Institut for miljøvidenskab

Aarhus Universitet

ERHVERVSAKADEMI SJÆLLAND

PRAKTIKRAPPORT

20/07-16 – 30/09-16

Skrevet af:

Morten Toudahl

Indhold

Virksomheden

Aarhus Universitet

Aarhus Universitet er overordnet set en uddannelses- og forskningsinstitution.
I toppen af deres organisationsdiagram sidder bestyrelsen. Deres opgave er, at fastlægge universitetets strategi for udvikling og drift. Herunder strukturering af hele organisationen.

Under bestyrelsen er der universitetsledelsen. Den består af rektor, prorektor, universitetsdirektøren samt dekanerne for de fire fakulteter.
Deres opgave er universitetets daglige drift, jf. bestyrelsens beslutninger. Under dem, er der så diverse ledere og mellemledere, for de forskellige institutter, centre, afdelinger og andre faglige enheder.

- Information fra: http://www.au.dk/om/uni

Institut for Miljøvidenskab

"Institut for Miljøvidenskab driver grundlagsskabende og problemorienteret forskning inden for økologiske, kemiske og fysiske sammenhænge i miljøet, samt i økonomiske, politiske og sociale forhold i samspillet mellem miljø og samfund. Instituttet udfører forskningsbaseret myndighedsbetjening og leverer forskningsbaseret rådgivning."
- Citat: http://envs.au.dk/instituttet

Instituttet som jeg arbejdede på ligger i Risø, og er en af de få afdelinger, der ikke har undervisningspligt. Derfor var min oplevelse af at arbejde der, også hvad man kan forvente fra et almindeligt arbejde. Jeg havde i hvert fald ikke oplevelsen af, at være på en uddannelsesinstitution.

Forslag til opgaver

- *Præsentation af "online" måleresultater fra målestationer på hjemmesiden*
- *Præsentation (med grafik) af aggregerede data på hjemmesiden*
- *Oprettelse af services til udtræk af data fra luftmålinger og meteorologimålinger*
- *Dataindsamling fra arktiske målestationer, modelsammenligninger og datapræsentation*
- *Præsentation af data fra modelkørsler*
- *Udvikling af app til "Luften på din vej"*
- *Logningssystem til stationsbesøg, Windows- og Webapplikation*

- *Citat velkomst præsentation*

Ovenstående er listen over opgaver som jeg blev præsenteret for da jeg var til møde ude på Risø, inden jeg startede. Jeg valgte, at komme et par dage før praktikken rigtig gik i gang. Dem brugte jeg på at få sat systemer op, samt overveje hvilke(n) opgave(r) der gav mest mening for mig, at arbejde med. I sidste ende valgte jeg, at arbejde med servicen til udtræk af data, grundet min erfaring fra tidligere projekter.

Analyse af opgaver

Når man kigger på de forslåede opgaver, så står det klart, at de alle sammen gør brug af en database. Derfor gav det mest mening for mig, at man skulle fokusere på adgangen til den først, da denne så vil kunne genbruges af alle deres ønskede udviklingsprojekter. Og da en af projekterne skal være en app til brug af "luften på din vej" – der præsentere modelleret data omkring luftforurening – stod det klart for mig, at den service som de også ønsker burde udvikles først, da man så kan nøjes med at skrive sit data access layer en enkelt gang og derefter lade de andre projekter konsumere denne service, i stedet for f.eks. at have direkte database adgang. Da dette ville betyde at man skulle skrive den samme kode igen og igen, til hvert enkelt projekt. Og når man så kom til appen, ville man alligevel skulle lave en web service.
Ud fra beskrivelsen af opgaverne, kom jeg til den konklusion, at det bedste valg til denne service var at udvikle det som en RESTfull service, og ikke som SOAP.

Grunden til dette er, at f.eks. WS-Security, WS-AtomicTransaction og WS-ReliableMessaging ikke er nødvendige i den type projekter som Instituttet ønsker at udvikle.
WS-Security: Der bliver ikke overført sensitiv data. Alt er offentligt domæne. Derfor er det tilstrækkeligt at implementere SSL på forbindelsen samt gøre brug af tokens, i tilfælde af andet en Read operationer.
WS-AtomicTransaction: Med mindre forbindelsen til Stationen bliver afbrudt, så vil TCP give samme effekt. Og i så fald vil teknikere køre ud til stationen for at reetablere forbindelsen.
WS-ReliableMessaging: Igen som ovenstående. Alle HTTP metoderne giver en form for respons. Men hvis forbindelsen til stationen ikke eksistere er der ikke andet at gøre end at sende teknikere ud til den.
Ud over dette, så er der også en stor overhead ved at bruge SOAP i forhold til at bruge et REST API.
Og da der potentielt skal kommunikeres med API'et fra Station Nord i Grønland, er det vigtigt at holde mængden af overhead til et minimum. At sende data hjem fra Grønland er nemlig voldsomt dyrt.

Råd og Vejledning

Vi havde to møder i løbet af de første tre dage, hvor jeg – jf. ovenstående analyse – tegnede og fortalte om, hvorfor den ønskede webservice skulle prioriteres, og hvorfor de skulle lave deres planlagte arkitektur om til en serviceorienteret arkitektur, med et web API som center for alt data.

Da de andre praktikanter ikke var startet endnu, bad de også om min mening til måden, at samarbejde om opgaverne på. Grundet opgavernes små størrelser og deres afhængighed af API'et anbefalede jeg, at vi alle sad sammen, da det så ville gøre det muligt for os, at snakke sammen hurtigt omkring evt. fejl ved, eller ønsker til API'et. Dette burde lede til hurtigere udvikling, samt hurtigere reaktion ved brug for ændringer. Dette følger mentaliteten i udviklingsmetoden eXtreme Programming, og i planlægningsværktøjet Scrum.

Jeg anbefalede dog også, at vi skulle arbejde på hver vores opgave – igen pga. opgavernes små størrelser. Grunden til dette var, at hvis vi skulle sidde tre og arbejde sammen om en opgave, mente jeg, at vi ikke ville være lige så effektive, og at der ville være potentiale for, at vi ikke hver især ville få mulighed for at udfolde os, og vise vores kunnen. Dette er ikke noget jeg nødvendigvis vil anbefale i forbindelse med et almindeligt job. Men da dette er en praktikplads, syntes jeg at det var passende.

Der var enighed fra Instituttets side, og det blev derfor vedtaget, at vi skulle arbejde på denne måde.

Udviklingsmetode og miljø

Da Instituttet ikke er et software producerende firma, og der kun er to ansat til at udføre hvad jeg vil kalde ad hoc programmering, har de ikke nogen decideret udviklingsmetode. Deres ræsonnement for dette kom sig af, at de stort set altid kun har arbejdet alene på en opgave, og at det derfor ikke er nødvendigt. Af samme årsag bliver der heller ikke brugt nogen versionskontrol systemer.
Det blev dog besluttet for et par år siden, at alt skulle skiftes over til Microsoft produkter. Dvs. at alt udvikling

forgår i C#, og at der gennemgående bliver brugt Microsoft teknologier. Såsom Internet Information Services til hosting, og MSSQL databaser.
De havde også et ønske om at tilegne sig ny viden fra os studerende. Både om nye måder at arbejde på, samt nye produkter og teknologier.

Udviklingsmetode

Da jeg skulle sidde og udvikle alene på noget kode, og 'blot' samarbejde om integration med de to andre praktikanter mente jeg, at det ikke ville være nødvendigt med den helt store udviklingsmetode. Hvorfor f.eks. Unified Process ikke passede ind. Faktisk syntes jeg ikke, at nogen af de udviklingsmetoder eller planlægningsværktøjer vi er blevet undervist i kom til sin fulde ret i forbindelse med mit projekt.

Dette skal selvfølgelig ikke forstås som, at jeg så bare tog hovedet under armen og hamrede løs på tastaturet til der skete noget.

Hvis vi f.eks. kigger på Unified Process, så gælder følgende:

Remember that all UP artifacts are optional, and avoid creating them unless they add value. Focus on early programming, not early documenting.

- *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Side 32*

Unified Process' force er efter min mening grundig dokumentation, der udvikler sig som arbejdet skrider frem. Den slags dokumentation som de forskellige Unified Process artifacts udgør, var dog ikke noget jeg kunne se værdien i, for dette projekt.

eXtreme Programming følger strengt de agile principper, blandt andet kan man læse følgende i en af Robert C. Martins bøger.

Simplicity - the art of maximizing the amount of work not done - is essential.
Agile teams do not try to build the grand system in the sky. Rather, they always take the simplest path that is consistent with their goals. They don't put a lot of importance on anticipating tomorrow's problems; nor do they try to defend against all of them today. Rather, they do the simplest and highest quality work today, confident that it will be easy to change if and when tomorrow's problems arise.

- *Agile principle nr. 10,*
  Agile Principles, Patterns, and Practices in C#, side 10

Som jeg tolker dette, så er både Unified Process og eXtreme Programming enige om, at der skal bruges de værktøjer der er nødvendige til at få udført arbejdet, og ikke mere. Processer og procedurer skal ikke blot implementeres og følges, "fordi det er det, vi plejer at gøre" eller "det står der i vejledningen".

Man kan derfor sige, at jeg har lænet mig mest op af eXtreme Programming i min tilgang til programmeringen og design. Det gav dog heller ikke mening for mig at følge eXtreme Programming slavisk.

User stories: Da det var klart defineret hvad det var jeg skulle levere, dvs. et API, der kan give samme data som bliver præsenteret på den nuværende hjemmeside, så jeg ingen grund til at lave User stories.
Iteration plan, release plan and acceptance tests, planning game: Har alle User Stories som en nøgle komponent, så derfor er de også fravalgt. De ville heller ikke have tilføjet nogen værdi til projektet.
Pair programming: Ej muligt, da jeg arbejdede alene på projektet.
Test-Driven development: Jeg overvejede kraftigt, at forsøge mig med TDD., da der er mange fordele ved at bruge denne teknik. Vi har dog aldrig fået undervisning i emnet, så hvis jeg skulle forsøge mig med dette, ville jeg have

brugt for mange kræfter der, hvilket udviklingen af funktionerne og API'et som helhed ville have lidt under. Jeg genovervejede TDD flere gange under udviklingen. Men kom altid frem til samme konklusion.
Metaphor: Blev til dels brugt, når jeg skulle forklare mit projekt til andre, og f.eks. da jeg ville overtale mine chefer til at ændre i deres arkitektur.
Collective ownership: Dette blev på en måde brugt. Vi havde undervejs i processen gentagne møder med andre ansatte, der havde interesse i projekterne vi lavede, hvor jeg f.eks. forklarede struktur samt kodestumper fra mit projekt, og de to andre praktikanter gjorde tilsvarende for deres.
Whole team: Min kontaktperson ved instituttet, Keld Mortensen og hans kollega Rune Keller var altid tilgængelige til at afklare tvivlsspørgsmål, eller give detaljer om det nuværende system. Så det var en af de ting fra eXtreme Programming, som jeg gjorde brug af.
Short cycle, Continuous integration, sustainable pace: Alle vigtige komponenter, som jeg også gjorde brug af. Dog lige med den modifikation, at der som sådan ikke var nogen korte cykluser, da jeg ikke arbejdede i iterationer – men konstant producerede i et, for mig, passende tempo. Dette gjorde også, at jeg kontinuerligt opdaterede det API, som de andre praktikanter gjorde brug af i deres projekt.
Open workspace: Som nævnt under *Whole Team,* så er døren altid åben hos mine chefer, og jeg sad desuden i samme kontor som de to andre praktikanter. Så de, og jeg, var nemt tilgængelige for alle der havde brug for det.
Simple design, refactoring: Disse to går hånd i hånd, og er efter min mening, noget af det vigtigste under udvikling af software. Uagtet hvilken udviklingsmetode man vælger at bruge.

De ting jeg har beskrevet ovenfor, synes jeg generelt er den bedste måde at udvikle på, når man arbejder alene. Så længe man ikke bliver doven, og springer over at lave f.eks. diagrammer, hvis det kan hjælpe. Og så er det også altid en god ide, at tegne den overordnede arkitektur, da det så er nemmere at visualisere, om der er noget der burde gøres anderledes.

Til at planlægge mit arbejde, brugte jeg en notesblok og en blyant. Jeg syntes igen ikke, at der var nogen grund til at bruge et stort framework, som f.eks. nogen af de planlægningsmetoder der bliver beskrevet i Unified Process. Ej heller noget lettere som f.eks scrum. Min notesblok, samt de ugentlige møder med Keld og Rune var mere end rigeligt til at holde styr på, at udviklingen skred fremad, samt planlægge arbejdet og prioritere opgaverne.

Udviklingsmiljø

Da instituttet har valgt, at der skal bruges Microsoft teknologier, lå valget lige for.
Mit udviklingsmiljø vedblev med at være Visual Studio Enterprise med Resharper Ultimate. Til versionsstyring, var jeg lidt ude og kigge på de forskellige muligheder.

Ved brug af versions kontrol har man nemlig mulighed for at gå frem og tilbage i forskellige versioner af ens kode. Dette gør det nemt at komme tilbage til en virkende version. Man har ydermere nemt ved at arbejde flere personer sammen, da man kan arbejde i hver sin forgrening af koden, og til sidst bruge nogle værktøjer – f.eks. kdiff – til at flette koden sammen til et virkende produkt. Af denne årsag fandt jeg det meget vigtigt at det enten blev indført generelt, eller at jeg om ikke andet brugte det personligt.
Som følge af min vejledning taler Keld og Rune nu om at begynde at gøre det til en fast del af deres arbejde.

Team foundation server: Her er der masser af smarte features og direkte integration i Visual Studio. De tilbudte features, som f.eks. hostet scrumboard, product backlog, etc. var som nævnt ovenover ikke noget jeg mente, ville være anvendeligt for mit projekt. Og integrationen i Visual Studio er ikke mere end en bekvemlighed. Minusset ved TFS er, at vi ikke har modtaget undervisning i dette.
Subversion: Jeg har tidligere arbejdet for et firma der brugte SVN. Om firmaet ikke kunne finde ud af at sætte det op og bruge det korrekt skal jeg selvfølgelig ikke kunne sige. Men det var en utrolig dårlig oplevelse at bruge det, og det forhindrede ikke konstante problemer med at få tingene til at spille sammen.
Git: Vi har brugt Git siden slutningen af første semester. At vi har den erfaring at trække på betyder at der ikke

skal bruges tid på at lære et nyt versions kontrol system og risikoen for fejl forårsaget af forkert brug vil være meget lav. Gits integration i Visual Studio er dog heller ikke her mere end en bekvemlighed.

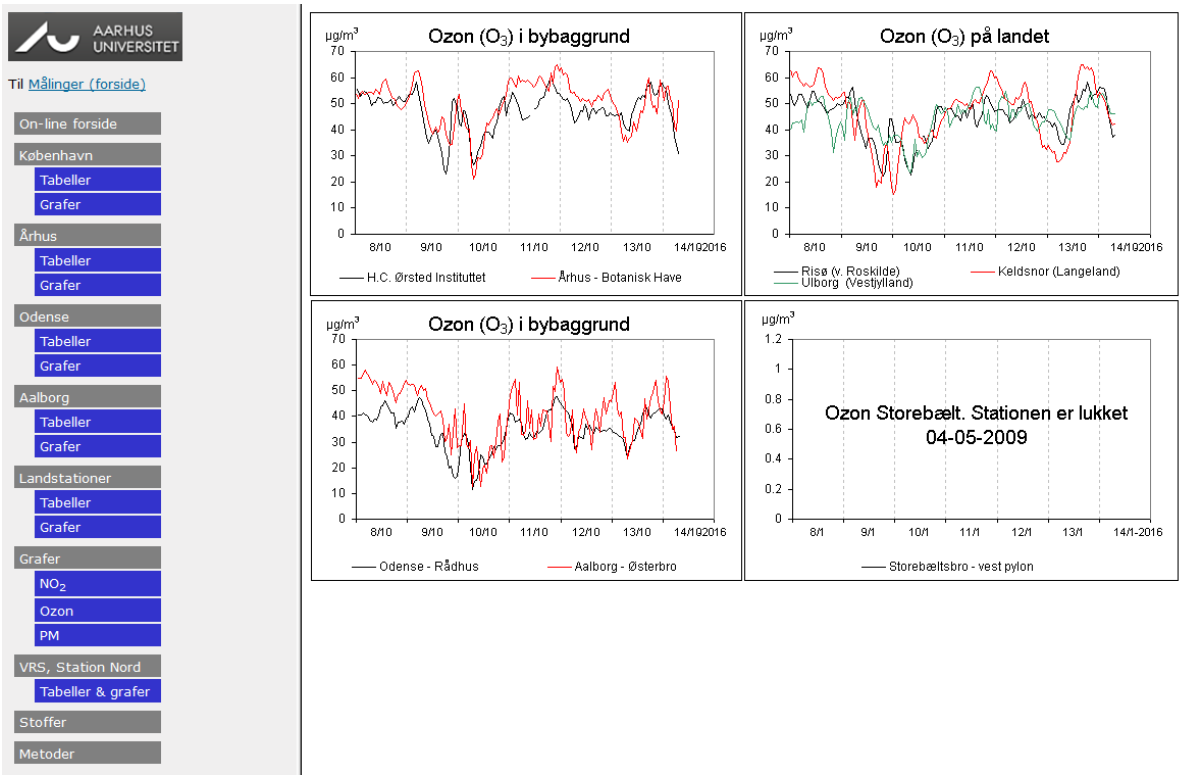
Valget faldt derfor på Git. Som host valgte jeg bitbucket. De tilbyder nemlig et ubegrænset antal private repositories ved max 5 brugere tilknyttet, hvorimod f.eks. github kræver betaling for et lignende tilbud.

Præsentation af analyse

Da jeg havde gjort mig ovenstående tanker, præsenterede jeg dem for Keld og Rune på et møde. Jeg forklarede mulighederne, og kom med min anbefaling. De valgte at følge min anbefaling, men anmodede om at få et crash course i Git. Jeg afholdte derfor en kort undervisningslektion i Git, og demonstrerede nogle af de forskellige features. Jeg viste dem også hvordan man bruger Bitbucket hjemmesiden, samt hvordan man navigerer rundt i denne og gør brug af nogle af de mere interessante features – såsom issue tracker og wiki.

Problematikken

En af instituttets opgaver er at *"Udføre den nationale overvågning af luftkvaliteten i Danmark"*. Dette gør de ved hjælp af 15 målestationer fordelt over hele landet. Disse stationer består af X antal måleapparater, der opsamler data om forskellige stoffer. Det kunne f.eks. være koncentrationen af NO^2 i luften. Målingerne bliver så samlet sammen af en lokal stationær computer. Denne PC sender så 1 gang i timen de opsamlede data tilbage til en server stående hos Instituttet via FTP.
Denne server kører derefter nogle programmer der læser de rå data, og lægger dem ind i deres database. På databasen, bliver der kørt nogle scripts som laver korrektion på de aflæste værdier, ud fra resultaterne af nogle kontrolmålinger, der bliver lavet en gang pr. måned.
De samme rå data filer bliver også brugt til at generere statiske billeder og tabeller, der bliver brugt på den hjemmeside, som instituttet stiller til rådighed for offentligheden. Hjemmesiden er lavet i pascal, og er meget svær at opdatere og redigere.
F.eks. bliver der stadig genereret kolonner i tabeller, og billeder til data, der ikke længere bliver målt, på trods af, at det i enkelte tilfælde er over 7 år siden at ændringen er sket.



Screenshot fra http://www2.dmu.dk/atmosphericenvironment/byer/forside.htm

Løsningen

Instituttet bad derfor om at få fornyet deres hjemmeside, så den ville være lettere at vedligeholde og opdatere. Jeg endte derfor med at lave et API hvis output kan styres af information i web.config. Et konfigurationsværktøj til at redigere sektionen i web.config der styrer outputtet, samt gemmer sektionen i web.config'en som hører til hjemmesiden, jeg også lavede. Hjemmesiden genererer så indhold ud fra denne sektion. F.eks. genererer den dynamiske menupunkter, med dynamisk genererede links, der sender de korrekte parametre til de korrekte actions.

Man kan i web.config oprette 'Stoffer', 'Områder' og 'Stationer'. Hver station har attributter der angiver by, område type, og hvilke stoffer der kan bedes om fra pågældende station.
Selv om en web.config teknisk set blot er en xml fil, og derfor er let læselig, mener jeg, at det vil være en kilde til fejl hvis man manuelt skal sidde og redigere i den og derefter kopiere sektionen over til en anden konfigurations fil.

Af den grund lavede jeg et konfigurationsværktøj i WPF, som kan åbne en såkaldt 'master config' – det vil sige API'ets web.config – og samtidig pege på en såkaldt 'slave config'. Denne 'slave config' en anden web.config f.eks. den, som høre til præsentations laget. Alle de ændringer man laver, bliver så gemt i begge filer. Hvilket får både hjemmesiden og API'et til at ændre adfærd.

API'et har også en controller som giver informationer direkte fra web.config'en. Dette gør at der heller ikke skal laves opdateringer her, for at holde API beskrivelsen up-to-date.
Her kan man f.eks. bede om en liste af mulige stoffer. Eller man kan få at vide, hvilke stationer der er i et givet område eller by, eller hvilke stoffer, man kan se på en given station.

Alt dette resulterer i at en ændring i deres målestationer tager få minutter at effektuere på deres hjemmeside og det offentlige API.

Produkter

API

Da jeg havde valgt at starte tidligere end de andre, havde jeg et par dage til at lægge byggestenene for API'et.
Jeg valgte at starte med forbindelsen til databasen. Her faldt valget naturligt på Entity Framework. Årsagen til dette var todelt. At arbejde med et Object Relational Mapping framework, er nemmere end at sidde og skrive SQL, for efterfølgende selv at overføre resultaterne til objekter. Og da der ikke var nogen af projekterne på deres to-do liste over projekter de vil have lavet, som er af tidssensitiv karakter, så jeg ingen grund til ikke at bruge et ORM framework.
Tidssensitivt skal forstås som, f.eks. at et system til handel med aktier er tidsensitivt, da der opstår en vis overhead ved brug af ORMs. Det er heller ikke altid, at det SQL der bliver genereret og eksekveret op mod databasen er lige så effektivt som det en dygtig SQL udvikler kan skrive.

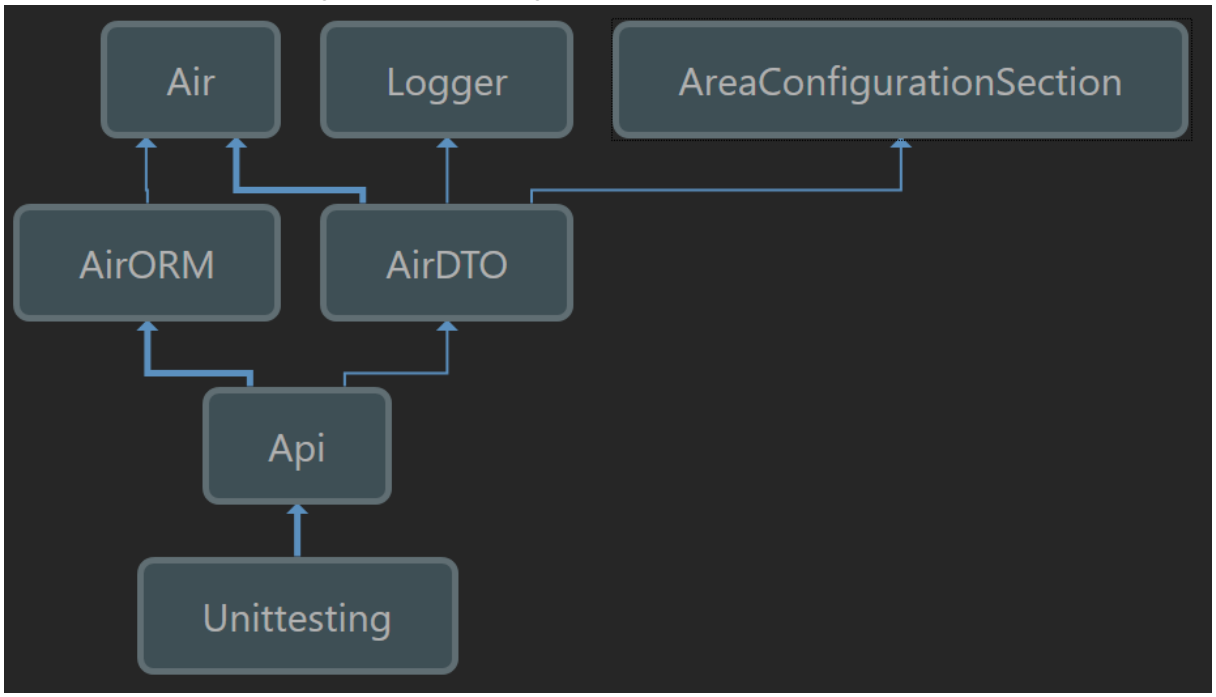
Til at starte med valgte jeg et såkaldt "database first approach". Det gjorde, at jeg hurtigt kunne oprette en controller og sætte routing op, for at trække data ud af databasen. Der var dog en masse ting som jeg ikke var tilfreds med, i den kode der blev genereret til mig af entity framework.
Af den årsag valgte jeg at slette den edmx fil som jeg havde fået, og valgte i stedet et "code first approach". Dette gjorde mig i stand til nemmere at lægge dataannotations på klasserne og disses properties, så jeg kunne få den ønskede effekt. F.eks. gjorde det mig i stand til at bruge engelske navne i klasserne, og navngive properties bedre end databasen dikterede ved en "database first approach". Disse ting er også mulige at gøre, når man bruger designeren. Men hvis jeg alligevel skulle bruge en masse tid på at konfigurere outputtet, så jeg ingen grund til at bruge en wizard først.

Derefter lavede jeg en implantation af design mønsteret Facade. Denne facade betyder at koden til at tale med databasen er gemt væk, og det derfor er meget nemmere at trække data ud fra databasen. Inde i facaden arbejder jeg med typen IQueryable<DataWarehouse>. Dette gør, at den logik der kører inde i den, ikke bliver eksekveret op mod databasen, men blot bygger en SQL query som der bliver udvidet for hvert metodekald der forgår internt i facaden. Når facaden er færdig med sit arbejde, afleverer den igen et IQueryable<DataWarehouse> objekt.
Ud over, at facaden er et design pattern i sig selv, så benytter jeg også constructor dependency injection. Dette gør jeg for at være i stand til at unit teste, og for at det senere skal være muligt at udskifte entity framework med et andet ORM hvis behovet skulle opstå, uden at man skal ændre i koden på selve API'et.

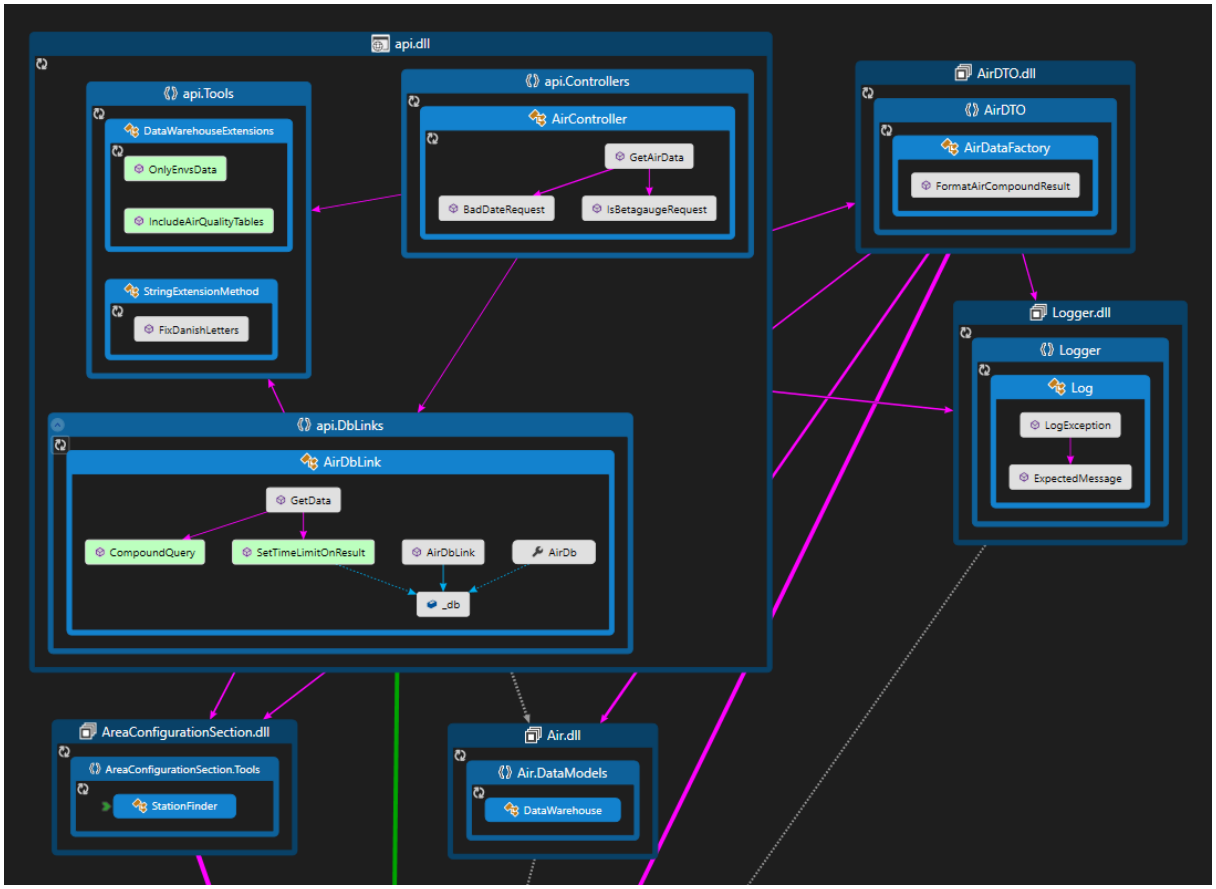
Objektet fra facaden, bliver givet videre til et factory. Dette factory iterere hen over listen og omdanner denne til ExpandoObjects. Det er først når eksekveringen når til dette punkt at den query der er blevet opbygget af facaden bliver evalueret, og at der bliver trukket data ud fra databasen.
ExpandoObject er en klasse defineret af Microsoft. Den fungerer som et dynamisk objekt efter man har lavet en instans. Den har dog også den fordel at den arver fra IDictionary<string, object>. Dette gør mig i stand til dynamisk at oprette properties, i modsætning til andre implementationer. Hvilket er nødvendigt hvis man skal undgå at skulle rette, recompile og redeploy API'et ved hver ændring af stationer etc. i web.config.

Jeg oprettede også en wrapper til Microsofts Trace klasse. Ud over at undgå gentagelser i koden, gør det også, at såfremt instituttet på et senere tidspunkt vælger at bruge et custom logging framework, så kan de blot ændre koden i metoderne i wrapperen og se effekten over alt i API'et.
Mit valg faldt på Trace klassen da den kunne det den skulle, og samtidig er konfigurerbar fra web.config. Hvilket igen betyder, at de kan ændre ting ved API'ets opførsel uden at skulle recompile.
For at gøre det muligt at konfigurere API'et uden at skulle recompile, besluttede jeg mig for at lægge oplysninger om stationer i web.config. Dette krævede, at jeg oprettede en klasse der arver fra ConfigurationSection. Denne AreaSection klasse beskriver så ved hjælp af properties den del af sektionen som indeholder informationerne omkring stoffer, områder og stationer. For at lette kommunikationen med web.config oprettede jeg igen en facade til det formål.

Alt dette er delt ud på adskillige projekter i løsningen.



Den overordnede arkitektur



Her kan man se et nærmere blik på hvordan koden afhænger af hinanden.

Til sidst implementerede jeg også throttling, for at undgå at instutittets servers bliver for overbelastede. Denne er også mulig at konfigurere via web.config. Jeg kan dog ikke tage æren for denne del, da det er en nuget pakke som jeg fandt.

API'et vil blive sat i drift så snart IT får løst nogle infrastruktur problemer.

Website

Undervejs i praktikperioden blev cheferne oprigtigt bekymret for, om de to andre praktikanter kunne nå at blive færdige med præsentationslaget. De havde nemlig planlagt, at hjemmesiden skulle skiftes ud med vores produkter.

I løbet af weekenden besluttede jeg mig for at lave noget i min fritid, for at sikre mig at de i sidste ende ville have en ny hjemmeside der levede op til deres krav.

Da jeg var fast besluttet på at det skulle være muligt for instituttet at opdatere og vedligeholde websitet også, valgte jeg at importere den dll jeg oprettede i API'et der definerede sektionen i web.config med information omkring stationer etc.
Dette gjorde at jeg ud fra informationerne der i kunne genere indholdet af menupunkter samt de nødvendige parameter til at generere tabeller og grafer dynamisk. Det vil sige at jeg oprettede to index fil og to partial views til grafer og tabeller.
Index filerne bruger så en viewmodel de får tilsendt til at generere det korrekte antal tabeller/grafer. De to partial views får så en anden viewmodel tilsendt som indeholder den information de skal bruge til at påsætte labels og bestemme antal kolonner.

Tabellerne er ganske almindelige HTML tabeller, stylet med bootstrap.

AARHUS UNIVERSITET | Forside | Tabeller | Grafer | Information | Stoffer | Metoder | API

Resultaterne er ikke kvalitetskontrollerede. Tidsstemplerne vises i tidszonen UTC+1. Klokken er nu: 09:16 (UTC+1)

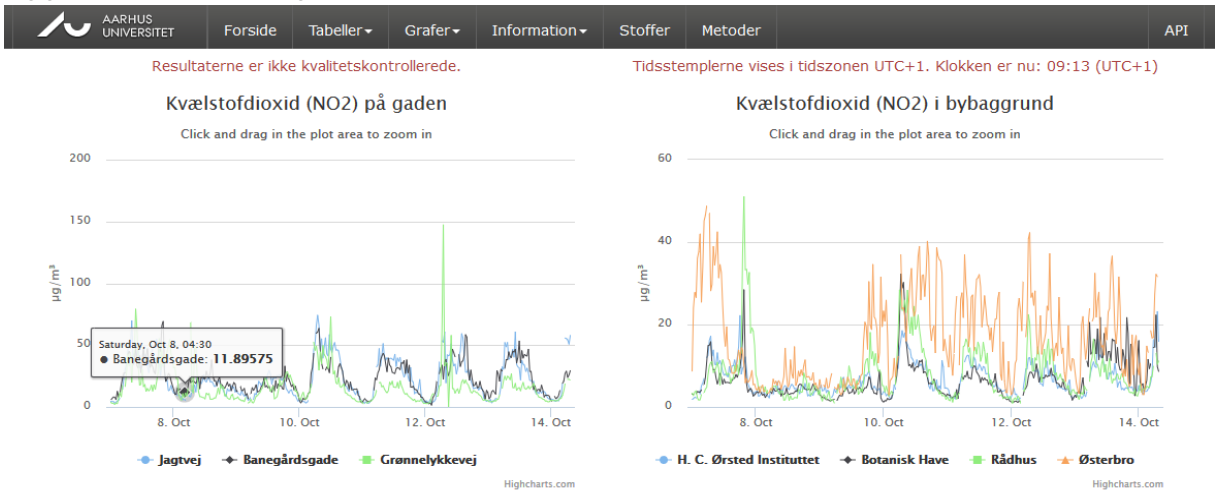
Copenhagen - H. C. Ørsted Instituttet

Measured	CO	NOx	NO2	O3
14-10-2016 07:30:00	0,17463	21,47738	15,93112	35,04449
14-10-2016 07:00:00	0,162988	30,3705	23,16037	26,74238
14-10-2016 06:30:00	0,162988	16,77263	14,4585	34,52561
14-10-2016 05:30:00	0,139704	19,98562	16,33275	35,26402
14-10-2016 05:00:00	0,139704	8,721	7,956	39,29533
14-10-2016 04:30:00	0,139704	7,057125	6,00525	43,50626
14-10-2016 04:00:00	0,139704	5,527125	4,838625	45,40218
14-10-2016 03:30:00	0,139704	4,417875	4,3605	47,03865
14-10-2016 03:00:00	0,139704	3,538125	3,805875	
14-10-2016 02:30:00	0,139704	3,805875	3,767625	48,37577
14-10-2016 02:00:00	0,139704	3,5955	3,691125	51,68863
14-10-2016 01:30:00	0,139704	2,94525	3,17475	55,93947
14-10-2016 01:00:00	0,139704	4,85775	4,8195	54,94162
14-10-2016 00:30:00	0,139704	3,691125	4,01625	56,45835
14-10-2016 00:00:00	0,139704	5,221125	5,182875	56,0193
13-10-2016 23:30:00	0,139704	3,920625	3,844125	58,11478
13-10-2016 23:00:00	0,139704	3,997125	4,2075	57,05706
13-10-2016 22:30:00	0,139704	4,131	4,2075	57,27659
13-10-2016 22:00:00	0,139704	4,3605	3,920625	56,77767
13-10-2016 21:30:00	0,139704	4,876875	4,85775	53,92382
13-10-2016 21:00:00	0,151346	4,9725	4,685625	52,78627
13-10-2016 20:30:00	0,151346	5,01075	4,9725	52,68648

Measured	PM10b	PM25b
13-10-2016 00:00:00	12,5	10
12-10-2016 00:00:00	12,7	9
11-10-2016 00:00:00	10,4	20,1
10-10-2016 00:00:00	7,2	6,5
09-10-2016 00:00:00	7,3	7,2
08-10-2016 00:00:00	5,5	5,7
07-10-2016 00:00:00	8,1	7,2
06-10-2016 00:00:00	10,8	5,9
05-10-2016 00:00:00	9,6	5,5
04-10-2016 00:00:00	5,2	2,2
03-10-2016 00:00:00	4,1	0
02-10-2016 00:00:00	9,8	5,8
01-10-2016 00:00:00	19,7	14,3
30-09-2016 00:00:00	13,7	6,7
29-09-2016 00:00:00	22,7	14,1
28-09-2016 00:00:00	33	16,3
27-09-2016 00:00:00	25,2	16,1
26-09-2016 00:00:00	24	16,1
25-09-2016 00:00:00	20,4	14,2
24-09-2016 00:00:00	15,2	8,7
23-09-2016 00:00:00	25,3	16
22-09-2016 00:00:00	18,6	11,9

Tabeller i nyt præsentations lag.

Og graferne er lavet vha. highcharts.



Grafer i nyt præsentations lag.

Udviklingstiden var på ca. 6-7 dage. Så koden er ikke så nem at vedligeholde som i API'et. Instituttet har dog indgået en aftale med mig om, at arbejde videre på hjemmesiden under mit hovedeforløb, da de planlægger at sætte den i drift så snart de har løst nogle infrastruktur problemer.

Konfigurationsværktøj

Den sidste ting jeg arbejdede på var egentlig ikke noget Instituttet havde bedt om. Men jeg følte at det var en naturlig fortsættelse af projekterne.

Værktøjet er bygget i WPF, og læser web.config direkte som en almindelig XML fil. Denne bliver så først undersøgt for fejl og mangler, der ville kunne forårsage exceptions i kodeafviklingen på API og website. Den håndhæver desuden nogle regler der skal overholdes under redigeringen af informationen i web.config.

Dette værktøj vil udelukke, at der kan ske fejl der forårsager nedbrud af services som følge af manuel redigering af konfigurationsfilen.

Refleksioner og konklusion

Jeg har opnået viden omkring den daglige drift i Instituttet gennem de forskellige møder jeg har deltaget i, desuden mener jeg også at have demonstreret de ønskede færdigheder samt kompetencer der er defineret i læringsmålene.

Dette ses i sektionerne 'Virksomheden', 'Forslag til opgaver' og 'Råd og vejledning'.
Dog har punktet "tilegene sig ny viden… " under kompetencer været svært at leve op til for mig.

Problematikken har bestået i, at instituttet som nævnt ikke er et software producerende firma, og derfor ikke på alle punkter har samme niveau som en datamatiker. Derfor har de relevante personer ikke været i stand til at lære mig noget rent programmeringsfagligt eller teknisk. Jeg har tværtimod fået at vide et par gange, at mine løsninger var for avancerede.
Jeg har dog tilegnet mig noget ny viden.

F.eks. vidste jeg ikke, hvordan man skulle oprette sin egen sektion til web.config, og jeg havde heller ikke brugt highcharts før. Og da highcharts er baseret på javascript, fik jeg også mulighed for at lære det. I hvert fald til

husbehov. Jeg skrev også en custom validator til den konfigurations sektion jeg lavede, hvilket gjorde at jeg blev opmærksom på en fejl i implementeringen af noget af Microsofts framework.
Jeg valgte også at lave konfigurationsværktøjet i WPF, da jeg ikke har noget erfaring med dette.

Da API'et skulle hostest virkede det ikke som det plejede at gøre når Instituttet skulle lægge et website online. Derfor måtte jeg også dykke ned i opsætning af Internet Information Services, for konfigurere application pools samt bruger rettigheder, så serveren også kunne køre MVC, og ikke kun webforms.

Omkring planlægning og samarbejde, syntes jeg ikke at have lært noget nyt. Jeg er blevet grundigt forberedt af skolen på at arbejde med denne type opgaver, og da jeg har haft en god del jobs inden jeg begyndte på uddannelsen har jeg massere af erfaring med at være i et professionelt miljø.

Overordnet set, synes jeg alligevel at det har været en rigtig god oplevelse at være i praktik. Det var en rigtig dejlig arbejdsplads, og jeg nød at arbejde efter princippet "frihed under ansvar".

Bibliografi

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)
Forfatter: Craig Larman
ISBN-13: 978-0131489066
ISBN-10: 0131489062

Agile Principles, Patterns, and Practices in C# 1st Edition
Forfattere: Robert C. Martin og Micha Martin
ISBN-13: 978-0131857254
ISBN-10: 0131857258

Udtalelse

AARHUS
UNIVERSITY
DEPARTMENT OF ENVIRONMENTAL SCIENCE

Recommendation

Enviromental social science

Date: 06 September 2016

We hereby confirm that Morten Toudahl has attended a company internship program at Aarhus University, Department of Environmental Science during the period 20 July 2016 to 6 September 2016. The internship is expected to be completed by October 1, 2016.

Morten has been working on the development of applications for the presentation of air measurements on our web portal. The system is based on MS SQL databases and MS Internet Information Servers.

Morten has demonstrated great knowledge and experience regarding the implementation of the used frameworks (Microsoft.Net, ASP.NET, MVC, Web API, Entity Framework), and he has during the internship managed to organize a solution using up-to-date strategies and conventions.

Morten has worked intensively on the task and he has followed all agreements. Furthermore he has taken active part in meetings and social activities.

Morten has worked both independently and as part of a developer group and he has contributed as an important partner, and shown great helpfulness in the group.

We highly recommend Morten for future development tasks.

Sincerely yours

Keld Mortensen
IT employee
System Developer

Thomas Ellermann
Senior Scientist
Head of section