BACHELOROPPGAVE:

OLKWEB
SØKNADSMODUL

FORFATTERE: IVAN LÉ HJELMELAND, EMIL KJELSRUD OG OLE CHRISTIAN MELBOSTAD

Dato: 15.05.2013

SAMMENDRAG

Tittel:	OLKweb søknadsmodul	Dato : 15.05.2013
Deltaker(e)/	Ivan Lé Hjelmeland Emil Kjelsrud Ole Chistian Melbostad	
Veileder(e):	Anders Sundnes Løvlie	
Oppdragsgiver:	Innit AS	
Stikkord:	OLKweb, PHP, E-learning, REST, Modul	
Antall sider/ord: 69/13851	Antall vedlegg: 15 sider	Publiseringsavtale inngått: ja

Etter 2-årig yrkesfaglig utdannelse skal elevene gjennomføre opplæring gjennom en lærlingperiode i bedrift. I 2012 var det kun 29% av elever med yrkesfaglig utdannelse som gjennomførte sin læreplan, i Oppland kommune. Noen av grunnen til dette er blant annet dårlig informasjon og kommunikasjon mellom aktører, høyt press på elever som selv er ansvarlig for å finne lærlingplass i bedrift samtidig som fristen for rettigheter til videregående opplæring går ut.

Innit AS som allerede har en fot innenfor problemområdet, ved å tilby programvare for administrasjon og oppfølging av lærlinger, men ønsker en egen løsning for selve søknad- og formidlingsprosedyren. Produktet av dette prosjektet skal fungere sammen med de allerede eksisterende løsningene, for å bidra til å danne en komplett plattform for kvalitetssikring av hele den yrkesfaglige utdanningen. For konsulenter som jobber under opplæringskontorene skal systemet erstatte manuelle rutiner og samtidig gi dem effektive verktøy i den daglige driften.

Innit har gjennom prosjektperioden gitt oss stort spillerom for å avdekke produktets krav, noe som har resultert i et unikt læringsutbytte vedrørende innhenting og analyse av informasjon, samt konsultasjon med kunder. Oppgaven har også utfordret oss på teknisk nivå, da løsningen utnytter sofistikert arkitektur og implementasjon.

ABSTRACT

Title:	OLKweb Application Management	Date : 15.05.2013
Participants/	Ivan Lé Hjelmeland Emil Kjelsrud Ole Christian Melbostad	
Supervisor(s)	Anders Sundnes Løvlie	
Employer:	Innit AS	
Keywords	OLKweb, PHP, E-learning, REST, Module	
Number of pages/words: 69/13851	Number of appendix: 15 pages	Availability: Open

Student of vocational education is required to complete their program through an apprenticeship period, but only 29% of the students registered their completion in 2012. The reason for this apostasy includes poor communication and information flow between the students, consultants and companies, as well as the unnecessary strain on the student to take responsibility for the application and hiring procedure.

Innit AS whom already have an established footing in the E-learning industry, by providing trainee management software, wants to extend their services by offering a complete dissemination and application management module. The final product will work in conjunction with the already existing systems to enable a complete vocational education platform. For consultants who work under training offices, the system should replace manual procedures while giving them efficient tools in daily operations.

Throughout this project our employer has provided us with space to specify the product requirements, which has enabled great learning value in regard of information gathering and analysis, as well as customer consultation. The assignment has also proven to be a great technical challenge, by requiring sophisticated architectural concepts and implementations.

Forord

Takk til oppdragsgiver Innit AS, veileder Anders S. Løvlie og Høgskolen i Gjøvik, som har muliggjort dette prosjektet. Lenke til løsning: http://aspirant.test6.innit.no

Innholdsfortegnelse

Innledning

Problemområde

I følge Thomas Bergersen(personlig kommunikasjon, 7.februar 2013) var det kun 29% av lærlingene som gjennomførte hele lærlingeutdannelsen i Oppland kommune. Det finnes i dag ingen gode systemer for å se hvor det finnes ledige læreplasser og i hvilke fag det finnes ledige læreplasser. I tillegg er opplæringskontorenes rolle under søknadsprosedyren uklar, noe som medfører til økt behandlingstid. Opplæringskontorene og vår oppdragsgiver Innit AS har sett behovet for et slikt system, som kan føre til at en større andel av lærlingene gjennomfører den yrkesfaglige utdannelsen.

Involverte parter

Elev

Etter 2-årig yrkesfaglig utdannelse skal elevene gjennomføre opplæring i bedrift. De er selv ansvarlige for å søke lærlingplass for å kunne gjennomføre utdannelsen frem til fagbrev.

Opplæringskontor

Er bindeleddet mellom fylkeskommunen og bedriftene i tilhørende fylke. Videre er de ansvarlige for å kvalitetssikre elevens lærlingperiode, samt tilordning av lærlingplasser.

Bedrift

Åpner for inntak av lærlinger, samt forplikter seg til å gi lærlingen faglig kompetanseoppbygging frem til fagbrev. Alle bedrifter tilknyttet denne ordningen får månedlig tilskudd fra fylkeskommunen i gjengjeld.

Oppdragsgiver

"Innit AS er et it-selskap lokalisert på Hamar som spesialiserer seg på programvare- og webutvikling, rådgivning, konsulenttjenester, virtualiseringsløsninger, driftstjenester, kommunikasjonsverktøyer og backup / sikkerhetsrutiner.
Selskapet ble etablert i 2000, og har i dag 16 ansatte med bred erfaring og høy kompetanse innen tjenestene de tilbyr. Mangfoldet i klientporteføljen vitner til Innits gode evne til å gjøre hverdagen enklere ved hjelp av moderne teknologi" (Innit AS, 2013).

Innit AS har utviklet OLKweb(OLKweb, 2013) som er en nettbasert oppfølgingsplattform for lærlinger, opplæringsbedrifter og opplæringskontor med fokus på lærlingen. Det finnes per i dag ingen funksjonalitet i OLKweb som gir eleven mulighet til å søke - og bli formidlet lærlingplass på en enkel og oversiktlig måte.

Oppgavedefinisjon, avgrensning og problemstilling

Vi har fått i oppdrag å lage søknadsportalen som har blitt døpt Aspirant. Aspirant skal dekke den komplette prosessen fra utlysning til tildelt lærlingplass. Målet med modulen er å skape en helhetlig oversikt over ledige plasser og reelle søkere.

For konsulenter som jobber hos opplæringskontorene, skal Aspirant gjøre det mulig å fungere som kontrollorgan for opplæring av fremtidens ansatte, ved å definere hvilke rolle disse konsulentene skal fylle i søknadsprosessen, og samtidig gi dem effektive verktøy i den daglige driften.

For bedrifter som ønsker lærlinger skal Aspirant gi bedrifter mulighet til å enkelt profilere seg og annonsere at de ønsker lærlinger. Ulikt tidligere da det ikke finnes tilsvarende arena eller plattform. Aspirant skal fungere som elevens inngangsport til OLKweb.

Problemstilling

Hvordan lage en webbasert søknads- og formidlingsmodul, som en effektiv inngangsport til et eksisterende E-learning system.

Rammer

Som krav fra Innit ble vi pålagt å anvende teknologi basert på åpne standarder. Dette vil si at både plattformen og verktøyene ikke skal ha proprietært opphav. Grunnen til dette er for både å støtte Open Source miljøet, i tillegg vil vi ikke være avhengige av tredjepart innstikk i nettlesere for å tolke innholdet. Konsekvensielt vil løsningen bli stabil og kryssplattformkompatibel.

Utover disse punktene fikk vi som ramme av Innit å benytte oss av det internt utviklede rammeverket RBASE, som blant annet skal erstatte grunnmuren i OLKweb ved utrulling av neste versjon. Det ble derfor et krav fra Innit at vi måtte sette oss inn i RBASE slik at vi på best mulig måte kan utnytte funksjonalitet som allerede finnes i rammeverket. Ved å benytte RBASE gjør vi det enklere for Innit å endre og videreutvikle koden vi har skrevet senere.

Forretningsmessige mål, suksesskriterier og risikoer

Oppdragsgiver mål er at systemet skal benyttes av samtlige opplæringskontor i Norge. Dette anser vi å være en mulig ambisjon da det per dags dato ikke eksisterer et system som dekker behovene som er avdekket i vårt problemområde.

Risikoer

Den forretningsmessige risikoen er å miste troverdighet i offentlig sektor som er hovedkunden til Innit per dags dato.

Kartlegging

Intervjuprosessen

Vi ønsket tidlig i prosjektet å finne ut hvorfor så mange elever ikke fullfører den yrkesfaglige utdanningen. Vi bestemte oss derfor for å gjennomføre intervjuer for å få et bedre overblikk over dagens situasjon. Intervjuobjektene vi bestemte oss for å fokusere på var varierende, men vil på lik linje kaste lys over problemstillingen og peke prosjektet i riktig retning.

I første rekke inkluderer dette elever, opplæringskonsulenter, fagansvarlig VGS, samt lærlinger som allerede er i praksisperiode. Vi ble tipset av oppdragsgiver om å ta kontakt med Thomas Bergersen som er faglærer ved Ringsaker VGS og Marianne Stepperud som er konsulent ved opplæringskontoret for håndverk- og industrifag i Gjøvik. Vi tok kontakt med begge to og fikk

satt opp et møte.

Det første møte hadde vi med Thomas Bergersen og elever ved Ringsaker VGS. Det vi ønsket svar på med dette møtet, var å kartlegge hvilken rolle faglærer har i søknadsprosessen, samt hvilke utfordringer elever kan møte ved å søke lærlingplass. Erfaringene vi kan trekke inn i prosjektet er at fagansvarlig har i større grad en veilederrolle for eleven enn det vi først hadde forutsett. Mye eller all informasjon eleven trenger for å finne bedrift, søke til lærlingstilling og annen støtteinformasjon formidles per nå gjennom fagansvarlig. Dette ser vi på som en flaskehals i dagens løsning med tanke på at eleven da er hundre prosent avhengig av en dyktig og engasjert lærer i denne sammenheng. En bi-effekt av dette er også at prosessen går langsommere når informasjon må behandles og formidles manuelt mellom aktørene gjennom ulike kanaler.

Marianne hjalp oss med å kartlegge hvilken rolle opplæringskontorene har i søknadsprosessen og med dette gav oss et innblikk i hvilke utfordringer de ansatte har i sitt arbeid. Denne informasjonen er essensiell for å kunne automatisere og forbedre dagens rutiner.

Slik det kommer frem i intervjuene oppfatter vi at hovedgrunnene til den labre gjennomførelsen til yrkesfaglige utdannelsen er:

- For dårlig informasjon og kommunikasjon mellom aktørene(Opplæringskontor, bedrift og elever og lærere).
- Dårlig samarbeid mellom skole og bedrift.
- At elevene må selv oppsøke bedrift, uten å vite om bedriften tilbyr lærlingplass.
- At elevene føler press til å ta videreutdanning da fristen for rettighetene til videregående opplæring går ut.
- Manglende motivasjon fra bedrifter til å tilby lærlingplass.
- Mye usikkerhet gjennom hele søknadsprosessen. Dette resulterer i unødvendig lang behandlingstid, som i mange tilfeller strekker seg opp til en halvannen måned.

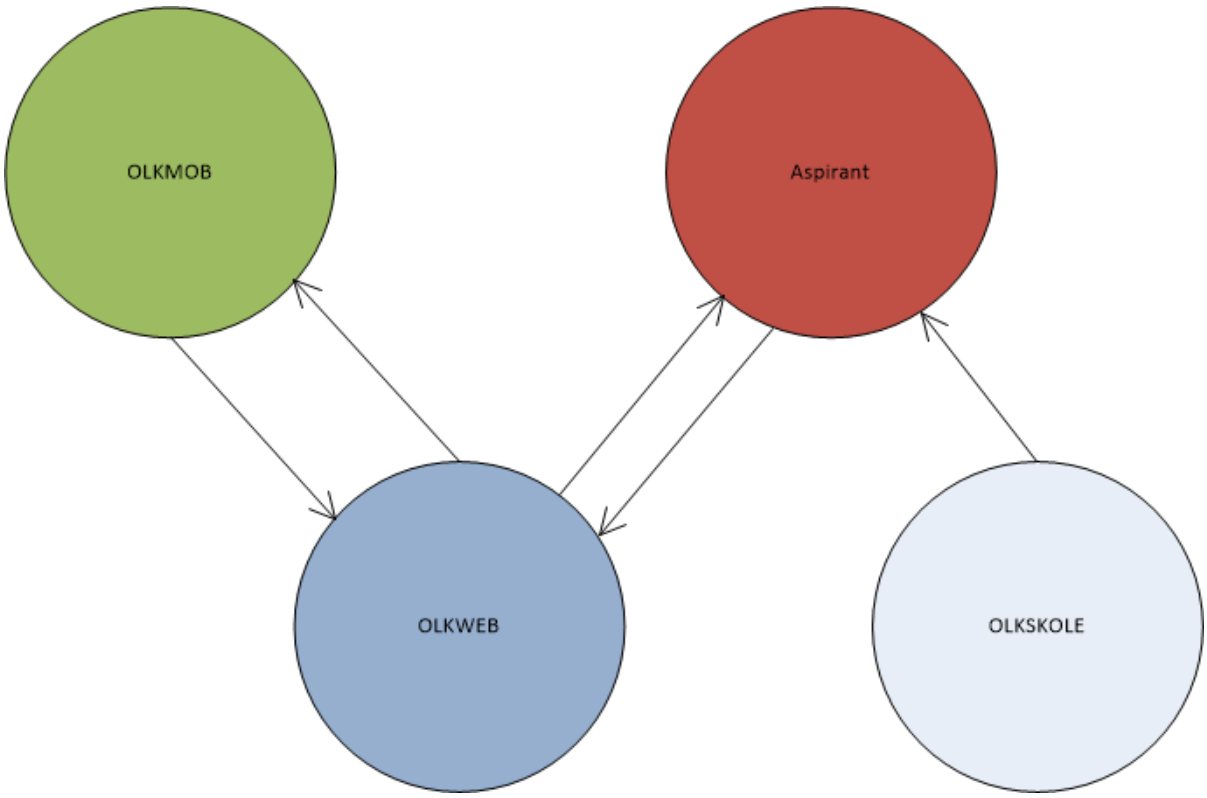
OLKweb og andre eksisterende løsninger

OLKweb er en nettbasert oppfølgingsplattform for lærlinger, opplæringsbedrifter og opplæringskontor med fokus på lærlingen. Dette systemet fungerer som et verktøy som binder lærlingen, opplæringsbedriften og opplæringskontoret sammen på en oversiktlig og praktisk måte.

OLKmob er utviklet som en mobilapplikasjon for dokumentasjon av lærlingens hverdag.

OLKskole er modul som skal utvikles av Innit i fremtiden. Tanken bak denne er å fungere som en opplæringsportal for elever ved videregående opplæring. Hvis Aspirant har mulighet til å motta informasjon fra OLKskole, blir søknadsprosessen enklere for eleven ved at personalia og kvalifikasjoner blir automatisk hentet inn. For opplæringskontoret blir arbeidet med kvalitetssikring enklere, da informasjonen blir hentet inn er fra en pålitelig kilde.

Aspirant sin rolle i dette E-learning systemet, er som nevnt tidligere en inngangsport for eleven til OLKweb.



Figur 1: Oversikt over moduler i OLKweb

Prosjektmål

Arbeidsgivers mål

I slutten av prosjektperioden skal det foreligge et system for søknad, formidling og tildeling av lærlingplass i en bedrift knyttet opp mot et opplæringskontor. Systemet skal kunne utvides uten store komplikasjoner og utveksling av data mellom tilknyttede systemer skal være mulig.

Læremål

Bedre forståelse for utvikling av større web-applikasjoner. Med fokus på utvikling av integrerbar programvare skal gruppen også få økt forståelse for arkitektur og implementering av slike systemer. Gjennom denne prosessen ønsker vi også å lære mer om konsultering med både kunder, brukergrupper og arbeidsgiver.

Målgrupper

Rapporten er utformet etter anbefalt oppsett for informatikkbachelorrapporter utgitt av HiG, og målgruppen inkluderer Innit AS, sensor og andre som har interesse for oppgaven. Produktets målgruppe er Innit og deres kundegruppe.

Øvrige roller

Representanter fra Innit: Kim Sandvold og Simen Mørch
Veileder: Anders Sundnes Løvlie
Gruppeleder: Emil Kjelsrud
Utviklere: Ole Christian Melbostad, Ivan Lé Hjelmeland og Emil Kjelsrud

Simen Mørch og Kim Sandvold er våre kontaktpersoner hos Innit. Simen og Kim har hjulpet oss med systemarkitektur, brukervennlighet, forstå kode og funksjoner i RBASE og andre utfordringer vi har støtt på i forbindelse med oppgaven.

Gruppens faglige bakgrunn og kompetanse

Alle i gruppen studerer Webutvikling ved Høgskolen i Gjøvik, og i løpet av studiet av studiet har vi jobbet mye utviklingen av web -og mobile applikasjoner.

Ole Christian

Hovedinteressen til Ole Christian innen informatikk er programmering og systemutvikling. Programmeringsspråk han har jobbet med gjennom studietiden er: PHP, JavaScript, Java, og C++. Gjennom studiet har han også oppnådd 20 poeng innen systemutvikling. Han trives best med å jobbe med utfordrende oppgaver der han kan bidra til å utvikle programvarearkitektur.

Emil

Emil har relativt bred erfaring i ulike programmeringsspråk som PHP, JavaScript (klient- og serverside), Java og C++. Han har også gjennom mange år opparbeidet en grundig forståelse for arkitektur og programmeringsmønstre ved å aktivt bidra til Open Source miljøet, samt løsninger av lukket natur. Han har i senere år jobbet som webutvikler for individuelle klienter, i tillegg til samarbeidsprosjekter med Høgskolen i Gjøvik, TagStudio AS og Innit AS. Emil streber etter effektive, vedlikeholdbare, stabile og robuste løsninger.

Ivan

Ivan fullfører i skrivende stund bachelorgraden i Webutvikling. Hovedinteressen i studiet har vært alt fra design til programmering, men Ergonomi i Digitale Medier var faget Ivan likte aller best. I tillegg har han også fullført et årsstudium i geografiske informasjonssystemer. Forrige sommer hadde Ivan sommerjobb i Statens Kartverk hvor han var teknisk ansatt i avdeling for universell utforming. Som Ole Christian har Ivan også oppnådd 20 studiepoeng innen systemutvikling, i to store innleveringer var problemstillingen knyttet rundt Scrum som systemutviklingsmodell.

Aspirant skal utvikles i programmeringsspråket PHP og JavaScript. Alle i gruppen har god kompetanse i disse programmeringsspråkene, siden vi har utviklet applikasjoner i disse språkene tidligere.

Forventet kunnskapsnivå hos leseren

På bakgrunn av problemstillingens tekniske natur, vil rapporten hovedsakelig rettes mot personer med teknisk bakgrunn innen webteknologier og informatikk. Anvendt terminologi vil dog være tilgjengelig i vedlegg hvis noe er uklart.

Arbeidsprosess

Da vi alle tre hadde ulike fag ved siden av bacheloroppgaven ble vi enige om at den første dagen i uken alltid ville gå bort til forelesninger og eventuelt arbeid rundt disse fagene. Vi jobbet derfor sammen på skolen eller i oppdragsgivers lokaler på Hamar innenfor kjernetiden 09:00 til 16:00. Noen uker benyttet vi også søndager om dette var nødvendig.

Møtevirksomhet

I løpet av utviklingsperioden har det vært essensielt å sette av møter, både internt i utviklingsgruppen og med oppdragsgiver, for å danne en felles oversikt over prosjektets fremgang. I disse møtene har vi diskutert hvilke utfordringer som er aktuelle for nåværende periode, i tillegg til å utveksle idéer for neste periodes arbeid. Internmøter har blitt utført sporadisk der det har vært behov, mens møter med oppdragsgiver har blitt holdt ved faste ukentlige tidspunkt. Vår møtepraksis har etter beste evne imitert den virksomheten som blir praktisert internt hos Innit. Produktet av disse møtene vil videre utdypes under kapittelet kvalitetssikring.

Møter med veileder ble avholdt etter behov. Under disse møtene ble det konkretisert at prosessarbeidet må dokumenteres omfattende gjennom hele utviklingsperioden.

Valg av systemutviklingsmodell

Ved prosjektets start fantes det lite dokumentasjon og spesifikasjoner for systemet vi skulle utvikle. I tillegg ønsket oppdragsgiver at viktige nøkkelfunksjoner ble prioritert framfor en fullstendig løsning. Videre valgte vi derfor å benytte den agile utviklingsmetoden Scrum. Grunnlaget for valget var å støtte opp for eventuelle endringer eller ønsker fra oppdragsgivers side, men også å kunne gi både sistnevnte og gruppens medlemmer bedre innsikt i prosjektet ved å ha en oversiktlig og tilgjengelig produktkø og holde faste sprint planning- og Review Meetings. Produktkøen ville også være med på å forsikre oppdragsgiver om at nøkkelfunksjoner ville bli prioritert, dette ble gjort ved at oppdragsgiver selv fikk sette en prioritetsverdi på hvert element i produktkøen.

Vi valgte å benytte sprinter med varighet på 14-dager hvor gruppen i sprintene skulle være isolert for endringer fra oppdragsgiver. Sistnevnte skulle være med å velge ut hvilke elementer fra produkkøen det skulle jobbes med i neste sprint. Etter hver sprint blir det holdt et sprint Review- og Retrospective Meeting. Dette møtet skulle brukes til å definere hva vi hadde fått gjort, hva som ikke har blitt gjort, samt refleksjon over hvordan prosessen har vært. For å støtte opp for et

kollektivt eierskap og forståelse i prosjektet valgte vi utover det som er nevnt å trekke inn parprogrammering - et konsept hvor to personer jobber sammen med den samme koden.

Siden Scrum er et rammeverk som baserer seg på empirisk prosesskontroll og mangler krav til dokumentasjon eller fastsatte krav, valgte vi å trekke inn noen artefakter fra RUP. RUP er et arkitektonisk rammeverk med en rekke artefakter som legger til rette for en stabil arkitektur. Siden Aspirant skulle bli en del av et større system med en allerede etablert arkitektur så vi det hensiktsmessig å anvende relevante artefakter fra RUP. Dette inkluderer Use Case, Logical view, Sekvensdiagram og Domenemodell. Ved å trekke inn så formelle artefakter vil vi til en viss grad kunne miste den gevinsten SCRUM gir, eksempelvis behovet for å kunne forutsi endringene. På tross av dette vil prosjektet fortsatt være tilpasningsdyktig innenfor de rammene som er fastsatt.

Ved å benytte denne kombinasjon av utviklingsmetoder i vårt prosjekt mener vi at vi legger til rette for at både oppdragsgiver og sluttbrukere blir involvert i prosjektet. Samtidig legger vi også vekt på at viktig dokumentasjon og krav fastsettes og blir en del av rapporten i prosjektet.

For å definere når et element i Sprint Backlog er "ferdig" velger vi å benytte oss av "Quality gates". I artikkelen Scrum + Engineering Practices Experiences of Three Microsoft Teams (Williams, Brown, Meltzer, Nagappan, 2013), beyntter Microsoft teamene "Quality gates" med stor suksess, . Vi har derfor stor tro på at disse portene skal kvalitetssikre arbeidet vi gjør og forsikre oss om at elementer som har passert portene er klar til levering.

Quality gates - Rapport:

- Skal være fri for skrivefeil.
- Må følge Harvard sine retningslinjer for kildehenvisning

Quality gates - Kode:

- Skal tilfredsstille Innits retningslinjer for kodestandard
- Skal passere alle egendefinerte tester
- Alle offentlige grensesnitt må dokumenteres
- All kode som dyttes inn i produksjon må være fri for feil og advarsler på høyeste nivå

Kravspesifikasjon

Produktet som vil bli et resultat av dette prosjektet er for omfattende å ferdigstille i løpet av prosjektperioden. Vi jobber derfor etter beste evne å ferdigstille de mest kritiske delene av systemet, samt tilrettelegge for enkel overdragelse til oppdragsgiver på sluttdato. Den valgte systemutviklingsmodellen passer også utmerket etter denne planen da Product Backlog til enhver tid vil representere de elementene med høyest prioritet.

Omgivelser

Aspirant er en webapplikasjon der elever og opplæringskontor får tilgang ved å identifisere seg med brukernavn og passord. Brukerne av systemet trenger ikke å laste ned noen programvare for å bruke Aspirant, det eneste de trenger er en moderne nettleser.

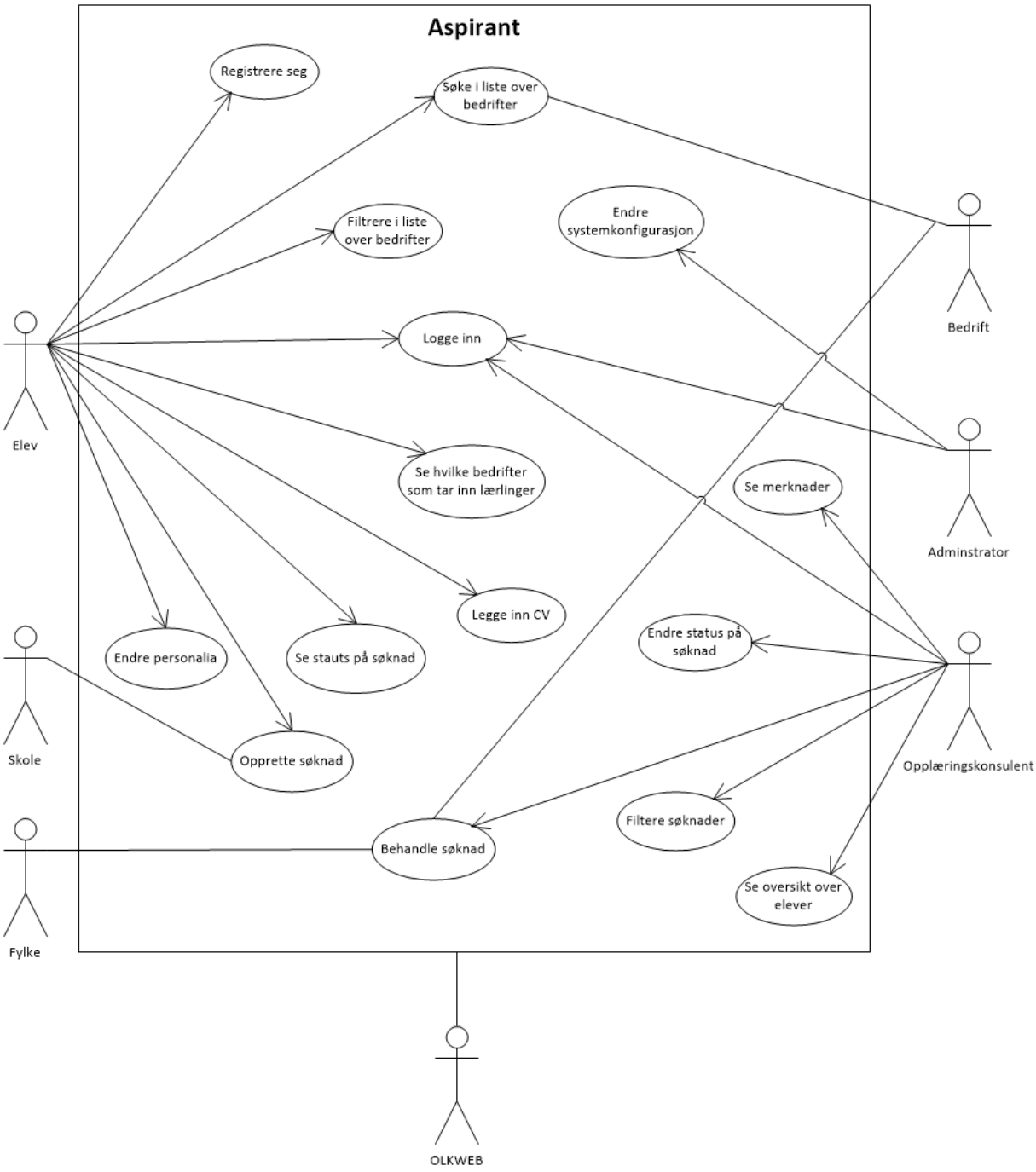
Systemets brukere

Brukerne av systemet OLKweb er: Administrator, opplæringskonsulenter, lærling og bedrift. Aspriant innfører en ny bruker som er elev. Elev er steget før man blir lærling, hvilke handlinger en elev kan utføre i systemet blir vist i Use Case diagrammet. Når eleven blir tildelt lærlingplass endres brukeren fra elev til lærling, han eller hun blir da automatisk flyttet over til OLKweb.

Administrator har ansvaret for å endre systemkonfigurasjonen i Aspirant. Opplæringskonsulentens ansvar er å tildele elever lærlingplass, endre status på søknader og se oversikt over elever.

Use Case / Brukerhistorier

Scrum har som nevnt tidligere ikke noen definert måte å lage Use Case på. Grunnen til at vi bestemte oss for å lage dette diagrammet, var for å få en oversikt over hvilke brukerhistorier som er de mest essensielle i Aspirant. For å beskrive diagrammet har vi laget detaljerte Use Case beskrivelser.



Figur 2: Use Case diagram

OLKweb

I OLKweb blir det opprettet bedrifter og opplæringskonsulenter, som Aspirant benytter seg av. Det er derfor ikke nødvendig i vårt system og ta høyde for at bedrifter og opplæringskonsulenter skal kunne endre sine opplysninger.

Fylke

Fylke har en indirekte rolle ved at opplæringskontoret har plikt til å melde i fra til fylke om nye lærlinger. Det er per dags dato ikke avklart om dette kan gjøres automatisk.

Detaljert Use case beskrivelser

Use Case Navn	Registrere seg	
Aktør	Elev	
Beskrivelse	Eleven registrerer seg for å få tilgang til å se -og søke på tilgjengelige lærlingplasser.	
Pre-krav		
Post-krav	1. All informasjon om eleven lagres i Aspirant sin database.	
Normal hendelseflyt	1. Eleven fyller inn E-post, fulltnavn og passord for å registrere seg.	2. Systemet bekrefter om det er en gyldig E-post adresse og at ikke noen med samme E-post adresse har registrert seg før.
		3. Systemet starter en veiviser og ber eleven å skrive inn lærefag, telefonnummer, adresse, postadresse og postnummer.
	4. Eleven fyller inn de påkrevde feltene.	
		5. Systemet bekrefter at feltene som eleven har fylt inn er gyldig.
		6. Systemet lagrer informasjonen som eleven har fylt inn. Eleven får valget mellom å gå til profilsiden eller å gå til formidlingssiden.
	7. Eleven velger å gå til formidlingssiden	
		8. Systemet omdirigerer brukeren til formidlingssiden.
		9. Systemet avslutter Use Caset
Alternativ hendelseflyt	7. Eleven velger å gå til profilsiden 8. Systemet omdirigerer brukeren til formidlingssiden. 9. Systemet avslutter Use Caset	
Unntak	**A.1 - Eleven avslutter veiviseren** 4. Eleven avslutter veiviseren ved å lukke nettleseren. 5. Eleven må gjennomføre veiviseren neste gang han/hun autentiserer seg.	
Prioritet	Høy	
Bruksfrekvens	Hyppig i søknadsperioden.	

<table>
<tr><td>Use Case Navn</td><td colspan="2">Opprette søknad</td></tr>
<tr><td>Aktør</td><td colspan="2">Elev</td></tr>
<tr><td>Beskrivelse</td><td colspan="2">Eleven finner ønsket bedrift i listen over bedrifter. Eleven trykker på send søknad og får opp én veiviser der han/hun må laste opp CV og/eller vedlegg</td></tr>
<tr><td>Pre-krav</td><td colspan="2">• Eleven har registrert seg som bruker
• Eleven har blitt autentisert
• Det finnes bedrifter med ledige lærlingplasser eller PTF-plasser</td></tr>
<tr><td>Post-krav</td><td colspan="2">• Søknaden sendes til rett opplæringskontor</td></tr>
<tr><td>Normal hendelseflyt</td><td>1. Eleven velger ønsket bedrift.</td><td>2. Systemet sjekker om det finnes ledige lærlingplasser eller PTF-plasser hos denne bedriften.</td></tr>
<tr><td></td><td></td><td>3. Systemet bekrefter at brukeren har lastet opp CV.</td></tr>
<tr><td></td><td></td><td>4. Systemet ber brukeren laste opp påkrevde vedlegg som kreves for at søknaden skal være gyldig</td></tr>
<tr><td></td><td>5. Eleven laster opp det påkrevde vedlegget.</td><td>6. Systemet verifiserer at et vedlegg er lastet opp.</td></tr>
<tr><td></td><td></td><td>7. Systemet sender inn søknaden til riktig opplæringskontor og setter status på søknad til ubehandlet.</td></tr>
<tr><td></td><td>8. Eleven får en bekreftelse på at søknaden er sendt inn.</td><td>9.Systemet avslutter Use caset.</td></tr>
<tr><td>Alternativ hendelseflyt</td><td colspan="2"></td></tr>
<tr><td>Unntak</td><td colspan="2">A.1 - Eleven har ikke lastet opp CV(Steg 3)
1. Systemet viser en feilmelding: «Du har ikke lastet opp CV».
2. Systemet spør eleven om han/hun ønsker å laste opp CV nå eller avbryte.
3. Eleven velger å laste opp CV nå.
4. Systemet verifiserer at CV er lastet opp.
5. Systemet fortsetter normal hendelseflyt fra steg 5.
A.2 – Eleven laster ikke opp vedlegg(Steg 5)
1. Systemet viser en feilmelding: «Du må laste opp påkrevde vedlegg for å søke lærlingplass hos denne bedriften»
2. Eleven velger å ikke laste opp vedlegg.
3. Systemet avslutter Use Caset.</td></tr>
<tr><td>Prioritet</td><td colspan="2">Høy</td></tr>
<tr><td>Bruksfrekvens</td><td colspan="2">Hyppig i søknadsperioden.</td></tr>
</table>

<table>
<tr><td>Use Case Navn</td><td colspan="2">Endre status på søknad</td></tr>
<tr><td>Aktør</td><td colspan="2">Opplæringskonsulent</td></tr>
<tr><td>Beskrivelse</td><td colspan="2">Opplæringskonsulenten endrer status på mottatte søknader</td></tr>
<tr><td>Pre-krav</td><td colspan="2">1. Opplæringskonsulenten har blitt autentisert
2. Det må foreligge én eller flere søknader</td></tr>
<tr><td>Post-krav</td><td colspan="2"></td></tr>
<tr><td>Normal hendelseflyt</td><td>1. Opplæringskonsulenten velger en søknad fra listen over søknader.</td><td></td></tr>
<tr><td></td><td></td><td>2. Systemet gir brukeren mulighet til å endre status fra åpen til: Under behandling, godkjent, avist.</td></tr>
<tr><td></td><td>3. Opplæringskonsulenten setter status til godkjent.</td><td></td></tr>
<tr><td></td><td></td><td>4. Systemet endrer status på søknaden og sender en E-post til eleven som opplyser om statusendringen.</td></tr>
<tr><td></td><td></td><td>5. Systemet avslutter Use Caset</td></tr>
<tr><td>Alternativ hendelseflyt</td><td colspan="2">Status: Under behandling
3. Opplæringskonsulenten setter status til under behandling
4. Systemet endrer status på søknaden, men ingen E-post blir sent.
5. Systemet avslutter Use Caset.
Status: Godkjent
3. Opplæringskonsulenten setter status til godkjent.
4. Systemet endrer status på søknaden. Sender en E-post til både bedrift og elev.
5. Systemet avslutter Use Caset
Status: Avist
3. Opplæringskonsulenten setter status til avist
4. Systemet ber om opplæringskonsulenten skrive inn begrunnelsen for at søknaden ble avist.
5. Systemet fortsetter normal hendelsesflyt fra steg 4.</td></tr>
<tr><td>Unntak</td><td colspan="2">A.1 – Opplæringskonsulenten skriver ikke inn begrunnelse for at søknaden ble avist
1. Systemet viser en feilmelding «Du har ikke skrevet begrunnelse for hvorfor søknaden ble avist. Status vil ikke bli endret.
2. Systemet avslutter Use Caset.</td></tr>
<tr><td>Prioritet</td><td colspan="2">Høy</td></tr>
<tr><td>Bruksfrekvens</td><td colspan="2">Hyppig i søknadsperioden.</td></tr>
</table>

Deployment View

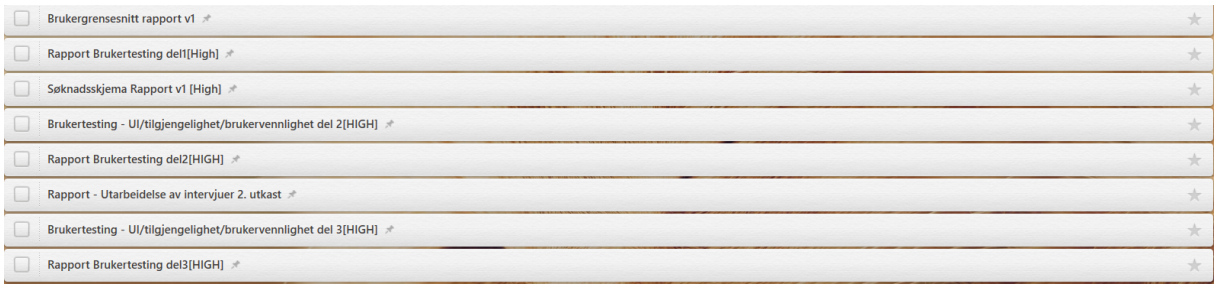
På bakgrunn av uavklaringer internt hos oppdragsgiver, vil vi ikke ha mulighet til å illustrere oversikten over serverstruktur og -kommunikasjon. Vi forholder oss derfor til en enkel modell med to typer klienter (elev og opplæringskontor), samt én sentral server som representerer hele løsningen. Dette vil dog endres når løsningen skal implementeres i miljøet sammen med OLKweb og potensielt OLKskole.

Produktkø

Tidlig i prosjektet ønsket vi å benytte OnTime Scrum som et verktøy for å holde oversikt over elementer i produktkøen. Dette gikk vi bort fra da dette opplevdes for omfattende og komplisert. Avgjørelsen falt da oppdragsgiver ytret ønske om bruk av Wunderlist[4] som et gyldig alternativ. Å bruke dette verktøyet fullt ut ville vært alt for ressurskrevende og kunne ha påvirket resultatet i en negativ retning.

Wunderlist er et styringsverktøy for å få oversikt over hvilke oppgaver som skal gjøres til hvilken tid. Dette verktøyet blir brukt daglig av Innit i deres utviklingsprosjekter.

Vi satt opp produktkøen som en liste med oppgaver i Wunderlist som vist i bildet under.

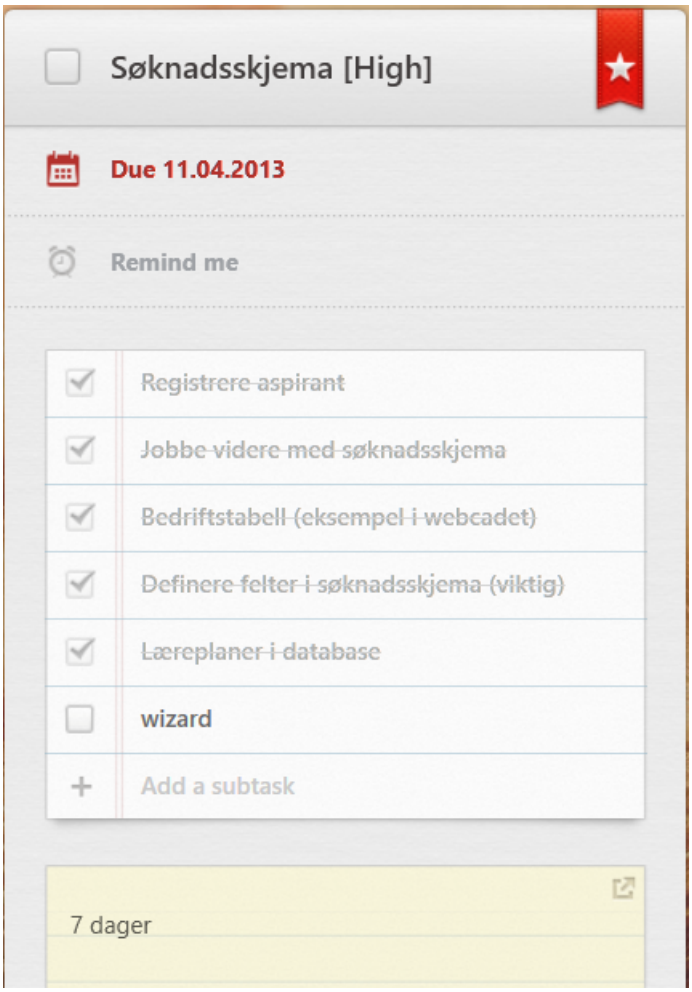


Figur 3: Utklipp av hvordan en del av produktkøen ser ut i Wunderlist

I begynnelsen av hver Sprint setter vi merkelapp i Wunderlist på hvilke elementer som skal gjøres i løpet av den gjeldende Sprinten. Prioriteten av hvert element i produktkøen blir fastsatt av oppdragsgiver etter hvor viktig de er for systemet som skal leveres.

Figur 4: Utklipp av et markert element av produktkøen i Wunderlist

For å se hvilke deloppgaver som må gjøres for at et element i produktkøen skal bli ferdigstilt, har Wunderlist en detaljert visning av hvert element som vist i bildet under. Når en deloppgave er ferdig markeres elementet, og det er dermed enkelt å se hvilke deloppgaver som er ferdig og ikke. Estimatet til hvert element i produktkøen har vi skrevet nederst i den detaljerte visningen, vi kan dermed enkelt velge ut hvilke elementer vi har tid til å gjennomføre i hver Sprint.



Figur 5: Utklipp av detaljert produktelement i Wunderlist

Supplementær spesifikasjon

Sikkerhet

For å få tilgang til de ulike tjenestene i Aspirant kreves det at bruker eller administrator må autentiseres med brukernavn og passord som sjekkes opp mot en database. Denne funksjonaliteten er knyttet opp mot allerede eksisterende klasser i RBASE, som vi utvider med egendefinert funksjonalitet. Person klassen tar for seg registrering av brukernavn og passord, samt aksesskontroll og mulighet for å tilknytte ulike roller.

Logg

Feil og unntak skal kunne logges til en egen fil, men i senere tid også direkte til en konsulent. Dette er funksjonalitet som allerede eksisterer i RBASE, men krever til gjengjeld at vi spesifiserer gode feilmeldinger.

Andre ting som skal logges er endring av brukerinformasjon. Med dette menes det at brukernes interaksjon med sine egne data vil loggføres slik at aktuelle opplæringskonsulenter enkelt kan følge disse.

Implementasjon

I dagens løsninger har Innit AS valgt å benytte en tradisjonell MySQL-database. Denne kjøres på toppen av en Debian 6 server som er sikret med backup-generator i tilfelle strømmen skulle gå.

Vi skal utvikle i de samme utviklingsverktøyene som Innit AS benytter, det vil si NetBeans som IDE og PHP og JavaScript som programmeringsspråk.

Menneskelig grensesnitt

Aspirant skal benytte standarder definert fra World Wide Web Consortium, blant annet HTML5 og CSS3. Dette er for å støtte de mest brukte nettlesere tilbake til Internet Explorer 7. Med dette mener vi blant annet Safari, Mozilla Firefox, Google Chrome og Opera. Vi skal også benytte Modernizer.js for å bedre kompatibiliteten så langt det er mulig.

Responstid

Da endringer og oppdatering på løsningen vil finne sted på egne testservere vil ikke systemet oppleve mye nedetid på grunn av dette. Store oppdateringer kan medføre at systemet er nede i små perioder, men dette vil da være utenfor arbeidstid.

Ved bruk av en relativt god nettlinje (1Mbit) skal de tyngste oppgavene i Aspirant ha en responstid på maks 3 sekunder. Oppdragsgiver har fastsatt maks total nedetid på 12 timer i løpet av et år.

Konfigurasjon

Brukere av Aspirant skal kunne endre sine personalia og annen informasjon som er knyttet opp mot sin egen bruker. Utover dette ligger all konfigurasjonsendring hos enten administrator eller konsulent hos Innit AS.

Juridiske krav om personvern

Personvernloven § 13 sier følgende: "Den behandlingsansvarlige og databehandleren skal gjennom planlagte og systematiske tiltak sørge for tilfredsstillende informasjonssikkerhet med hensyn til konfidensialitet, integritet og tilgjengelighet ved behandling av personopplysninger." Aspirant inneholder store mengder sensitiv informasjon, og for å hindre uvedkommende tilgang og eventuelt misbruk av disse opplysningene er det kun bestemte roller i Aspirant som har innsyn. Det vil kun være personer som har skrevet under taushetsklausul som har mulighet til å logge seg inn med rettigheter til å se opplysninger om andre personer enn seg selv.

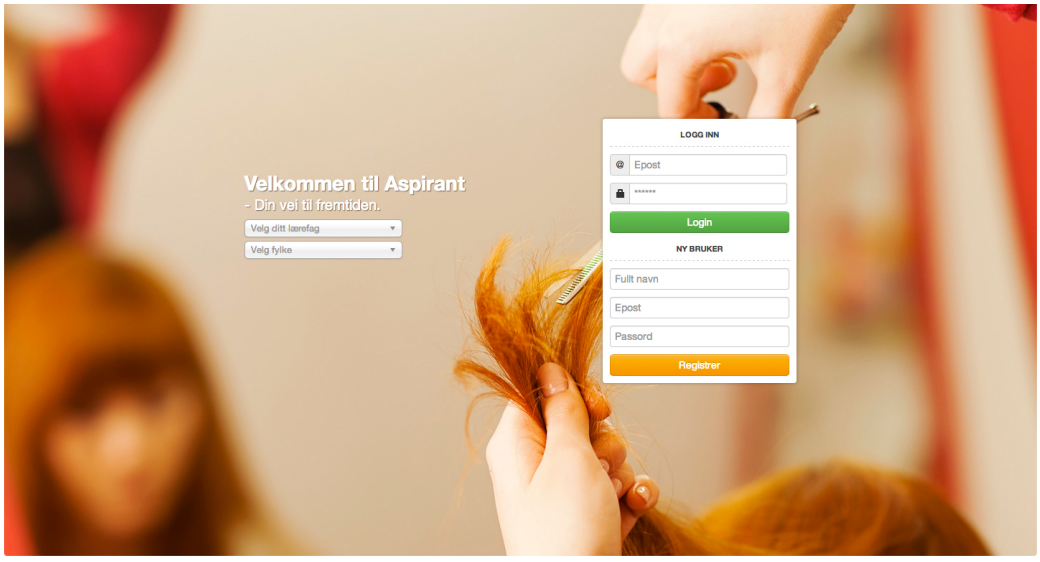
Presentasjon av løsningen

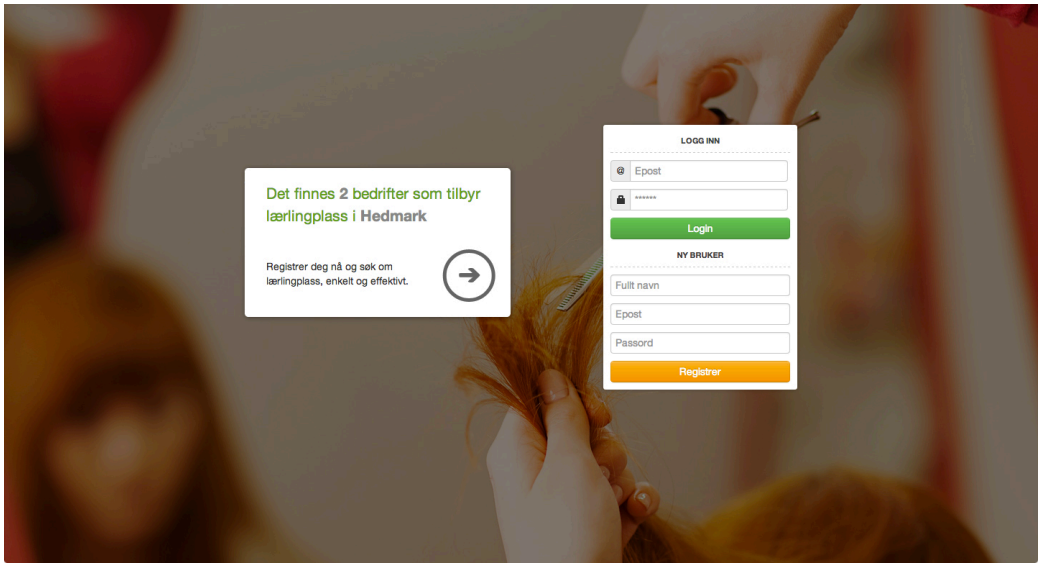
Som nevnt tidligere var det uttrykt ønske fra både elever ved Ringsaker videregående skole og konsulenter ved opplæringskontoret i Oppland om en god løsning på formidling av lærlingplasser. Noen av problemene i dagens situasjon er blant annet at elever føler stor usikkerhet rundt mulighetene for lærlingplass i bedrift og mangel på informasjon rundt selve søknadsprosessen. For konsulentens perspektiv ble det ytret stort ønske om å kunne behandle søknader på en mer effektiv og ryddig måte - både søkere og søknader skal være lett å finne fram i systemet for så å bli behandlet. Det ble også spesifisert at en stor andel av elevene har lese- og skrivevansker, det var ønskelig med en enkel og oversiktlig løsning for å forhindre at selve søknadsprosessen skulle bli vanskeligere enn nødvendig.

Innlogging og registrering

Innlogging- og registreringssiden er det første en bruker blir presentert når de besøker løsningen vår, det er derfor kritisk at denne siden gir den informasjonen brukeren trenger for å bli overbevist til å registrere seg i systemet. Her vil det også være viktig at terskelen for registrering ikke er for høy, da dette kan medføre at brukeren opplever dette som vanskelig eller utfordrende, spesielt om denne prosessen inneholder mye tekst. Måten vi har valgt å løse dette på er derfor å lage en innbydende og fristende presentasjon for brukeren med store knapper og mye luft. Eleven har så to valg, enten å logge inn eller å registrere seg. Registreringen består av kun tre felt; et navn, en e-postadresse og et ønsket passord, mer skal ikke til for å registrere en bruker i Aspirant.

For at eleven skal få et lite innblikk i hva Aspirant kan tilby blir en oppfordret til å velge et lærefag og et fylke. Gjør en dette vil bakgrunnen bli mørkere og informasjon om antallet bedrifter som tilbyr lærlingplass innenfor valgt lærefag og fylke vil dukke opp. Vi har valgt å kalle dette for en smakebit av det Aspirant tilbyr, brukeren får nemlig ikke informasjon om hvilke bedrifter dette gjelder før han/hun har registrert seg og fylt ut en brukerprofil. Denne typen smakebit er hyppig brukt i flere av dagens webløsninger, blant annet dating-tjenester, Twitter, Facebook-applikasjoner og markedsplasser på nett.

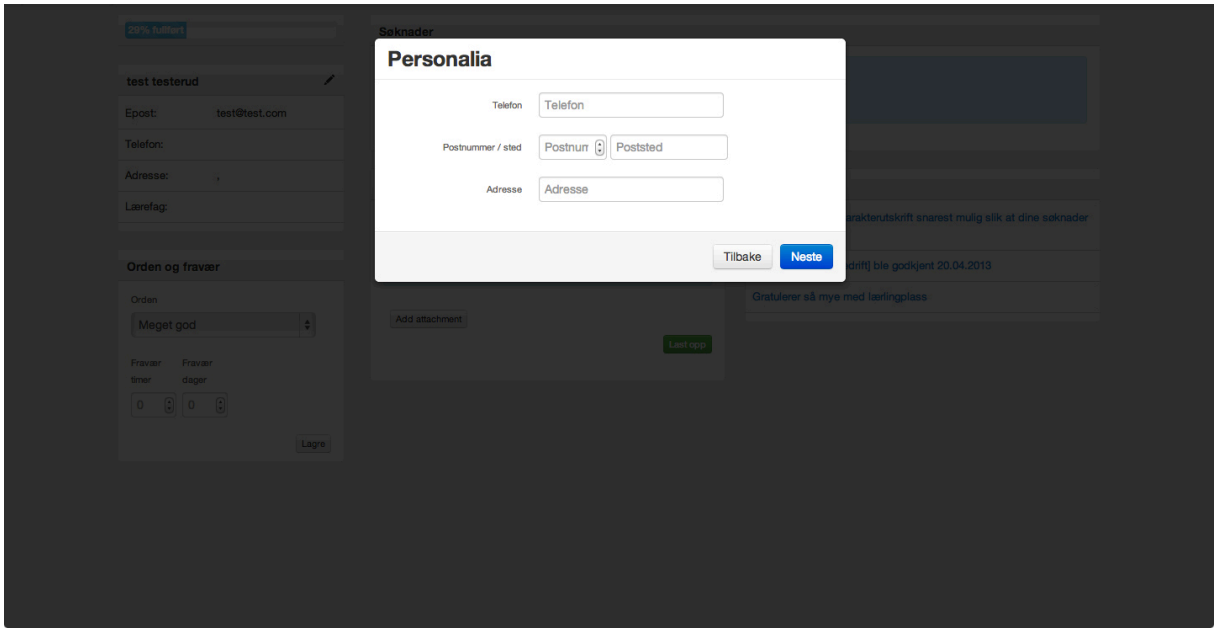




Figur 6: Innlogging- og registeringsside

Veiviser

Etter at en bruker er registrert og logget inn må informasjon om brukeren fylles ut, dette skjer ved at bruker steg of steg fyller ut informasjon om seg selv i en veiviser. Grunnen til at vi valgte å gå for en veiviserløsning er at vi da har muligheten til å dele opp teksten i mindre deler for å gjøre det enklere og mer oversiktlig for brukeren å fylle inn informasjon. Ved å samle inn denne informasjonen tidlig vil store deler av en eventuell søknad allerede være fylt ut, noe som kan føre til at søknadsprosessen oppfattes som enklere. En annen fordel med dette er at noen bedrifter kan kreve mer informasjon enn andre, for eksempel helseinformasjon eller vandelsinformasjon. Ved at mye an informasjonen er utfylt allerede vil det da kun være bedriftspesifikk informasjon som må fylles ut før en kan sende en søknad. Etter at denne informasjonen er utfylt vil bruker få to valg, enten få oversikt over sin egen brukerprofil eller oversikt over bedrifter som tilbyr lærlingplasser.

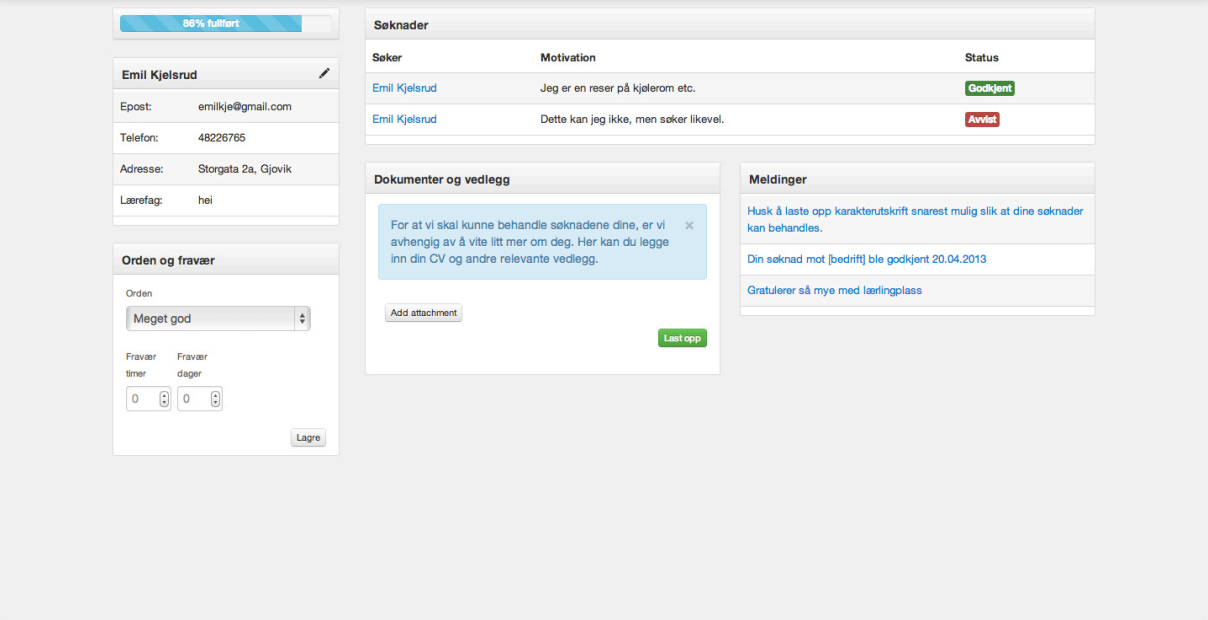


Figur 7: *Veiviser*

Profilside

Profilsiden er den siden som vil være unik for hver bruker. Denne inneholder personalia, søknader og informasjonen som ble fylt ut i veiviseren. Hvis veiviseren av en eller annen grunn ikke blir fullført vil det vises en progresjonsbar på profilsiden. Denne indikerer hvor mye av informasjonen som er fullført for at en søknad kan sendes.

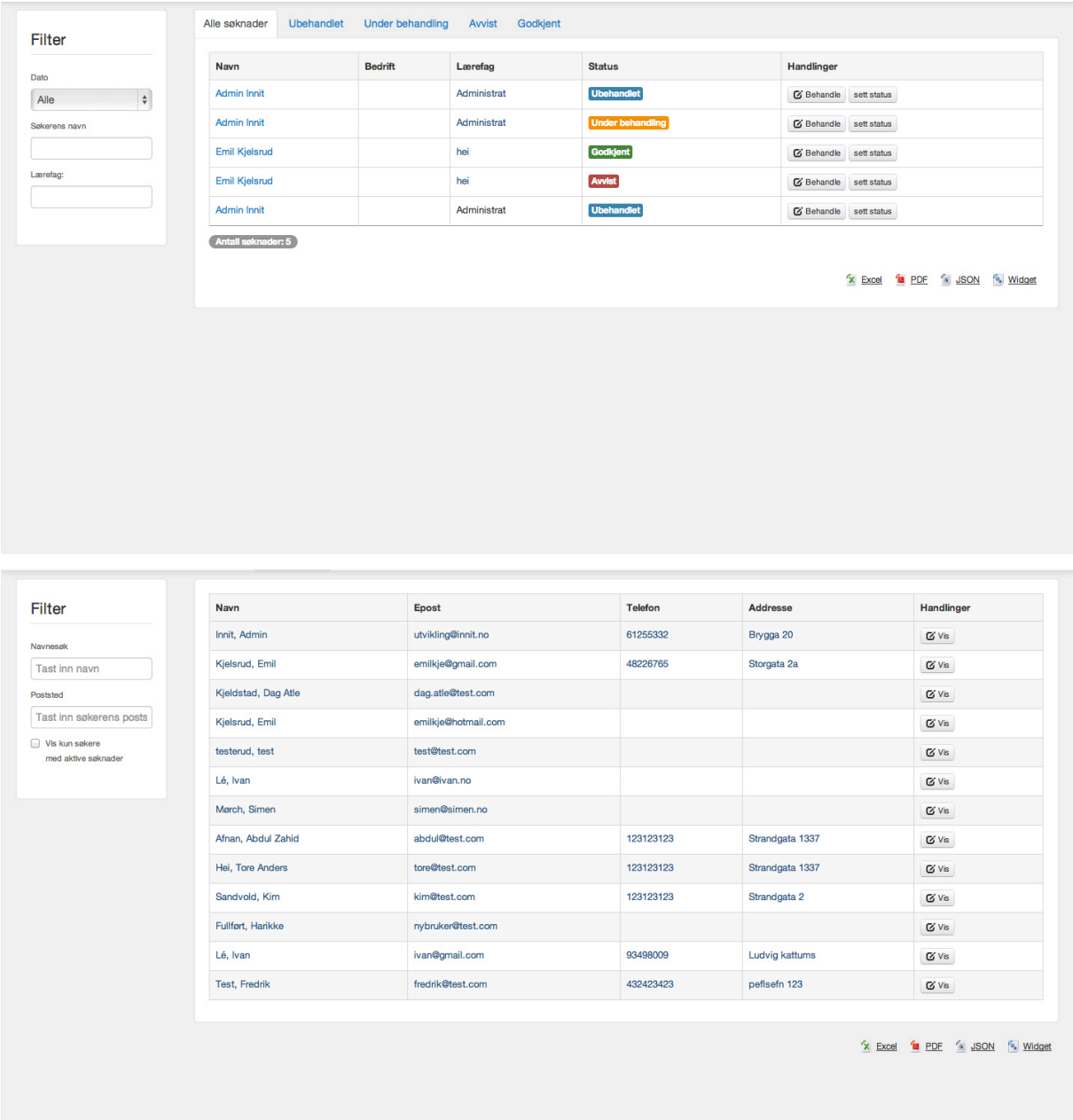
Så snart veiviseren er helt gjennomført vil profilsiden bli brukerens oversikt over egne søknader og brukerinformasjon. Her vil han/hun kunne sjekke status på søknader, endre personlig informasjon, laste opp vitnemål og eventuelt andre nødvendige vedlegg. Brukeren vil også kunne lese meldinger som er sendt fra opplæringskontoret.



Figur 8: Profilsiden

Administrasjon av søkere og søknader

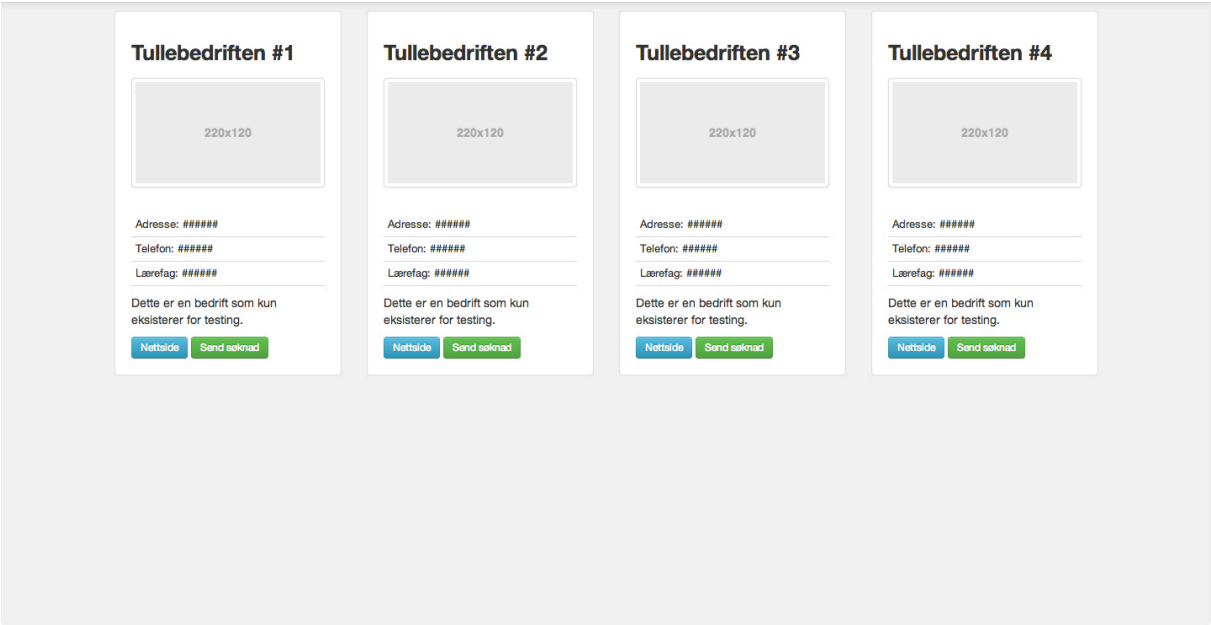
For å løse utfordringen med usikkerhet og lang behandlingstid gjennom søknadsprosessen har vi gitt både bruker og opplæringskonsulent bedre oversikt. Opplæringskonsulenten har fått en oversiktlig visning av både søkere og søknader, søknadene er sortert på hvilken status søknaden har, enten avvist, godkjent, under behandling eller ubehandlet. På denne måten kan konsulenten enkelt se hvilke søknader det vil være aktuelt å jobbe med og har i tillegg muligheten til å endre status på flere søknader på samme tid. Brukeren har muligheten til å følge søknaden sin i profilsiden og vil hele tiden kunne se en oppdatert status på søknaden. Som et ekstra tillegg er det også lagt opp for en egen filtrering etter ønske, det kan enten være i form av navn eller en bestemt dato.



Figur 9: Søknadsoversikt og søkere

Bedriftsoversikt

Bedriftsoversikten er en side som viser hvilke bedrifter som tilbyr lærlingplasser ut i fra brukerens kriterier; enten på geografisk begrensning eller innenfor et bestemt lærefag. Oppdragsgiver ytret at dette gjerne kunne bestå av en mockup og skulle heller ikke prioriteres framfor andre deler av løsningen. Målet med siden er å kunne vise aktuelle bedrifter for brukeren.



Figur 10: Stilling- og bedriftsformidling

Design

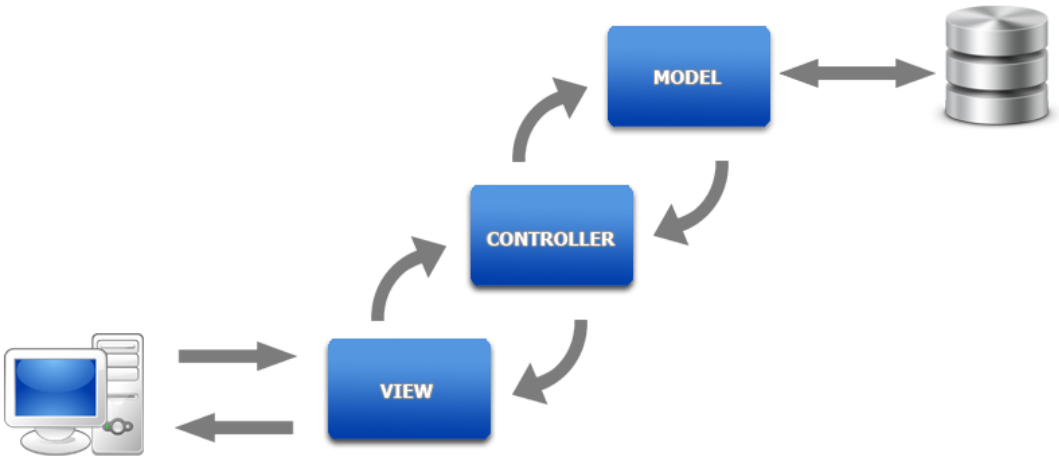
Overordnet arkitektur

Lagdeling

MVC er en forkortelse for Model, View, Controller som er et prinsipp, eller designmønster, for separering av kode i større systemer. Standarden legger ingen føringer for implementasjon, men tilrettelegger for en konvensjon som bidrar til et ryddigere og mer oversiktlig konseptuelt bilde av systemets arkitektur.

I korte trekk defineres tre ulike komponenter som utfører sine respektive oppgaver:

- Models for lagring av data, gjerne med tilordning til databasesystemer
- Views er et presentasjonslag i form av et grensesnitt (bruker- eller maskinsentrert)
- Controllere fungerer som et mellomlag mellom Models og Views for å behandle, kontrollere, og validere datautveksling.



Figur 11: MVC modell

Ved å dele opp systemet i komponenter med tydelige definerte roller vil vi som en konsekvens få ryddigere kildekode som er enklere å feilsøke, teste og vedlikeholde.

I dette prosjektet vil vi bruke et rammeverk utviklet av Innit AS som konkretiserer disse prinsippene, dog med noen justeringer i terminologi og struktur.

Template (View) - Presentasjonslaget

Er en representasjon av data i vilkårlige formater. JSON, HTML, CSV og PDF er allerede innebygget i rammeverket, men det gis gode muligheter til å utvide disse og evt utvikle adaptere til nye formater selv.

Handler + Processor (Controller) - Applikasjonslaget

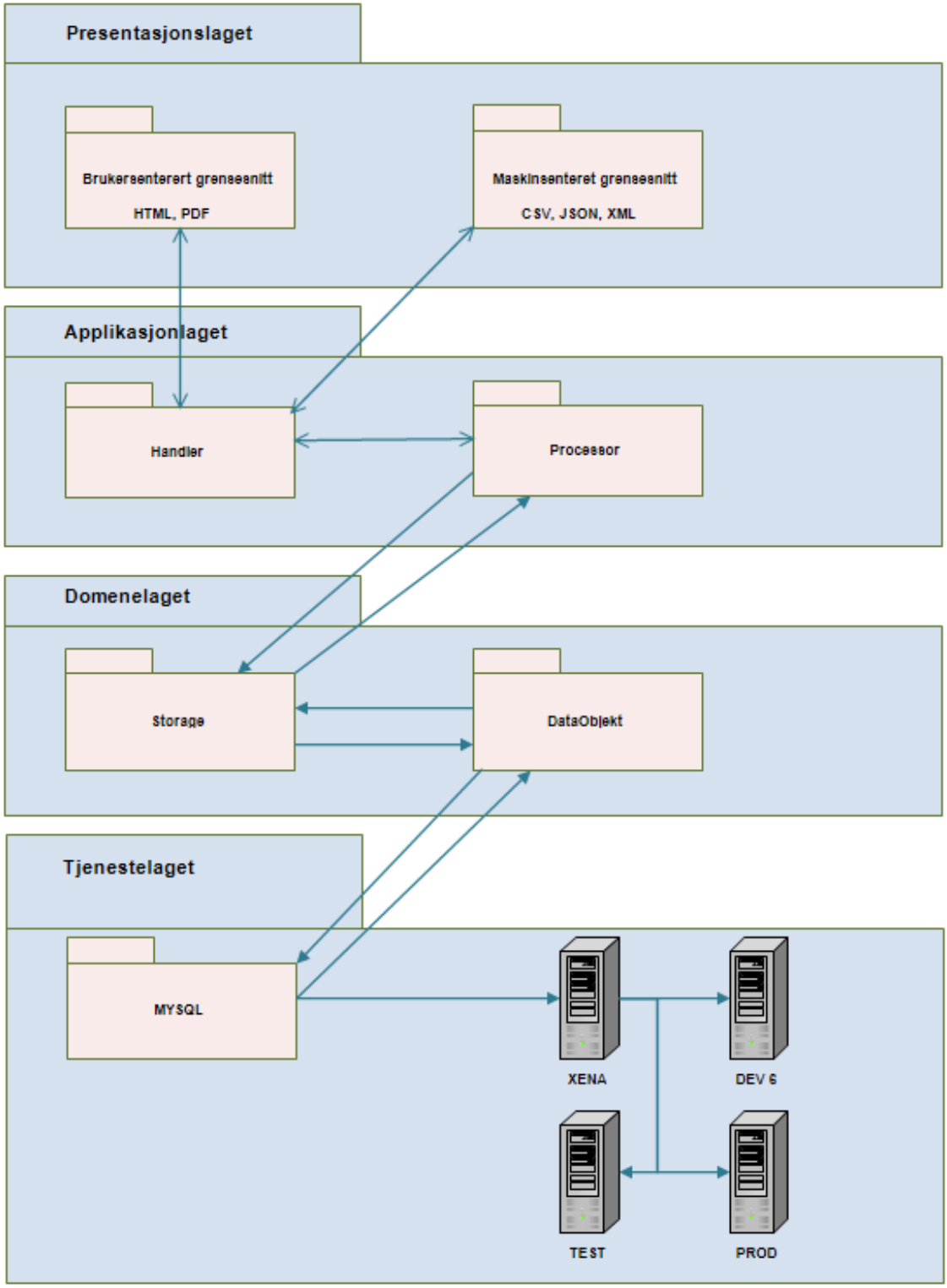
I stedet for å ha én klasse for logikk vil vi i RBASE operere med en sammensetting av Handlere og Prosessorer. Disse vil i utgangspunktet styre aksesskontroll/ruting og ytterligere kjernefunksjonalitet respektivt.

Storage (Model) - Domenelaget

Er klassene som håndterer database og relasjoner mellom data. Et Storage behandler et DataObject som er en direkte representasjon av tuplene i den aktuelle databasetabellen.

Tjenestelaget

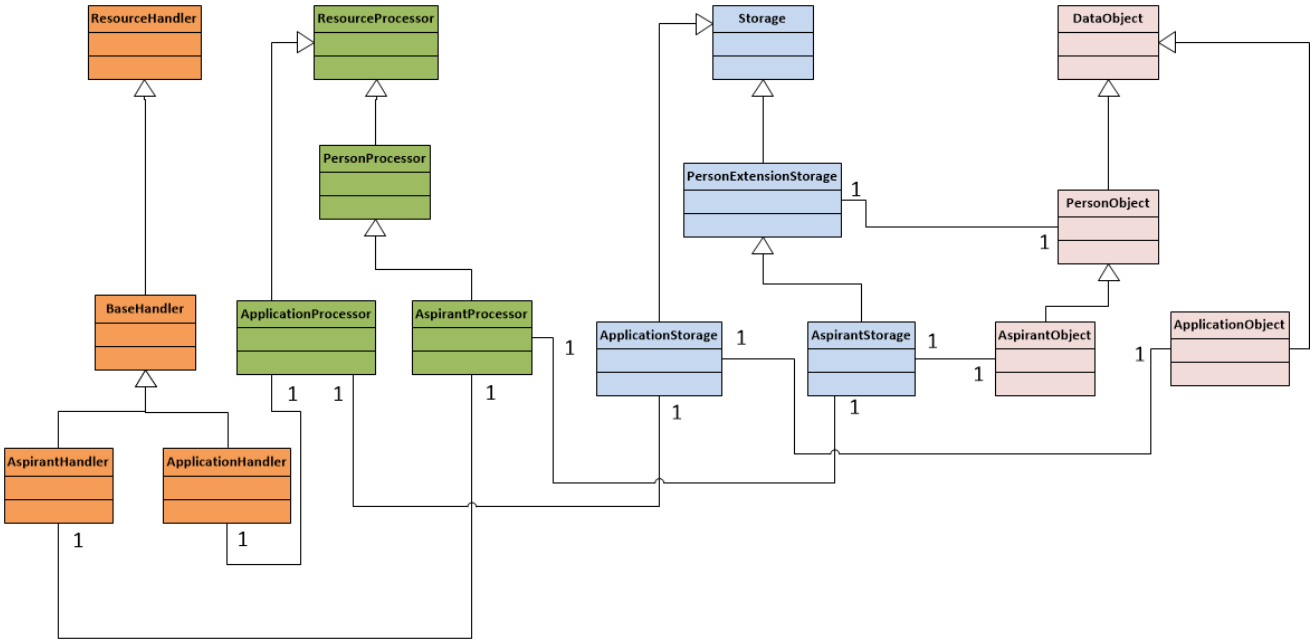
I tjenestelaget finner vi databasen og serverne til Innit AS.



Figur 12: Illustrasjon av lagdelingsmodell

Domenemodell

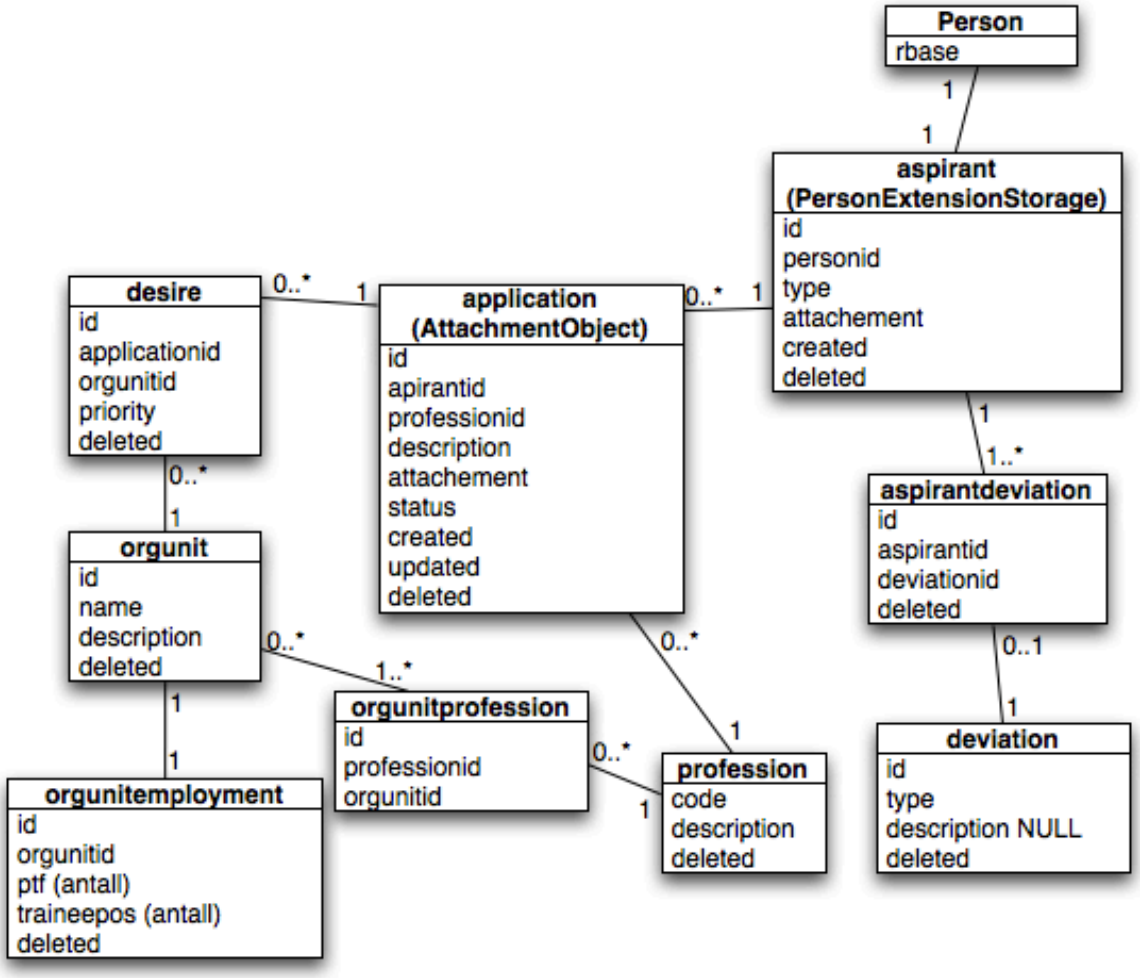
Punktene i foregående kapittel er representert i illustrasjonen under. Fargene representerer de ulike lagdelingene.



Figur 13: Domenemodell

Databasemodell

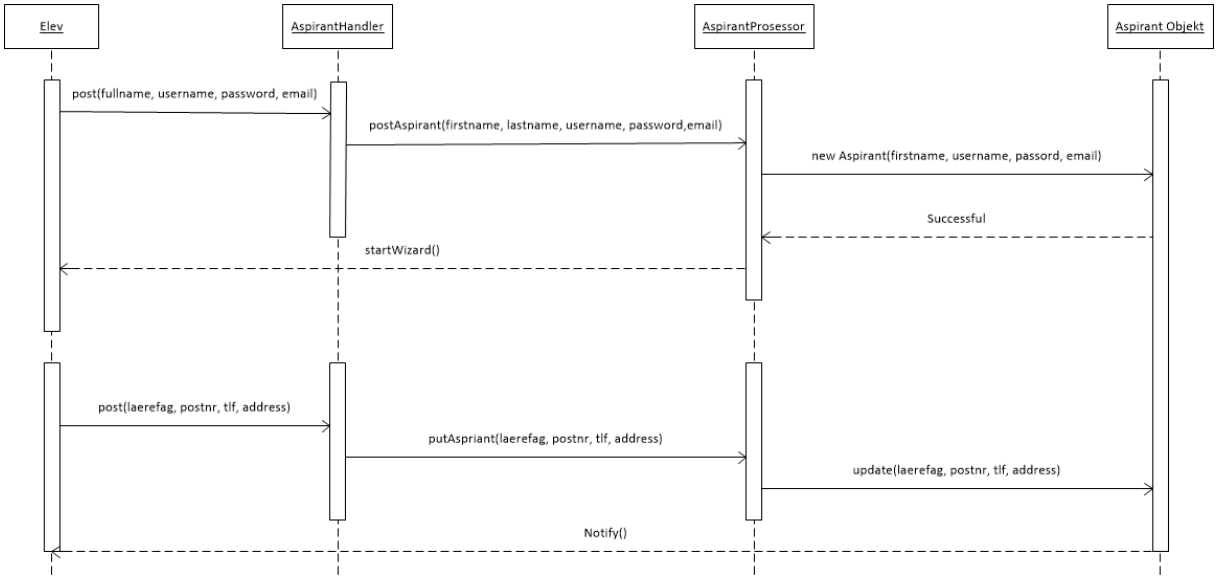
Databasen som beskrevet over er representert i illustrasjonen under.



Figur 14: Databasemodell

Sekvensdiagram

Sekvensdiagrammet nedenfor illustrerer flyten ved registrering av en bruker i Aspirant.



Figur 15: Sekvensdiagram

For at et Aspirant-objekt skal opprettes sjekkes det om det finnes et objekt registrert med samme E-postadresse. Hvis et slikt objekt eksisterer i persontabellen får bruker en feilmelding om dette. Hvis ikke vil et nytt Aspirant-objekt opprettes.

Brukergrensesnitt/GUI

Gjenkjennbart

Aspirant skal som nevnt tidligere fungere som en søknadsmodul til OLKWEB. Det er derfor viktig at brukere av OLKWEB vil kunne kjenne seg igjen i Aspirant.

Bootstrap

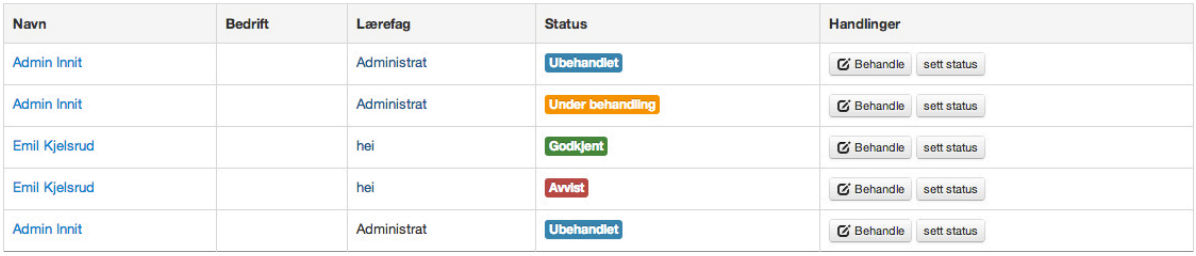
Bootstrap er et CSS- og JavaScript rammeverk med en del ferdige komponenter for å kunne konstruere et web-template. Siden oppdragsgiver allerede benyttet Bootstrap i sine løsninger var det også et naturlig valg å benytte dette i Aspirant. Dette kan også medføre at brukere fra andre løsninger Innit AS tilbyr vil kunne kjenne seg igjen.

Tilgjengelighet

I løsningen vår støtter vi opp under WAI sin WCAG 2.0 (2012) sine prinsipper for innhold på web (Web Content Accessibility Guidelines). Dette gjør at vi ikke ekskluderer brukere men heller tilrettelegger for både hjelpemidler og verktøy brukere eventuell skulle benytte.

Fargevalg

For å øke lesbarheten av tekst har vi valgt å benytte svart tekst på hvit bakgrunn, det finnes noen unntak. Vi har benyttet flere ulike farger for å gi bruker en naturlig oppfattelse av hva vi prøver å utrykke i løsningen. Et eksempel på dette ser vi på status til søknader. Her har vi brukt grønn for godkjent, rødt for avvist, oransje for søknader som er under behandling og blått for søknader som er ubehandlet. På lenker har vi valgt å beholde blå farge, men uten understreking.



Figur 16: Eksempel på bruk av farger (statusindikasjon)

Fontvalg

Vi har valgt å benytte sans-serif fonten Open Sans som er en Open Source Google Font. Den ble opprinnelig laget for både print, web og mobile grensesnitt, men er også kjent for dens lettleselighet. Grunnlaget for valget vårt er at den er luftig, moderne og ryddig. Det andre alternativet vi vurderte er Helvetica/Arial, fonter som dominerer på web, men vi valgte Open Sans da denne har større mellomrom mellom bokstavene, noe som kan gjøre tekstens lesbarhet bedre, ifølge Tim Ahrens (2012). En annen ting som var med på å avgjøre fontvalget var fordelene ved å bruke en webfont kontra en normalfont. Webfonten hentes fra Google Fonts, noe som tilsier at alle brukerne vil oppleve fonten likt uavhengig av hvilket operativsystem som benyttes. Om vi hadde valgt å gå for Helvetica/Arial vil denne kunne oppleves ulikt da den ligger installert på klienten. På slutten av 2012 valgte det populære publiseringssystemet WordPress å velge bort Helvetica/Arial til fordel for Open Sans (2012). Et siste alternativ som ble tatt opp til vurdering var å benytte fonter av typen som er beregnet for dyslektikere, OpenDyslexic(2012) var

en av disse. Dette ble også valgt bort da det ikke fantes noen eller god nok dokumentasjon på at lesbarheten var bedre for mennesker med lese- og skrivevansker.
Open Sans sammenlignet med Arial

Figur 17: Open Sans sammenlignet med Arial

Implementasjon

I dette kapittelet skal vi utrede for hvordan vi har valgt å angripe de ulike praktiske utfordringene. Først og fremst litt om hvordan vi har jobbet, deretter hvilke verktøy og hjelpemidler som har blitt anvendt. I tillegg skal det foreligge en teknisk utredelse av systemet og applikasjonen med fokus på arkitektur, visningstilpassninger, samt gjenbruk av kode. Disse punktene skal eksemplifiseres konkret mot prosjektet der vi beskriver hva som skal løses, hvordan vi har løst det, samt en refleksjon rundt forbedringer for hvert respektive punkt.

Verktøy og arbeidsmetodikk

Et av de mer kritiske aspektene for et vellykket prosjekt innebærer å sette opp et fornuftig og effektivt utviklingsmiljø. Dette miljøet skal som minstekrav muliggjøre for samarbeid mellom flere utviklere, i tillegg til å etablere en stabil og konsistent plattform som utviklerne kan jobbe på. For å oppnå dette har vi anvendt en rekke verktøy og programvare både lokalt og på en sentral server.

Integrated Development Environment

På våre lokale arbeidsstasjoner har vi valgt å bruke NetBeans IDE som er et Open Source verktøy for å produsere kode av ulike slag. På nettsiden deres er det mulig å laste ned ulike versjoner som er optimalisert for de mest populære programmeringsspråkene, Oracle (2013). I følge rammene spesifisert av Innit skal Aspirant bygge på PHP og vi så det derfor hensiktsmessig å laste ned versjonen som er tilpasset denne plattformen. Dette gir oss funksjonalitet som syntaksutheving, StackTrace (feilsøking) og autofullføring ut av boksen.

Versjonskontroll / versjonsstyring

For å forenkle samarbeid og samtidig holde styr på revisjoner, så vi det hensiktsmessig å bruke en form for versjonskontroll. Det finnes i dag flere systemer for å håndtere dette, deriblant Subversion, Git og Mercurial som, i følge Chris Nagele (2013), er de mest populære løsningene. Vi har på gruppen god erfaring med Git fra tidligere prosjekter og ønsket derfor å anvende dette, men siden Innit har en etablert arbeidsflyt med SVN, benyttet vi det i stedet.

I retrospekt ser vi at dette har resultert i en god del utfordringer; det er blant annet spesielt vanskelig å utføre store lokale endringer uten å stadig dytte koden til/fra hverandre, på grunn av SVNs sentraliserte natur. Vi så det også som et problem at hele gruppen var engstelige for å skape sammenslåingskonflikter som vi ikke hadde klart å gjenopprette grunnet manglende kompetanse med det aktuelle styringsverktøyet.

Servermiljø

For å danne en stabil og konsistent utviklingsplattform uavhengig av lokalt arbeidsmiljø, installerte alle medlemmene i prosjektgruppen lokale virtuelle maskiner som speiler spesifikasjonene til produksjonsserveren. På denne måten er vi relativt sikre på at koden ikke vil brekke så snart den rulles ut i produksjon.

Det skulle dog vise seg at skrivebordsmiljøet på den aktuelle Linux-distribusjonen Debian 6 ikke oppførte seg optimalt under den virtuelle plattformen. Med regelmessige systemfrys og dårlig ytelse, valgte vi å droppe skrivebordsmiljøet og satte heller opp en Samba-forbindelse mellom vertsmaskinen og den virtuelle maskinen og eliminerte dermed problematikken.

Arkitektur

Applikasjonens arkitektur er ett av de viktigste punktene vi har måtte ta hensyn til under utviklingen av prosjektet. Problemstillingen tro er vi avhengig av et system som er både tilpasningsdyktig, forutsigbart og oversiktlig. Med dette menes det at programvaren bør struktureres etter gitte konvensjoner, slik at Innits overdragelse vil skje forholdsvis smertefritt. Med forutsigbarhet menes det at annen programvare, slik som OLKweb og OLKskole, skal ha mulighet til å kommunisere med Aspirant gjennom et intuitivt grensesnitt, samt forvente konsekvente maskinlesbare responsmeldinger for alle forespørsler.

For å muliggjøre slik funksjonalitet har vi anvendt en rekke høyt verdsatte prinsipper, som skal forklares i nærmere detalj. Vi ser det ikke hensiktsmessig å legge ved for mye kode i denne rapporten, men noen utdrag vil bli presentert for å illustrere de aktuelle eksemplene.

Representational state transfer (REST)

Nettlesere kan sende forespørsler til serveren på ulike måter. Dette inkluderer GET, POST, PUT, DELETE og OPTION, W3C (2011). Den vanligste metoden er GET som vi alle bruker når vi navigerer oss rundt på nettet ved å trykke på lenker osv. POST brukes når vi skal sende med data til serveren som den er avhengig av for å behandle forespørselen korrekt. Et konkret Use-Case til dette er når man må taste inn brukernavn og passord for å logge inn. PUT, DELETE og OPTION har frem til dags dato blitt forholdsvis lite brukt og nettlesere oppdateres stadig for å utbedre støtte for disse metodene. PUT, DELETE og OPTION skiller seg ikke nevneverdig ut fra POST rent teknisk, men det er her REST kommer inn i bildet.
Det REST-prinsippet prøver å takle er å definere hvilke forespørsler som skal sendes for å utføre ulike handlinger mot ressursen. GET skal hente en ressurs, POST skal opprette en ressurs, PUT skal endre en ressurs og DELETE skal slette en ressurs. OPTION brukes i bakgrunn for å kommunisere med serveren hvilke metoder som er støttet. Hvis brukeren prøver å sende en PUT forespørsel til serveren vil moderne nettlesere først sende en OPTION forespørsel for å korrespondere serveren om klienten har lov til å sende PUT mot denne ressursen. Serveren vil deretter svare med en HTTP statuskode 200 (OK) som forteller nettleseren om den kan sende PUT eller ei. Vi kommer ikke til å gå nærmere inn på tekniske detaljer rundt forespørselmetoder da dette ligger utenfor rammene i dette prosjektet.
I Aspirant anvender vi REST for å forenkle URL'er ytterligere ved å utføre ulike handlinger mot en ressurs, basert på forespørseltypen. Stien /person/1 kan derfor utføre ulike modifikasjoner i

systemet basert på hvilke forespørsel som er sendt. GET vil gi oss personen i retur for visning, PUT vil endre personen med de dataene vi sender med, og DELETE vil slette personen. POST-forespørsler sendes mot lister for å opprette en entitet i denne listen. Dette prinsippet er gjennomgående i hele systemet og gjør det enkelt å både forstå og lære seg hvordan systemet fungerer. Vi vet da med andre ord at en PUT mot /person/1/søknad/43 vil endre søknaden med id 43, i stedet for noe helt vilkårlig som for eksempel en POST forespørsel mot /endre_søknad?id=43, som i dag er vanlig praksis i mange web-systemer.
Sett i lyset av problemstillingen der Aspirant skal integreres med andre løsninger og kommunisere med eksterne systemer, vil et enhetlig, oversiktlig og funksjonelt grensesnitt som dette utgjøre stor forskjell (Apigee 2012, API Design: Third Edition).

Ressursregisteret

Ressursregisteret i databasen er sentral for REST-implementasjonen da dette forteller hvilke URL'er som gir tilgang til spesifikke ressurser, samt hvilke templates som skal assosieres. Her styres også tilgangskontrollen til hver enkelt ressurs slik at vi har kontroll på hvilke brukere som har tilgang til ressursene som blir forespurt. Fordelen med et slik oppsett er at en spesifikk ressurs kan ha ulike templates og tilgang basert på kontekst. Vi kan illustrere dette med et konkret eksempel der vi tar for oss søknader:

```
/person/${personid}/søknad
```

Denne stien fører til en liste over søknader som hører til en person. Her er det kun eieren av søknadene som har tilgang, og visningen er tilpasset til denne konteksten.

```
/søknad
```

Dette er også en liste over søknader, men i motsetning til eksemplet over så hører ikke søknadene til en spesifikk person. Her er det kun opplæringskonsulenter som har tilgang for administrering og visningen er tilpasset slik at søknadenes eiere enkelt kan identifiseres.

Uten å gå for mye i detalj vil vi også nevne at det sentrale ressursregisteret også forteller systemet hvilke Handler (se lagdelingsmodell) som skal håndtere forespørselen. Dette gjør at både /person/${personid}/søknad og /søknad kan benytte samme prosessor og vi kan dermed

gjenbruke koden som henter søknadene fra databasen og behandler de før dataene sendes til det spesifiserte templatet.

MVC

Vi har tidligere snakket litt om programvaredesign-mønsteret MVC, og vil her trekke frem et utdrag fra prosjektet for å illustrere implementasjonen fra et praktisk ståsted. I dette eksemplet skal vi se hvordan vi henter frem en liste med brukere i tillegg til uthenting av enkeltbruker. Først og fremst opprettes tilknytningen til ressursen i databasen slik at forespørselen bli rutet riktig:

Id	url	handler	object	template	accesread	acceswrite
101	/bruker	userHandler	User	index.php	admin, employee	admin, employee
102	/bruker/{user_id}	userHandler	User	single.php	admin, employee, me	admin, employee, me

Figur 18: Ressurstabell

Vi ser øyeblikkelig at vi starter på id #101 i ressurstabellen. Id #0 - #100 er reservert for å unngå konflikt med interne ruter i RBASE. Url-feltet definerer hvilke adresse ressursen skal registreres mot. I dette eksemplet vil listen med brukere være tilgjengelig på /bruker og dataene for en enkelt bruker vil ligge på /bruker/{brukerens unike id}. Feltene Handler, Object og Template viser til filene i filsystemet som skal håndtere denne forespørselen på hvert respektive lag - i dette tilfellet vil userHandler og User-objektet håndterer begge forespørslene, mens template vil variere slik at vi kan tilpasse visningene uavhengig av hverandre. Accessread og Accesswrite styrer hvilke roller som har tilgang til denne ressursen, dog vil vi ikke eksemplifisere aksesskontroll i denne rapporten.

Etter at ressursene er registrert, vil neste naturlige steg være å definere handleren slik at vi kan navigere oss inn på adressen og begynne å teste koden vi skriver underveis. Kodeblokken under avsnittet viser hvordan en slik handler kan se ut.

```
#/handler/userHandler.php
class UserHandler extends Handler {

      function get($request, $response) {
            //Henter processor
```

```
            $p = new UserProcessor($request);

            //Hvis forespurt enkeltbruker
            if(isset($request->param("user_id")))
                  return $p->getUser($response);

            //Ellers forespurt listevisning
            return $p->getUserList($response);
      }
      function put($request, $response) {//behandle put requests}
      function post($request, $response) {//behandle post requests}
      function delete($request, $response) {//behandle delete
requests}
}
```

Det er umiddelbart åpenbart at ved å utvide RBASE-klassen Handler som inneholder en mengde hjelpefunksjoner, vil vi med enkelhet kunne opprette en handler som tar seg av brukerhåndtering uten store utfordringer. På denne måten har vil tilgang til ressursparametere og annen relevant informasjon i forespørselen som kan være nyttig for hvordan vi behandler den. Videre ser vi at vi oppretter en ny Processor-instans, og returnerer deretter prosessoren sin getUser eller getUserList avhengig av forespørselen. På dette tidspunktet er det naturlig å definere prosessoren som følger:

```
#/processor/UserProcessor.php
class UserProcessor extends Processor {
      function __construct($request) {
            $this->request = $request;
            $this->storage = new UserStorage();
      }

      function getUser($response) {
            $user_id = $this->request->param("user_id");
            $user = $this->storage->getById($user_id);
            $response->setObject($user);
            return $response;
      }

      function getUserList($response) {
            $users = $this->storage->getIndex();
            $response->addObject($users);
            return $response;
      }
}
```

I dette Use Case'et har prosessoren definert funksjoner for å hente både liste med brukere, samt enkeltbrukere fra Storage-klassen. Poenget med prosessoren er å holde datasamling og -manipulasjon på ett sted, slik at vi enkelt kan gå inn å justere disse funksjonene etterhvert som

applikasjonen vokser. Det vil for eksempel være aktuelt å knytte en indirekte kobling til brukerens søknader i getUser funksjonen. Selve uthentingen av dataene fra databasen skjer i Storage-objektet som vi har kalt UserStorage:

```
#/data/UserStorage.php
class UserStorage extends Storage {

      function getById($id) {
            return parent::getById($id);
      }

      function getIndex() {
            return parent::getIndex();
      }
}
```

UserStorage trenger ikke mye kode da superklassen Storage innehar all funksjonalitet som foreløpig er nødvendig. Det er i UserStorage vi knytter direkte koblinger mellom tabeller ved hjelp av egendefinerte SQL setninger og/eller innebyggede hjelpefunksjoner. For at superklassen Storage skal vite hvilke tabeller i databasen som er aktuelle, er vi avhengig av å definere et data-objekt som representerer datasettet i databasen. I ressursregisteret vårt har vi spesifisert User som utgangspunkt for datamodellen. Data-objekter defineres ofte som en utvidelse av klassen DataObject, som er en generisk innpakning for enkle datasett uten relasjoner. RBASE gir også mulighet for å utvide ytterliggere klasser som eksempelvis AttachmentObject som igjen åpner for enklere filbehandling av binære data. Dette vil dog ikke være nødvendig for den enkle brukerklassen per dette eksemplet:

```
#/data/User.php
class User extends DataObject {
      public $firstname;
      public $lastname;
      public $email;
      …

      function getFirstname() {
            return $this->firstname;
      }

      function setFirstname($value) {
            $this->firstname = $value;
            return $this;
      }
}
```

Dette dataobjektet har et par *public* variabler som samsvarer med feltene i databasetabellen den representerer. Det er også definert "gettere" og "settere" for disse attributtene, som gjør at vi kan lenke sammen konstruksjon av nye ressurser på følgende måte:

```
$user = new User();
$user -> setFirstname("Ola")
      -> setLastname("Normann")
      -> setEmail("ola@normann.no");

$storage->save($user);
```

De overnevnte klassene er minstekravet for å lage en data-sentrert applikasjon med RBASE. Vi har ikke definert noen visning for forespørselen, men dataene vil være tilgjengelige i maskinlesbare formater på url'ene /bruker.json og /bruker/#id.json.

Levende informasjon og visningstilpassning

I moderne web-applikasjoner kan man styrke interaksjonsmønsteret ved å utvide grensesnittet med levende oppdateringer, Linda Dailey Paulson (2005). Teknologien ofte referert til som AJAX, Asynchronous JavaScript and XML, muliggjør at brukeren kan kommunisere med serveren uten å måtte laste hele websiden på nytt. Vi har utnyttet denne teknologien på blant annet forsiden der brukeren har mulighet til å spesifisere lærefag og lokasjon, for å få en indikasjon på hvor mange bedrifter som er aktuelle for lærlingen.

Denne implementasjonen anvender JavaScript-biblioteket jQuery for å hente informasjonen fra serveren, og deretter formattere dataene for visning i et dedikert vindu som popper opp så snart dataene er klare.

```
$.get("http://aspirant.dev/available.json",
  {
    profession: $("input.profession").val(),
    county: $("input#county").val()
  }, render(data));
```

Som vi kan se i forespørselen utnytter vi JSON-representasjonen av dataene, slik at vi med enkelhet kan behandle resultatet i JavaScript uten å måtte parse dataene manuelt. Vi sender med parameterne *profession* og *county* som serveren er avhengig av for å filtrere søket. Deretter behandles dataene i en egendefinert render funksjon som tar seg av konstruksjonen av visningen.

Denne implementasjonen fungerer bra siden vi henter data med vanlig get forespørsel, men i et større perspektiv kan dette forbedres ytterligere. På bakgrunn av REST-konvensjonen vet vi at alle forespørsler og responser forekommer i et forutsigbart format og det vil derfor være mulig å abstraktere mye av denne koden. I fremtiden ser vi det fordelaktig at det utvikles et fullverdig front-end bibliotek for RBASE som vil åpne for utvikling av slike løsninger på en enklere og ikke minst mer konsekvent måte. APIet til biblioteket kan eksempelvis se slik ut:

```
config = {
  baseUrl: "http://aspirant.dev",
  format: "json",
}

rbase = new Rbase(config);

//hente ressurs
rbase.get("ressurs", {}, "#template-id");

//opprette ressurs
ressurs = rbase.create("ressurstype", data);
ressurs.save();

//oppdatere ressursen
ressurs.set("attributt", "verdi");
ressurs.save(function(response) {
  if(response.status == rbase.statuscode.OK)
    console.log(response.object) //her kan vi behandle resultatet
});

//slette ressursen
ressurs.delete();
```

Det er åpenbart at et slikt oppsett vil gjøre utvikling av tyngre ajax-applikasjoner mye enklere. Det er blant annet mulig å bytte *baseUrl* fra testserveren til produksjonsserveren globalt, i stedet for å måtte navigere gjennom hele strukturen og bytte flere steder. Responsen kan også kobles opp mot et JavaScript template-system (som f.eks. jQuery-tmpl, Underscore eller Handlebars) slik at vi kan gjenbruke visninger og samtidig kompilere de i hurtigbuffer for bedre ytelse. Funksjonen save() bør også optimalt sett håndtere unntak som manglende internettforbindelse ved å lagre forespørselen i en kø for så å synkronisere med serveren så snart forbindelsen er oppe igjen.

Templatesystemet

Som nevnt i tidligere kapitler, så håndterer rammeverket og ressursregisteret hvilke template/visning som skal benyttes på hver enkelt ressurs. Dette fungerer slik at referansen til den aktuelle template-filen defineres i ressursregisteret og vil dermed automatisk bli returnert til brukerens nettleser. Til tross for at template-filer ikke er annet enn vanlige PHP-filer, gjør de en meget viktig oppgave i applikasjonen - de separerer logikk og data fra selve visningen. Det vil derfor være utrolig enkelt å endre applikasjonens utseende, uten å røre internmekanismer.

Visningsfilene har også to innebyggede modi: Standard- og widgetvisning. Standardvisningen er ment for statisk representasjon av dataene, mens widgetvisningen er ment for å kunne bli inkludert som et enkeltstående element i en større komposisjon. Dette gir oss mulighet til å la visningene representere ulik data som ikke nødvendigvis assosieres med hverandre. I Aspirant benytter vi oss av denne funksjonaliteten blant annet på brukerprofilen, der søknadene inkluderes dynamisk.

Templatesystemet fungerer ganske godt, og vi hadde ingen store problemer med å anvende dette i prosjektet. Dog fant vi det noe kløne på bakgrunn av dets rigide struktur. Ved tilfeller der man trenger å tilpasse visningen etter brukerens rolle eller andre dynamiske parametere, feiler templatesystemet å separere logikk og presentasjon. Dette resulterer i at flere template-filer inneholder tester på tilgangsrettigheter og andre uforutsigbare variabler.

En mulig løsning på dette vil være å definere visningen i handler fremfor ressursregisteret. Ved å gjøre dette vil vi kunne teste på de ulike kriteriene før vi definerer hvilke visning som skal returneres, og dermed slippe slik testing i selve template-filene.

Kodestandard og gjenbruk

I prosjekter av signifikant størrelse er det viktig å produsere effektiv kildekode, spesifikt hvordan vi kan praktisere prinsipper som DRY "Don't Repeat Yourself" (Andrew Hunt og David Thomas, The Pragmatic Programmer s. 30) og KISS "Keep It Simple, Stupid". Denne praksisen har flere unike fordeler ved utvikling av programvare, både på forretningsnivå og teknisk nivå. Gjenbruk er for det første billig for tjenestetilbyder ved å unngå å finne opp hjulet på nytt, eller starte på bar bakke hver gang et nytt prosjekt settes i gang. Det er også lettere for utviklere å jobbe effektivt når han/hun kan utnytte eksisterende biblioteker og verktøy for å utføre oppgaver

som ellers ville ha kostet tid og ressurser å bygge fra bunn. Gjenbruk krever også til en viss grad bevisstgjøring av struktur og standarder for å frikoble kode slik at de ulike verktøyene kan fungere uavhengig av hverandre. Denne bevisstgjøringen vil konsekvensielt forene utviklere i en felles forståelse av hvordan kode skal bygges og danner en plattform der samarbeid kan blomstre.

Prosjektet med Innit har gitt oss en unik innsikt i hvordan vi med enkle prinsipper og arkitekturløsninger kan bygge programvare med kommersiell standard. Vi har tidligere i rapporten demonstrert at vi har hatt stort fokus på lagseparasjon mellom applikasjon, data og presentasjon. I tillegg til dette har vi taktisk utnyttet arveegenskapene til programmeringsspråkene som er anvendt for å maksimere gjenbruk også på objekt- og klassenivå. Et konkret eksempel på dette er implementasjon av en global handler med autentisering- og rettighetskalkulasjoner som alle handlere arver fra.

Til tross for denne innsatsen er det fortsatt rom for forbedringer. For å kunne utnytte gjenbruk og kodeintegrasjon i et bredere perspektiv er vi avhengig av å følge en standard som er akseptert blant utviklere over hele verden. Dette resulterer i blant annet triviell integrering av tredjepart-kode. Det er naivt å tro at alle utviklere blir enige om å bygge kode rundt èn konvensjon, dog har det blitt gjort store fremskritt i denne forbindelse. Open Source prosjektet Composer [referanse], laget av Nils Adermann, Jordi Boggiano, har blitt mer eller mindre akseptert som industristandard for håndtering av kodepakker, moduler og klasser i PHP på lik linje som NPM for node.js og Gems for Ruby [referanse]. Composer forutsetter at kode er strukturert etter PSR-0 standarden (PHP-fig 2012) definert av PHP Framework Interop Group slik at alle moduler kan sømløst integreres med det aktuelle prosjektet. Arbeidsflyten under utviklingen kan også forbedres ved å anvende Composer til å installere pakker som håndterer automatisert testing (unit og spec-testing med PHPUnit og PHPSpec), samt verktøy for distribusjon/utrulling og vedlikehold - noe vi har savnet under selve utviklingen.

Test av løsning

Unit testing

Unit testing går enkelt ut på å kvalitetssikre enheter i applikasjonen i isolasjon. Dette vil si at hver enkelt funksjon skal kunne kvalitetssikres med automatiserte tester uavhengig av hverandre (PHPUnit, 2013).

Under utviklingen av dette prosjektet har det vært problematisk å gjennomføre enhetstesting på grunn av hvordan internrammeverket rent teknisk er bygget opp. For å muliggjøre dette er vi avhengig av å implementere en form for Factory-pattern for å konstruere objektene vi jobber med, ikke direkte instansiering av klasser slik det blir gjort i dag. En annen løsning vil være å utnytte et system for avhengighetsinjeksjon der vi kunne ha byttet ut de reelle objektene med mockups i et dedikert unit-testmiljø (PHPUnit).

Brukertesting

Aspirant sin brukergruppe har varierende datakunnskaper. Vi har derfor valgt å designe en brukertest som imiterer disse forholdene, ved å lage et scenario med oppgaver som to testsubjekter med ulik bakgrunn må gjennomføre individuelt.

Scenario

Eleven skal søke lærlingplass.

Oppgaver:

- Registrer deg som bruker, men stop når du kommer til det siste steget i veiviseren. Vi vil si ifra når du når dette steget. Var denne prosessen enkel og var den som du forventet?
- Finn en bedrift og søk lærlingplass, men stop når du skal legge inn CV. Vi vil si ifra når du når dette steget. Hvordan opplevde du denne prosessen? Er det noe informasjon du synes mangler/eller du ønsker å se?
- Gjennomfør søknadsprosessen. Hvordan opplevde du denne prosessen og hva synes du om søknadsprosessen i sin helhet?

Andre spørsmål:

Når du kommer inn på forsiden av nettsiden, hva er det første du tenker på?

Test 1

Den første testpersonen vi oppsøkte var en medelev. Det ble testet både elev- og opplæringskonsulentrollen. Vi ønsket å teste en medelev fordi vedkommende har tilsvarende bakgrunn som vi har og kan legge merke til ting vi har glemt eller oversett. Det vi først ble gjort oppmerksomme på når testpersonen kom inn på forsiden var at den opplevdes som pen og oversiktlig. Det ble også nevnt at det manglet en god del informasjon som skal introdusere tjenesten og redegjøre for hva denne tjenesten kan tilby brukeren.

Testesubjektet startet så med å registrere seg som bruker, og trykket på nedtrekkslistene for å velge fylke og lærefag, som illustrert tidligere i rapporten. Etter at modalvinduet dukket opp, med resultatene for kriteriene som angitt, trykket vedkommende på den pilen med sirkel rundt som peker mot der man kan registrere seg. Testsubjektet sa at dette kan lett misforstås og det var forvirrende at modalvinduet ikke kan lukkes hvis man har lyst til å velge et annet fylke eller lærefag. Vedkommende gjennomførte resten av søknadsprosessen og synes at denne var enkel, oversiktlig og logisk.

Neste steg var å teste konsulentrollen, spesifikt hvordan man fikk oversikt over søknader og hvordan endret status på disse etter eventuell behandling. Faneløsningen for å sortere på søknadsstatus fungerte bra, men ytterligere filtre var ikke implementert på nåværende tidspunkt og ble derfor ikke testet.

Test 2

For å dekke brukertest nummer to oppsøkte vi en elev som er midt i søknadsprosessen om lærlingplass. Denne avgjørelsen ble tatt på det grunnlag at eleven vil med stor presisjon kunne representere den virkelige målgruppen til applikasjonen. Testsubjektet hadde lavnivå datakunnskaper, men hadde til gjengjeld erfaring med bruk av samordna opptak.

Tilbakemeldingene vi fikk var overraskende gode, men testsubjektet opplevde noe utfordring rundt oversikt over vedlegg. Mer spesifikk opplevde eleven at det var vanskelig å se om en CV var lastet opp. Testsubjektet stilte spørsmål rundt valg av terminologi, spesielt med tanke på at CV ikke er noe som bli brukt i skolesammenheng. Oppsummert opplevde ikke eleven at søknadsprosessen var verken tung eller vanskelig, men heller enkel og moderne.

Refleksjon av brukertest

Vi er enige med testperson 1 i at pilen som henviser til registreringsskjema lignet på et interaktivt element og vi har derfor bestemt at vi fjerner sirkelen rundt pilen for å fjerne indikasjonen på at dette er en knapp man kan trykke på. Et alternativ vil være å sette klikk-funksjonalitet på pilen som markerer første felt i skjemaet. I tillegg ble enige om at modal-vinduet bør kunne lukkes slik at brukeren kan angi lærefag eller område på nytt ved eventuelle feil. Når det gjelder manglende informasjon på forsiden, så har vi tatt det opp med Innit og vi er i gang med å produsere en fullstendig introduksjon som skal eliminere eventuelle uklarheter rundt tjenesten og hva som tilbys.

Etter å ha diskutert resultatene fra test nummer to, ble vi enige om at et bedre alternativ til CV ville vært å benytte ordet vitnemål da de fleste elevene i klassen til testsubjektet hadde større forhold til denne benevnelsen.

Test med oppdragsgiver

Det ble i tillegg gjennomført en funksjonell test med oppdragsgiver der vi fikk tilbakemeldinger og tips på implementeringer av funksjonalitet. Blant tilbakemeldingene i søknadsprosessen ble vi gjort oppmerksom på at postnummer kan hentes ut fra RBASE og poststed kunne dermed hentes inn automatisk avhengig av hvilke postnummer som ble spesifisert. Vi kunne også forenkle behandling av binære data (vitnemål og andre vedlegg) ved å benytte oss av hjelpeklasser som håndterer opplasting og lagring. I tillegg til dette hadde vi en diskusjon angående bruk av ikoner og tekst, der oppdragsgiver mente det var redundant med både tekst og ikoner for å beskrive funksjoner på knapper og andre interaktive elementer. Integrasjonstester med reelle data fra OLKweb har blitt vellykket gjennomført for å bekrefte systemets samarbeid med de eksisterende løsningene.

Kvalitetssikring

Scrum - Quality gates

Som nevnt i introduksjonen benyttet vi oss av Quality gates for å definere når et element i produkt backlog er "ferdig". Vi synes det å benytte seg av disse portene var en bra ting, da vi enkelt var sikre på at de elementene som kom gjennom alle portene kunne defineres som ferdig. Dette sparte oss for mye tid og unødvendig hodebry.

Kvalitetssikring gjennom møtevirksomhet

Etter hver Sprint hadde vi møter med Kim og Simen der vi presenterte hva vi hadde fått til i løpet av Sprinten. De gav oss mye konstruktiv kritikk som vi opprettet som nye produktelementer i produktkøen. Det å være utplassert hos Innit en gang i uken synes vi var en fin måte å jobbe på, da vi alltid hadde muligheten til å spørre om hjelp hvis det var noe vi lurte på. Vi lærte også mye om hvordan en utviklerbedrift jobber med kunder og hvordan teamet jobber internt med ulike prosjekter, noe vi synes var veldig lærerikt.

Evaluering av arbeidsprosess og utviklingsmodell

Den største utfordringen vi støtte på i bruken av Scrum-prinsipper var å samkjøre arbeidsprosessen med hvordan oppdragsgiver ønsket at vi skulle jobbe. I stedet for å benytte sprinter på to uker med utvalgte produktkøelementer, ønsket oppdragsgiver at vi delte prosjektet inn i fire store deler med påfølgende tidsfrister. Dette strider imot prinsippene for en smidig utvikling. Måten vi løste dette på var at oppdragsgivers ønsker om ferdigstilt produkt til bestemt tid ble avgjørende for hvor høy prioritering produktkøelementene fikk, altså ikke helt slik vi hadde sett for oss i starten av prosjektet. Det var også utfordrende å involvere produkteier i like stor grad som ønskelig da denne rollen aldri ble ordentlig utfylt, selv om oppdragsgiver var tilgjengelig.

OnTime Scrum ble tidlig valgt bort da dette opplevdes for omfattende og komplisert. Avgjørelsen falt da oppdragsgiver ytret ønske om bruk av Wunderlist som et gyldig alternativ. Å bruke dette verktøyet fullt ut ville vært alt for ressurskrevende og kunne ha påvirket resultatet i en negativ retning.

Internt i utviklingsgruppen opplevde vi fordeling av arbeidsoppgaver som noe utfordrende da det var vanskelig å forutse både ressursbruk og tidsestimering uten praktisk erfaring med prosjekter av denne størrelse. Det at vi ofte måtte tilbringe arbeidstiden i skolens fellesområde medførte at kommunikasjon og konsentrasjon ble redusert, noe som igjen førte til at dagene i oppdragsgivers lokaler føltes mer produktive. Disse dagene fungerte også de daglige Scrum møtene mye bedre enn da vi avholdt disse i skolens lokaler eller over kommunikasjonstjenesten Skype. Noen dager valgte vi å jobbe hver for oss, da dette ofte var enklere fordi timeplanen vår var ulik. Disse dagene var også de som opplevdes som minst produktive. Parprogrammeringen fungerte som

forventet og medførte at utviklingsgruppen opplevde et kollektivt eierskap og en felles forståelse av produktet.

Ved å trekke inn RUP-artefaktet kravspesifikasjon fikk vi et effektivt hjelpemiddel for å rette oppmerksomhet på hva som skulle utvikles. Spesielt med tanke på den mindre utfylte produkteierrollen.

Oppsummert er gruppen enig om at bruken av Scrum i kombinasjon med RUP-artefakter og parprogrammering kunne fungert bedre om gruppens medlemmer hadde vært samlet under ett og samme tak til hver samling. Alle opplevde til tider usikkerhet rundt arbeidsoppgaver og avgjørelser, spesielt de dagene vi jobbet individuelt.

Kritikk av oppgaven

Oppgaven var litt utradisjonell utover det som er normalt for en mediebacheloroppgave. Problemstillingen i seg selv var veldig spennende å jobbe med, men i retrospekt ser vi at prosjektets omfang ble noe mer omfattende enn først antatt. Som nevnt i Kravspesifikasjonen var det ikke definert på forhånd hvilken funksjonalitet som skulle ferdigstilles i løpet av prosjektperioden. Dette førte til mye uklarheter rundt hva som faktisk skulle være ferdig på sluttdatoen, og gjorde dermed at planleggingen ikke ble optimal.

Videre arbeid

Til tross for at vi har fått gjort mye på den tiden vi har hatt disponibel, så kan produktet forbedres på flere punkter.

I forhold til utseende og brukergrensesnitt er det flere punkter som kan forbedre eller utvide Aspirant. Vi var lenge inne på tanken om å opprette et kart slik at brukere av Aspirant kunne få en helhetlig oversikt over hvor de ulike bedriftene hadde sin geografiske plassering, her var det også muligheter for å lage et fylkeskart som hadde som mål å vise antall plasser i hvert fylke. Vi måtte dessverre velge bort begge alternativer da det ikke ble prioritert.

Aspirant fortjener en grafisk profil, med dette tenker vi på en logo som kan kjennes igjen samt en fargeprofil som kan skape en identitet for seg selv samtidig som den er del av noe større. Det vil

etterhvert også være behov for en egen meny, eventuelt et alternativ til navigering mellom de ulike sidene i Aspirant.

Videre bør det også fokuseres på å gjøre Aspirant tilgjengelig på mobil- og nettbrettplattformen. Det bør også vurderes å legge til et alternativ i fontvalg, for eksempel å bytte til en fonttype beregnet for dyslektikere som nevnt tidligere. En forstørring eller forminskning av innhold er også noe som bør vurderes, det skal dog nevnes at de fleste moderne nettlesere har en egen zoom-funksjon for dette.

Som nevnt i kapitlene om implementering, så kan vi forbedre systemet på flere områder. Vi har sett at AJAX-kall og utvikling av dynamiske elementer kan forbedres ved å lage et klientsiderammeverk i JavaScript som kommuniserer med serveren gjennom det grensesnittet som er fastsatt. Dette muliggjøres med den strenge REST-implementasjonen.

I tillegg til dette har vi forbedringspotensialer på serversiden, blant annet med en bedre ORM implementasjon. Det er i dag noe tungvint å sette sammen tabeller med egendefinerte SQL setninger gjennom tilsynelatende vilkårlige hjelpefunksjoner i RBASE. Et alternativ kan være å utnytte en form for relasjonsdefinisjon i konstruktøren til DataObject instansene. Med dette så menes det at relasjoner bør kunne defineres på en lesbar måte:

```
$object->hasMany('ressurs', $options)
$object->hasOne('ressurs', $options)
$object->belongsTo('ressurs', $options)
```

der man med $options vil kunne sende med parametere som sier hvilke primærnøkler og fremmednøkler ORM motoren skal lete etter for å sette sammen tabellene. Det bør også være mulig å spesifisere hvilke type *join* som skal anvendes, for mest mulig fleksibilitet. Dette er dog noe som må endres i kjernen i RBASE og vil måtte tas opp med utviklerne hos Innit for å gjennomføre. Det kan også være at det å utvikle dette fra bunn vil være lite hensiktsmessig da det allerede finnes frittstående Open Source ORM implementasjoner med høy kvalitet slik som Eloquent (http://laravel.com/docs/database/eloquent).

Som et resultat av brukertestingen skal forsiden i fremtiden fylles med utdypende informasjon og terminologien endres fra CV til vitnemål. Det skal også utformes og implementeres gode tilbakemeldinger i systemet for å eliminere usikkerhet hos brukeren.

Konklusjon

I løpet av det siste semesteret har vi jobbet med bacheloroppgaven som en avslutning på tre års utdanning på Høgskolen i Gjøvik, og resultatet av dette arbeidet kan vi se oss meget fornøyd med. I denne rapporten har vi argumentert for de valgene som har blitt gjort, og vi har på systematisk vis jobbet mot å tilfredsstille problemstillingen underveis.

Det er på nåværende tidspunkt vanskelig å bekrefte at resultatet vil fungere som en effektiv inngangsport til OLKweb som problemstillingen sikter mot, da dette krever empiriske data gjennom en hel søknadsperiode. Til tross for dette har vi etter beste evne utviklet programvaren slik at den enkelt kan integreres med de eksisterende løsningene til oppdragsgiver. Konsekvente maskinelle grensesnitt for datautveksling, gjennomgående konvensjoner for både kodestruktur og menneskelige grensesnitt er tiltak som har blitt gjennomført for å rette prosjektet mot dette målet. Tilpasningsdyktigheten har blitt bekreftet gjennom tester med oppdragsgiver.

Gjennom hele prosjektperioden har vi fått anvendt tilegnet kunnskap gjennom studiet. I tillegg har vi også lært mye om hvordan vi jobber med eksterne oppdragsgivere og kunder, arkitektoniske valg med hensyn til integrasjon, samt hvordan vi dokumenterer og styrer et prosjekt med betraktelig størrelse over lengre tid.

Selv om denne perioden er overstått, vil produktet videreutvikles av oppdragsgiver og vil med tiden avdekke prosjektets suksessrate. Uansett har vi levert et produkt vi er stolte av.

Litteraturliste

Bergersen, Thomas. 7. februar 2013. Ringsaker VGS.

Innit AS(2013) Om Innit AS [online]
url: http://www.innit.no/om_innit (15.5.2013)

OLKweb(2013) Om OLKweb [online]
url: http://www1.olkweb.no/om-olkweb/ (15.5.2013)

Williams, Brown, Meltzer, Nagappan (2013) Scrum + Engineering Practices Experiences of Three Microsoft Teams[online] url:

http://dl.acm.org/citation.cfm?id=2082758.2083397&coll=DL&dl=ACM&CFID=157489638&CFTOKEN=50436891 (15.05.2013)

Scrum (2013), *Scrum.org Improving the Profession of Software Development* [online]
url: http://www.scrum.org/ (15.05.2013)

IBM RUP (2013), *RUP: Best practices for design, implementation and effective project management* [online]
url: http://www-01.ibm.com/software/rational/rup/ (15.05.2013)

Wunderlist (2013), *Wunderlist 2: Your beautiful and simple online to-do list app* [online]
url: https://www.wunderlist.com/#/login (15.05.2013)

Modernizr (2013), *Modernizer: the feature detection library for HTML5/CSS3* [online]
url: http://modernizr.com/ (15.05.2013)

Web Accessibility Initiative (2012) *How to meet WCAG 2.0* [online].
url: http://www.w3.org/WAI/WCAG20/quickref/ (15.05. 2013)

Tim Ahrens (2012) *A Closer Look At Font Rendering* [online].
url: http://www.smashingmagazine.com/2012/04/24/a-closer-look-at-font-rendering/
(1505.2013)

Wordpress (2012) *Open Sans, how do we love thee?* [online].
url:http://en.blog.wordpress.com/2012/10/09/open-sans-how-do-we-love-thee-let-us-count-the-ways/ (15.05. 2013)

Selena Ross (2012) *New Font Helps Dyslexics Read Clearly* [online].
url: http://thebottomline.as.ucsb.edu/2012/10/new-font-helps-dyslexics-read-clearly
(15.05.2013)

PHPUnit (2013) *Chapter 4: Writing Tests for PHPUnit* [online]
url: http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html (15.05.2013)

Oracle (2013) *NetBeans IDE Download* [online]

url: https://netbeans.org/downloads/index.html (15.04.2013)

Chris Nagele (2013) *An introduction to version control* [online]
url: http://guides.beanstalkapp.com/version-control/intro-to-version-control.html (15.04.2013)

World Wide Web Consortium (2011) *REST* [online]
url: http://www.w3.org/2001/sw/wiki/REST (14.05.2013)

Line Dailey Paulson (2005) *Building Rich Web Applications with Ajax* [online]
url: http://userpages.umbc.edu/~dhood2/courses/cmsc433/fall2005/Miscellaneous/ajax.pdf
(14.05.2013)

The jQuery Foundation (2013) jQuery [online] url: http://jquery.com (15.05.2013)

DocumentCloud (2013) *Underscore.js* [online] url: http://underscorejs.org (14.05.2013)

Wycats (2013) *Handlebars.js* [online] url: http://handlebarsjs.com (14.05.2013)

Andrew Hunt og David Thomas (1999) *The Pragmatic Programmer: From Journeyman to Master*. Utgivelsessted: Addison-Wesley Professional.

Apigee (2013) *API Design: Third Edition* [online]
url: http://apigee.com/about/api-best-practices/api-design-third-edition (14.05.2013)

Composer (2013) *Introduction* [online] url:
http://getcomposer.org/doc/00-intro.md (15.05.2013)

Node NPM (2013) *Node Package Modules* [online]
url: https://npmjs.org (15.05.2013)

Ruby Gems (2013) *Your Community Gem Host* [online]
url: http://rubygems.org/ (15.05.2013)

PHP Framework Group (2013) *Welcome PHP Developers* [online]

url: http://www.php-fig.org/ (20.04.2013)

PHP Framework Group (2012) *fig-standards* [online]
url: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md (20.04.2013)

Vedlegg

Terminologi

- Åpne standarder - Standarder som sikrer at spesifikasjonen kan implementeres på en avgiftsfri måte
- Innstikk - Tredjepartsprogramvare som kan integreres i et system
- Kryssplattformkompatibel - Programvare som skal fungere på tvers av ulike plattformer
- Rammeverk - Et sett med verktøy og retningslinjer som fordrer en konvensjon for utvikling
- E-learning - Systemer for læring på elektronisk plattform
- Modul - Programvare som hører til en større kontekst
- Open Source (Åpen kildekode) - Utviklingsmodell der filosofien dikterer at kilden skal ha universell tilgang med lisenser uten kommersielle restriksjoner
- Sprint - Er en tidsperiode avgrenset med en gitt lengde
- Use Case
- Product Backlog - Er en liste med produkter som utviklingsteamet må forholde seg til
- WCAG 2.0 - Retningslinjer for tilgjengelig webinnhold
- Samordna Opptak - administrerer opptaket av nye studenter til norske universiteter og høgskoler
- IDE (Integrated Development Environment) - Integrert utviklingsmiljø som hjelper utviklere med å skape og feilsøke kode.
- Linux-distribusjon - et medlem av Unix-familien av operativsystemer som er bygget på toppen av Linux-kjernen.

Arbeidslogg

Dato	Hva	Timer	Sted	Sum Timer
09.01.2013	Møte med oppdragsgiver og veileder	4	HiG	561
10.01.2013	Jobbet med forprosjekt	6	HiG	
11.01.2013	Lynkurs og møte med oppdragsgiver	6	HiG	
13.01.2013	Skrive forprosjekt	8	HiG	
15.01.2013	Møte med veileder + jobbing med forprosjekt	8	HiG	
16.01.2013	Jobbet med forprosjekt	7	Frusethenga	
17.01.2013	Framstilt problemstilling	6	HiG	
18.01.2013	Klargjøre intervju	7	HiG	
21.01.2013	Arbeidet med forprosjekt	9	HiG	
22.01.2013	Intervjugjennomgang og forberedelse	8	HiG	
24.01.2013	Definere mål for intervju og forprosjekt	7	HiG	
25.01.2013	Workshop i PubSub og JS	7	Brygga 20, Innit	
27.01.2013	Omstrukturering og omskriving	9	Forprosjekt, HiG	
28.01.2013	Felles lesning, forprosjekt	7	HiG	
30.01.2013	Sprint gjennomgang, oversikt	8	HiG	
31.01.2013	Kravspek	9	HiG	
07.02.2013	Intervju og møte med Ringsaker VGS	6	Thomas Bergersen	
08.02.2013	Møte med OLK - Oppland	5	Marianne Stepperud	
10.02.2013	Søndagsarbeid, rapportplanlegging	6	HiG	
11.02.2013	Sprint Retrospective Meeting og arbeid	8	HiG	
13.02.2013	Database, felles multiplum	8	HiG	
14.02.2013	Sette opp utviklingsmiljø	8	Brygga 20, Innit	
15.02.2013	Fikset problem med miljø, XDebug	7	HiG	
18.02.2013	Emil hjalp Ivan med RBase, Søknadsskjema	7	HiG	
20.02.2013	Sette opp Database	8	HiG	
21.02.2013	Statusoppdatering, diskusjon rundt valg i utviklingsfase	4	Brygga 20, Innit	
22.02.2013	Felles jobbing på Bachelornettsiden, ferdigstilt	10	HiG	
25.02.2013	Sprint Review og Retrospective, arbeid	8	HiG	
26.02.2013	Møte med veilder + arbeid	6	HiG	
28.02.2013	Sprint Planning Meeting og start på Kravspek del 2	8	Brygga 20, Innit	
01.03.2013	Startet på GUI, Rapportskriving	8	HiG	
04.03.2013	Rapportskriving, og jobbet med innlogging	9	HiG	
06.03.2013	Rapportskriving, og googlet PHP-feilmelding med retting	8	HiG	
07.03.2013	Jobbet med design	8	Brygga 20, Innit	
08.03.2013	Implementering av dataobjekter	8	Brygga 20, Innit	
11.03.2013	Utarbeidet datautveksling mot objektene (storage)	6	HiG	
12.03.2013	Utarbeidet skisser for profilsiden	8	HiG	
14.03.2013	Satt opp proof of concept for profiloppbygging + rapport	8	Brygga 20, Innit	
15.03.2013	Opprydding i databasen for bedrifter og søknader	7	Brygga 20, Innit	
16.03.2013	Søndagsøkt: Hentet >3000 lærefag fra utdanningsdirektoratet	8	HiG	
18.03.2013	Puttet lærefagene inn i databasen + rapportskriving	7	HiG	
19.03.2013	Refleksjon og skissering av innloggingsprossess	8	HiG	
21.03.2013	Implementering av responsiv bakgrunn og innlogging/registreringskjema	7	Brygga 20, Innit	
03.04.2013	Idémyldring rundt "lokkebiff" for registrering + rapportskriving	8	HiG	
04.04.2013	Utviklet AJAX widget på forsiden for å presentere ledige plasser	8	Brygga 20, Innit	
05.04.2013	Finpusset widget med bedre interaksjonskomponenter	8	HiG	
09.04.2013	Jobbet med forsiden/loggin	7	HiG	
10.04.2013	Jobbet med registrering og JSON-data fra Orgunit	8	HiG	
11.04.2013	Jobbet videre med registering og JSON-data fra Orgunit	8	Brygga 20, Innit	
12.04.2013	Jobbet med veiviseren	8	HiG	
16.04.2013	Videre arbeid med veiviseren	8	HiG	
17.04.2013	Videre arbeid med veiviseren + rapportskriving	8	HiG	
18.04.2013	Presentasjon av veiviseren. Jobbet med profilsiden	7	Brygga 20, Innit	
19.04.2013	Videre arbeid med profilsiden. Vedlegg funksjonalitet	8	HiG	
23.04.2013	Videre arbeid med vedlegg funksjonalitet	8	HiG	
24.04.2013	Arbeidet med søknadsprosessen	7	HiG	
25.04.2013	Presentasjon av søknadsprosessen. Begynte arbeidet med opplæringskonsulentes rolle	8	Brygga 20, Innit	
26.04.2013	Jobbet videre med filtrering av søknader	8	HiG	
29.04.2013	Jobbet med behandling av søknad	8	HiG	
30.04.2013	Jobbet videre med behandling av søknad	9	HiG	
01.05.2013	Presentasjon av opplæringskonsulentens rolle. Ferdigstilling av utviklingen	8	Brygga 20, Innit	
02.05.2013	Rapportskriving	9	HiG	
03.05.2013	Rapportskriving	8	HiG	
05.05.2013	Rapport skriving	5	Hjemmearbeid	
06.05.2013	Rapportskriving	8	HiG	
07.05.2013	Rapportskriving	8	HiG	
08.05.2013	Rapportskriving	7	HiG	
09.05.2013	Rapportskriving	9	HiG	
10.05.2013	Rapportskriving	7	HiG	
12.05.2013	Rapportskriving	6	Hjemmearbeid	
13.05.2013	Rapportskriving og korrekturlesing	10	HiG	
14.05.2013	Rapportskriving og korrekturlesing	12	HiG	
15.05.2013	Ferdigstilling av rapport	15	HiG	

Utover timene angitt i tabellen, har vi jobbet individuelt sporadisk gjennom utviklingsperioden.

Forprosjekt

1. Mål og rammer
    - 1.1 Prosjektmål
    - 1.2. Rammer
        - Tidsrammer
        - Juridiske rammer
        - Teknologiske rammer
2. Omfang
    - 2.1. Oppgavebeskrivelse
    - 2.2. Problemstilling
    - 2.3. Bakgrunn for problemstilling
    - 2.4. Avgrensning
3. Prosjektorganisering
    - 3.1. Oppdragsgiver
    - 3.2 Ansvarsforhold og roller
    - 3.3. Arbeidsrutiner
    - 3.4. Grupperegler
4. Planlegging, rapportering og oppfølging
    - 4.1. Valg av SU modell - Scrum
5. Kvalitetssikring
    - 5.1. Dokumentasjon, kodestandard og kildekode
    - 5.2. Konfigurasjonsstyring

1. Mål og rammer

1.1 Prosjektmål

Arbeidsgivers mål

I slutten av prosjektperioden skal det foreligge et system for søknad, formidling og tildeling av lærlingplass i en bedrift knyttet opp mot et opplæringskontor. Systemet skal kunne utvides uten store komplikasjoner og utveksling av data mellom tilknyttede systemer skal være mulig.

Læremål

Bedre forståelse for utvikling av større web-applikasjoner. Med fokus på modulutvikling skal gruppen også få økt forståelse for arkitektur og implementering av slike systemer. Gjennom denne prosessen ønsker vi også å lære mer om konsultering med både kunder, brukergrupper og arbeidsgiver.

1.3. Rammer

Tidsrammer

Prosjektet skal ferdigstilles 15.05.2013. På dette tidspunktet skal det foreligge en ferdig testet og funksjonell formidlingstjeneste skreddersydd for norske opplæringskontor, samt potensielle lærlinger. Oppdragsgiver har også etterspurt en fungerende prototype med fokus på lærlingsplassformidling og søknadsregistrering i februar.

Juridiske rammer

Løsningen behandler til en viss grad sensitive data slik som personalia, sertifikat- og utdanningsopplysninger, derfor vil det være kritisk at disse dataene blir behandlet på en forsvarlig og sikker måte.

Teknologiske rammer

- Det er en forutsetning at Innit AS skal videreutvikle systemet etter endt prosjektdato. Det er derfor gitt at prosjektgruppen følger tekniske standarder. Innit jobber også utelukkende med Open Source programvare, og det vil være naturlig at prosjektet bygger på samme teknologi. Følgende teknologier vil derfor være aktuelle:
- Server: Apache 2.2 (Debian 6)
- Programmeringsspråk: PHP 5.3

- Database: MySQL 5.5
- Presentasjon: HTML5 og JavaScript

2. Omfang

2.1. Oppgavebeskrivelse

Oppgave som beskrevet fra Innit AS:
OLKWEB er en applikasjon som brukes av en rekke opplæringskontor i Norge. Dette er et fullverdig administrasjons og opplæringssystem for lærlinger. Applikasjonen håndterer alt som har med lærlinger å gjøre i opplæringsperioden deres. applikasjonen er bygd over en 6 års periode og er nå inne i generasjon 3 av systemet. Erfaringene vi har gjort så langt tilsier at denne delen av virksomheten til opplæringskontoret er mer eller mindre fullkommen.

Inntak av lærlinger er en annen viktig del av opplæringskontorene sin virksomhet samt formidling av nye læreplasser er også en viktig del. Uten inntak og formidling stopper utdanning av nye fagarbeidere i Norge. Dette er i dag manuelle rutiner og tidkrevende prosesser for opplæringskontorene. Innhenting av opplysninger, behandling av søknad, kontrakts inngåelse osv. er noe av det som gjøres i forbindelse med formidling og inntak.

Pr i dag finnes det ingen gode systemer for å vise hvor det finnes ledige læreplasser og i hvilke fag det finnes ledige læreplasser. I tillegg er opplysningene som blir innhentet mangelfulle og av dårlig kvalitet.

Opplæringskontorene og Innit har derfor sett behovet for å få utviklet en formidlings portal som viser ledige læreplasser i Norge, slik at antall søkere øker og at vi få mindre frafall. I tillegg ønsker opplæringskontorene et system for behandling av søknader for å kunne opprettholde kvalitet og rasjonalisering av denne prosessen.

Oppgaven vil derfor bestå i å lage et formidlings og søknadssystem hvor formidlingsdelen vil være en som viser hvor det er ledige læreplasser, samt en administrasjonsdel hvor opplæringskontorene vil behandle mottatte søknader. Dette innebærer alt fra behovsanalyse og annet forarbeid, til utvikling av selve løsningen.

I dette prosjektet skal gruppen utnytte rammeverk for strukturering av kode, samt utnytte etablerte designmønstre og standardiserte formater for integrering mot tredjeparts systemer. Analyse av kundebehov og krav vil stå sentralt for at dette prosjektet skal lykkes. Det er viktig at studentene velger en systemutviklingsmodell og verktøy for samarbeid for å gjennomføre dette prosjektet på en best mulig måte.

2.2. Problemstilling

Hvordan lage en webbasert søknads- og formidlingsmodul, som en effektiv inngangsport til et eksisterende E-learning system.

2.3. Bakgrunn for problemstilling

Det finnes i dag ingen gode systemer for å se hvor det finnes ledige lærlingplasser og i hvilke fag det finnes ledige lærlingplasser. I tillegg er opplæringskontorenes rolle under søknadsprosedyren uklar, noe som medfører til økt behandlingstid. Opplæringskontorene og vår oppdragsgiver Innit AS har sett behovet for et slikt system, som kan føre til at en større andel av lærlingene gjennomfører den yrkesfaglige utdannelsen.

2.4. Avgrensning

Prosjektet produkt skal ikke være en erstatning for Vigo som er en tjeneste for videreutdanning levert av fylkeskommunene i Norge, men som et supplement for å forenkle manuelle rutiner utført av opplæringskontorene.

3. Prosjektorganisering

3.1. Oppdragsgiver

Innit AS er et it--selskap lokalisert på Hamar som spesialiserer seg på programvare- og webutvikling, rådgivning, konsulenttjenester, virtualiseringsløsninger, driftstjenester, kommunikasjonsverktøyer og backup / sikkerhetsrutiner.

Selskapet ble etablert i 2000. INNIT AS har i dag 14 ansatte med bred erfaring og høy kompetanse innen tjenestene de tilbyr. Mangfoldet i klientporteføljen vitner til Innits gode evne til å gjøre hverdagen enklere ved hjelp av moderne teknologi.

3.2 Ansvarsforhold og roller

Representanter fra Innit: Kim Sandvold og Simen Mørch
Veileder: Anders Sundnes Løvlie
Gruppeleder: Emil Kjelsrud
Utviklere: Ole Christian Melbostad, Ivan Lé Hjelmeland, Emil Kjelsrud

3.3. Arbeidsrutiner

Ukentlige møter med veileder og oppdragsgiver. Arbeidsplass vil variere mellom Campus, Innit (brygga 20) og evt. hjemme hos en av gruppemedlemmene ved behov
Vi prioriterer å jobbe under samme tak så langt det lar seg gjøre

3.4. Grupperegler

Om det skulle oppstå en uenighet mellom gruppens medlemmer skal det avholdes en diskusjonsrunde. Her vil prosjektleder ha rikelig med makt til å løse denne konflikten på egenhånd. Hvis ikke konflikten kan løses skal veileder kontaktes.

Alle gruppens medlemmer har retten til å signere på vegne av gruppen.

Dersom det skulle påløpe kostnader skal disse fordeles likt blant gruppens medlemmer. Men en kostnad skal alltid godkjennes av alle medlemmer før en avtale inngås.

Ved fravær fra avtalte møter skal det gis beskjed til gruppeleder snarest, med begrunnelse.

Ved forsinkelser på over 30 minutter utover avtalt skal vedkommende spandere kaffe samme dag som forbrytelse er vedkjent. Eventuell sykdom skal meldes til alle gruppens medlemmer.

Dersom medlem ikke utfører en tildelt oppgave skal det diskuteres hvorfor oppgaven ikke har blitt utført. Om lignende tilfeller dukker opp i framtiden skal både veileder og oppdragsgiver bli informert.

4. Planlegging, rapportering og oppfølging

4.1. Valg av SU modell -

Systemet som skal utvikles skal utvikles ved bruk av en smidig utviklingsmetode, da dette gir oss muligheten til:

- Å utvikle systemet med en iterativ tilnærming.
- Mer synlighet i prosjektet
- Å være åpen for skiftende kundekrav
- Økt kundetilfredshet ved rask levering av fungerende programvare

Skulle systemet ha blitt utviklet på en sekvensiell måte ville ikke dette gitt oss muligheten til å utvikle systemet slik som nevnt i listen over. Siden systemet som skal utvikles har skiftende kundekrav, mener vi det er best å utvikle systemet ved å benytte en smidig utviklingsmetode.

Valget av systemutviklingsmodell sto mellom Scrum og en "lettvekter" versjon av RUP. Valget falt til slutt på Scrum, da RUP er beregnet på større bedrifter med store budsjetter og mange ansatte. RUP definerer også mange roller, noe ikke vi har nok medlemmer i gruppen til å oppfylle.

Scrum

Scrum prosjektet skal bli systematisert på følgende måte:
Product Owner: Kim Sandvold
Scrum Master: Emil Kjelsrud
Utviklingsteam: Emil Kjelsrud, Ivan Lé Hjelmeland, Ole Christian Melbostad

Product backlog for prosjektet er vedlagt. Estimeringsteknikken vi har valgt å bruke kalles Planning Poker. Ved å anvende denne teknikken oppfordres det til diskusjon ved at alle gruppemedlemmene presenterer sitt estimat; som forhåpentligvis fører til et mer presist anslag. Vi velger å benytte oss av to ukers sprinter. Sprintene starter på tirsdager og avsluttes tirsdag to uker senere. I Product Backlog deler vi opp i tre forskjellige prioriteringskategorier: Høy, medium og lav. Elementer i Product Backlog med høy til og med medium prioriteringsgrad er de elementer som må gjennomføres for at systemet skal tilfredsstille arbeidsgivers krav.

Elementer fra Product Backlog som overføres i Sprint Backlog velges på bakgrunn av prioriteringsgrad og sannsynlighet for gjennomførelse på gitt tidspunkt.

I starten av hver arbeidsdag skal det holdes et Daily Scrum Stand Up Meeting som varer 10 minutter. Hensikten med disse møtene er å holde alle gruppemedlemmer oppdatert på hva som har blitt gjort, og hva som eventuelt byr på utfordringer. Ved eventuelle utfordringer hos en av gruppemedlemmene, vil vi utnytte konseptet Par-programmering der vi ser det hensiktsmessig. På grunn av manglende grupperom har vi valgt å benytte oss av en programvare som heter OnTime Scrum som skal hjelpe oss med å styre Scrum prosjektet. OnTime Scrum hjelper oss med alltid å ha lett tilgjengelig Product Backlog, Sprint Backlog og Burndown chart på ett sted. Disse konseptene bidrar til gi gruppen bedre oversikt, samt effektiv monitorering av arbeidsmengde. Vi vil med dette oppdage tids- og organisatoriske avvik tidlig i prosjektet.
Sprint Review holdes i slutten av hver Sprint før møte med veileder. Sprint Retrospective Meeting holdes etter hver Sprint Review de tre første sprintene og deretter etter annenhver sprint. Sprint Review Meeting og Sprint Retrospective Meeting skal hver for seg ha en varighet på 30 minutter.
For å definere når et element i Sprint Backlog er "ferdig" velger vi å benytte oss av "Quality gates". Disse portene skal kvalitetssikre arbeidet vi gjør og forsikre oss om at elementer som har passert portene er klar til levering.

Quality gates - Rapport:

- Skal være fri for skrivefeil.
- Må følge Harvard sine retningslinjer for kildehenvisning

Quality gates - Kode:

- Skal tilfredsstille Innits retningslinjer for kodestandard
- Skal passere alle egendefinerte tester
- Alle offentlige grensesnitt må dokumenteres
- Alle kode som dyttes mot felles repository må være fri for feil og advarsler på høyeste nivå

Vi har også valgt å utnytte oss av disse artefakter fra andre systemutviklingsmodeller:

Use Case

- Logical View og Deployment View fra RUP
- Domenemodell
- Par-programmering fra XP

5. Kvalitetssikring

5.1. Dokumentasjon, kodestandard og kildekode

Dokumentasjon

En wiki med API referanser, kjøreregler, samt rammeverk- og modulkildekode vil bli vedlikeholdt gjennom hele prosjektet. Dette vil fungere som en felles kompetanse- og dokumentasjonsplattform for alle deltakere i prosjektet.

Kodestandard

Wikien vil også si noe om definerte konvensjoner for hvordan koden skal bygges opp og fungere på både mikro- og makronivå. Eksempler på slike konvensjoner kan være standardiserte returneringsverdier for funksjoner (mikro) og standarder for hvordan moduler og submoduler snakker sammen på overordnet nivå (makro). Det benyttede rammeverket vil være med på å både definere og støtte opp under definerte standarder og konvensjoner.

Kildekode

Ved å holde strenge konvensjoner for kodestandard, samt disiplin, vil det det naturligvis bli mindre behov for å dokumentere i kildekode. På bakgrunn av systemets natur og størrelse ser vi det som hensiktsmessig å holde kildekoden fri for mye dokumentasjon. Det vil derfor være kritisk for prosjektets utfall å opprettholde konkrete design- og arkitekturprinsipper definert i wikien.

5.2. Konfigurasjonsstyring

Det vil foreligge lokale Git repository med hele prosjektets kildekode. En ekstern tjener vil også synkronisere disse klientene slik at alle gruppemedlemmer oppdatert kode til enhver tid. Innit har også gitt oss tilgang til et egenutviklet rammeverk for hyppig applikasjonsutvikling, som vil symbollenkes fra ekstern tjener grunnet sikkerhetsmessige årsaker. Det vil også bli satt opp en SVN-proxy slik at vi kan dytte kildekode opp til Innit ved behov for dette.

Filstruktur

```
/rbase/
/module/
 data/
   AspirantStorage.php
   Aspirant.php
   ApplicationStorage.php
   Application.php
   OrgunitStorage.php
   Orgunit.php
 handler/
   aspirant/
     AspirantHandler.php
     AspirantIndexHandler.php
   application/
     ApplicationHandler.php
     ApplicationIndexHandler.php
   orgunit/
     OrgunitHandler.php
     OrgunitIndexHandler.php
   processor/
     AspirantProcessor.php
     ApplicationProcessor.php
     OrgunitProcessor.php
   template/
     aspirant/
       aspirant.index.phtml
       aspirant.phtml
       form/
         aspirant.post.phtml
         aspirant.put.phtml
         aspirant.fields.phtml
```

```
    application/
      application.index.phtml
      application.phtml
      form/
        application.post.phtml
        application.put.phtml
        application.fields.phtml
      orgunit/
        orgunit.index.phtml
        orgunit.phtml
/config.ini
/www/
 js/
 css/
 index.php
```

Intervju 07.02.2013 - Ringsaker VGS

Vi intervjuet tre elever ved Ringsaker VGS. Vi ønsket å få svar på følgende spørsmål:

Hvordan søker du lærlingplass?

Hva fokuserer du på når du søker lærlingplass(Geografisk lokasjon? og spennende bedrift?)

Erfaringer fra familie og venner?

Føler du at du har fått tilstrekkelig informasjon om aktuelle bedrifter og søknadsprosess?

Har du noen bekymringer rundt søknadsprosessen?

Er det noe med søknadsprosessen du skulle ønske var annerledes?

Testperson 1

Jeg kontaktet ulike bedrifter. Fikk hjelp fra lærer for å finne aktuelle bedrifter.

God faglærer er viktig - Hovedkontakten til arbeidslivet.

Erfaringer om det å være lærling -og kunnskapsdatabase finnes ikke.

Praksis er den mest vanlige inngangen til lærlingplass.

Geografisk lokasjon er ikke viktig.

Arbeidsmiljø er viktig.

Testperson 2

Kjennskap til folk i bedrift er nøkkelen.

Tilbyder krever som regel anbefaling fra faglærer.

Lite informasjon om bedrifter

Viktig at arbeidsmiljø er bra.

Lokasjon er lite viktig.

Historier og erfaringer fra lærlinger er bra.

Testperson 3

Bør være i nærheten (lokasjon viktig)

Tidligere erfaringer fra lærlinger er positivt.

Portalen bør være lett å bruke, og bør inneholde nok informasjon om bedriftene.

Viktig å kunne finne en bestemt bedrift.

Info om tilgjengelig praksis er et pluss.