitesurfer har Jakob jævnligt udfordringen med at finde det ideelle sted at kitesurfe på det tidspunkt, hvor lysten og muligheden er der, og så inden for en overkommelig afstand. Ud over afstanden er vindretning, vindstyrke og strøm de vigtigste parametre, når man skal finde de gode surfspots.

Så han stillede sig selv den konkrete opgave at udvikle en service, der skulle samle globale vejrdata fra forskellige kilder, knytte dem til brugeroprettede surfspots og præsentere de enkelte kitesurfere for vejrdata fra de spots, der ligger i nærheden af kitesurferen.

Samtidig med, at Jakob beskrev den teoretiske ramme for implementationen i specialet, gik han i januar 2009 i gang med at udvikle welovewind.com, en lokationsbaseret webservice med et mashup af meteorologiske data. For hurtigt at komme i gang med den konkrete implementation valgte han at udvikle servicen ved hjælp af Google App Engine (GAE).

– Min oprindelige idé var at udvikle i Java med et Java-framework til at understøtte applikationen. Men det ville betyde, at jeg skulle sætte en Java-host op på min egen server, hvilket både ville koste tid, bekymringer og penge, og som jeg alligevel ikke ville få nogen "point" for. Så jeg valgte GAE, der er nem at gå til og gratis i sin basisudgave, fortæller han.

GAE stiller en komplet udviklingsplatform til rådighed. For Jakob betød det, at han kunne

op i skyen

gå direkte i gang med at løse selve opgaven frem for at rode med serverproblemer.

GAE den rigtige platform

– Arbejdet med welovewind.com har bekræftet mig i, at GAE var den rigtige platform at vælge, selvom der selvfølgelig er nogle begrænsninger. Den helt afgørende effekt af at bruge en platform som GAE er, at der er meget kortere vej, fra man får en god idé, til webservicen ligger ude i skyen, siger han.

Jakob kendte nogle af begrænsningerne i GAE, inden han begyndte på sit projekt. Andre stødte han på, efterhånden som han gravede sig dybere ned i detaljerne. Ved projektets start var Python det eneste supporterede sprog, den maksimale varighed på en request var 10 sekunder, og enhver processering kunne kun initieres, når en http-klient kaldte op. Og så stødte han på begrænsningen, at GAE kun tillader ulighedsfilter på én egenskab i database queries. Det var kritisk, da det var helt afgørende, at webservicen filtrerer data, så de rent geografisk er relevante for brugeren.

Da det kun er muligt at benytte ét ulighedsfilter, kan der ikke filtreres på basis af både længdegrad og breddegrad. Løsningen blev at konvertere geografiske punkter til en værdi i én dimension ved brug af en metode, der bevarer lokaliteten nogenlunde, en såkaldt space-filling curve. Herefter kan geografisk data filtreres til et bestemt område ved brug af kun ét ulighedsfilter. Fordelen ved denne tilgang, ud over at det fungerer på GAE, er, at forespørgslerne efter geografiske punkter kan understøttes af et normalt indeks.

Jakob præsenterede sit projekt på årets JAOO-konference i Århus på tracket "Cloudy Computing". Og her valgte han netop at fokusere på de udfordringer, han var løbet ind i på GAE, og de løsninger, han havde implementeret.

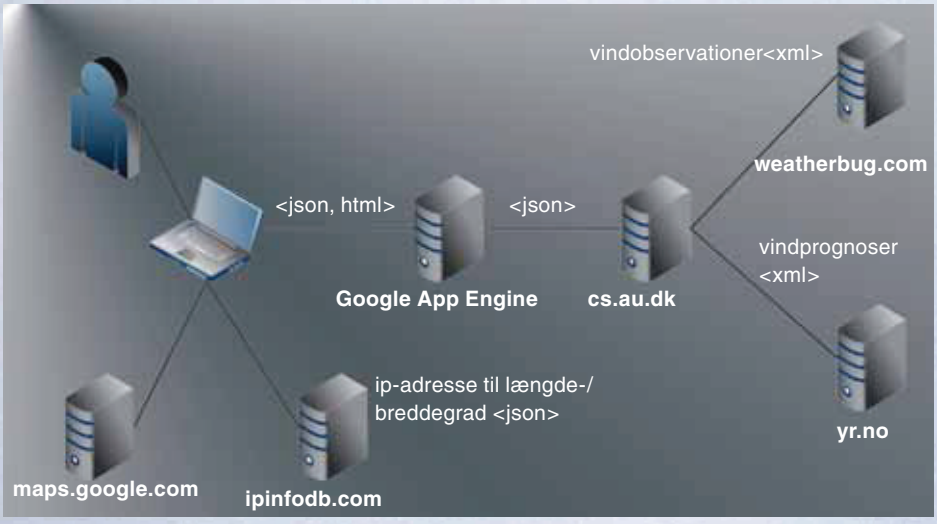
Valgte REST

Som en del af specialeprojektet skulle Jakob kontinuerligt integrere data fra adskillige eksterne resurser og stille dem til rådighed for en Ajax-klient i brugerens browser. Der var således brug for en webservice. Den arkitektoniske stil, der ligger til grund for webservicen, er REST (Representational State Transfer). REST udmærker sig blandt andet ved at kræve, at webservicen er brudt op i ressourcer, der alle, i en web-kontekst, er eksponeret ved en separat URI.

– Jeg valgte REST, som jeg opfatter som en model for, hvordan webbet burde virke. Specifikationer i REST er selvfølgelig lidt komplekse at overholde i alle detaljer, men hvis man gør det, får man sin webapplikation til at arbejde med og ikke imod den grundlæggende filosofi i webbet, siger Jakob.

Der er gennem mange år lagt lag på lag oven på http for at opnå specifik funktionalitet, men Jakob mener at kunne se en tendens nu til, at man med arkitektoniske stilarter som REST ønsker at komme tættere på selve protokollen, da det grundlæggende er til gavn for applikationerne.

Welovewind.com er efter endt speciale indtil videre et fritidsprojekt, hvor der vil blive tilføjet lidt nyt en gang imellem, når tiden tillader det. Fremtiden vil vise, om der skulle være et kommercielt potentiale i projektet, men indtil videre er Jakob godt beskæftiget med sine to store fritidsinteresser, kitesurfing og håndbold, og så naturligvis jobsøgning.



Overblik over arkitekturen af welovewind. Vejrdata holdes kontinuerligt opdateret via cronjobs, der kører uden for Google App Engine (GAE). Dette skyldes, at det tidligere ikke var muligt at køre cronjobs på GAE. Når en bruger tilgår mashuppet, konverteres der fra brugerens IP-adresse til et geografisk punkt, længde-/breddegrad ved hjælp af ipinfodb. Herefter henter browseren prognoser og surfspots i nærheden af dette punkt fra webservicen på GAE. (Grafik: Jakob Aarøe Dam)

Relevante links

welovewind.com
Lokationsbaseret webservice med et mashup af meteorologiske data for kitesurfere.
docs.google.com/present/view?id=dgm8prkk_98ftst8txz
Præsentation: Case Study: Wind Sports Mashup on Google App Engine
code.google.com/intl/da/appengine/docs/whatisgoogleappengine.html
Introduktion til Google App Engine

REST - Representational State Transfer

REST hviler på fem grundprincipper:

- Alle ressourcer har et id (URI på web)
- Alle ressourcer eksponerer det samme interface (GET, POST, PUT, DELETE,... på web)
- En ressource kan have flere repræsentationer (HTML, JSON, XML,...)
- Repræsentationer forbinder ressourcer til hinanden (links)
- Interaktion med ressourcer sker tilstandsløst. Serveren behøver ikke holde yderligere tilstand for hver klient

REST gør i de fleste tilfælde webapplikationen eller webservicen mere skalerbar, pålidelig, og hurtig. Samtidig er det nemmere og hurtigere at komme i gang med, da infrastrukturen, d.v.s. webbet, allerede er på plads, og mange eksisterende webframeworks kan benyttes til at implementere applikationen.