Bachelorprosjekt

Web services og web grensesnitt til QS Manager EE

Forfattere: **Odd Amund Wik**
Chris Stian Melby

Dato: **19. Mai 2009**

Sammendrag

Tittel	Web services og web grensesnitt til QS Manager EE
Dato:	2009-05-19.
Forfattere:	Odd Amund Wik og Chris Stian Melby.
Veileder:	Høyskolelektor Harald Liodden, IMT, Høyskolen i Gjøvik.
Oppdragsgiver:	IT-tjenesten ved HIG i samarbeid med QS Manager AS.
Kontaktpersoner:	Jon Langseth (HIG) og Ole Martin Frydendal (QS Manager AS)
Nøkkelord:	QS Manager EE, nettjeneste, SOAP, Microsoft C# ASP.NET, MS-SQL
Antall sider:	144
Antall vedlegg:	80
Tilgjengelighet:	Begrenset
Sammendrag:	QS Manager AS med sin brede erfaring innen kvalitetssikring og dokumentasjon har spesialisert seg *i*, óg bidratt til utvikling *av*, fagområdet ARI Management (Asset/Resource/Inventory Management); dvs. forvaltning av alle typer ressurser/driftskritisk utstyr. QS Manager EE er deres erfaring satt i system. QS Manager® bærer de administrative konsekvensene av den posisjon teknologien har inntatt i virksomheten. Prosjektet er gjennomført på oppdrag av IT-tjenesten i samarbeid med QS Manager AS. Det er demonstrert et behov for en nettløsning av Windows-løsningen QS Manager EE. Respektive utvikling skal danne et grunnlag for videreutvikling av nettløsningen; dette ved spesifisering og detaljering av tjeneste- og presentasjonslag for avgrenset funksjonalitet. QS Manager EE systemet benytter en kompleks database, over blant annet objekter og respektive aktiviteter, som nettløsningen og nettjenester skal anvende. Løsningen av prosjektoppgaven har bestått i å lage nettjenester for operasjoner mot databasen og nettside som brukergrensesnitt for nettjenestene. Systemet er utviklet i et Microsoft Visual Studio 2008 C# ASP.NET miljø.

Forord

Prosjektoppgaven ble utformet og presentert ved Høyskolen i Gjøvik, november 2008. Utviklingsoppgaven går mot en nettløsning av QS Manager EE, med It-tjenesten i samarbeid med QS Manager AS som oppdragsgiver.

Denne oppgaven appellerte til oss pga. oppgavens dybde og bredde, samt. føringen om Microsoft ASP.NET plattform pga. dens tyngde og utbredelse blant bedrifter. Ingen av de nåværende deltakerne i prosjektgruppa har hatt noe særlig erfaring med dette, så dette ble sett som en spennende faglig utfordring og et "must" å ha kunnskap om, med tanke på hvor ofte kunnskap om C#/ASP.NET/MS-SQL etterspørres. Dessuten er ARI-managment et stort område under stadig utvikling, så vi anser ikke denne oppgaven som noen form for "plankeoppgave".
I januar ble det holdt hyppige møter med veileder med utskiftinger av gruppedeltakere. Og blant annet et møte med oppdragsgiver der vi fikk presentert oppgaven mer i detalj og i samråd avgrenset oppgaven.

Gjennom prosjektet har vi hatt møte med veileder og høyskolelektor Harald Liodden annenhver uke. Liodden har vært til stede for oss så ofte vi har trengt, og har fulgt opp for å sikre fremgang gjennom hele prosjektet. Dette er noe vi har satt stor pris på, og vil spesielt takke han for interessen og de positive tilbakemeldingene han har gitt oss gjennom de siste månedene.

Vi har hatt regelmessige møter med oppdragsgiver, både i form av it-tjenesten og QS Manager AS, hvor vi har dratt nytte av god teknisk veiledning, gode innspill og stort engasjement i arbeidet. Blant annet nevnes oppsett av QS Manager EE, innføring i databasen og koding av en klasse for dekoding av lisensnøkkel. For dette rettes det stor takk til representantene Jon Langseth (HiG), Ole Martin Frydendal (systemutvikler, QSM) og Tor Sætrang (daglig leder, QSM).
Vi ønsker forøvrig å rette takk til Stian Husemoen for teknisk konsultasjon vedrørende localhosts kjøring av asp-sider.

Gjøvik, 2009-05-19.

.....	.....
Odd Amund Wik	Chris Stian Melby

Innhold

Figurer

1 Innledning

1.1 Oppdragsgivere

Denne bacheloroppgaven har It-tjenesten ved Høyskolen på Gjøvik (heretter HiG) på vegene av QS Manager AS som oppdragsgiver.

QS Manager AS med hovedkontor på Hamar har utviklet windowsapplikasjonen som skal modelleres som nettløsning i prosjektet.
Den opprinnelige ideen var å drifte å effektivisere driften av et dataanlegg, dette har utviklet seg videre med tett samarbeid ut mot kunden for å kunne lage et mer effektivt program. QS Manager EE legger veldig tung vekt på dokumentasjon og har en oversikt over alle objekter som er lagt til.
QS Manager AS jobber med kvalitetssikring og dokumentasjon. De tilbyr et program som hjelper til å holde oversikt over ditt firma sine objekter og objektets relasjoner og aktiviteter.
QS Manager har vært med på å utvikle ARI Management(Asset/Resource/Inventory Management). ARI Management er stort og meget omfattende, det bygger på følgende elementer [2]:

- Identifisere:
  Kartlegge ressursene bedriften er avhengig av for å opprettholde driften. Her må det også identifiseres hvilke rutiner og planer som må iverksettes.

- Registrere:
  Benytter seg av de identifiserte ressursene og beskriver dem så godt som mulig. Slike spesifikasjoner er blitt viktig siden et av de største effektiviseringspotensialet ligger i å finne raskt og enkelt den informasjonen en leter etter når en trenger dem.

- Synliggjøre sammenhenger og avhengighetsforhold:
  I dagens samfunn er det vanskelig å fortelle hva som er avhengig av hva eller hvem. Det å beholde denne oversikten gjør det mye enklere å styre virksomheten på en god og sikker måte.

- Vedlikeholde:
  Opprettholde den informasjonen som finnes. Bruk enkle verktøy, enkle definisjoner og fordeling av ansvar. Benytte felles verktøy for hele bedriften.

IT-tjenesten bruker per dags dato QS Manager EE og har komplett oversikt over så godt som alle ressurser/objekter under høgskolen.
Det er IT-tjenesten som har initiert oppdraget og har også forbeholdt seg retten til å videreutvikle produktet.

1.2 Problemområdet

Det har vist seg i dag at det er lønnsomt å holde en oversikt over alt av objekter som finnes i en bedrift. Disse objektene kan variere fra veggkontakt til for eksempel en gravemaskin. Alle objekter kan ha en tilhørighet til et annet objekt, som enten bruker eller blir brukt av det andre objektet. Det kan være en dataskjerm som er koblet på en datamaskin.
Det registreres også aktiviteter som skal utføres. Det vil da være en oppgave som skal utføres, som igjen involverer forskjellige objekter.

All den informasjonen som ligger lagret i databasen er kun tilgjengelig via en windowsapplikasjon, noe som gjør at du må være tilkoblet det lokale nettverket til databasen for å få tilgang til den. Oppgaven vil gjøre det mulig å koble til den databasen via internett.

Dette kan by på en rekke andre problemer. I selve tilkoblingen er det viktig at sikkerheten ivaretas slik at en inntrenger/lytter ikke har mulighet til å få tak i verdifull informasjon som utveksles. Kravet til pålogging vil være meget sentral og oppgaven tar ikke høyde for å lage nye brukere. Dermed skal vi kun utvikle en påloggingsfunksjon der sikkerheten blir sentral.
QS Manager EE fungerer i dag slik at det er en begrensning på antall brukere som kan være pålogget samtidig, avhengig av hvilken pakke du abonnerer på. Denne begrensningen skal også fungere på lik linje som windowsapplikasjonen og brukeren skal ikke få tilgang om det ikke er noen tilgjengelige brukerlisenser ledige på det tidspunktet.

Systemet må være robust og fungere på en enkel og oversiktlig måte som gjør brukeren fornøyd med løsningen vi har utviklet. Det må også legges fokus på responstid fra databasen. Slik windowsapplikasjonen er kan den bli noe treg i forbindelse med større databaser. Dermed må vi fokusere på dette problemet for at løsningen vår tar høyde for responstid.

Et annet viktig punkt vil være selve webløsningen og brukergrensesnittet. Etter samtale med oppdragsgivere ble vi enige om at webløsningen vil ha en del fellestrekk som windowsapplikasjonen, men med en fornyet utseende som gjør den litt mer opp til dagens dato når det gjelder design og utvikling.

1.3 Målgruppe

Målgruppen for dette prosjektet vil være hovedsaklig oppdragsgivere QS Manager AS og IT-tjenesten ved Høyskolen i Gjøvik, da prosjektet er avgrenset slik at systemet ikke er klar til utgivelse. Ansatte ved IT-tjenesten vil kunne anvende implementerte enkeltelementer i systemet; som en initiativtaker til dette prosjektet. Rapporten dokumenterer systemet overordnet og enkelte tekniske detaljer til bruk for videre utvikling. Ved videreutvikling vil prosjektet inkludere eksisterende brukere av QS Manager.
I tillegg kommer vanlig målgruppe som inkluderer studenter, sensor, veileder, og andre ansatte ved HiG. Rapporten legges til grunn for evaluering og karaktersetting, og kan være til nytte for senere studentprosjekter inkludert videreutvikling av dette systemet.

1.4 Formål

Formålet med oppgaven vil være å lage en webløsning til windowsapplikasjonen QS Manager EE. Det har lenge vært et ønske fra HiG og QS Manager å ha muligheten til å aksessere databasen uten om å være koblet på internettverket der databasen er plassert. Med dette prosjektet skal denne løsningen utvikles og gi brukeren internettilgang via nettjenester og nettsiden i systemet.

Gjennom prosjektet skal vi også lære oss å jobbe som en gruppe og prosjektet er en større tverrfaglig oppgave som er meget relevant i jobbsammenheng.

1.5 Egen bakgrunn

Vi har begge jobbet med C++, java-script og SQL databaser i forbindelse med emner ved HiG. Vi har også hatt faget www teknologi som også vil være noe sentralt i forbindelse med at det skal lages en egen webside.
I oppgaven anvendes C# ASP.NET, som fortoner seg som en blanding av programmeringsspråket C++ og java-script. Tatt i betraktning det nye språket så ble det mye å sette seg inn i starten av hovedoppgaven, men progresjonen tiltok etter hvert som arbeidet gikk sin gang.
Hele prosjektet ble utviklet i Visual Studio 9 som også er et program som ikke er så mye brukt i tidligere sammenhenger. Dette er et oversiktlig og greit program som presenterer filene og systemet godt, som gjorde det forholdsvis greit å ta i bruk.

Systemet som skal utvikles skal også bruke SOAP meldinger som er grunnlaget for overføring av XML dokumenter. Det var et ønske fra oppdragsgiver at pros-

jektet skal utvikles med denne protokollen. Ingen i gruppen har brukt den tidligere så det ble en læringsprosess før den ble tatt i bruk.

Det å jobbe med et større prosjekt som dette, og utvikle noe for en reell oppdragsgiver, er ikke noe som noen av oss har vært med på tidligere. Med det så har vi også måtte forholde oss til oppdragsgiverne på en god måte, som også har vært en god læringsprosess gjennom hovedoppgaven.

1.6 Rammer og begrensninger

- Prosjektutviklingen skal skje i tidsrommet fram til 25. Mai, 2009.
- Nettapplikasjonen skal fungere både i Windows- og Linux-baserte nettlesere.
- Nettjenestene skal kjøre mot en MS SQL 2005 database.
-

1.7 Øvrige roller

Vi har også hatt On-site customer som har vært Magne Reinsborg. Han kar kommet med ønsker om grunnfunksjonalitet.
Stian Husemoen ble også et hjelpemiddel i forbindelse med forsøksmessig oppsett av localhost som kjører asp-sider.

1.8 Arbeidsformer

Gruppen har ikke hatt et fast grupperom gjennom prosjektet. Dette begrunnet i at "lagring av utstyr" ikke er grunn nok til grupperom. Dessuten forutsetter utviklingen ingen "stasjonære" aktiviteter. Prosjektet vil bli utviklet på våre egne bærbare datamaskiner, noe som gjør oss meget mobile og adaptive mht. arbeidform og -sted.
Meste av jobbingen har foregått på skolen der vi har møttes på et ledig grupperom. Begge har hatt fag ved siden av bacheloroppgaven så arbeidstiden har måtte tilpasse seg litt på grunn av det.

Generelt har dette gått ganske greit da vi har refordelt oppgaver og ansvarsområder ved behov. I forprosjektet forutsatte vi å bruke parprogrammering som et element i utviklingen, men etter hvert ble differens i kompetansenivå mht. selve programmeringen indikert og arbeidsfordelingen ble gradvis refordelt. Som konsekvens

avsluttes parprogrammeringen; en tar ansvaret for programmering, mens den andre tar ansvaret for utvikling av systemdokumenter, samt. testing under validering av inkrement.

1.9 Dokumentstandard og rapportorganisering

Dokumentets standardformatering mht. skrifttyper som nevnt i forprosjektrapporten **??** ble irrelevant som resultat av en revurdering av verktøy for dokumentering. Det var ønskelig med versjonsstyring også på rapporten for å forenkle arbeid mot det samme dokumentet; for å oppnå dette ble det anvendt TeXMaker i kombinasjon med SVN.

1.9.1 Innledning

Denne seksjonen gir en kort innledning til rapporten.

1.9.2 Kravspesifikasjon

Denne seksjonen inneholder en generell kravspesifikasjon som gjelder for hele prosjektet.

1.9.3 Design

Denne seksjonen inneholder en oversikt på hvordan systemet skal oppføre seg og fungere som en helhet.

1.9.4 Testing

Denne seksjonen beskriver testresultater og hvordan vi tolket de under testing av systemet som en helhet.

1.9.5 Implementering

Denne seksjonen vil beskrive hvordan innføringen av systemet fungerte som en helhet.

1.9.6 Iterasjon 1

Iterasjon 1 gikk ut på å lage en innlogging der brukeren logger seg på databasen. Denne delen viser hvordan vi gikk frem for å få ordnet dette på en god måte. Den

vil også beskrive det visuelle oppsettet og hvordan selve påloggingen foregår.
Vi tar også for oss det å hente ut et objekt fra databasen.

1.9.7 Iterasjon 2

Iterasjon 2 gikk ut på at kunden skulle kunne hente ut et objekt og i tillegg få mulighet til å se på det objektet sine relasjoner og aktiviteter. Det skal også gis mulighet til å søke etter aktiviteter etter gitte søkekriterier.

1.9.8 Iterasjon 3

Iterasjon 3 var den siste fase i API utviklingen. Denne fasen skulle vi bruke et uthentet objekt til å kunne foreta endringer på det og legge det tilbake i databasen. Det ble også mulighet til å lage helt nye objekter og lagre de i databasen.

1.9.9 Iterasjon 4

I denne delen går vi nærmere inn på hvordan oppsettet på webløsningen er og hvordan det ble seende ut.

1.9.10 Iterasjon 5

I denne delen beskriver vi oppsett av en https localhost server og viser indikert resultat.

1.9.11 Avslutning

Beskriver hvordan vi ser på resultatet av oppgaven og hva som kunne gjøres annerledes.
Det er også gjort en oppsummering og konklusjon av oppgaven som beskrives her.

1.9.12 Vedlegg

Her finnes alle vedleggene i forbindelse med hovedoppgaven.

2 Kravspesifikasjon

2.1 Introduksjon

Denne seksjonen beskriver hvordan kravene til systemet er. Kravspesifikasjonen ble i hovedsak utviklet i første møtet med oppdragsgivere. Siden det er to oppdragsgivere i forbindelse med oppgaven måtte det et møte til og løpende dialog for å få satt opp realistiske mål, praktiske mål og rammer for prosjektet. Kravspesifikasjonen ble mer tilspisset via møter og samtaler med både QS Manager og IT-tjenesten ved HiG i starten av prosjektet.

2.1.1 Krav til systemet

Slik situasjonen er nå har QS Manager allerede en windowsapplikasjon som kan brukes til å hente de dataene som brukeren trenger. Systemet vi skal lage vil gjøre det mulig å få tak i de dataene på databasen via internett på en trygg og effektiv måte. Dermed kan brukere av databasen jobbe videre uten å må være koblet til det lokale nettverket databasen er satt opp i. Får å få til det må en bruker ha mulighet til å logge seg på systemet via internett. Når den tilkoblingen er satt opp skal brukeren kunne hente informasjon om de objektene som er aktuelle.
Systemet må være robust, stabilt og tilfredsstille de vanligste handlingene som windowsapplikasjonen støtter.

2.1.2 Systemets brukere

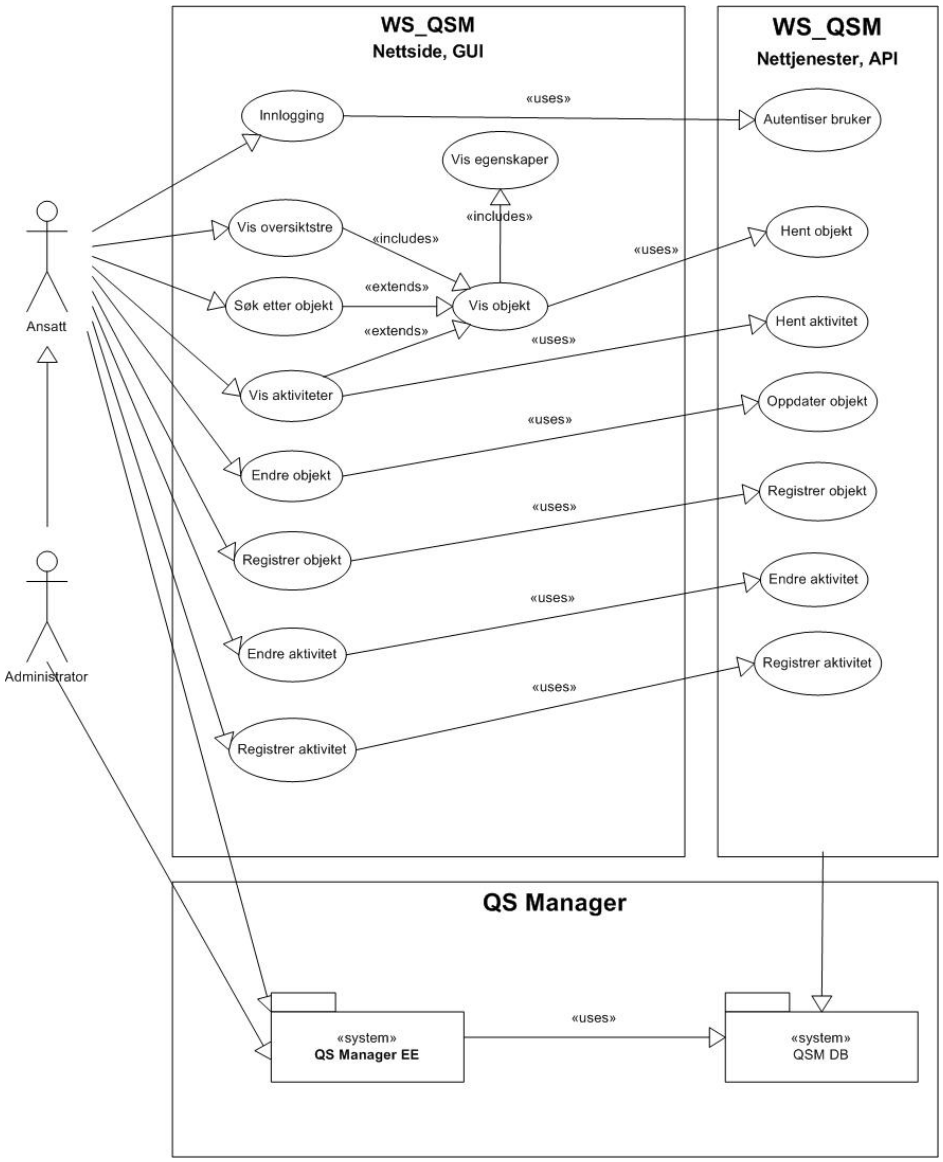
I første rekke vil det kun være brukere relatert til ansatte i IT-tjenesten, HiG og QS Manager AS. Disse kan koble til via webløsningen. Denne brukeren vil he en "vanlig" brukerrolle som evt. har tilgang til å lese og endre på dataen som er lagret på databasen. Det er ingen ekstra administratorfunksjonalitet innebygd, bortsett fra at admin. Som har tilgang til alle objekter.
Ved evt. fullførelse av systemet vil en ha brukere i et utall forskjellige bedrifter fra "gulvet og opp".

2.1.3 Systemets omgivelser

Høgskolen i Gjøvik vil antagelig sette opp en egen virtuell windowsserver for det uferdige (* etter avgrensinger) systemet til eget bruk. Når systemet er ferdigutviklet kan QS Manager implementere systemet i en installasjonspakke til kundene for å settes opp på en windowsserver med MS SQL 2005 eller bedre.
Databasen til QS Manager EE må være satt opp på serveren der systemet befinner seg.

2.2 Overordnede funksjonelle krav

2.2.1 Use case diagram



Figur 1: Use case diagram

2.2.2 Innlogging

Aktør: Brukeren.
Brukeren må logge seg inn med brukernavn og passord, som defineres enten ved klientens windowsbruker eller manuell innskriving i respektive felt, for å få tilgang til systemets brukergrensesnitt. Dette skal gjøres på en egen innloggingsside. Når brukeren ikke er innlogget skal systemet redirigere til innloggingssiden ved adressering av systemets overordnede brukergrensesnitt. Hvis brukeren adresserte eller har vært innlogget på en spesifikk side, vil brukeren redirigeres tilbake til den ved suksessiv innlogging.
Suksessiv innlogging forutsetter at brukernavn og passord er korrekt, brukeren finnes i databasen, databasen er i riktig versjon, lisensnøkkelen i databasen er gyldig, lisensen ikke er utløpt og samtidige brukere i databasen ikke overskrider en gitt verdi.

2.2.3 Søk objekt

Aktør: Brukeren.
Forutsetter at brukeren er innlogget for å utføre denne handlingen.
Brukeren ønsker å hente ut et aktuelt objekt som ligger i databasen. Brukeren får opp en søkelinje og kan velge et søkekriterie. Søket går gjennom databasen og returnerer de aktuelle objektene som oppfyller søkekriteriets krav.

2.2.4 Vis objekt

Aktør: Brukeren.
Forutsetter at brukeren er innlogget.
Brukeren ønsker å vise informasjon om et aktuelt objekt. Etter at et søk er gjennomført vil brukeren ha mulighet til å klikke på det objektet, og objektet vises til skjerm.
Et objekt kan også hentes ved å bruke trestrukturen. Da går man gjennom listen og kan hente ut et objekt ved å klikke på det.

2.2.5 Endre objekt

Aktør: Brukeren.
Forutsetter at brukeren er innlogget og at det er hentet ut et objekt for å utføre denne handlingen.
Brukeren ønsker å foreta endringer på et aktuelt objekt. Objektet blir oppdatert med de aktuelle dataene som brukeren ønsker, og legger det oppdaterte objektet

tilbake i databasen.

2.2.6 Registrer objekt

Aktør: Brukeren.
Forutsetter at brukeren er innlogget for å utføre denne handlingen.
Brukeren ønsker å registrere et nytt objekt til systemet. Brukeren får beskjed om å taste inn objektets data og objektet blir laget og oppdatert med de aktuelle dataene og relasjonene som medfører.

2.2.7 Vis egenskap

Aktør: Brukeren.
Forutsetter at brukeren er innlogget og at det er hentet ut et objekt for å utføre denne handlingen.
Brukeren ønsker å vise egenskapene til et objekt. Når et objekt er hentet er det mulig å gå inn på det objektet for å se på dets egenskaper.

2.2.8 Vis aktivitet

Aktør: Brukeren.
Forutsetter at brukeren er innlogget for å utføre denne handlingen.
Brukeren ønsker å hente ut en aktuell aktivitet. Det blir opprettet en søkemulighet og brukeren søker etter en aktivitet, de aktuelle aktivitetene som oppfyller søkekriteriet vises til brukeren.

2.2.9 Endre aktivitet

Aktør: Brukeren.
Forutsetter at brukeren er innlogget og at det er hentet ut en aktivitet for å utføre denne handlingen.
Brukeren ønsker å oppdatere en aktivitet. Den aktiviteten som er hentet ut vises til brukeren og gir mulighet til å oppdatere de detaljene om aktiviteten.

2.2.10 Registrere aktivitet

Aktør: Brukeren.
Forutsetter at brukeren er innlogget for å utføre denne handlingen.
Brukeren ønsker å registrere en ny aktivitet. Brukeren taster inn den nødvendige informasjonen for en aktivitet og lagrer den på databasen.

2.3 Detaljerte Funksjonelle krav

De detaljerte funksjonelle kravene finnes i vedlegg **??** som utvidet use cases.

2.4 Andre krav

2.4.1 Responstid

Selve windowsapplikasjonen har vist seg at den kan være treg når den brukes på større databaser. Dette er et problem når det brukes større databaser som blant annet HiG sin. Den er blant de største som bruker QS Manager EE til kvalitetssikring. Dermed vil det være fokus på responstid for å få et system som fungerer godt, selv på store databaser.

2.4.2 Sikkerhet

Det skal følgelig logges inn via et nettverk slik at det er viktig at påloggingen foregår på en sikker måte. Spesielt hvis kommunikasjonen foregår over internett eller åpne lokal-nett. For slik kommunikasjon forutsettes det at løsningen befinner seg på en https/SSL server slik at kommunikasjonen foregår kryptert.
Forøvrig må brukeren for å autentiseres ha tilgang til databasen og være definert som bruker i databasen.

2.5 Begrensninger

2.5.1 Generelt

Utviklingen avgrenses primært til nettjenesteplattformen, et API mot databasen. Og sekundert et nettbasert brukergrensesnitt som anvender et utvalg av primærfunksjonaliteten fra windowsapplikasjonen.

Den primære funksjonaliteten i brukergrensesnittet utgjøres i første rekke (i prioritert rekkefølge) av tilgang til I. Objekter og II. Aktiviteter med følgende funksjonalitet a) Oppslag, b) Endring og c) Registrering i databasen med fokus på respons.

Romertall I og II med funksjonalitet bokstav a og b fremstår som et minimum av funksjonalitet i webbrukergrensesnittet. I tillegg forutsettes en innloggingsfunksjonalitet med brukernavn og passord relatert til antall samtidige brukere i databasen iht. lisensnøkkel og dennes utløpsdato. Funksjonaliteten i API skal reflekteres i funksjonaliteten til brukergrensesnittet.

2.6 Iterasjoner

2.6.1 Iterasjon 1

Denne iterasjonen er definert for utvikling av innloggings og objekthentingsfunksjonalitet i nettjenestelaget. Dette inkrementet inkluderer også utvikling av codebehind for en enkel html-implementering for testing.
Påloggingen skal skje på en sikker måte, forespørslene som går over internett skal være krypterte, på den måten vil ikke brukernavn og passord bli sendt i klartekst der noen kan fange opp pakkene, og lese de som de er.

QS Manager EE produktet har også et gitt antall pålogget brukere som kan være pålogget på samtidig. I databasen ligger lisensnøkkelen lagret, og i den igjen ligger informasjonen om hvor mange som kan få tilgang samtidig. Systemet vårt må ta høyde for at den øvre grensen ikke overstiges.

Etter samtaler med oppdragsgivere ble vi også enige om at det kan være greit å ha en "'husk meg"' knapp som kan benyttes ved innlogging. Denne knappen vil da huske den siste brukeren som har logget seg på. Den vil være meget praktisk å bruke på private datamaskiner som kobler seg til med dette systemet ofte, men bør brukes litt med forsiktighet med tanke på sikkerhet.

2.6.2 Iterasjon 2

De fleste objekter har en relasjon til et annet objekt, og noen har aktiviteter knyttet opp mot seg. Dette er en viktig del av QS Manager sin tankegang. Ved å knytte disse objektene sammen gir det en mer helhetsoversikt og gjør det enkelt å identifisere hvert enkelt objekt.
Dermed er det viktig at disse relasjonene og aktivitetene blir ivaretatt.

I forbindelse med uthenting av et objekt vil det alltid vise om det objektet har andre objekter relatert til seg.

Det skal også være mulig å søke etter aktiviteter. Det vil være meget behjelpelig for de som utfører selve aktiviteten. På den måten kan de koble seg på å lete opp

de aktuelle aktivitetene som vedkommende har på sin dagsorden.

2.6.3 Iterasjon 3

Som vi har vært gjennom tidligere, skal en bruker ha mulighet til å hente objekter fra databasen. Dette objektet skal også kunne endres på. Dette er for at brukere av QS Manager skal kunne holde en oppdatert database.

Dermed er det viktig at objektet blir lagret på riktig form, etter at det er endret/lagt til egenskaper. Objektets egenskaper blir lagret i en tekststring i databasen. I denne tekststringen ligger egenskapene fortløpende og skilles med tegnet þ. Dermed vil det også bli viktig at det blir lagret på den måten. Et eksempel på dette er:
H.toalettþþþþþþþA002þþþþþþþ .
Dette eksemplet viser: At det er et herretoalett med en objektID A002. Der det kommer þ etter hverandre vil det si at de feltene i tabellen er tomme. Disse feltene kan også fylles ut, men vil ikke inneholde noen informasjon av viktighet for akkurat det objektet.

```
OBJEKTTYPE SCRPLASSERING DBPLASSERING FELTNAVN FORMATKODE
WINDOWCONTROL LISTINNHOLD SUMFORMAT
Rom;1;1;Romnr./navn;@s30;Entry;;NULL
Rom;3;8;Plassering;@s20;Link;~K:O00~G:Lokasjoner~;NULL
Rom;4;3;Veggkontakt;@s20;Link;~K:U01~T:Veggkontakt~;NULL
Rom;7;4;Arbeidsstasjon;@s20;Link;~K:U110~T:Arbeidsstasjon~;NULL
Rom;8;10;Laptop;@s20;Link;~K:U110~T:Laptop~;NULL
Rom;9;5;Skjerm;@s20;Link;~K:U110~T:Monitor~;NULL
Rom;10;9;Skriver;@s20;Link;~K:U10~T:Printer~;NULL
Rom;11;12;Projektor;;Link;~K:U10~T:Lysbildeframviser~;NULL
Rom;12;6;Telefon;@s20;Link;~K:U11~T:Telefon~;NULL
Rom;13;2;Bruker;@s20;Link;~K:U11~T:Person internt~;NULL
Rom;14;11;Docking station;@s20;Link;~K:U11~T:Docking station~;NULL
Rom;15;13;Programvare/RIS;;Link;~K:U10~T:RIS~;NULL
Rom;16;7;Programvare;;Link;~K:U11~T:Programmer~;NULL
Rom;17;14;OL;;Link;~K:O11~;NULL
Rom;18;15;UL;;Link;~K:U11~;NULL
```

DB plassering viser posisjonen i tilleggsinformasjonen, som er relatert til scrplasering som viser rekkefølgen det blir presentert for brukeren. Listinnhold K viser det er en kobling mot et annet objekt. Dette er mer beskrevet i møtereferat **??**.

Det skal også kunne registreres nye objekter. Ved opprettelse av et nytt objekt er det viktig at relasjoner blir knyttet riktig opp mot andre objekter.
Ved feil på noen av disse kan det ha fatale følger i ettertid.

2.6.4 Iterasjon 4

Websiden skal være enkel og grei å bruke. Det vil bli viktig å ha tydelige kjennetegn mellom den og windowsapplikasjonen.

Etter samtaler med oppdragsgivere skal utseende være mer moderne, men samtidig ha flere kjennetegn som gjør det enkelt for brukeren å ta i bruk webløsnignen.

Hovedfokuset på oppgaven vil ligge på utviklingen av API'et, og utviklingen av webløsningnen vil andre prioritet.

2.6.5 Iterasjon 5

Det eneste kravet er at serveren krever HTTPs kryptering for å gi tilgang til websiden som prøver å koble seg på.

3 Analyse og Design

3.1 Introduksjon

Dette er en generell innføring i hvordan systemet vil fungere, og hvordan det ble oppnådd.
Mer i detalj kommer i beskrivelsen for hver iterasjon.

3.2 Overordnet arkitektur

Det første som skjer når webapplikasjonen tas i bruk er at brukeren må logge seg på med brukernavn og passord. Dermed blir brukeren logget inn og får tilgang til databasen. Brukeren kan på den måten gå gjennom databasen og hente ut informasjon om de objektene som brukeren selv ønsker. Det vil også være mulighet til å foreta endringer og oppdatere informasjonen om et objekt.
I forbindelse med objektene er det også relaterte aktiviteter som skal utføres. Disse aktivitetene er knyttet mot objektet i seg selv eller brukeren kan søke opp det på gitte søkekriterier.
For oversikten sin del er det også laget en trestruktur der brukeren kan bla seg gjennom og klikke på de objektene som er aktuelle.

3.2.1 Database

Databasen er allerede der i forbindelse med windowsapplikasjonen. Dermed vil bruken bli den samme som med windowsapplikasjonen, der behandling av objekter vil bli gjort på en lignende måte. Det blir da også mulig å bruke noen av de samme spørringene som er laget.
Siden databasen allerede er en ferdig oppsatt måtte vi sette oss skikkelig inn i hvordan databasestrukturen til QS Manager var organisert. Dermed ble det et møte med QS Manager på Hamar den 11. februar der vi fikk en detaljert gjennomgang. Det som ble det sentrale her var hvordan navngivingen på tabellene er laget, og hvilke tabeller som inneholder den informasjonen som vi er ute etter. I vedlegget **??** finnes mer detaljert hva vi gikk gjennom på det møtet.
Veldig forenklet var det bare å finne en måte å koble seg på databasen, og ta den i bruk.

3.2.2 Utseende

Selve utseende vil være i likhet med windowsapplikasjonen, men med en litt mer moderne og fornyet utseende. Etter samtale med QS Manager var det også noen

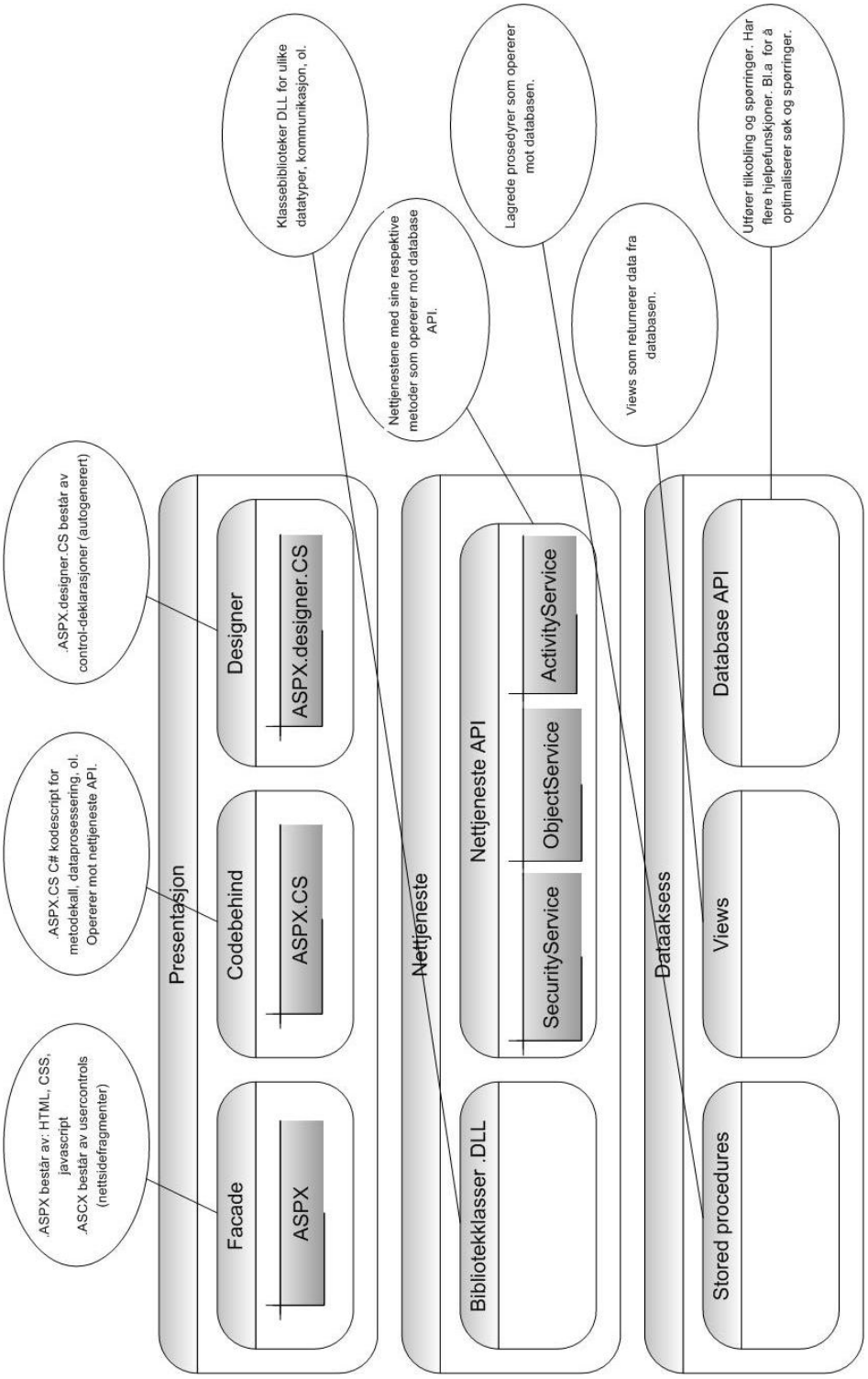
av kundene deres som også hadde påpekt at windowsapplikasjonen så litt gammel ut, og et mer moderne design ville falle i smak.
Ved å gjøre webløsningen og windowsapplikasjonen forholdsvis like vil det bli mye enklere å ta i bruk. Brukerne vil finne det de leter etter på samme måte som de har brukt før. Siden de har store likhetstrekk vil ikke kreve noen form for opplæring for brukerne å ta i bruk systemet.
Windowsapplikasjonen ble et sentralt hjelpemiddel som ble flittig brukt når det kom til selve utseende til webløsningen. Gjennom store deler av prosjektet ble det hele tiden sjekket hvordan windowsapplikasjonen hadde satt opp og organisert sin data. På den måten gjør det enklere for brukeren å kjenne igjen hvor de forskjellige knappene og tilgangene finnes.

Når det kom til en mer modernisering av programmet var vi inspirert av noe mer kurver på boksene og et mer moderne utseende. Dette kommer vi nærmere tilbake i beskrivelsen hver iterasjon.

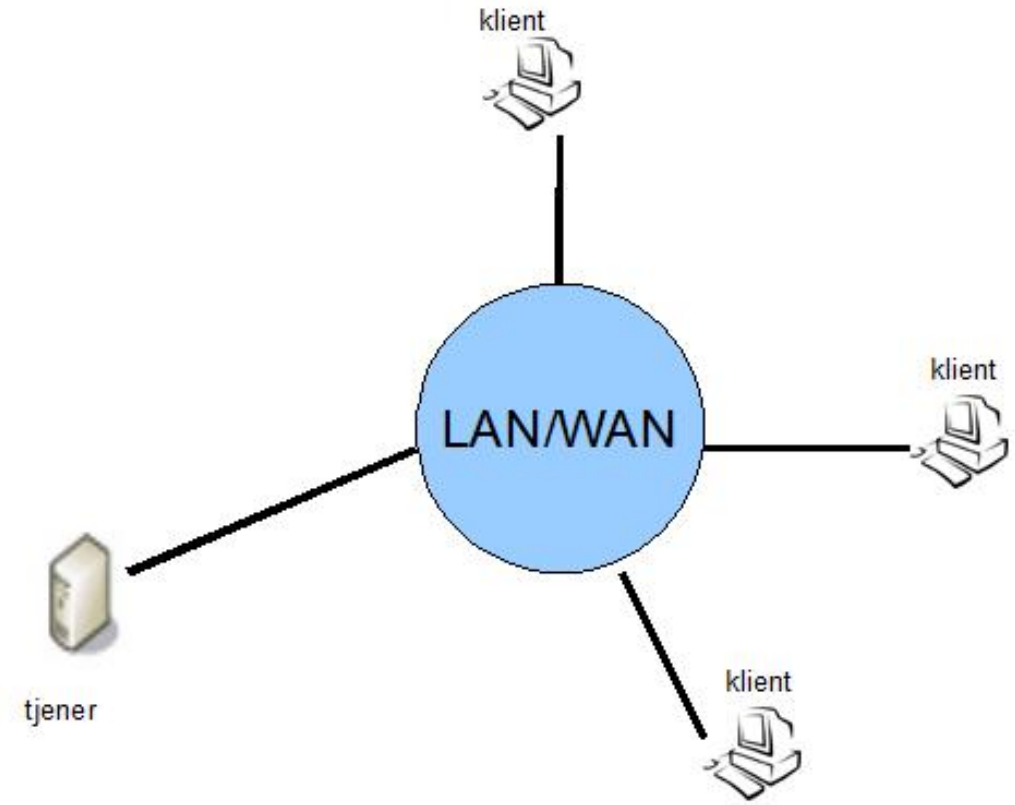
3.3 Logisk oversikt

I QSM Web har vi valgt en enkel trelagsstruktur som gjengir resulterende systemlag. Dette med bakgrunn i kravspesifikasjonen som indirekte gir slik struktur. Noe som er hensiktmessig i forhold til de tre hoveddelene nettsiden, nettjenestene og databasen.

Nettverksinndelingen er som figuren nedenfor. Serversiden, Nettjenestelaget og dataaksesslaget befinner seg alle på serveren. Klienten etterspør serversiden og serveren poster den til klienten.



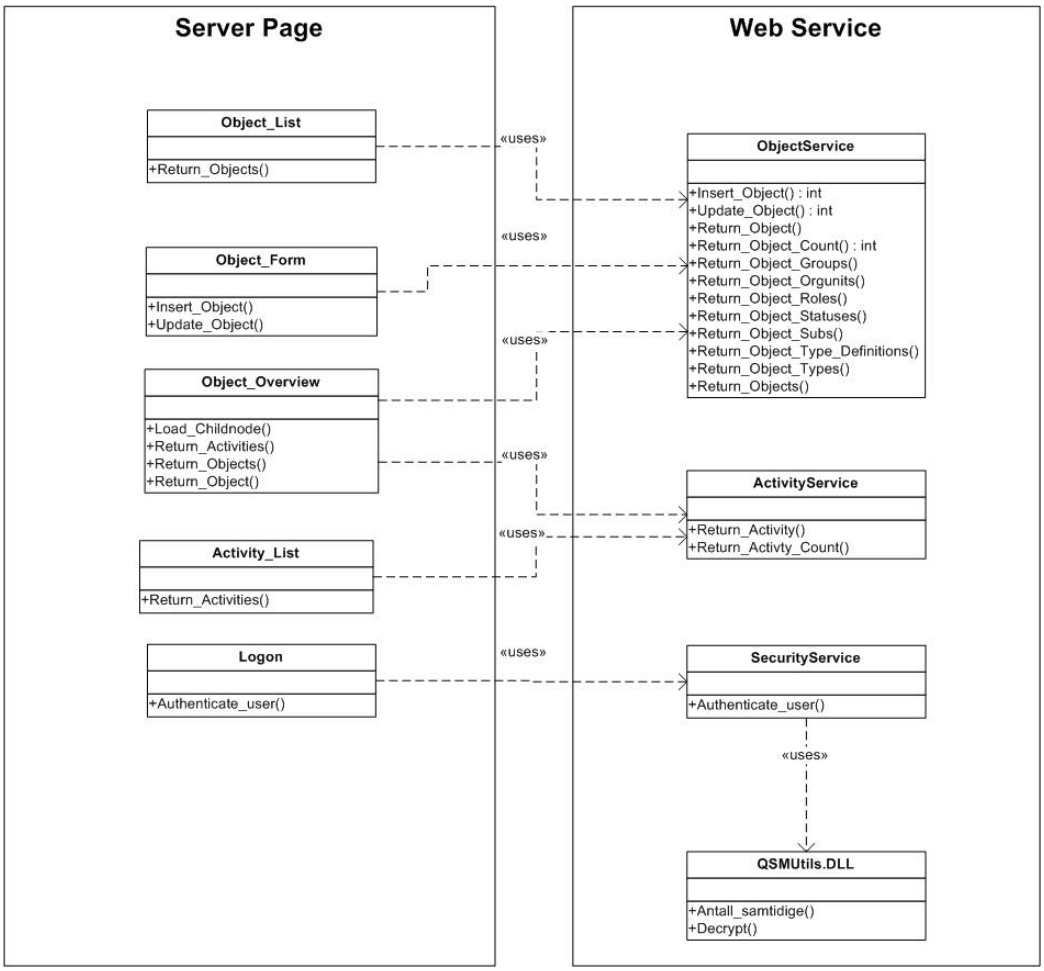
Figur 2: Logisk laginndeling



Figur 3: Nettverksinndeling.

3.3.1 Teknologi

Arkitekturmessig vil utviklingsmiljøet (C# ASP.NET) gjøre at systemet deles i ulike filtyper, som diskutert i følgende avsnitt. Presentasjonslagets klassers bruk av enkelte nettjenester er demonstrert ved et enkelt klassediagram:



Figur 4: Klassediagram

3.3.2 Presentasjon

Dette laget inneholder all funksjonalitet som er relatert til visning av nettsiden.

Fasade: Fasadepakken inneholder selve presentasjonsstrukturen (HTML) og kan for så vidt også inneholde javascript, CSS og C# script, men C# scriptene holdes i egne ASPX.CS filer for bedre oversikt. ASCX filene inneholder brukerkontrollere

(nettsidefragmenter), for eksempel en innloggingskontroller som kan knyttes til alle ASPX sider.

CodeBehind: I Codebehind (ASPX.CS) gjøres all metodekall til nettjenestene og prosessering av data som blir returnert. Prosesseringen innebærer til siste slutt og dynamisk genererer nye elementer i ASPX siden. En innlogget bruker har nødvendigvis ikke tilgang til alle objekter i databasen; dette blir håndtert i codebehind ved at returnerte objekter ikke vises dersom brukeren ikke har tilgang.

Designer: ASPX.designer.CS er autogenerert og utgjør sammenlinkingen mellom ASPX og ASPX.CS filene. Klassefunksjonene i .CS filene linkes opp mot objektene i .ASPX filene, som refererer til respektive funksjoner.

3.3.3 Nettjeneste

I nettjenestelaget ligger all funksjonalitet som tar hånd om kommunikasjonen mellom presentasjonslagets codebehind og databasen.

Klassebibliotek DLL (Dynamic Link Library): Dette er ulike kodebibliotek som blant annet brukes til å deklarere datatyper, kommunikasjonsklasser, ol. For eksempel er QSMUtils.DLL et klassebibliotek laget av QS Manager AS som tilbyr funksjonalitet for dekryptering og validering av lisenskoden.

Nettjeneste API: Denne inneholder alle klasser og metoder som presentasjonslaget benytter seg av. Blant metodene er blant annet Return_Object(string object_id) som returnerer et objekt med en gitt object_id.

3.3.4 Dataaksess

I dataaksesslaget ligger all funksjonalitet som tar hånd om SQL views, spørringer, utføring av "stored procedures", mm.

Stored Procedures: Her ligger all funksjonalitet i form av funksjoner. Stored procedures er subrutiner som ligger i databasesystemet og kan benyttes til å utføre et kall. Prosedyren utfører kallet med hensyn til evt. argumenter, utfører evt. instruksjoner (update, insert, etc), og returnerer data.

Views: Her ligger alle egendefinerte views for å returnere ofte utvalgt data.

Database API: Her håndteres selve tilkoblingen(e), spørringene, og egne hjelpefunksjoner.

4 Testing

4.1 Introduksjon

Testingen av systemet beskrives i dette kapitlet med referanse til utviklingsmetoden speilet i gantt-skjemaet. Disse testene er utviklet med grunnlag i utvidete use-case.
Formålet med testene er å sjekke at det fungerer, pålitelighet, mulighet for rette opp evt. feil, brukervennlighet og for å godkjenne den aktuelle iterasjonen.

Tester som er blitt utført er utført spesielt med tanke på:

- Fungerer, at programmet gjør som det skal.
- Robusthet, tollerer feiltastinger.
- Dokumentasjon, viser at systemet fungerer.

Uttesting av løsningen er veldig viktig med hensyn til generell og spesiell anvendbarhet. Med generell anvendelighet menes et gjenkjennelig system som intuitivt kan brukes uten treghet. Den spesielle anvendbarheten gjenspeiler ulik funksjonalitet i løsningen.

4.2 Testmiljø

I forbindelse med testing har vi satt opp et testmiljø der vi tester funksjonaliteten til systemet. Det er da viktig at vi kjører en database med relevante data og har løsningen installert på testmaskinene. Vi har brukt egne maskiner på testingen hvor vi tilfredsstiller kravene for testing.

4.3 Gradering av feil

- **Høy:** Retting iverksettes umiddelbart, og feilsøking pågår til problemet er korrigert.
- **Middels:** Bør undersøkes og rettes, om det viser seg at feilene påvirker systemet i en eller annen form.
- **Lav:** Mindre feil, ser etter løsninger kun der det er nødvendig.

4.4 Utførte tester

Det er utført en rekke tester gjennom prosjektet som er knyttet opp mot hver iterasjonsom vi har utviklet. De mer spesifikke testene for hver iterasjon finnes i avsnitt 5.4.

4.4.1 Komponent- og inspeksjonstesting

Utviklingsprosessen er inndelt i iterasjoner på ulike systemkomponenter som utvikles og testes individuelt, før de testes og evt. valideres som en helhetlig del av systemet. Lavere lag ble utviklet først, slik at det helhetlige systemet alltid var kompilerbar. I kritiske deler som påloggingslagene er det anvendt inspeksjonstesting, der kodeutvikleren setter det andre gruppemedlemmet inn hva koden skal gjøre, slik at denne kan vurdere og teste koden.

4.4.2 Whiteboxtesting

I påloggingslagene som er utgjør en kritisk del av systemet er det utført spesielt grundige tester. Dette (whitebox-testing) innbefatter testing mot og med kjennskap til kodelogikk i lagene. Som f.eks vurdering og uttesting av lagenes robusthet ved forsøk på SQL-injeksjon, feilinntasting og manuell overstyring av variabler.

4.4.3 Blackboxtesting

Det ble utført Blackbox-testing i den forstand at enkelte inkrementvalideringer ble utført ved testing mot og av QS Manager EE. Med andre ord sammenliknes uthentet/presentert data i vår nettløsning med den eksisterende windows-løsningen, hvilket sin kildekode er ukjent.

4.4.4 Stresstesting

Av kravspesifikasjonen er det spesifisert at systemet skal ha en rimelig ytelse selv når det opererer mot store databaser. For å kunne avgjøre om kravene er møtt, ble det testet mot høyskolen på Gjøviks relativt store database. TreeView-testingen ble utført uformelt i den forstand at navigeringen ble brukt og vurdert med hensyn på forsinkelse mellom museklikk og visuell respons.

4.4.5 Akseptanse- og systemtest

Systemet ved prosjektslutt ble testet med hensyn elementer nevnt i seksjon 4.4.2.

5 Implementering

5.1 Generelt

Det vil være enkle kodeeksempler for å beskrive tankegangen når det ble programmert, og hvordan kommenteringen er gjennomført i selve kildekoden. Det skal forklares generelt om hendelsesforløpet når en prosedyre blir kalt. Det kan være når brukeren utfører et søk vil det søket kalle på en gitt funksjon som henter ut objektene som oppfyller det kravet.
Dermed vil det bli enklere å videreutvikle systemet, noe som er ønskelig fra HiG sin side.
Kommenteringsspråket i programmet vil være engelsk, som er en grei standard å bruke med tanke på videreutvikling i større sammenhenger.
Det er også laget en overordnet struktur som viser hvordan systemet fungerer som en helhet. Denne strukturen finner du mer om i figur 2.

5.2 Programvare

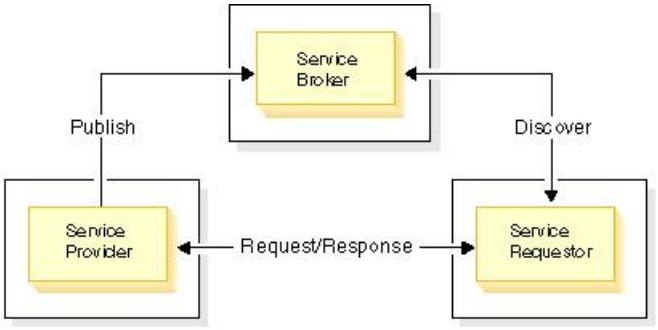
Microsoft Visual Studio 2008/9.0 er utviklingsrammeverket for prosjektet; som gitt i kravspesifikasjonen med bruk av ASP.NET teknologi. Vi valgte forøvrig C# som kildekode. Dette er et stort og omfattende rammeverk som kan brukes i oppsett av hele systemet. Både API'et og selve webløsningen ble utviklet i det programmet. På grunn av rammeverket omfang og kompleksitet krevde det en del tid lære seg godt nok. Jevnt over gikk dette ganske greit selv om vi støtte på noen utfordringer gjennom prosjektgangen.
En annen utfordring var webløsningen. Ingen av medlemmene hadde noen spesiell bakgrunn innen webløsninger så dette ble også en læringsprosess for å få satt opp den på en god måte. Dette vil vi komme nærmere inn på i inkrementet webløsning.

Microsoft SQL Server.2005 Managment Studio utgjorde hovedplattformen ved utforsking av databasen, og ved initiell testing av spørringene.
En annen utfordring var webløsningen. Ingen av medlemmene hadde noen spesiell bakgrunn innen webløsninger; denne utfordringen resulterte i en prosess som det ble lært mye av. Dette vil vi komme nærmere inn på i inkrementet webløsning.
Vi har også brukt windowsapplikasjonen, QS Manager Enterprise Edition, for å teste hvordan det eksisterende systemet fungerer. Vi har fått lisenser til QS Manager EE med tilgang til en database opp til 5000 objekter, og gyldig tilgang ut juni for å kunne bruke det i prosjektperioden.
Vi har også brukt windowsapplikasjonen, QS Manager Enterprise Edition, for å teste hvordan det eksisterende systemet fungerer. Vi har fått lisenser til QS Manager EE med tilgang til en database opp til 5000 objekter og gyldig tilgang ut juni

for å kunne bruke det i prosjektperioden.

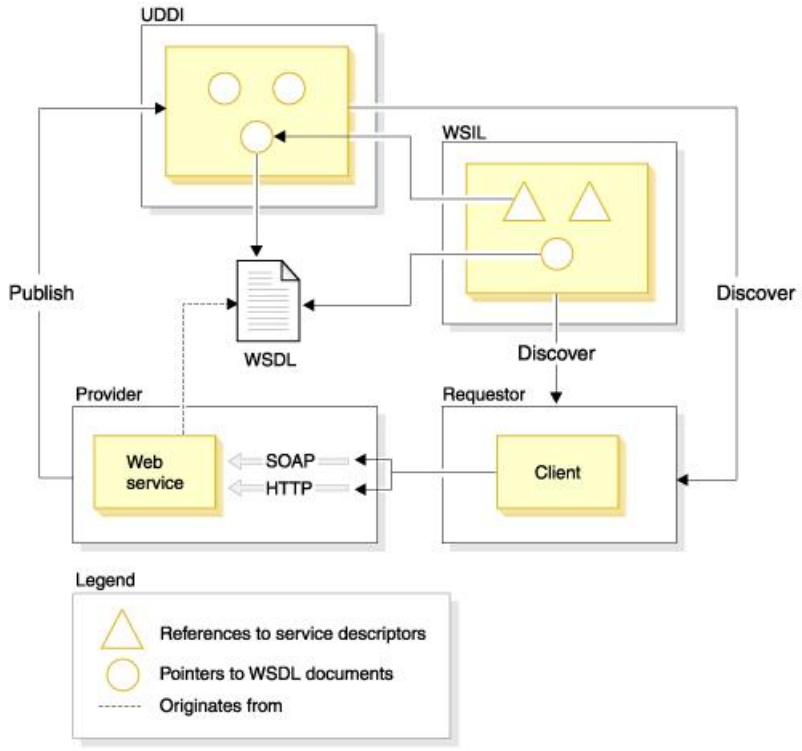
5.3 Arkitektur

Det som er utviklet er et API som henter informasjon fra en database, og som gjør det mulig for en bruker å koble seg på via internett. For å få til dette brukes det web-services for å kommunisere mellom systemet og databasen. Web-services fungerer som en protokoll, og brukes til å overføre XML-baserte meldinger. Der kommer SOAP inn som er grunnlaget for web-servicet på grunn av at det bruker XML som standard.



Figur 5: Serviceroller og interaksjon.

Figuren ovenfor viser sammengengen mellom serveren som yter tjenesten og klienten som etterspør den. Det er også en tjenestemegler som brukes av tjenesteyteren til å publisere nettjenesten, og av tjenestesøkeren å finne den. [3]



Figur 6: Sammenheng mellom SOAP, UDDI, WSIL and WSDL.

En tjenesteyter er vert for nettjenesten og gjør den tiljengelig med bruk av protokoller som SOAP/HTTP eller SOAP/JMS. Web Servicet er beskrevet av et WSDL-dokument som er lagret på tjenesteyterens server eller i et spesiellt område. WDSL dokumentet kan refereres av "UDDI business" registeret og WSIL dokumenter. Disse inneholder pekere til nettjenestens WSDL-filer. [3] Siden sikkerhet er en viktig del av prosjektet, vil trafikken gå over HTTPS ved ekstern bruk.

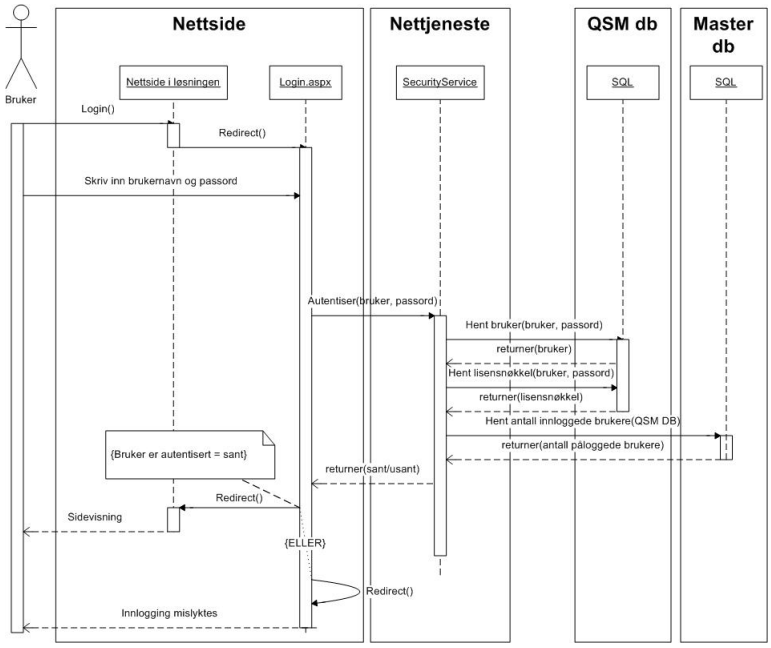
5.4 Iterasjoner

5.4.1 Iterasjon 1

Denne iterasjonen er definert for utvikling av innloggingsfunksjonalitet og metode for returnering av objekt i nettjenestelaget. Dette inkrementet inkluderer også utvikling av codebehind for en enkel html-implementering for testing.

5.4.1.1 Sekvensdiagram for innlogging

Systemet krever innlogging for tilgang til brukergrensesnittet. Når bruker går til en av nettsidene inkludert i systemet, eller session er utløpt vil brukeren automatisk blir redirigert til Login.aspx-siden for innlogging. Brukerautentiseringen skjer ved et nettjenestekall til SecurityServices metode Authenticate_User() som sjekker om brukeren eksisterer i databasen, og om lisensnøkkelen er gyldig.



Figur 7: Sekvensdiagram for innlogging.

Innloggingssekvensen er som detaljert i figuren ovenfor.

5.4.1.2 Design

Figur 8: Innlogging.

Som vist ovenfor sees selve innloggingen og feilmeldingen etter et mislykket innloggingsforsøk. Her er både "'husk meg"' og "'overstyr pålogget bruker"'.
Husk meg knappen gjør det mulig for kunden å slippe å taste inn brukernavn og passord for hver gang systemet tas i bruk. Noe som forenkler innloggingen, men som utgjør en sikkerhetsrisiko ved urutinert bruk av mer "offentlige" tilgangspunkt.
Når "overstyr pålogget bruker" ikke er valgt, vil systemet forøke og logge seg på med kredentialene tilhørende klientens påloggede bruker; windows-autentisering. Ved å hake av for overstyring vil brukeren kunne skrive inn brukernavn og passord og logge på med SQL-pålogging.
For at brukeren skal få tilgang til systemet så sjekkes det at brukeren har tilgang til databasen, finnes som bruker i "QSMDB". I tillegg valideres databaseversjonen og lisenskoden mht. formatering samt. utløpsdato, i tillegg til at antall brukere ikke er overskredet på innloggingstidspunktet.

5.4.1.3 Testing

Den viktigste formen for testing i denne fasen har vært at brukernavn og passord er korrekte.
Begge tastene "'husk meg"' og "'overstyr pålogget bruker"' ble kjørt noen runder gjennom testsystemet for å se at de fungerte tilstrekkelig.
QS Manager EE fungerer også med et gitt antall brukere som kan være pålogget samtidig. Dette ble testet med å sette antallet til å være 1 og testet med å få tilgang både med windowsapplikasjonen og webløsningen samtidig.
Feilmeldingen "The user is not associated with a trusted SQL Server connection." oppstod under utvikling av inkrementet. Dette ved SQL-pålogging så en mulig

årsak var at databasen ikke var konfigurert for både windows- og sql-pålogging. Men etter å ha sjekket både med SQL Managment Studio og regedit, er databasen korrekt oppsatt. Etter en reboot av Windows, fungerer systemet som normalt. Systeminkrementet ble validert ved modul- og systemtest.

Testcase 1

Testid/nr 1	Testnavn Innlogging		
Dato 2009-02-27	Tidspunkt 12:00	Testtype system, anvendlighets- og akseptansetest	
Estimert tidsforbruk 60 minutter		Reellt tidsforbruk 120 minutter	
Formål med testen Sjekke at innloggingen fungerer som den skal.			
Systemets tilstand Systemet i gang med tilkobling til databasen.			
Beskrivelse av testen 1. Logg inn med inkorrekt burkernavn og/eller passord. 2. Test av «husk meg» og «overstyr bruker» funksjoner. 3. Test av antall påloggede brukere. 4. Test av webbrowsere. Godkjenningskriterier: • Bruker logges ikke inn, og mottar feilmelding mht inkorrekt brukernavn eller passord. • Knappene fungerer. • Antall påloggede brukere ikke overstiger antall tilatte brukere ihht lisensen fra QS Manager. • Innlogging fungere som normalt i de mest brukte nettleserne(firefox, explorer og opera).			
Beskrivelse av feil Ved bruk av explorer og opera nettlesere fungerer ikke windowspålogging og må overstyre bruker for å skrive inn brukernavn og passord manuelt.		Feilgradering lav	
Tidsfrist 1. mars 2009			
Beskrivelse feilretting Rettet opp slik at det fungerer.			
Notater Tester suksessfull, rettet feilen som ble oppdaget. Innlogginge er sikker og brukervenlig.			
Test utført av Odd Amund Wik og Chris Stian Melby			

Figur 9: Test case for innlogging

For nærmere informasjon om bakgrunnen for denne testen om innlogging se utvidet use case i vedlegg **??**.

Mot slutten av inkrementet har vi hentet ut et objekt fra databasen som er klart for testing.

Testcase 2

Testid/nr 2	Testnavn Søk objekt		
Dato 2009-03-06	Tidspunkt 11:00	Testtype system, anvendlighets- og akseptansetest	
Estimert tidsforbruk 20 minutter		Reellt tidsforbruk 60 minutter	
Formål med testen Søke opp et objekt som ligger i databasen.			
Systemets tilstand Systemet i gang og logget på databasen.			
Beskrivelse av testen 1. Søke opp et objekt fra databasen. Godkjenningskriterier: • Objekt søkt opp og hentet ut fra databasen.			
Beskrivelse av feil		Feilgradering lav | middels| høy	
Tidsfrist 1. mars 2009			
Beskrivelse feilretting			
Notater Test suksessfull, objekt oppsøkt og hentet ut.			
Test utført av Odd Amund Wik og Chris Stian Melby			

Figur 10: Test case for søk objekt

For nærmere informasjon om bakgrunnen for denne testen, se utvidet use case i vedlegg **??**.

Etter første inkrement var vi ca en uke på etterskudd som følge av at det var mye nytt å sette seg inn i. Dermed ble det oppført to tester i forbindelse med hent ob-

jekt.
Den første testen var ikke gjennomførbar siden vi ikke var kommet så langt i henhold til planleggingen. Dermed jobbet vi videre med oppgaven og fikk fullført testen ca en uke etter planlagt.
Dette var noe vi var klar over på forhånd kunne skje. Det var mye nytt å sette seg inn i, og det krevdes en god del jobbing og konsentrasjon for å komme over den første kneika. Men så fort vi kom over den gikk det noe lettere og vi fikk hentet oss inn igjen i løpet av inkrement 2.

5.4.1.4 SOAP meldinger

Kommunikasjonen går som nevnt over http(s) protokollen med bruk av POST og GET av SOAP XML-meldinger som representerer serialiserte objekter. Følgende er et eksempel på en POST til en nettjeneste "Return_Objects(string conditions)" via SOAP1.2:

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <Return_Objects xmlns="http://microsoft.com/webservices/">
      <conditions>brukernavn='Chris Stian Melby'</conditions>
    </Return_Objects>
  </soap12:Body>
</soap12:Envelope>
```

Figur 11: SOAP 1.2, POST eksempelmelding.

Dette metodekallet sender en Soap-forespørsel via. http(s)-protokollen til nettjenestemetoden ReturnObject og søker å finne objekter med et brukernavn iht. input fra brukergrensesnitt. Et brukernavn er ikke unikt mot for eks. objektid, så denne forespørselen kan gi flere rader i retur.
Og returmeldingen via SOAP1.2 er som følger forutsatt HTTP/1.1 200 OK:

```
<?xml version="1.0" encoding="utf-8"?>
  <soap12:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
    <soap12:Body>
      <Return_ObjectsResponse
       xmlns="http://microsoft.com/webservices/">
        <Return_ObjectsResult>
              <xsd:schema id="NewDataSet"
               xmlns="http://microsoft.com/webservices/"
               xmlns:xs="http://www.w3.org/2001/XMLSchema"
               xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
            <xsd:element name="NewDataSet"
             msdata:IsDataSet="true"
             msdata:UseCurrentLocale="true">
            <xsd:complexType>
              <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="Table">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="objektnavn"
                       type="xsd:string" minOccurs="0" />
                      <xsd:element name="underliggende"
                       type="xsd:int" minOccurs="0" />
                      <xsd:element name="objektid"
                       type="xsd:string" minOccurs="0" />
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:choice>
            </xsd:complexType>
          </xsd:element>
        </xsd:schema>
        <NewDataSet xmlns="http://microsoft.com/webservices/">
          <Table diffgr:id="Table1" msdata:rowOrder="0">
            <objektnavn>Chris Stian Melby</objektnavn>
            <underliggende>0</underliggende>
            <objektid>csm</objektid>
          </Table>
        </NewDataSet>
      </Return_ObjectsResult>
    </Return_ObjectsResponse>
  </soap12:Body>
</soap12:Envelope>
```

Figur 12: SOAP 1.2, GET eksempelmelding.

I resultatsettet er det returnert treff på objektnavnet i forespørselen. Videre har dette objektet med objektid “csm”, ingen underliggende objekter. De underliggende objektene brukes ved opprettelse av nye noder i trestrukturen “TreeView”.

5.4.1.5 SQL spørringer

Nettjenestelaget opererer mot dataaksesslaget med databasen. Uthenting av data fra databasen er gjennomført med a) egendefinerte, b) databasens views, eller c) stored procedures - spørringer. Med egendefinerte spørringer menes spørringer utover en enkel spørring f.eks mot et eksisterende view eller en tabell. Spørringen som resulterer i datasettet (figur 5.4.1.4), henter data fra et view med navn “qsmobjmu” som følger:

```
str_cmd = "SELECT objektnavn,underliggende,objektid
FROM qsmobjmu where " + conditions;
```

Figur 13: SQL eksempel, Return_Objects()

Som et eksempel på en egendefinert spørring har vi sql-kommandoen i figur 14 som blir brukt ved populering navigeringstreet. I henhold til ytelseskrav ble det utviklet en spørring som returnerer all nødvendig data for dannelse av en ny gren i treet; slik at klienten kun trenger å sende/motta én SOAP-melding per ekspandering av en ny gren.

```
str_cmd = SELECT (SELECT objektnavn FROM qsmobjmu
WHERE objektid=tilkoblingid) AS objektnavn,
(SELECT underliggende from qsmobjmu b
WHERE a.tilkoblingid=b.objektid) AS underliggende,
tilkoblingid FROM qsmkbl a
WHERE eierid='" + conditions + "'
ORDER BY objektnavn ASC";
```

Figur 14: SQL eksempel, Return_Object_Subs()

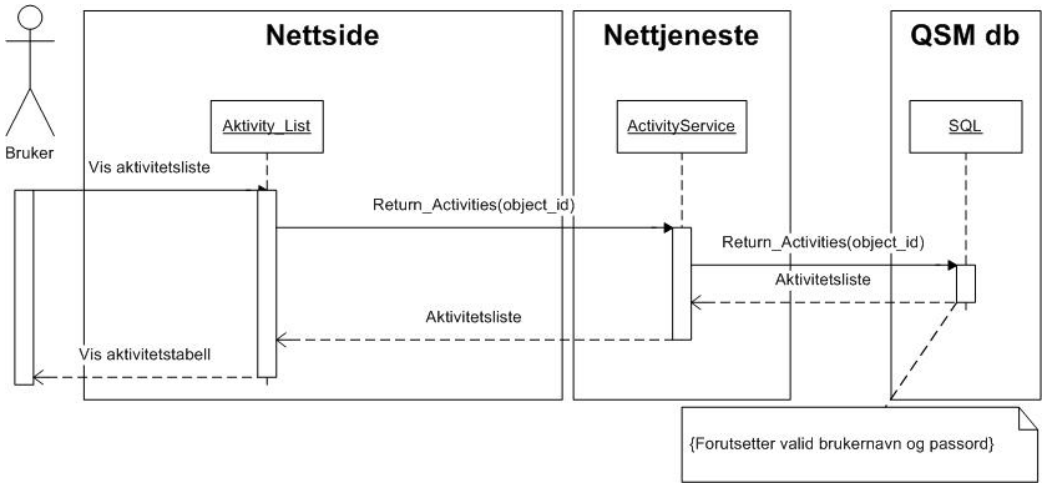
Denne spørringen returnerer objektnavn, antall underliggende objekter, og tilkoblingsid til et objekt mht. objektid sortert i objektnavns stigende rekkefølge; dvs. alfabetisk.

5.4.2 Iterasjon 2

Denne iterasjonen er definert for utvikling av aktivitetuthentingsfunksjonalitet og objektrelasjoner i nettjenestelaget. Dette inkrementet inkluderer også utvikling av codebehind for en enkel html-implementering for testing.

5.4.2.1 Sekvensdiagram for aktivitetforespørsel

Forespørsel etter og returnering av eventuelle aktivitet(er) skjer ved et nettjenestekall til ActivityServices metode Return_Activities som sjekker om klienten er innlogget og aksesserer databasen for returnering av alle aktiviteter tilknytte respektive objekt mht. objektid.



Figur 15: Returner aktiviteter, Return_Activities(object_id)

5.4.2.2 Testing

Den viktigste formen for testing i denne fasen har vært at aktivitetene til et objekt returneres som i Windowsapplikasjonen.
Feilmeldingen “Object moved here(...)” returneres ved metodekall og innloggingssiden html-struktur returneres i soap-meldingen istedenfor aktivitetsdatasettet. Dette grunnet i tilgangskonflikt, som løses ved å starte nettjenestene i en annen utviklingsserver.
Den viktigste formen for testing i denne fasen har vært at brukernavn og passord er korrekte.

Testcase

Testid/nr 1	Testnavn Søk og vis aktiviteter		
Dato 2009-03-19	Tidspunkt 10:00	Testtype system, anvendlighets- og akseptansetest	
Estimert tidsforbruk 60 minutter		Reellt tidsforbruk 120 minutter	
Formål med testen Test av innloggingen.			
Systemets tilstand Systemet i gang med tilkobling til databasen.			
Beskrivelse av testen 1. Søker etter aktiviteter. 2. Vise aktivitet til bruker. Godkjenningskriterier: • Når det kan søkes etter aktiviteter. • Aktivitetene vises til bruker.			
Beskrivelse av feil		Feilgradering lav | middels | høy	
Tidsfrist 22. mars 2009			
Beskrivelse feilretting			
Notater Test suksessfull.			
Test utført av Odd Amund Wik og Chris Stian Melby			

Figur 16: Test case Aktivitet

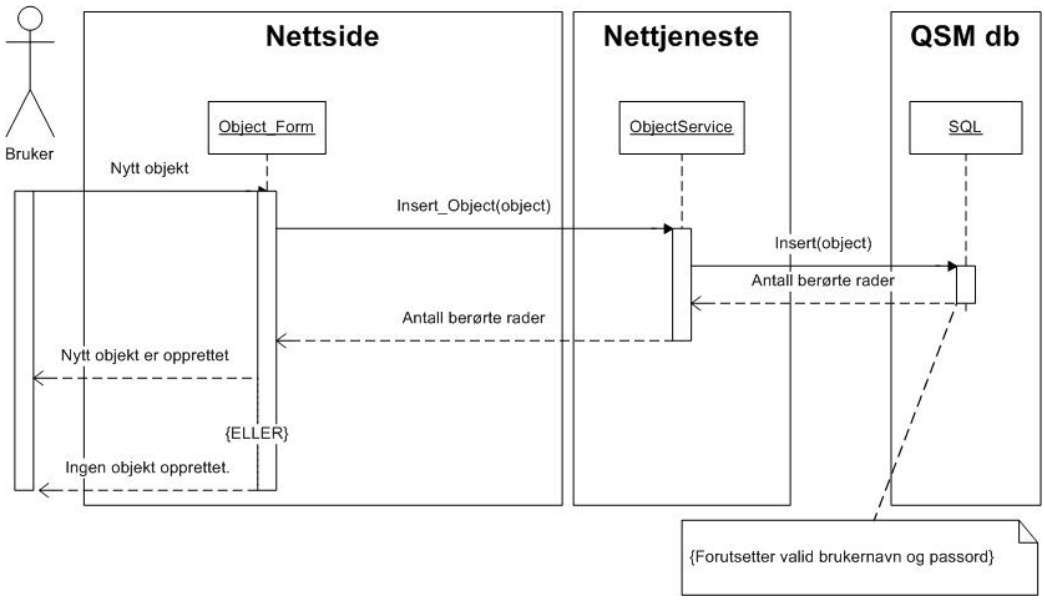
For nærmere informasjon om bakgrunnen for denne testen, se utvidet use case i vedlegg **??**.

5.4.3 Iterasjon 3

Denne iterasjonen er definert for utvikling av endring og opprettelse av et nytt objekt i nettjenestelaget. Dette inkrementet inkluderer også utvikling av codebehind for en enkel html-implementering for testing.

5.4.3.1 Sekvensdiagram for opprettelse av nytt objekt

Opprettelse av nytt objekt skjer ved et nettjenestekall til ObjectServices metode Insert_Object som sjekker om klienten er innlogget og aksesserer databasen for evt. å bruke "stored procedure" GET_OBJEKTID og å legge til det nye objektet.



Figur 17: Opprettelse av nytt objekt, Insert_Object()

Under utviklingen har rammeverket gitt litt hodebry mht. opprettelse og oppdatering av tjenestereferanser. Ved et par tilfeller var en autogenerert settings-fil som inneholdt assembly-referanser ikke oppdatert, og systemet ville ikke kompilere. Dette ble adressert ved sletting av settings-filen og rekompilering, med påfølgende oppretting av ny tjenestereferanse. Men en bedre løsning ble funnet å starte nettjenestene i en annen instans av Visual Studio og kompilere det før en legger til tjenestereferanse i den første instansen som inneholdt serversidene.

5.4.3.2 Testing

Testen har i hovedsak gått ut på å hente ut et objekt fra databasen, for så å foreta endringer på det før det blir dyttet tilbake igjen.

Resterende tester har gått med på å registrere et nytt objekt.
Forskjellen fra å endre et objekt og legge til et nytt objekt anser vi som samme problem og har kun et testcase som omhandler begge.

Testcase

Testid/nr 1	Testnavn Endring av et objekt		
Dato 2009-04-03	Tidspunkt 10:00	Testtype system, anvendlighets- og akseptansetest	
Estimert tidsforbruk 120 minutter		Reellt tidsforbruk 120 minutter	
Formål med testen Test av endring av egenskaper på et objekt fungerer.			
Systemets tilstand Systemet i gang med tilkobling til databasen.			
Beskrivelse av testen Etter å ha hentet et objekt skal det endres på egenskapene til objektet. 1. Endring av egenskapet til et objekt. 2. Lagre objektet på nytt i databasen med oppdaterte egenskaper. Godkjenningskriterier: • Når et objekt kan oppdateres med nye egenskaper og lagres i databasen.			
Beskrivelse av feil		Feilgradering lav | middels | høy	
Tidsfrist 12. april 2009			
Beskrivelse feilretting			
Notater Test suksessfull, henter et objekt og oppdaterer det. Objektet bli lagret riktig i databasen.			
Test utført av Odd Amund Wik og Chris Stian Melby			

Figur 18: Test case endre på objekt

For nærmere informasjon om bakgrunnen for denne testen, se utvidet use case i vedlegg **??**.

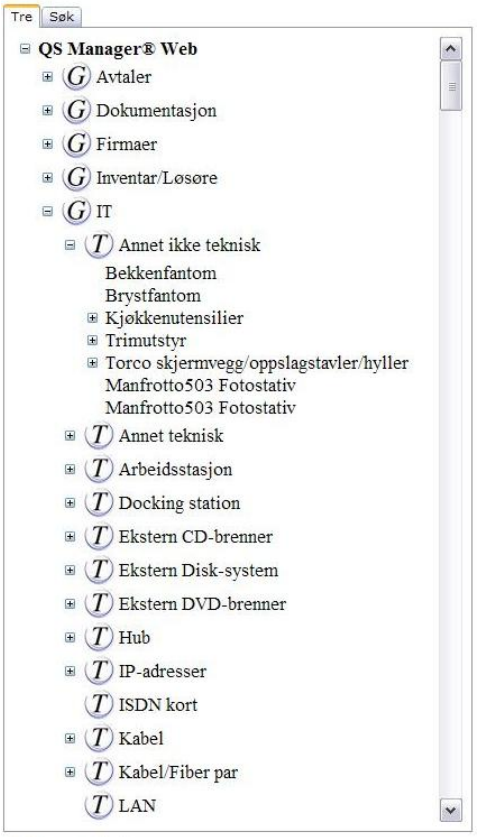
Testene har vist seg å fungere tilfredsstillende og fungerer som det skal.

5.4.4 Iterasjon 4

Denne iterasjonen er definert for utforming og utvikling av nettside, samt. å knytte denne til metodene i nettjenestelaget.

5.4.4.1 Brukergrensesnitt

Tidligere i rapporten har vi gjennomgått noe av funskjonaliteten bak trenavigeringen TreeView i Ojektoversikten. Dette er en del av standard funksjoner i vårt rammeverk. Etter en del utvikling er den på nivå med dette:



Figur 19: TreeView, bilde.

Under utvikling av nettsiden er det verdt å nevne en spesiell utfordring:
Ved utforming av siden med ajax-funksjonalitet ("UpdatePanel") slik at siden oppdateres asynkront; dvs at vi ikke gjør en full post-back, har dette før til at all annen dynamikk på siden slutter å virke og sidevalideringsfeil ("Invalid postback or callback argument (...)") ved sidebytte. Sidevalidering er en ASP.NET metode

for å sikre at sideinnholdet ikke er manipulert.
Opprettelse av nye tab-faner/“ajaxcontrolKit:TabPanel” i (“ajaxToolKit:TabControl”) [1] fra codebehind ga samme feil.

Etter lang tid og leting på forum for å finne hvordan dynamiske tab-faner kom jeg over løsningen på UpdatePanel-problemet (som noen til og med foreslo uløselig), og en stund etter så også også TabControl-problemet.
UpdatePanel-problemet ble løst ved den definering av en skjult <input> trigger som fyrer av BtnActivitiesTrigger_Click(Object sender, EventArgs e) i codebehind for en asynkron post-back.

```
<input id="btnActivitiesTrigger" runat="server" type="button"
style="display:none" onserverclick="BtnActivitiesTrigger_Click" />
```

Figur 20: UpdatePanel Trigger.

```
(...)
<ajaxToolKit:TabPanel ID="TabPanel_2_2" runat="server"
HeaderText="Aktiviteter">
  <ContentTemplate>
    <asp:UpdatePanel ID="UpdatePanel_2_2" runat="server"
    UpdateMode="Conditional">
      <Triggers>
        <asp:AsyncPostBackTrigger ControlID="btnActivitiesTrigger"/>
      </Triggers>
      <ContentTemplate>
      Vis aktiviteter hvor dette objektet er:
      <asp:DropDownList ID="Activity_DropDownList" runat="server"
      AutoPostBack="true" OnSelectedIndexChanged="Return_Activities"
      OnInit="Return_Activities" Height="24px" Width="134px">
        <asp:ListItem Value="4">Hovedobjekt</asp:ListItem>
        <asp:ListItem Value="3">Involvert</asp:ListItem>
        <asp:ListItem Value="2">Utførende</asp:ListItem>
        <asp:ListItem Value="1">Meldt av</asp:ListItem>
      </asp:DropDownList>
      <br /><br />
      <div style="height:700px; overflow:auto;">
        <asp:Table ID="Activities_Table_1" runat="server"
        Visible="false" BorderColor="#cccccc" BorderStyle="Solid"
        BorderWidth="1px" CellSpacing="0">
          <asp:TableHeaderRow>
            <asp:TableHeaderCell><div>Emne</div>
            </asp:TableHeaderCell>
            <asp:TableHeaderCell><div>Type</div>
            </asp:TableHeaderCell>
            <asp:TableHeaderCell><div>Registrert</div>
            </asp:TableHeaderCell>
            <asp:TableHeaderCell><div>Status</div>
            </asp:TableHeaderCell>
            <asp:TableHeaderCell><div>Oppgavenr</div>
            </asp:TableHeaderCell>
          </asp:TableHeaderRow>
        </asp:Table>
        <br />
        <asp:Label ID="Activities_Label_1" runat="server"
        Visible="false" ForeColor="Gray" Text="Ingen rader returnert.
        </asp:Label>
      </div>
    </ContentTemplate>
  </asp:UpdatePanel>
</ajaxToolKit:TabPanel>
(...)
```


Figur 21: Object_Overview.aspx - TabPanel_2_2 (Activities), del 1.

```
<br />
          <asp:Label ID="Activities_Label_1" runat="server"
          Visible="false" ForeColor="Gray" Text="Ingen rader returnert.
          </asp:Label>
        </div>
      </ContentTemplate>
    </asp:UpdatePanel>
</ajaxToolKit:TabPanel>
(...)
```

Figur 22: Object_Overview.aspx - TabPanel_2_2 (Activities), del 2.

I forrige figur ser en hvordan et TabPanel som her inneholder en enkel aktivitet-sliste er av utseende. Dette eksempelet er tatt fra siden Object_Overview.aspx og ut av <ajaxToolKit:TabContainer ID=TabContainer_2"(...)>(...)</ajaxToolKit:TabContainer>. TabPanel må opprettes fra "protected override void OnInit(EventArgs e)" som demonstrert nedenfor.

```
protected override void OnInit(EventArgs e)
{
  AjaxControlToolkit.TabPanel tp;
  XmlDocument ObjDoc = null;
  DataSet ObjDta = null;
  string[] split;
  string name = null;

  // Create new instance of ObjectService and return object
  // specified by argument (object_id) via dataset:
  ObjectServiceReference.ObjectServiceSoapClient autHdr =
  new QSM_WEB.ObjectServiceReference.ObjectServiceSoapClient();
  ObjectServiceReference.ObjectServiceSoapClient ObjSrv =
  new QSM_WEB.ObjectServiceReference.ObjectServiceSoapClient();

  authdr.Name = Cache["Name"];
  autHdr.Password = Cache["Password"];
  autHdr.SSPI = Cache["SSPI"].ToString();

  // Assign the SOAP header and call service:
  ObjSrv.authentication = autHdr;
  ObjDta = ObjSrv.Return_Object_Type_Definitions(
  Request["object_type"]);

  // Create new xml document and load dataset:
  ObjDoc = new XmlDocument();
  ObjDoc.LoadXml(ObjDta.GetXml());

  // Get the root xmlelement for the document:
  XmlElement XML_Root = ObjDoc.DocumentElement;

  // Create header row for the table:
  TableRow row = new TableRow();
```

Figur 23: protected ovveride void OnInit(EventArgs e), del 1.

```
    // Loop through the childnodes of the rootelement:
    XmlNode XML_Parent, XML_Node;
    for (int a = 0; a < XML_Root.ChildNodes.Count; a++)
    {
      XML_Parent = XML_Root.ChildNodes[a];
      XML_Node = XML_Parent.ChildNodes[5];

      if (XML_Node.InnerXml.ToString().Length > 0)
      {
        split = XML_Node.InnerXml.ToString().Split('~');

        tp = new AjaxControlToolkit.TabPanel();
        tp.ID = "DynTab_" + a.ToString();
        tp.HeaderText = split[2].Substring(
        split[2].LastIndexOf("T:") + 2);

        this.TabContainer_2.Controls.Add(tp);
        base.OnInit(e);
      }
    }
}
```

Figur 24: protected ovveride void OnInit(EventArgs e), del 2.

5.4.4.2 Testing

Det er ikke blitt utviklet et eget testcase på webløsningen. Dette er gjort på grunn av at det har vært delvis gradvis utvikling på utseende i sin helhet.
På slutten er det blitt gjort flere helhetstester av systemet. Både mot en mindre database, som vi har brukt under utvikling av produktet, og en større database.

5.4.5 Iterasjon 5

Denne iterasjonen er definert for å konfigurere en sikker (localhost-)server ved anvendelse av kryptert transportlag (SSL).

5.4.5.1 Oppretting av sertifikat

1. La til en virituell katalog med alias SSL som adresseres slik i nettleseren: http://localhost/SSL

2. Startet Internet Service Manager (ISM) som laster Internet Information Server snap-in for Microsoft Managment Console (MMC). Start -> Control Panel -> Administrative Tools -> Internet Service (IIS6.0) Manager

3. Navigerte til "Default Web Site", høyreklikker og velger Properties.

4. I "Directory Security"-fanen finnes knappen "Server Sertificate" under "Secure Communications" som klikkes og starter veilederen for "Web Server Certificate Wizard". Fortsetter (neste), velger opprettelse av et nytt sertifikat og fortsetter.

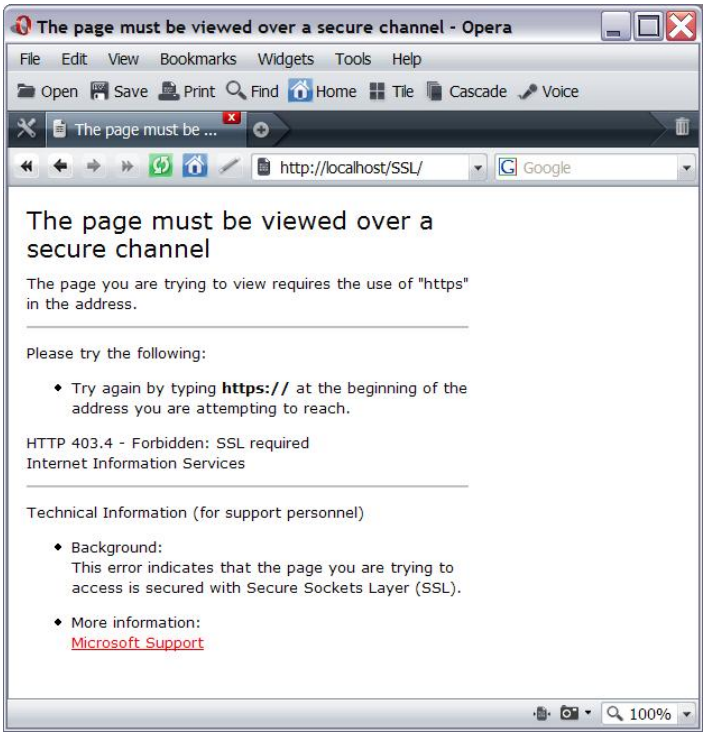
5. Velger å forberede forespørselen nå, men å sende senere. Fortsetter med å velge et navn for sertifikatet som settes til servernavnet og velger 1024 bit RSA/MD5.

6. Velger organisasjonsnavn og -enhet. Samt. servernavn som "common"-navn, og diverse land/lokasjonsdata.

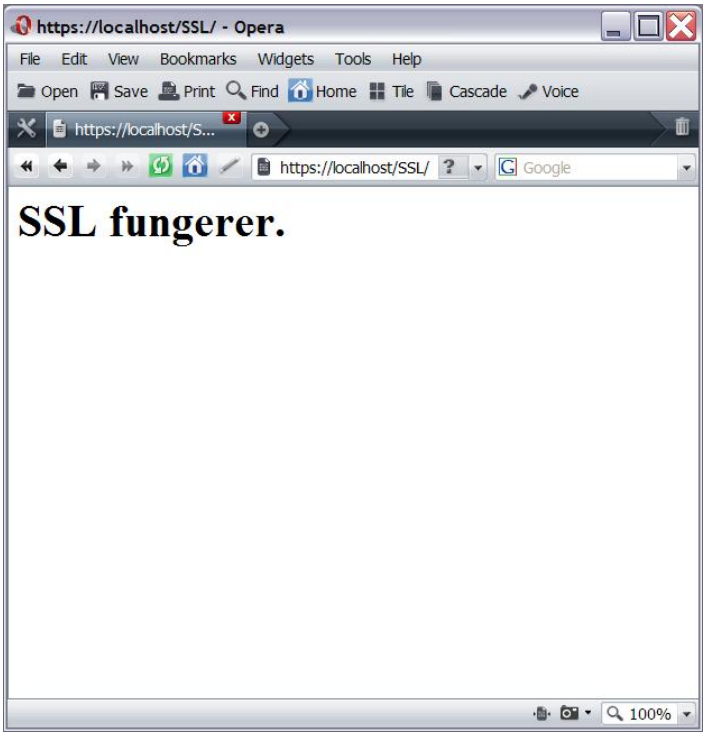
7. Lagrer sertifikatforespørselen og fortsetter med å fullføre forespørselen.

5.4.5.2 Testing

Uttesting av localhost ble foretatt i nettleser (Opera 9.64) i form av blackbox-testing ved adressering av en enkel html-testside. Blackbox-testing i den forstand at vi ikke kjenner til Operas kildekode eller detaljert implementering av krypteringsalgoritmen. Óg vi forutsetter at nettleseren behandler en kryptert server korrekt.



Figur 25: Adressering med http-protokoll.



Figur 26: Adrssering med https-protokoll.

I og med at sertifikatet ikke er godkjent av en utsteder av sertifikater, må en godkjenne det ved å klikke OK i varselboksen, eller noe til samme effekt når en går til adressen.

6 Avslutning

Dette kapittelet inneholder en oppsummering og vurdering av prosjektet, der vi gir uttrykk for vårt synspunkt på prosjektet mht. status per innlevering.

6.1 Evaluering av oppgaven

6.1.1 Produktmessig evaluering

Resultatsystemet bærer noe preg av at vi langt fra er erfarne nettapplikasjonsutviklere. Det ligger en mye større utfordring i en sømløs sammenknytting i en nettside mot for i et windowsprogram, og dette mestrer vi ikke til fulle. F.eks å definere tabellrader til å linke til oppdatering av et felt på nettsiden er ikke spesielt enkelt; forskjellen på server- og client-side script vanskliggjør problemstillingen. Men mener å ha funnet en akseptabel løsning på dette og liknende problem.
Siden vi ikke har rukket å teste systemet i praksis på en server, har vi gjort all testingen på våre egne pc'er ved kompilering til localhost. Dette burde være en rimelig tilnærming, selv om ytelsen på localhost alltid er optimal. Derfor er det ikke mulig å si noe om ytelsen i praksis.

6.1.1.1 Utviklingspotensiale

Systemet er som det fremgår av begrensninger i kravspesifikasjon 2 et ukomplett produkt. Det ligger et potensiale for forbedring og videreutvikling i spesiellt nettsiden. F.eks kan tabellene, og dropdownbox'ene gjøres bedre visuellt ved bedre definering av kolonnebredde, etc. Og samtidig en videreutvikling av nettjenestelaget. Underveis er det luftet muligheten for videreutvikling av systemet i et nytt bachelorprosjekt.

6.1.2 Utviklingsmessig evaluering

Systemutviklingen er gjennomført inkrementellt da systemet lett kunne deles i mindre moduler som kan ferdigstilles og kjøres hver for seg. Dette er en god tilnærming da en tidlig får synlige resultater og dermed tidlige tilbakemeldinger. Representantene fra QS Manager har stillt opp mer enn på forhånd projesert. Dette har lettet utviklingen, særlig i startfasen da databasesystemes er komplekst.
Verktøyene vi har brukt, deri forprosjektrapporten med Gannt-skjema, har bidratt til at vi har hatt god kontroll på utviklingen med hensyn til milepæler.

6.2 Evaluering av gruppearbeidet

Gruppeorganiseringen har vært uformell grunngitt gruppens størrelse på to mann. Gruppeleder er naturlig nok initiativtakeren til prosjektet. Avgjørelser har ofte blitt tatt ved bilateral enighet, men noen også ved gruppeleders insistasjon. Siden gruppeleder etterhvert ble den ledende i enmanns forstand på programmeringen, var det naturlig at administrative oppgaver ble refordelt til den andre part.

6.2.1 Arbeidsfordeling

Som nevnt så avviker den planlagte arbeidsfordelingen fra gjennomført arbeidsfordeling. Gruppeleder har tatt ansvaret for kodeutvikling på grunn av ulikt kompetansenivå; det andre gruppemedlemmet bidro i kodeutviklingen, men i en slik grad at beslutningen kan forsvares. Dette medlemmet fikk påfølgende alle administrative oppgaver fra gruppeleder, og trivdes bedre med dette.

6.2.2 Subjektiv helhetsvurdering

Dette er et utviklingsprosjekt av en slik størrelse at det måtte begrenses i omfang. Å begi seg inn på et slikt prosjekt for første gang kan være overveldene for de fleste. Spesielt med tanke på vår uerfarenhet med utviklingsverktøyet og respektive nettsideutvikling. Omfang- og utfordringsmessig er vi veldig fornøyde med dette prosjektet. Definitivt ikke en enkel oppgave; A/R/I er et stort og komplekst fagområde.

6.2.2.1 Korrigeringer underveis

Odd Amund Wik har kommet inn som et nytt tilskudd til gruppen. Gruppearbeidet ble heftet i starten pga. at den av medlemmene som ønsket å utsette prosjektet fikk innvilget 3 mnd. utsettelse og følgelig måtte ekskluderes fra gruppen; Christoph Roblin trukket seg på personlig grunnlag.
Selv etter den litt turbulente starten på prosjektet har prosjektet vist seg å gå veldig godt. Med unntak for inkrement 1, der vi kom en uke på etterskudd, har vi klart å holde fremdriftsplanen som ble planlagt i forprosjektet.

I forbindelse med planleggingen hadde vi opprinnelig 3 inkrementer og en webløsning. Etter å ha jobbet med prosjektet i fire måneder har vi sett at sammenhengen blir noe annerledes. Vi delte opp delen vi opprinnelig kalte webløsning inn i to deler: inkrement 4 og inkrement 5.
Inkrement 4 tar for seg webløsningen som en helhet og utseende, mens inkrement

5 tar for seg webserveren som måtte legges inn i løsningen for at den skal fungere.

6.3 Konklusjon

Som studenter har vi utviklet vår kompetanse når det gjelder utvikling i miljøet Microsoft Visual Studio 2008 ASP.NET med C# som kildekode. Læringskurven var i begynnelsen ganske bratt, men jeg som gruppeleder lå omtrent på linje. Vi har fått utvidet innsikt i fagområdene C# ASP.NET og nettjenester, MS SQL, men ikke minst A/R/I. Dette er en innsikt og kompetanse som er ettertraktet i arbeidsmarkedet.
Dette har vi oppnådd gjennom arbeidet med utvikling av vårt system QSM Web for å møte QS Manager's kundekrav. Nettløsningen vil forenkle operasjon mot systemdatabasen når brukeren er ute i felt; et ønske fra bl.a IT-tjenesten rettet mot QS Manager AS.
Vi har også fått tilbud om å ha en arbeidsdag på Hamar sammen med QS Manager. Om vi skulle gjort oppgaven på ny ville dette tilbudet blitt takket ja til, med tanke på at QS Manager sitter med en del spørringer og generell hjelp som vi kunne benytte oss av.

Vår sluttkommentar til dette prosjektet er dette: Vi stiller sterkere og bedre forberedt for oppgavene i arbeidslivet etter gjennomføring av dette prosjektet. Det har bidratt til både teknisk og administrativ forbedrelse og innsikt.

Referanser

[1] ASP.NET AJAX. Asp.net ajax control toolkit. http://www.asp.net/ajax/AjaxControlToolkit/Samples/, 2009.

[2] QS Manager AS. Qs manager hjemmeside. http://www.qsmanager.no/, 2009.

[3] IBM Corporation. Rational software architect — web services overview. http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.webser 2005.