Nexstep-integrasjon

Anders Fjelstad, Petter A. Helset og Didrik Sørensen

Bacheloroppgave
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2010

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Innhold

Figurer

1 Innledning

1.1 Organisering av rapporten

Rapportens innhold er inndelt som følgende:

1. Innledning - Innledende informasjon til resten av rapporten.
2. Kravspesifikasjonsdokument - Direkte krav til applikasjonen fra oppdragsgiver.
3. Designdokument - Utredelse om applikasjonens design.
4. Implementasjonsdokument - Hvordan løsningen har blitt implementert.
5. Testing - Testprosedyren av løsningen.
6. Avslutningsdokument - Resultat, drøfting og konklusjon.
7. Litteraturliste - Liste over brukt litteratur i hele prosjektet
8. Ordliste
9. Vedlegg

1.2 Oppgavebeskrivelse

Oppgaven går ut på å lage en applikasjon hvor de ansatte hos ASP kan registerere sine arbeidstimer fortløpende, enkelt og hurtig. Det skal være en integrasjon mot nåværende ordresystem, slik at hele faktureringsprosessen blir automatisert. Hver timeinnføring som skal registreres må inneholde dato, kunde, varetype, antall timer og timespris. Alternativt kan de inneholde kundeprosjekt og kontaktperson. Disse dataene skal hentes ut ifra oppdragsgivers system.

Det skal også kunne vises en oversikt over alle registrete timer, både ikke-fakturerte og fakturerte.

Denne løsningen skal avvikle oppdragsgivers nåværende løsning, som er excel-ark.

Mobilapplikasjonen skal ikke støtte innsending av data til Nexstep.

1.3 Målgruppe

Målgruppen for denne rapporten er oppdragsgiver og eventuelt andre medstudenter. Rapporten er også med i karaktergrunnlaget for bacheloroppgaven. Denne rapporten er ment å gi innsikt i løsningen, og kan være til nytte for oppdragsgiver om de ønsker å vidreutvikle systemet til andre formål.

1.4 Faglig bakgrunn

Gruppen vår består av Didrik Sørensen, Petter Helset og Anders Fjelstad. Didrik tar bachelor i programvareutvikling, mens Petter og Anders tar bachelor i dataingeniør. Alle har god erfaring

innenfor programmering.

1.5 Tidligere arbeid

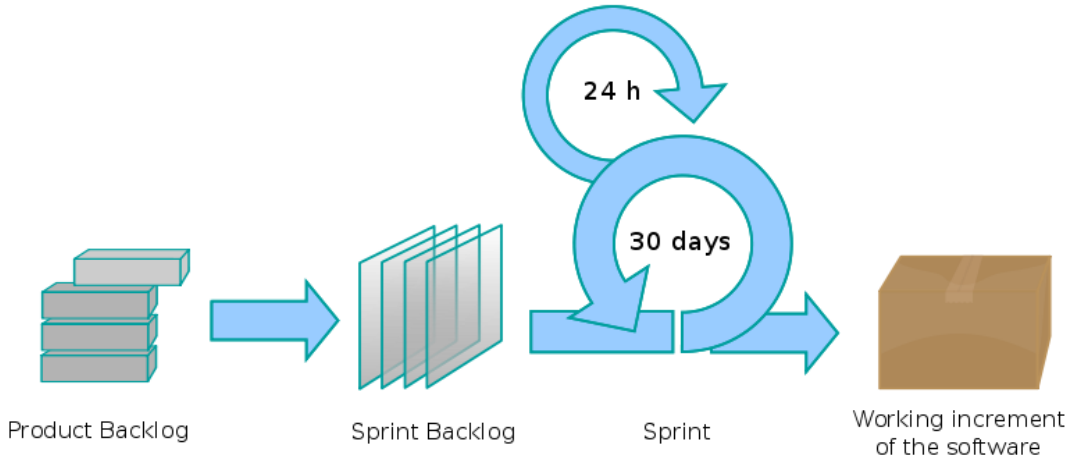
En av gruppens medlemmer har hatt et hobbyprosjekt om et fakturasystem. Dette har gitt et godt innblikk i hvordan en tenkt løsning kunne bli, i et tidlig stadie.

1.6 Valgt arbeidsform

Etter en grundig analyse av aktuelle utviklingsmodeller, falt vi tilslutt på Scrum. Scrum passet svært godt til prosjektet ettersom oppdragsgiver ønsket å teste systemet i faser, og Scrum leverer ferdigfungerende inkrementer per iterasjon. Denne modellen tar opp eventuelle endringer før hver iterasjon, noe som var nyttig i vårt prosjekt da vi visste det kom til å bli mye endringer i kravene underveis.

De daglige møtene har vært nyttige, og dette har gitt oss god kommunikasjon med oppdragsgiver og deres ønsker om systemet.

Det eneste negative ved utviklingsmodellvalget var at Scrum ikke spesifiserer hvordan dokumentasjonen skal foregå. Dette løste vi ved å implementere dokumentasjonskravene til RUP, Rational Unified Process.



Figur 1: Scrum prosessen

Vi delte inn utviklingsperioden i seks iterasjoner, også kalt sprints.

Sprint #1

- Web Service
  - Oppsett av AXIS2
  - JPA og Hibernate
  - Oppsett av kodestruktur

Sprint #2

- Web Service
  - Oppsett av kodestruktur
  - Input-/output-håndtering
  - Generell funksjonalitet
- Web-applikasjon
  - Layout
  - Generell funksjonalitet

Sprint #3

- Web Service
  - Generell funksjonalitet
- Web-applikasjon
  - Registrering av timer
  - Presentasjon av timer
- Mobil web-applikasjon
  - Layout
  - Generell funksjonalitet

Sprint #4

- Web Service
  - Rammeverk for kommunikasjon mot Nexstep
- Mobil web-applikasjon
  - Registrere timer
- Web-applikasjon
  - Generell statistikk
  - (Statistikkdiagrammer)

Sprint #5 (dobbel-sprint)

- Web Service
  - Fullfør kommunikasjon med NexStep
- Web-applikasjon
  - Fullfør statistikk
  - Fullfør timeregistrering
  - Fullfør timeliste (timeorganisering)
- Mobil web-applikasjon
  - Fullfør registrere timer
  - Fullføre timeliste (timeorganisering)

Sprint #6

- Web-applikasjon
  - Muligheter for prisendring
  - Innhenting av rabattmatrise

2 Kravspesifikasjon

2.1 Introduksjon

2.1.1 Hensikt

Hensikten med dette dokumentet er å beskrive kravene og omgivelsene til systemet. Dokumentet skal gi konkrete rammer rundt prosjektet, og gi en klarhet i hva som skal utvikles, og hvordan dette skal foregå.

Kravene i dette dokumentet har blitt formet sammen med oppdragsgiver.

2.1.2 Prosjektbeskrivelse

Prosjektet går ut på å lage et timeregistreringssystem. Dette innebærer blant annet å kunne registrere dato, antall timer jobbet, referanse til kunde, og lignende. I tillegg skal den registrerte timeinformasjonen kunne sendes videre til et eksternt system som håndterer selve faktureringen av kunden.

Løsningen skal utvikles for programvareselskapet Application System Partner AS.

2.1.3 Dokumentrevisjoner

Rev. #	Dato	Kommentar	sign. / navn
1	26. februar 2010	Opprettet dokument	Anders, Didrik og Petter.
2	29. mars 2010	Endringer i krav fra oppdragsgiver	Anders, Didrik og Petter.
3	14. mai 2010	Fullføring av dokument	Anders, Didrik og Petter.

2.2 Overordnet beskrivelse

2.2.1 Krav til systemet

Systemet skal kunne registrere følgende mot en innføring;
Systemet skal kunne registrere, oppdatere og slette en timeinnføring. Innføringene skal inneholde følgende;

- Dato
- Kunde
- Kundeprosjekt
- Kontaktperson
- Vare
- Varebeskrivelse
- Antall timer
- Timespris

Dette systemet skal gi hver ansatt god oversikt over hvilke timer som allerede har blitt fakturert eller timer som skal faktureres. Hver bruker kan når som helst endre på registrerte timer, så lenge de ikke allerede er sendt inn til ordresystemet. Dette forhindrer feilregistreringer, dobbelregistreringer og glemte registreringer.

Det skal være svært enkelt for hver bruker å kunne registrere timene sine, så denne registreringsprossen må gå hurtig.Systemet må i tillegg holde orden på fakturerte intern- og eksterntimer. Alle timeregistreringene skal kunne sendes inn til oppdragsgivers ordresystem, for å fakturere de respektive kundene.

Det skal være en kobling mot databasen til ordresystemet, slik at relevante databasetabeller kan hentes ut til systemet, og vises på skjermen, eller kan kalkuleres med. Dette skal være å gjøre en prisforespørsel på grunnlag av kunde og kundeprosjekt. Dette hentes ut ifra rabattmatriser hos oppdragsgivers nåværende system. Varetyper og -beskrivelser skal kunne hentes ut. Varebeskrivelsen på timeinnføringer skal være redigerbare. Systemet må også hente ut kunder og kundeprosjekter.

Systemet skal også inneholde en statistikkdel, hvor brukerene kan få se antall timer registrert per kunde, kundeprosjekt, dato, vare eller kontaktperson. Dette skal vises grafisk.

Denne løsningen skal være for stasjonære og mobile enheter. For de mobile enhetene skal det ikke støttes effektuering (innsending) av timer.

2.2.2 Systemets omgivelser

Systemet skal være nettbasert, både for stasjonære maskiner og mobile enheter. Det er valgfritt hvilken løsning brukerene ønsker å bruke, men det er mulighet for begge.

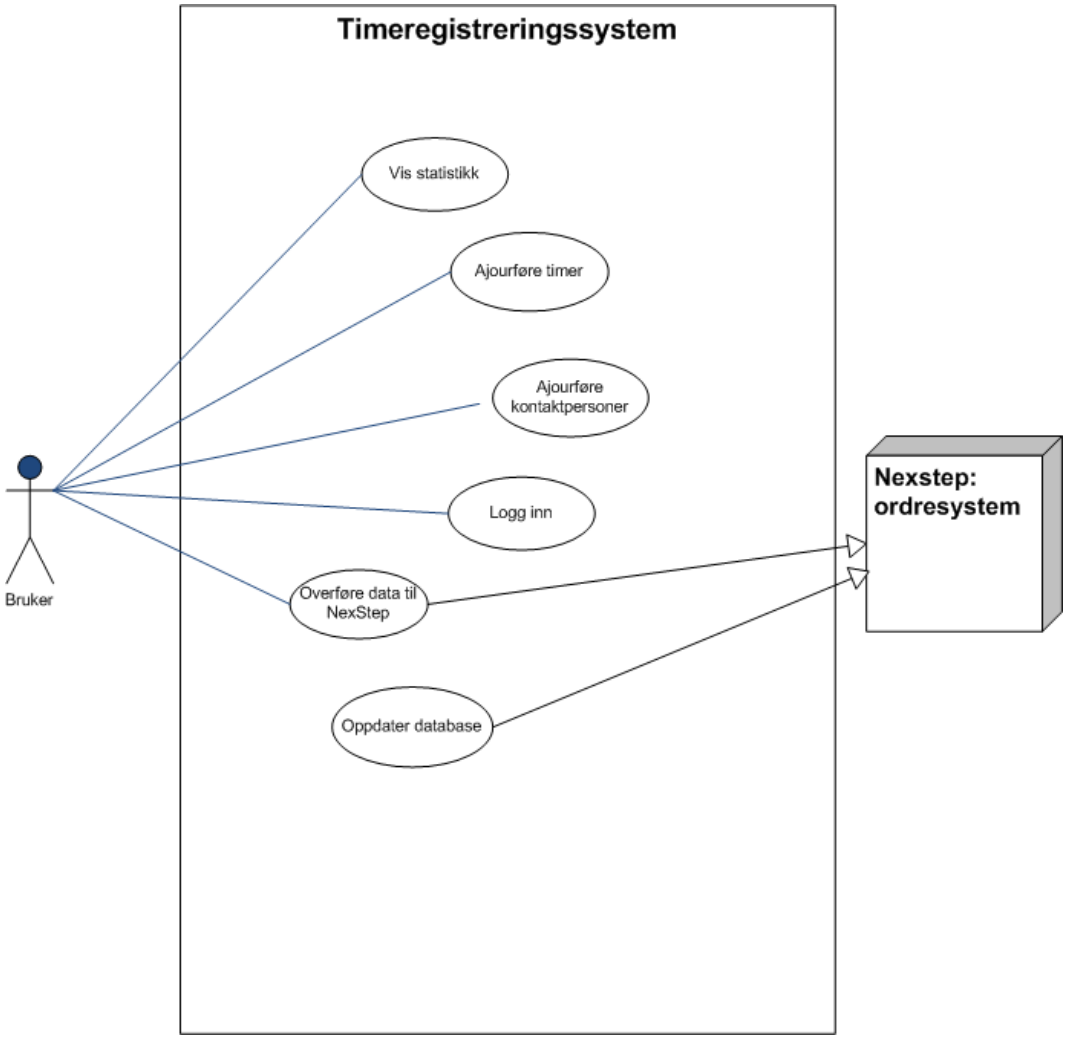
Løsningen vil ligge på serverene til oppdragsgiver, og ha direkte kobling mot ordresystemet i Nexstep.

2.2.3 Systemets brukere

Systemet vil kun ha èn type brukere, ettersom det ikke er behov for funksjonalitet utover timeregistreringen. Brukerene av systemet er de ansatte hos ASP.

- Konsulenter
- Utviklere
- Support

2.2.4 Use Case-modell



Figur 2: Use Case-diagram

- Denne modellen gir en oversikt over funksjonaliteten til brukerene av systemet, og hvordan dette kommuniserer med omverdenen. Den grå ruten er rammen innenfor systemet, som inneholder web-applikasjonen, web service og databasen.
- Brukeren logger først inn på systemet. Deretter kan vedkommende registrere de aktuelle timene, for å så velge å sende inn ønskede timer til ordresystemet. Dette kan være i slutten av en periode, eller på hvilket som helst vilkårlig tidspunkt. Timelisten vil deretter bli fakturert fra ordresystemet til oppdragsgiver.
- Brukeren kan når som helst velge å oppdatere den lokale databasen. Det vil si å hente inn ny data fra ordresystemet, slik som kunder, varer og kundeprosjekter. Dette gjør at systemet vil

være oppdatert til enhver tid.

- Statistikken viser brukerens timer registrert over en periode, per kunde, per kundeprosjekt, totalt eller per timetype (intern- eller ekstern time).

2.2.5 Overordnet Use Case-beskrivelse

Navn Ajouføre timer
Aktør Alle brukere
Mål Ajourføre arbeidstimer, slik at disse kan faktureres.
Beskrivelse Aktøren kan registere, editere eller slette arbeidstimer. Ved de to første mulighetene må dato (kan fylles ut automatisk), antall timer, kunde, vare, og eventuelt kontaktperson, kundeprosjekt og eventuell ny varebeskrivelse.

Navn Innlogging
Aktør Alle brukere
Mål Logge inn på systemet
Beskrivelse Aktøren får autentisere seg, slik at systemet åpnes. Brukernavn og passord sjekkes opp mot databasen til Nexstep.

Navn Ajourføre kontaktpersoner
Aktør Alle brukere
Mål Ajourfør èn eller flere kontaktpersoner til en kunde.
Beskrivelse Aktøren kan legge til, editere eller slette èn eller flere av en kundens kontaktpersoner.

Navn Oppdater database
Aktør System
Mål Oppdater databasen med relevant informasjon fra ordresystemet.
Beskrivelse Databasen oppdateres med brukere, kunder, kundeprosjekter og vare. Dette hentes inn fra ordresystemet, for å gjøre ny data tilgjengelig i systemet.

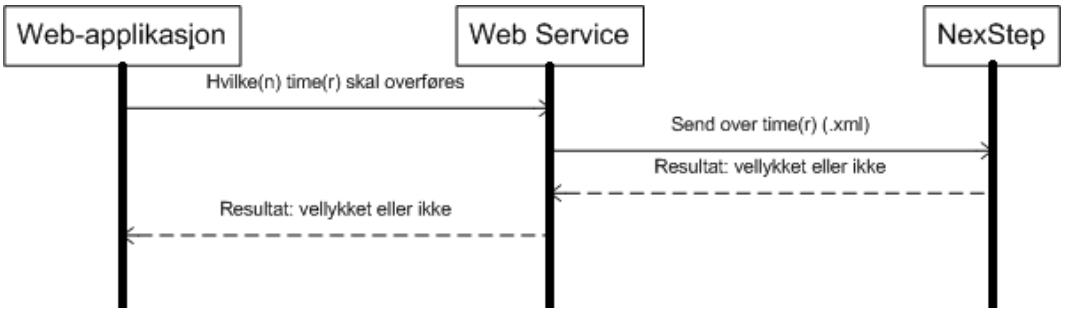
Navn Vis statistikk
Aktør Vanlige brukere
Mål Vis brukerens statistikk.
Beskrivelse Aktøren får en grafisk statistikk over antall timer som er registrert på vedkommende over en gitt periode, kunde, kundeprosjekt eller timetype.

Navn Overføre data til ordresystemet
Aktør Vanlige og administrative brukere
Mål Overføre data til ordresystemet.
Beskrivelse Aktøren ser over at periodens registrerte timer er komplett, og overfører deretter timene til ordresystemet i Nexstep, som fakturerer disse timene til de respektive kundene.

2.3 Spesifikke krav

2.3.1 Extended Use Case

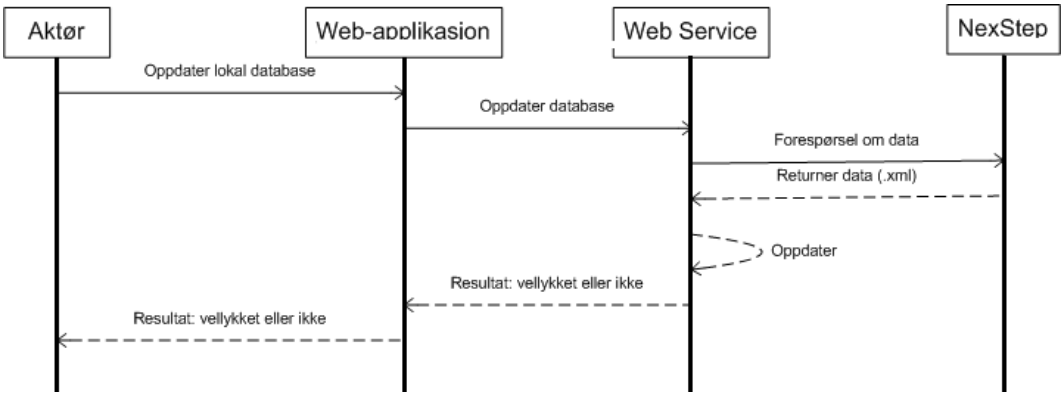
Navn	Oveføre data til ordresystemet
Aktør	Alle brukere
Mål	Overføre valgte timer til ordresystemet for fakturering.
Sammendrag	Systemet overfører timer til ordresystemet.
Type	Kritisk
Forhandling	Innlogging og eksisterende timeregistreringer.
Etterhandling	Ingen
Spesielle krav	Ingen
Hendelsesflyt Aktør	System
1. Velger timeregistreringer som skal sendes	2. Henter ut timer fra database 3. Timer samles i èn fil 4. Sender data på XML-format til ordresystemet 5. Gir beskjed om fullført overføring 6. Markerer overførte timer som fakturert
Alternativ hendelsesflyt #1 *Ingen kontakt med ordresystemet*	
	5. Får ingen kontakt, eller feilmelding fra ordresystemet 6. Viser feilmelding til aktør.
Alternativ hendelsesflyt #2 *Feil i overføring*	
	5. Kommunikasjon med ordresystemet blir brutt. 6. Viser feilmelding til aktør. 7. Forkaster data.



Figur 3: Sekvensdiagram for overføring av data til ordresystemet

Navn	Oppdater database
Aktør	System
Mål	Oppdater databasen med relevant informasjon fra ordresystemet.
Sammendrag	Databasen oppdateres med brukere, kunder, kundeprosjekte og vare, som hentes inn fra ordresystemet, for å gjøre dette tilgjengelig i systemet.
Type	Kritisk
Forhandling	Ingen
Etterhandling	Systemet er oppdatert med ny data.
Spesielle krav	Ingen

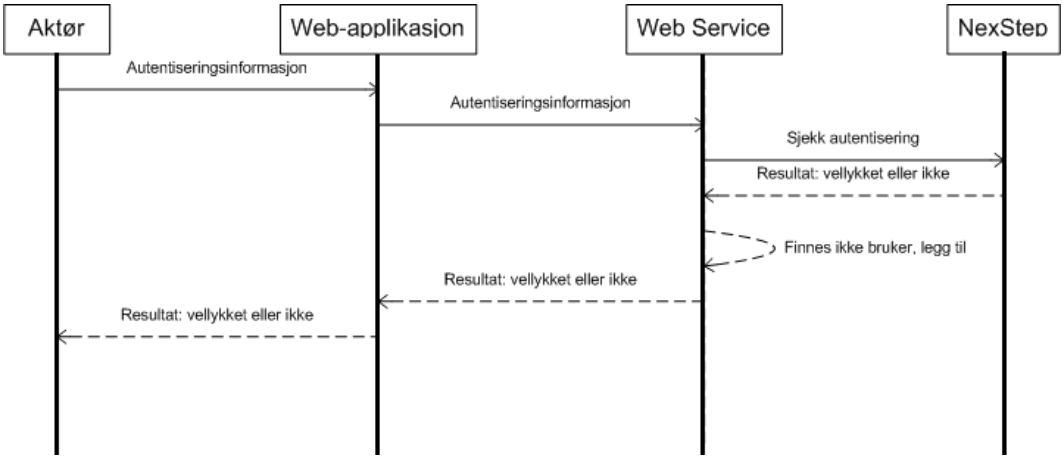
Hendelsesflyt	
Aktør	System
1 Ser etter nye - eller endringer i XML-filer	2. Henter ny data fra ordresystemet. 3. Oppretter eller oppdaterer egen database 4. Gir melding til aktør om fullført operasjon
Alternativ hendelsesflyt #1 *Feil ved forespørsel*	
	2. Gir melding til aktør om kommunikasjonsfeil 3. Logger feilmelding
Alternativ hendelsesflyt #2 *Ingen respons fra ordresystemet*	
	2. Ingen svar fra ordresystemet 3. Gir melding til aktør om kommunikasjonsfeil 4. Logger feilmeding
Alternativ hendelsesflyt #3 *Overføring avbrutt*	
	2. Mottar ufullstendige filer 3. Gir feilmelding til aktør om brutt kommunikasjon 4. Logger feilmelding
Alternativ hendelsesflyt #4 *Ingen kontakt med database*	
	3. Ingen respons fra databasen 4. Gir feilmelding til aktør om kommunikasjonsfeil 5. Logger feilmelding



Figur 4: Sekvensdiagram for oppdatering av database

Navn	Innlogging
Aktør	Alle brukere
Mål	Logge inn på systemet
Sammendrag	Sjekk aktørens autentiseringsinformasjon opp mot databasen i Nexstep.
Type	Viktig
Forhandling	Ingen
Etterhandling	Aktøren er logget inn på systemet.
Spesielle krav	Ingen

Hendelsesflyt	
Aktør	System
1. Skriver inn autentiseringsinformasjon	2. Klargjør og koble opp mot databasen 3. Sjekk om informasjonen stemmer overens 4. Vis melding om vellykket autentisering
Alternativ hendelsesflyt #1 *Ingen respons fra ordresystemet*	
	2. Ingen respons fra databasen 3. Vis melding til aktør om utilgjengelig tjeneste 4. Logger feilmelding
Alternativ hendelsesflyt #2 *Feil ved autentisering*	
	4. Informasjonen stemmer ikke mot databasen 5. Gi melding til aktør om autentiseingsfeil
Alternativ hendelsesflyt #3 *Bruker ligger ikke i lokal database*	
	4. Bruker ligger ikke i lokal database 5. Legg til brukeren i databasen 6. Vis melding til aktør om vellykket autentisering

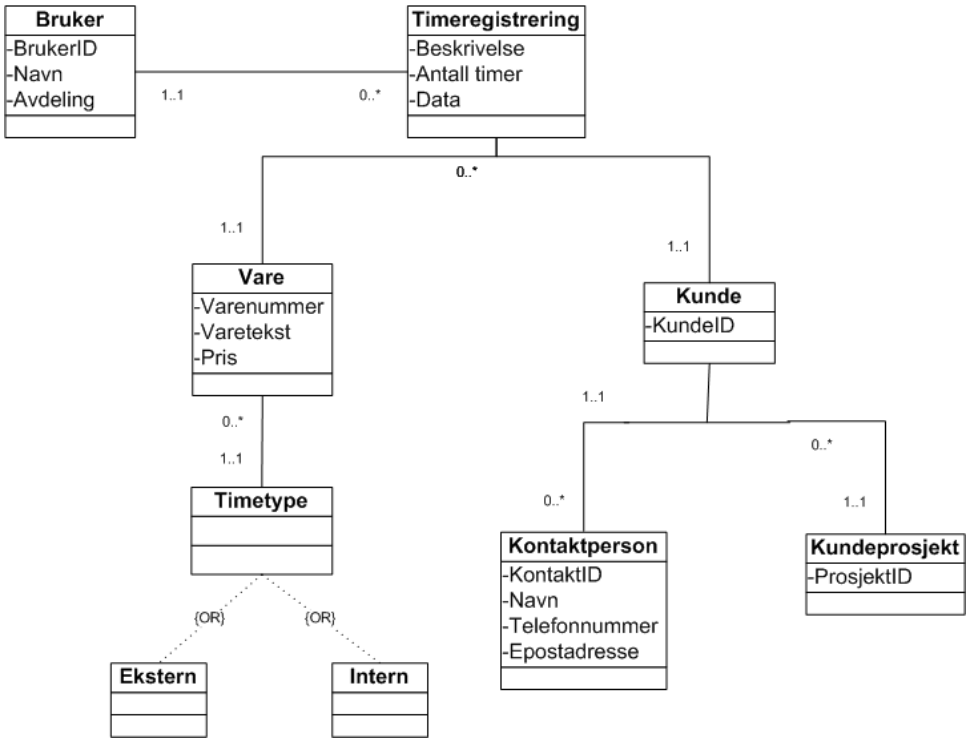


Figur 5: Sekvensdiagram for innlogging

2.3.2 Funksjonelle krav

Domenemodell

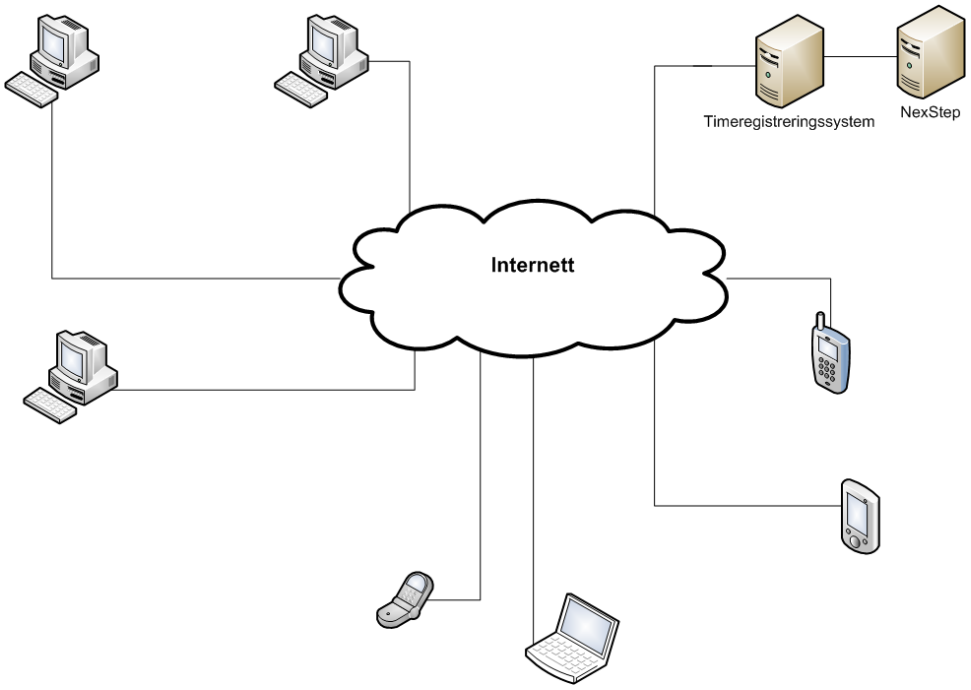
Denne modellen viser en logisk oversikt over systemet, og hvordan relasjonen er mellom objekter. Èn timeregistrering inneholder èn vare, og denne skal enten faktureres internt eller eksternt (mot kunden). Timeregistreringen inneholder èn kunde, som igjen inneholder ett kundeprosjekt. Hver kunde har opptil èn kontaktperson.



Figur 6: Domenemodell

Deployment view

Modellen gir et overblikk over hvordan systemet vil se ut fysisk. Timeregistreringssystemet kommuniserer mot Nexstep. Systemet er tilgjengelig på internett, som en web-applikasjon. Dette gjør det fritt for alle enheter å koble seg til denne, sett i at de kan autentisere seg.



Figur 7: Deployment view

Sikkerhet

Systemet skal ha følgende sikkerhetsløsninger integrert:

Autentisering av brukere

Autentiseringen går via databasen til Nexstep.

Kompabilitet

Systemet er plattformuavhengig, ettersom det kjøres som en web-applikasjon. Dette gjelder også for mobile enheter, ettersom løsningen skal få et brukergrensesnitt uten java-script, slik at den støttes i alle nettlesere. Dette gjør også at eldre maskiner kan kjøre systemet.

Sikkerhetskopi

Systemet vil ikke ha noe integrert backup-løsning, men systemet vil stå på oppdragsgivers server, og vil derfor følge Deres sikkerhetsløsninger.

2.3.3 Brukervennlighet

Opplæringsperiode

Det blir ikke avsatt ressurser til opplæring, bortsett fra dokumentasjonen vedlagt til systemet.

Brukergrensesnitt

Applikasjonen skal kjøre på samme utseende og oppsett oppdragsgiver har på nåværende hjemmeside.

2.3.4 Pålitelighet

Oppetid

Det er ikke kritisk om systemet går ned, i en mindre periode, ettersom det ikke er behov for å registrere timene umiddelbart etter utførelse. Vi setter derfor oppetidskravet til 99,5%, som utgjør maksimalt èn time i uken.

Vedlikehold

Oppetidskravet på 99,5% er inkludert eventuelt vedlikehold.

Reparasjon

Ved reparasjoner av systemet vil en nedetid på maksimalt fem timer være akseptabelt.

2.3.5 Ytelse

Responstid

Vanlige transaksjoner mot systemet skal gå umiddelbart, ettersom det er lite data som skal sendes og mottas. Denne responstiden er satt til maksimalt et halvt sekund, ved tunge operasjoner. Mer krevende funksjonalitet, slik som større oppslag mot databasen og generering av statistikk, vil ha en responstid på maksimalt tre sekunder.

Kapasitet

For å opprettholde respondstiden, så kan systemet håndtere opptil seks samtidige operasjoner per sekund. Hvis dette overskrides, vil respondstiden øke per ekstra operasjon som utføres i dette tidsrommet.

Systemet kan i prinsippet ha et meget høyt antall av brukere pålogget samtidig, uten at dette blir en flaskehals på systemet, slik at ytelsen faller.

Lagringskapasitet

Systemet står på oppdragsgivers server, så lagringskapasiteten tilgjengelig, og eventuelle oppgraderinger av dette, bestemmes av oppdragsgiver.

2.3.6 Supportability

Kodestandard

Følgende er kodestandarer innenfor prosjektet:

- Alle klassenavn og variable er på engelsk, for å slippe problematikk med nordiske bokstaver.
- Vanlig java-kodekonvensjoner følges.
- Alle kommentarer er på norsk.
- Det brukes tabulatorhopp.

2.3.7 Systemgrensesnitt

Brukergrensesnitt

Applikasjonen skal kjøre på samme utseende og oppsett oppdragsgiver har på nåværende hjemmeside.

Programvaregrensesnitt

Kjernen av systemet kjører som web service på en AXIS2 service container, som igjen kjører som en servlet på en Tomcat-server. Web-gresesnittet kjøres som en servlet på samme Tomcat-server.

Systemet skal kunne kommunisere med oppdragsgivers ordresystem.

Interopabilitet

Systemet skal logge feilmeldinger etter krasj, kommunikasjonssvikt, bugs, og andre eventuelle feil til en spesifikk loggfil på Tomcat-serveren.

3 Design

3.1 Introduksjon

Dette dokumentet skal gi innsyn i oppbygning og utvikling i prosjektet, og brukes som referanse ved utviklingen.

3.2 Arkitektur

Systemet skal være plattformuavhengig, og derfor må det støttes på alle stasjonære og bærbare maskiner, heretter omtalt som *gruppe 1*, og mobile enheter, slik som PDA og mobiltelefoner, heretter omtalt som *gruppe 2*. Dette skal fungere uavhengig av ønsket nettleser.

Valget av arkitekturmodell for systemet ble gitt fra oppdragsgiver, og er lagdelingsmodellen. Dette på grunn av at denne modellen blir brukt i deres eksisterende systemer, og det blir lett å implementere mer funksjonalitet i etterkant.

Denne modellen passer våres system svært godt, ettersom vi da slipper å lage to individuelle applikasjoner, til både gruppe 1 og gruppe 2.

Figur 8: Lagdelingsmodell

3.2.1 Presentasjonslaget

Dette laget inneholder det grafiske brukergrensesnittet for *gruppe 1* og *gruppe 2*. Her hentes det ut all relevant data som skal vises ifra det underliggende laget, domenelaget. Dette laget skal sikre korrekt visning av data til de respektive enhetene, med hensyn på forskjellen i skjermstørrelse.

3.2.2 Domenelaget

Domenelaget består av forretningslogikk. Her ligger all funksjonalitet innad i applikasjonen.

3.2.3 Tjenestelaget

I dette laget ligger databasen og alle de eksterne tjenestene som systemet trenger. Dette er blant annet tjenester som oppdragsgiver leverer, slik som kommunikasjon mot ordresystemet.

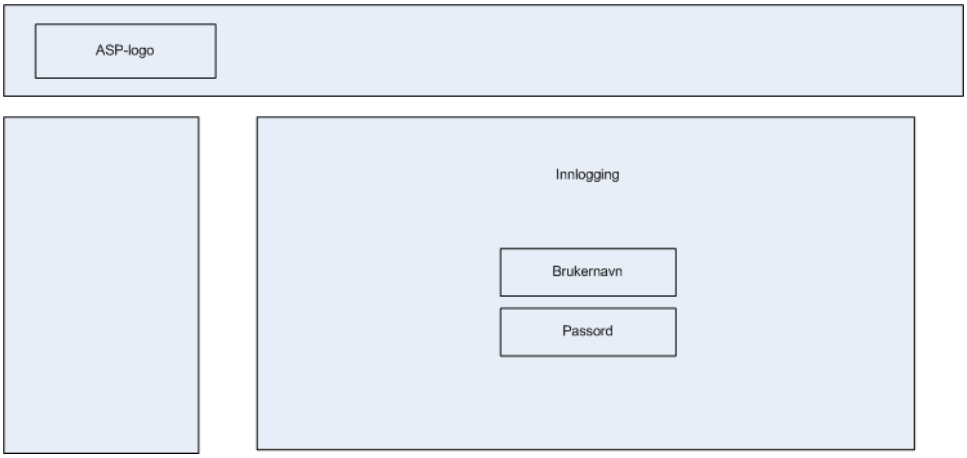
3.3 Brukergrensesnitt

Brukergrensesnittet skal være oversiktlig og enkelt å operere med. Dette er svært viktig for at ikke de ansatte faller tilbake på den gamle løsningen. Oppdragsgiver hadde et ønske om at vi holdt brukergrensesnittet på samme form som deres nåværende nettside.

Det tas også hensyn til størrelsen på applikasjonen for mobileenhetene. Dette brukergrensesnittet vil ha noe mindre funksjonalitet, men det blir ivaretatt hovedmålet, registrere, editere, vise og sende inn timer. Designmessig vil denne delen av applikasjonen bli svært simpel.

3.3.1 Brukergrensesnitt for *gruppe 1*

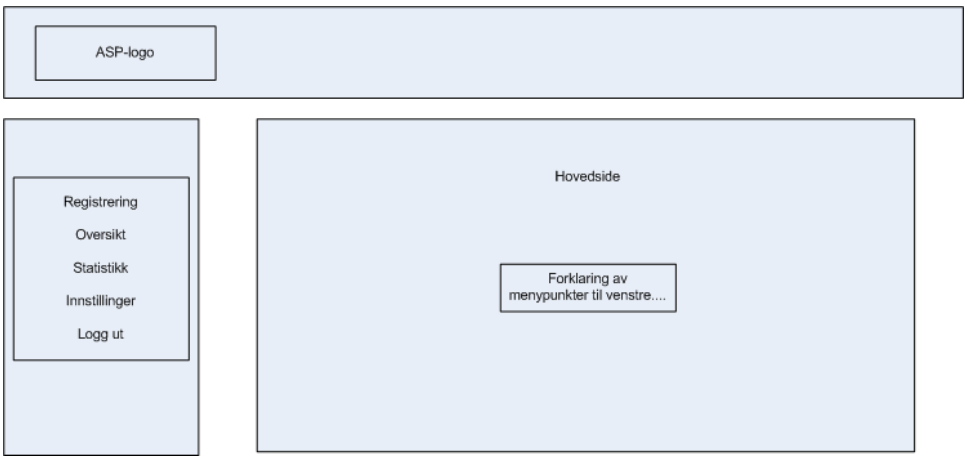
Innlogging



Figur 9: Skjemskudd av innloggingsbilde

Her må brukeren autentisere seg for å få tilgang til systemet.

Hovedside



Figur 10: Skjemskudd av hovedmeny

Brukeren får en kort beskrivelse av funksjonaliteten på siden.
Her kan brukeren velge mellom følgende operasjoner:

Registrering
Legge til en nye timer

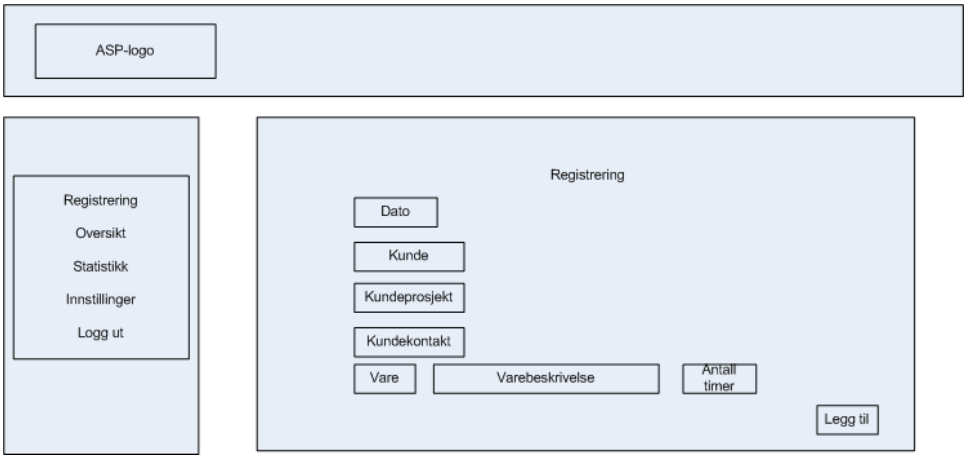
Oversikt
Hente timer fra valgfri dato. Se eller send inn timer til ordresystemet i Nexstep.

Statistikk
Enkel statistikk over egne timer. Det vil også bli vist forskjellige diagrammer.

Innstillinger
Her velger brukeren hvilke standarinnstillinger vedkommende ønsker.

Logg ut
Etter brukeren er ferdig, kan vedkommende logge seg ut av systemet.

Registrering



Figur 11: Skjemskudd av registrering

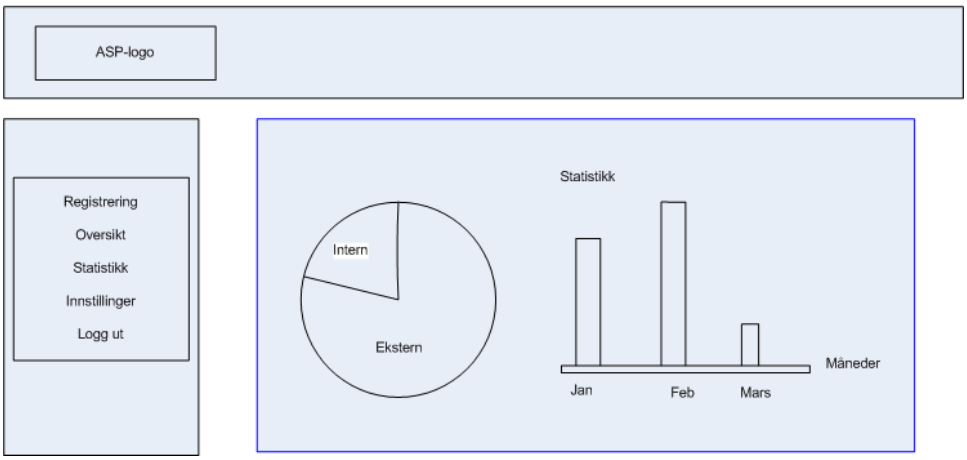
Her velger brukeren hvilken dato timen skal registreres på. Deretter hvilken kunde, kundeprosjekt og kontaktperson det skal være. Vare, en eventuelt egendefinering av varebeskrivelsen, og antall timer velges.

Oversikt

Figur 12: Skjemskudd av oversikt

Brukeren velger et ønsket datointervall, og får deretter listet opp alle registrete timer på vedkommende for denne perioden. Her kan brukeren velge å effektuere valgte timer, altså å sende de inn til ordresystemet i Nexstep. Timer som er allerede er effektuert kan ikke velges. Hvis en timeregistrering er feil, kan brukeren velge å slette eller å endre den.

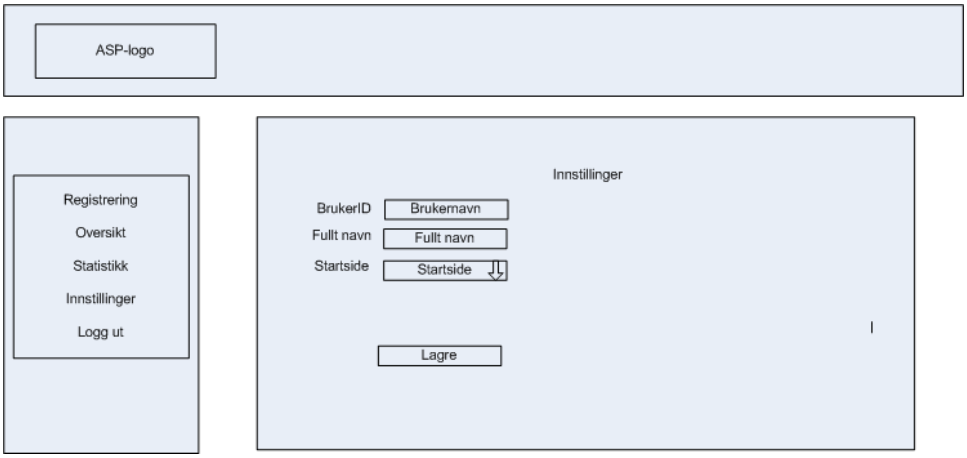
Statistikk



Figur 13: Skjemskudd av statistikk

Brukerene får se statistikk over antall timer vedkommende har registrert, sortert på totalt antall timer, timetype, per kunde og per kundeprosjekt. Dette vil vises ved hjelp av diagrammer.

Innstillinger

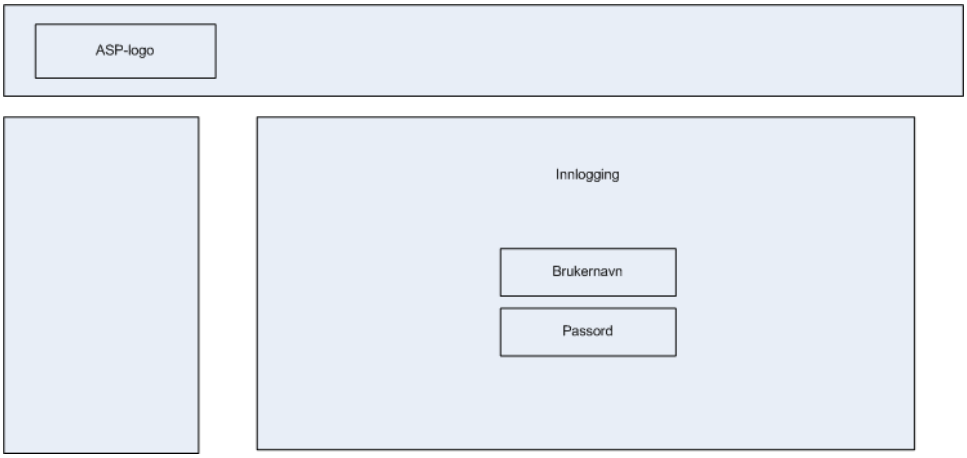


Figur 14: Skjemskudd av innstillinger

Her har brukeren diverse informasjon om seg selv, slik som brukerID og navn. Brukeren kan velge hva vedkommende vil se som startside. Dette kan være hovedside,statistikk, timeregistrering eller timeoversikt.

3.3.2 Brukergrensesnitt for *gruppe 2*

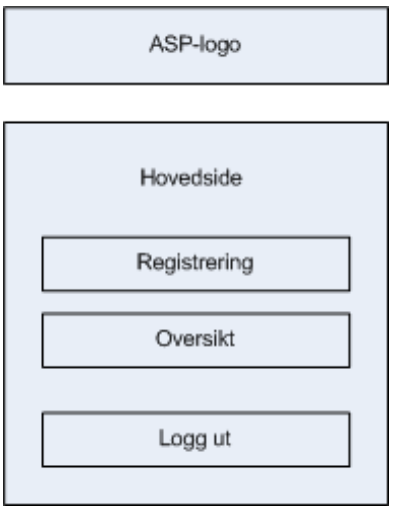
Innlogging



Figur 15: Skjemskudd av innloggingsbilde - mobil

Brukeren må autentisere seg for å få tilgang til systemet.

Hovedside



Figur 16: Skjemskudd av hovedmeny - mobil

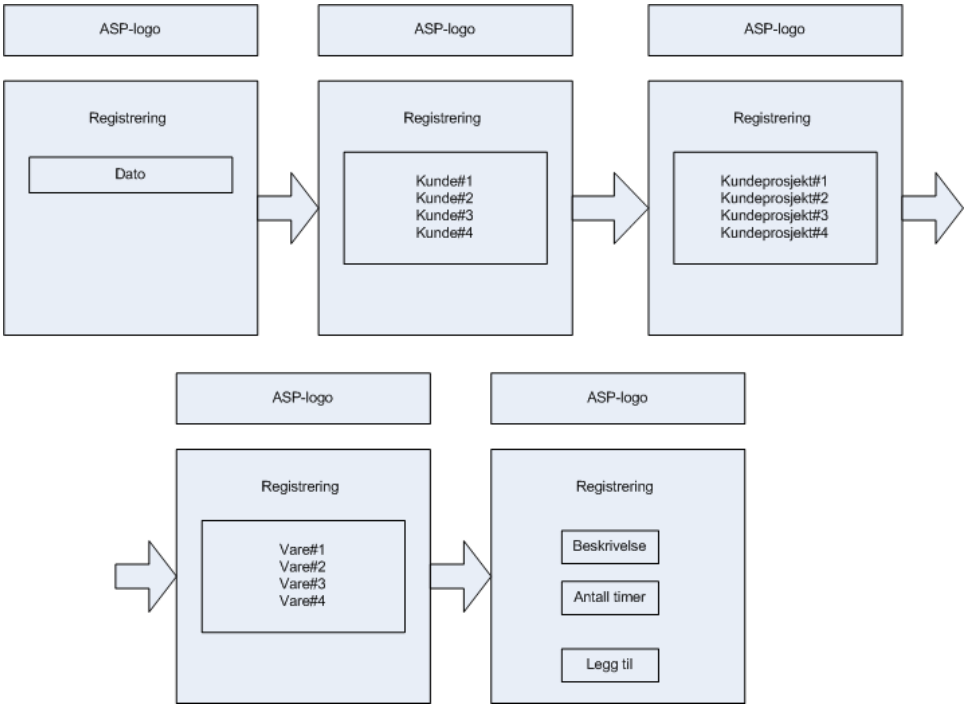
Her kan brukeren velge mellom følgende operasjoner:

Registrering
Legge til nye timer

Oversikt Hente siste registrerte timer.

Logg ut
Etter brukeren er ferdig, kan vedkommende logge seg ut av systemet.

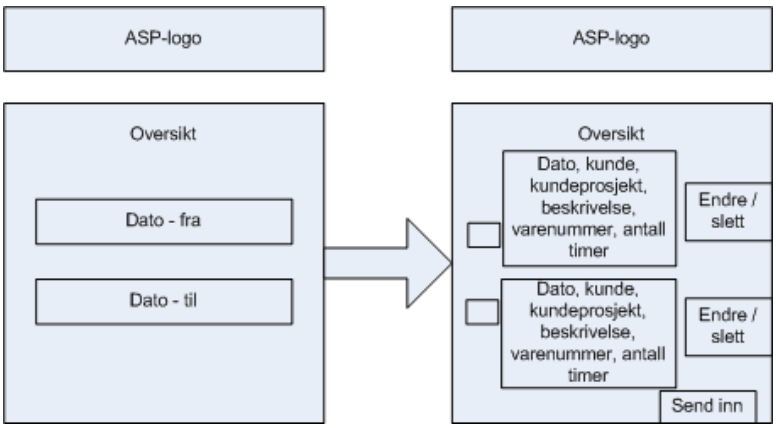
Registrering



Figur 17: Skjemskudd av registrering - mobil

Her velger brukeren hvilken dato timen skal registreres på. Deretter hvilken kunde, kundeprosjekt og kontaktperson det skal være. Vare, en eventuelt egendefinering av varebeskrivelsen, og antall timer velges. Disse operasjonene vises bilde for bilde, grunnet den lille skjermen på enhetene.

Oversikt

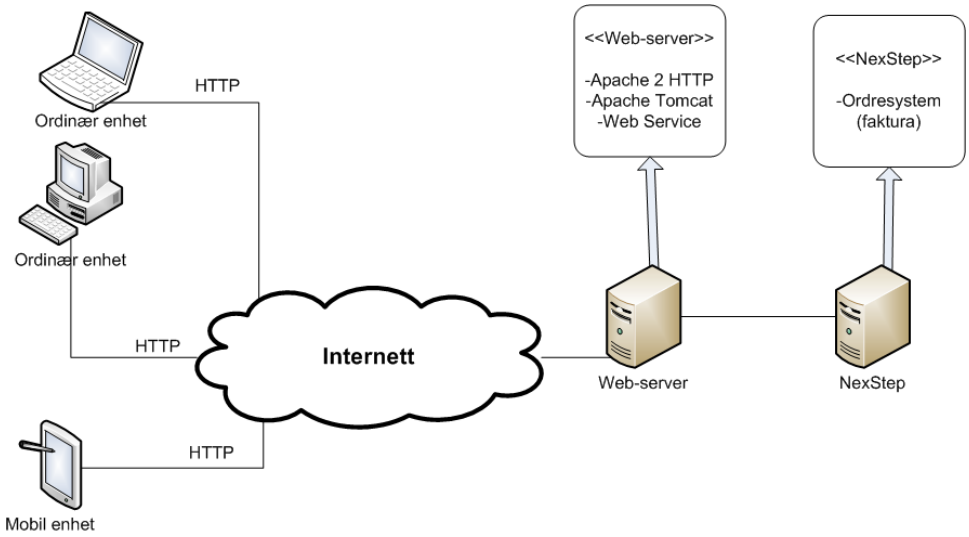


Figur 18: Skjemskudd av oversikt - mobil

Brukeren velger et ønsket datointervall., og får deretter listet opp alle registrete timer på vedkommende for denne perioden. Her kan brukeren velge å effektuere valgte timer, altså å sende de inn til ordresystemet i Nexstep. Timer som er allerede er effektuert kan ikke velges. Hvis en timeregistrering er feil, kan brukeren velge å slette eller å endre den.

Datosøk vises i et eget vindu, mens oversikten viser de ti første treffene. Brukeren kan velge å se flere treff, men dette vises over flere sider.

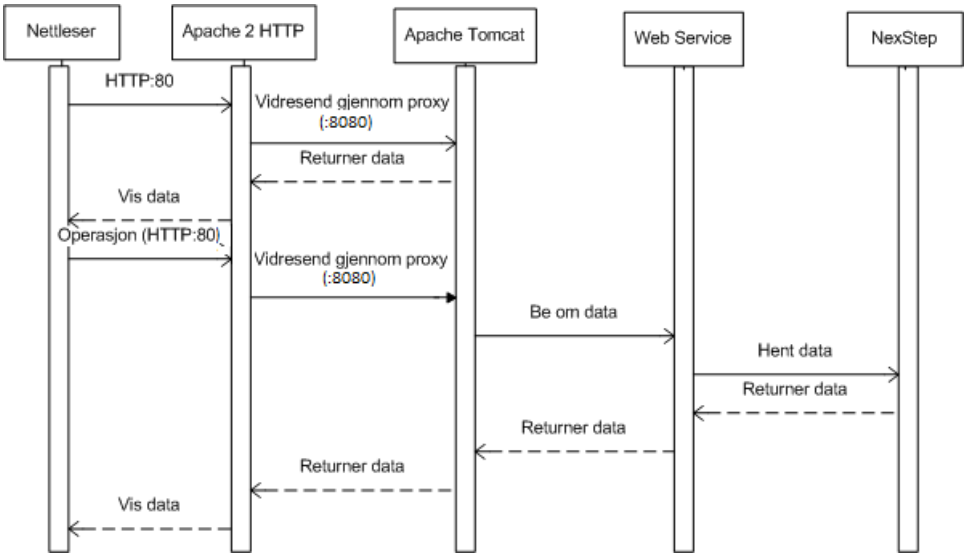
3.4 Utvidet deployment view



Figur 19: Deployment view

Enheter kobler seg opp mot web-serveren over internett, med HTTP. Web-serveren inneholder Apache 2 HTTP-server, Apache Tomcat-server og Web Service. Bak web-serveren står Nexstep-serveren, med blant annet ordresystemet.

3.5 Sekvensdiagram



Figur 20: Sekvensdiagram

Denne modellen viser en tenkt kommunikasjonsflyt innad i systemet.

Enhetene som kjører applikasjonen, nettleseren, kobler seg opp mot web-serveren med HTTP. Denne kommunikasjonen går inn til Apache 2 HTTP-serveren på port 80, og vidresender dette gjennom en proxy-tunell til Tomcat-serveren, på port 8080. Tomcat-serveren viser informasjonen på nettleserene hos enhetene, og henter informasjon fra Web Service, hvis brukeren gjør en operasjon. Web Service henter eller sender informasjon til eller fra Nexstep. Dette kan være innhenting av kunder, kundeprosjekter, kontaktpersoner, brukernavn, og annen relevant data. Sending til Nexstep er overføring av registrete timer, som er klar til fakturering.

4 Implementering

4.1 Utviklingsmiljø- og språk

Oppdragsgiver benytter seg av Java, så dette var et krav til utviklingsspråk. Serversiden av applikasjon er skrevet i Java, både på den ordinære og mobile delen av applikasjonen. På brukergrensesnittdelen er det brukt HTML, CSS og JavaScript, bortsett fra mobilapplikasjonen, som ikke benytter seg av JavaScript.

Vi benytter oss også av XSLT og SQL.

- CSS brukes for å forme stilen i forskjellige nettlesere.
- XSLT brukes for å transformere XML til HTML.
- SQL brukes for databasespørringer.
- HTML brukes for å vise data på applikasjonene (ordinær og mobil).

Vi benyttet oss ikke av noen ferdiglagede rammeverk, ettersom det ikke fantes noen som dekket våre behov. Dette ble derfor utviklet selv.

4.2 Verktøy

4.2.1 Utviklingsverktøy

Følgende utviklingsverktøy er brukt under prosjektet:

- Eclipse (programmering i overnevnte språk).
- Microsoft Office Visio (diagrammer og illustrasjoner)
- LaTeX (dokumentasjon)
- Planner (Gannt-diagram)
- phpMyAdmin (PHP-verktøy til mySQL)

4.2.2 Versjonsstyring

Fra prosjektstart har benyttet vi oss av et versjonsstyringsverktøy, Subversion. Dette har blitt brukt på både utviklingsdelen og dokumentasjonsdelen. Begge disse versjonshåndteringssystemene ble satt opp av IT-tjenesten på HiG.

Ved bruk av disse systemene har vi god sikkerhet av informasjonen. HiG har gode sikkerhetskopieringsrutiner, og ved benyttelse av overnevnte versjonshåndteringssystemer vil dette også fungere som en sikkerhetskopi av all informasjon på våres maskiner.

4.2.3 Læring

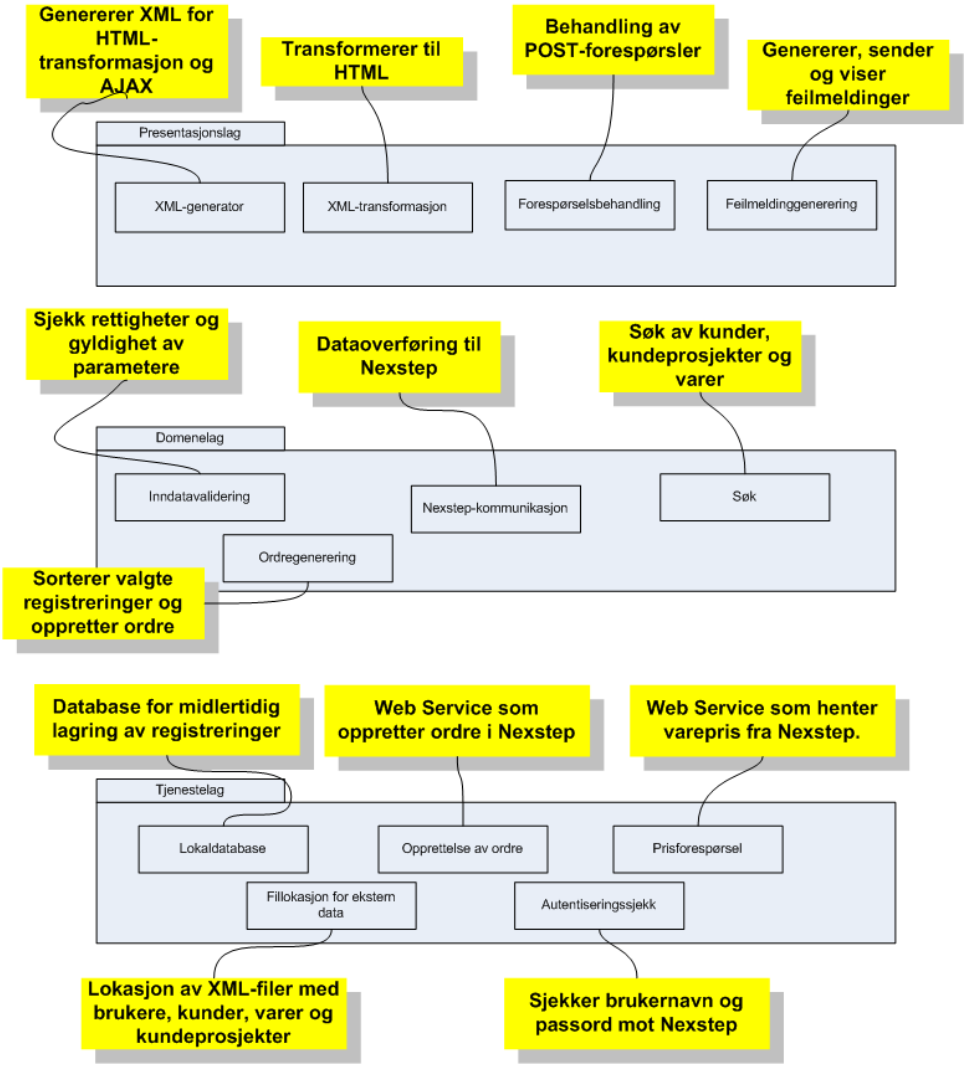
Èn av gruppemedlemmene har svært god erfaring innenfor Java. Dette har ført til at resterende medlemmer har hatt en bratt læringskurve innenfor språket. Ellers har det vært lite nytt innenfor HTML, SQL og CSS, men noe nytt i XSLT.

Vi har lært gode teknikker og rutiner for å implementere de andre utviklingsspråkene under Java, og vi har utviklet gode algoritmer for å få systemet til å kjøre hurtigst mulig.

Den måten vi arbeidet på har gjort oss godt kjent med valgt utviklingsmodell, og har gitt oss mye erfaring med arbeidsmåten.

4.3 Lagdelingsmodellen

Vi valgte å implementere lagdelingsmodellen som følger:



Figur 21: Lagdelingsmodell

4.3.1 Presentasjonslaget

Vi ønsker å bevare egenskapen til de stasjonære enhetene som har en forholdsvis stor skjerm, slik at vi kan vise mest mulig relevant data om gangen. For de mobile enhetene må vi ta hensyn til de små skjermene, og vil derfor vise minst mulig informasjon per skjermbildet. Her informasjonen bli vist på flere sider.

Presentasjonslaget sørger for å generere HTML fra XML og XSL, slik at nettleserene får vist informasjonen. JavaScript sørger for dynamisk henting av data, generering av lister og

tastatursnarveier.

Felles for begge enhetstypene er behandling av inndata, registreringer og søk.

4.3.2 Domenelaget

Domenelaget består av validering av inndata, midlertidig og permanent lagring av registreringer, søk etter data for bruk i lister, kommunikasjon mot Nexstep og generering av ordre.

Inndatavalidering sørger for at alle nødvendige parametere er med ved registrering av timer. Etter dette vil ordregenereringen samle alle registreringer i XML-dokumenter, og legge disse i aktuelle ordre. Kommunikasjonen mot Nexstep sender disse til ordresystemet, Nexstep, til fakturering. Denne kommunikasjonen henter også inn data fra Nexstep, som kunder, kundeprosjekt, varer og brukere.

Søkefunksjonaliteten filtrerer kunder, kundeprosjekter og varer etter innskrevet inndata. Denne søker opp data i databasen, i tjenestelaget.

4.3.3 Tjenestelaget

Tjenestelaget inneholder lokaldatabasen, som inneholder all data systemet lagrer. Her ligger det også fillokasjon for XML-filer fra oppdragsgiver, som inneholder all informasjon av brukere, kunder, kundeprosjekter og varer.

Dette laget benytter seg også av en Web Service hos ASP som oppretter nye ordre i Nexstep, sjekker autentiseringsinformasjon fra brukeren og prisforespørsel på varer.

4.4 Brukergrensesnitt

4.4.1 Stasjonære enheter

Innlogging

Autentiseringen går mot Nexstep. Brukernavn og passord sjekkes opp mot denne databasen, og legges til i den lokale databasen hvis autentiseringen er gyldig. Brukeren blir deretter logget inn ved hjelp av en session ID.

Figur 22: Skjemskudd innlogging

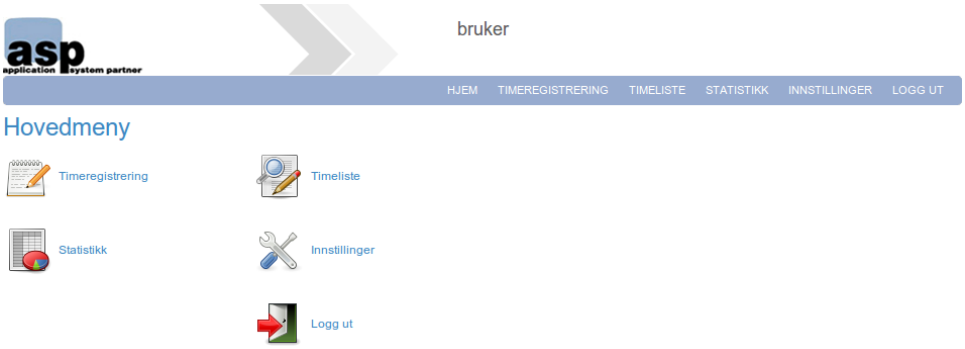
Listing 4.1: Autentisering mot Nexstep

```
1  public class LoginAction extends PostAction {
2    ...
3
4    @Override
5    public void doAction(FormContainer container) {
6      ...
7
8      String username = container.getParameter("username");
9      String password = container.getParameter("password");
10     String redirect = "/";
11
12     try {
13       User user = UserManager.checkLogin(username, password);
14       if (user != null) {
15         String lastVisit = SessionUtil.getLastVisit();
16         SessionUtil.invalidateSession();
17         SessionUtil.setUserPreferences(UserManager.getUserPreferences(username));
18         SessionUtil.setUser(user);
19         SessionUtil.setUserCredentials(WebservicesUtil.createUserCredentials(
               username, password));
20
21         if(MobileUtil.isMobile())
22           redirect = "/mobile";
23         else if (lastVisit != null)
24           redirect = lastVisit;
25         else
26         redirect = SessionUtil.getUserPreferences().getStartPage();
27         response.sendRedirect(redirect);
28       } ...
29     } ...
30   }
31 }
32
33 public static User checkLogin(String username, String password) throws ... {
```

```
34    UserCredentials credentials = new UserCredentials();
35
36    if(username != null)
37      credentials.setExternalUserId(WebservicesUtil.modelFactory.
            createUserCredentialsExternalUserId(username));
38
39    if(password != null)
40      credentials.setPassword(WebservicesUtil.modelFactory.
            createUserCredentialsPassword(password));
41
42    User user = WebservicesUtil.getPort().checkLogin(credentials);
43    return user;
44  }
```

Denne koden viser to utdrag; ett fra post-håndtering og ett fra autentiseringssjekken mot Nexstep-databasen.

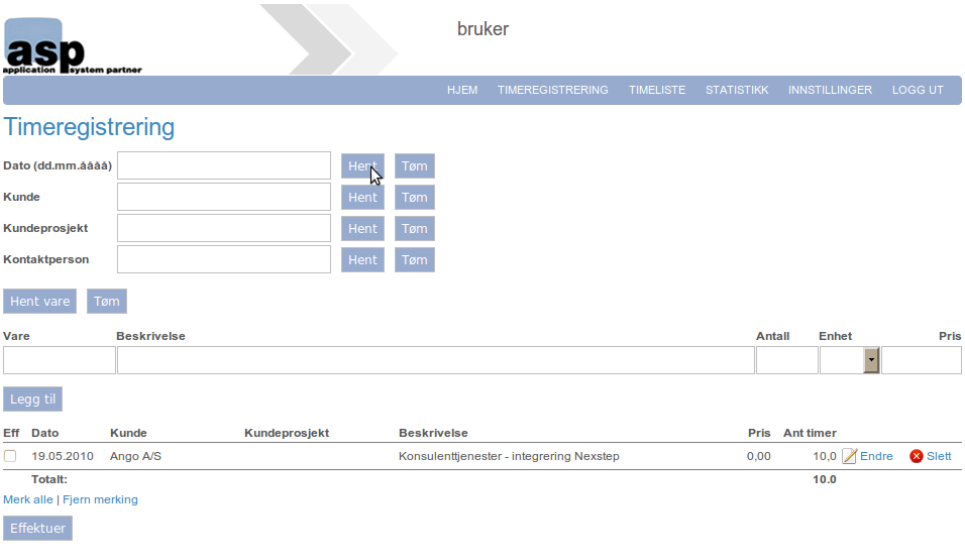
Forsiden / hovedmeny



Figur 23: Skjemskudd av hovedmeny

Dette bildet gir brukeren et overblikk over funksjonalitet i applikasjonen. Denne menyen kan også nås øverst på siden.

Timeregistrering



Figur 24: Skjemskudd av timeregistrering

Brukeren kan enten trykke på hent eller tøm i boksene, dato, kunde, kundeprosjekt, kontaktperson og vare. Dette er styrt med JavaScript. Alle hent-funksjonene benytter seg av AJAX, herunder XML. Ved å trykke hent eller enter kommer det opp et vindu som lar brukeren velge aktuell data. Hvis brukeren velger tøm, vil den aktuelle boksen tømmes ved hjelp av JavaScript.

Hver gang brukeren benytter seg av et tastetrykk, vil global event listener finne ut om noe benytter denne snarveien, og vidresende til aktuell funksjon, som igjen behandler event'et videre. Dette kan være enter, piltaster eller f-taster.

I beskrivelsefeltet hentes det ut standarbeskrivelsen utifra valgt vare. Denne teksten kan endres om brukeren ønsker det. Isåfall overskrives den standarbeskrivelsen beskrivelsen. Brukeren velger så antall timer for denne varen.

Prisen per time hentes ut fra en rabattmatrise på en Web Service hos oppdragsgiver. Denne prisen er basert på kunde, kundeprosjekt og vare, men det er kun vare som er påkrevd. Hvis ingen rabatt eksisterer, returneres NULL, og ingen pris blir satt opp i boksen. Denne prisen kan overstyres fra brukeren om ønskelig.

Ved å trykke legg til, eller enter i prisboksen. AJAX sender registrert data til web-klienten, som vidresender til Web Service. Web Service lagrer den aktuelle innføringen i databasen.

De siste registrete timene vises nederst på siden. Disse hentes ut ved hjelp av AJAX, og kommer i XML-format.

All XML som kommer i retur, parses over til JavaScript-objekter. Alle AJAX-forespørsler vil returnere resultatet eller feilmeldinger. Disse feilmeldingene vises på toppen av skjermbildet.

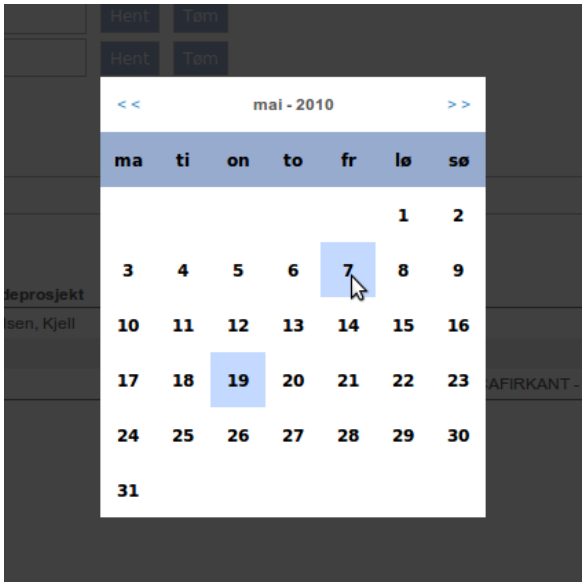
Brukeren kan velge å endre eller slette ønskede innføringer. Hvis brukeren velger å endre en time, vil AJAX hente ut informasjonen om innføringen og legge informasjonen i feltene ovenfor. Ved sletting av en innføring vil AJAX sende timeinnføringsID'en til web-klienten, som igjen sender den til Web Service. Denne sletter hele innføringen.

Ved å effektuere innføringer vil AJAX sende alle valgte innføringer til web-klienten. Denne vil igjen sende til Web Service, som igjen sorterer innføringene basert på kunde, kundeprosjekt og ordretype (interne,ikke-fakturerbare og fakturerbare timer). Disse sendes til en Web Service hos oppdragsgiver, som igjen legger innføringene inn i ordresystemet, Nexstep. Denne Web Service'en returnerer resultatet på operasjonen.

Listing 4.2: Hente kundeliste og lage XML

```
1  public static Element createTimeListXML(List<TimeElement> timeList) {
2    Element timeListXML = new Element("timeElementList");
3
4    // Returnerer tom XML om lista ikke inneholder elementer
5    if (timeList == null || timeList.isEmpty())
6      return timeListXML;
7
8    for (TimeElement timeElement : timeList)
9      timeListXML.addContent(createTimeElementXML(timeElement));
10   return timeListXML;
11 }
12
13 public static Element createTimeElementXML(TimeElement timeElement) {
14   Element timeElementXML = new Element("timeElement");
15
16   if(timeElement == null)
17     return timeElementXML;
18
19   timeElementXML.addContent(new Element("id").addContent(String.valueOf(timeElement
         .getId().getValue())));
20   timeElementXML.addContent(new Element("itemId").addContent(String.valueOf(
         timeElement.getItem().getValue().getId().getValue())));
21   timeElementXML.addContent(new Element("itemNr").addContent(String.valueOf(
         timeElement.getItem().getValue().getItemNr().getValue())));
22   timeElementXML.addContent(new Element("itemDescription").addContent(String.
         valueOf(timeElement.getDescription().getValue())));
23   timeElementXML.addContent(new Element("customerId").addContent(String.valueOf(
         timeElement.getCustomer().getValue().getId().getValue())));
24   timeElementXML.addContent(new Element("customerName").addContent(String.valueOf(
         timeElement.getCustomer().getValue().getName().getValue())));
25
26   ...
27
28   return timeElementXML;
29 }
```

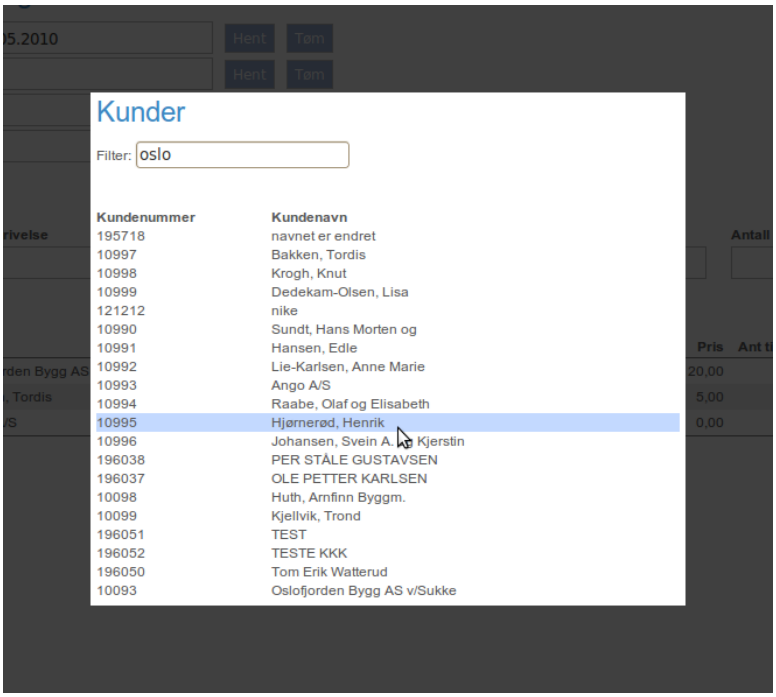
Kalender - valg av dato



Figur 25: Skjemskudd av kalender

Når brukeren trykker enter eller hent i datoboksen, vil dette vinduet komme opp. Brukeren kan navigere seg rundt med piltaster eller ved direkte trykk med musen. Navigasjonen foregår vertikalt eller horisontalt. Ved å navigere utenfor den aktuelle måneden, vil man nå forrige eller neste månede. Valgt dato merkes med en blå firkant rundt. Global event listener sender alle tastetrykk til event handler i kalenderen.

Kunde - innhenting av kunder



Figur 26: Skjemskudd av kundeliste

AJAX henter ut kundelisten basert på et tekstfilter fra web-klienten. Denne blir returnert som XML, parset om til objekter med kunder. Hvis en feil oppstår returneres det feilmeldinger i form av XML, som parses og printes på toppen av siden. Vinduet lukkes etter valgt kunde.

Listing 4.3: Hente kundeliste og lage XML

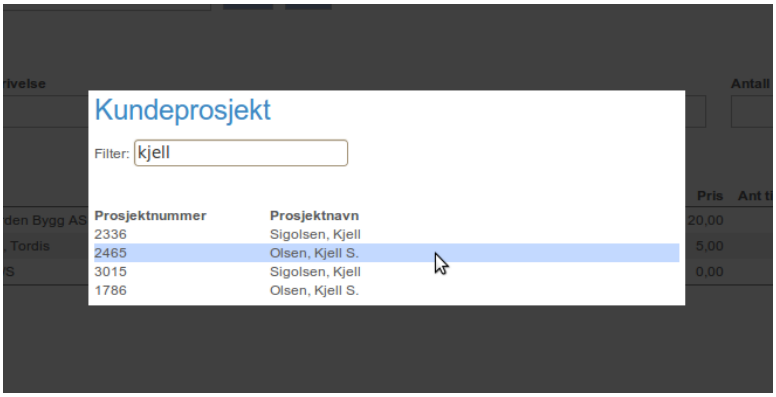
```
1  public class AjaxCustomerListAction extends SecureXML {
2    @Override
3    public void doAction() {
4      super.doAction();
5      List<Customer> customerList;
6
7      // Henter kundeliste fra ws
8      try {
9        customerList = CustomerManager.getCustomerList(SessionUtil.getUserCredentials
             (), 0, 20);
10       xmlRoot.addContent(XMLUtil.createCustomerListXML(customerList));
11     } catch (UserValidationException_Exception e) {
12       pageError.addError(ResourceBundleUtil.getString("exception_UserValidation"));
13       StackTracePrinter.debug(e);
14     } catch (NullPointerException
15       ...
16     } finally {
17       if (pageError.isErrors())
18         xmlRoot.addContent(pageError.writeAll());
19
20       writeXML();
21     }
```

```
22      }
23  }
```

Denne koden viser et utdrag av hvordan en kundeliste hentes ut ifra Web Service, og hvordan feilmeldinger blir behandlet.

Kundeprosjekt - innhenting av kundeprosjekt

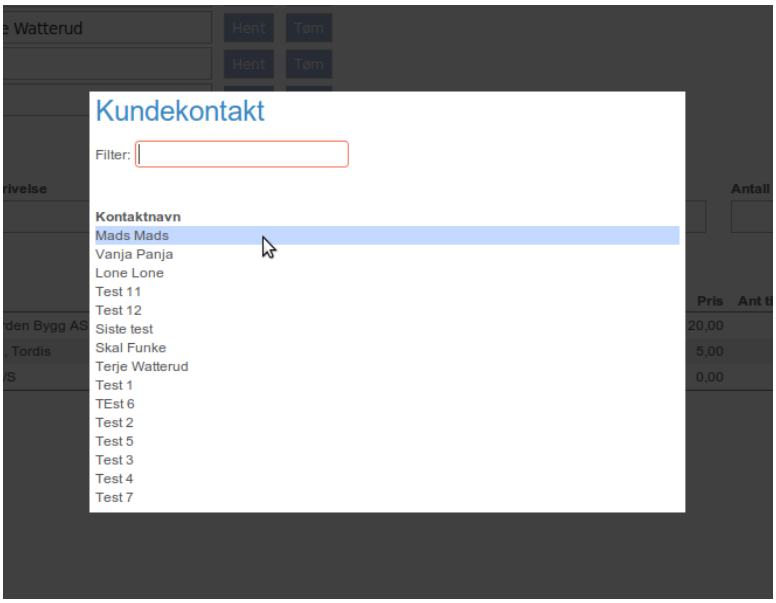
Kundeprosjektlista hentes på grunnlag av hvilke kunde som er valgt. Den generelle funksjonaliteten hel helt lik som kundelisten ovenfor.



Figur 27: Skjemskudd av kundeprosjektliste

Kontaktperson - innhenting av kontaktperson

Denne listen hentes basert på kunde, siden det kan være forskjellige kontaktpersoner pr. kunde. Generell funksjonalitet er lik.



Figur 28: Skjemskudd av kundekontaktliste

Timeliste

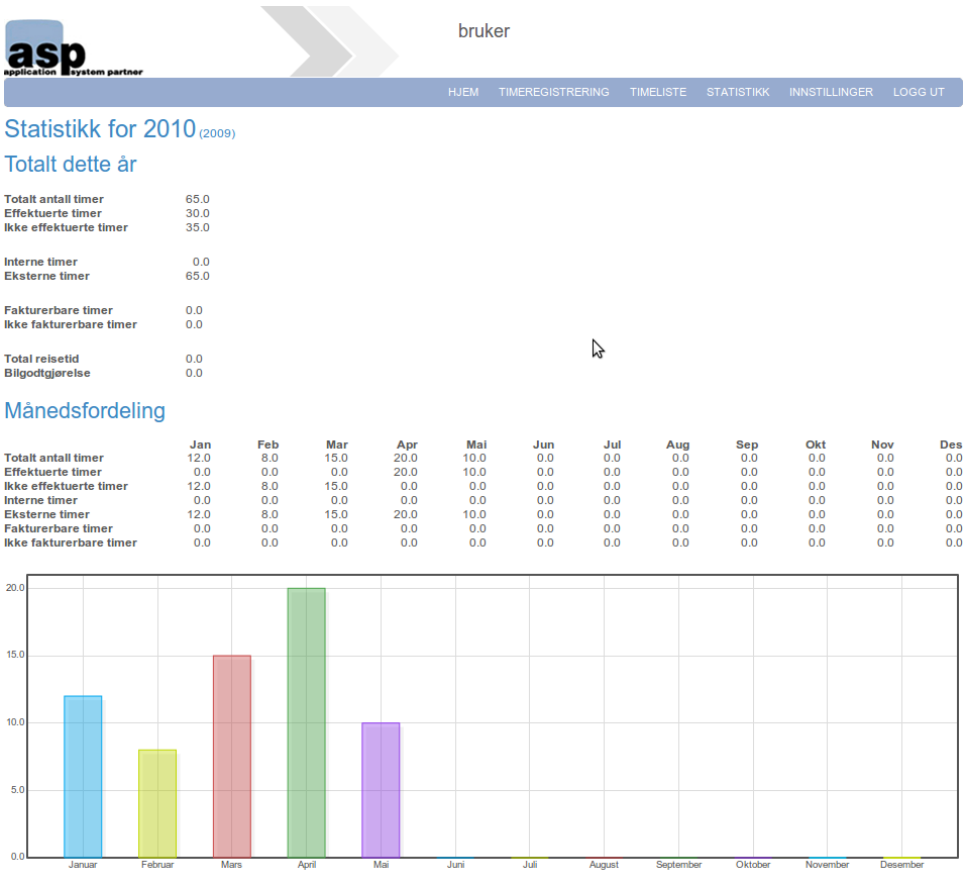
Figur 29: Skjemskudd av timeliste

Alle boksene fungerer likt som timeregistreringen. Brukeren kan hente ut innføringer basert på dato, kunde, kundeprosjekt, vare, effektuerte, ikke-effektuerte eller en sammensetning av flere av disse paramenterene.

Ved trykk på hent liste, vil AJAX sende parameterene til web-klienten, som igjen sender dette til Web Service. Web Service gjør oppslag mot lokaldatabasen med de gitte parameterene. Dette oppslaget returneres som XML, og genereres.

Brukeren kan her, som på timeregistreringen, endre, slette eller effektuere innføringene. Denne prosessen er lik som på timeregistreringen, bortsett fra ved endre vil brukeren bli sendt til timeregistreringen.

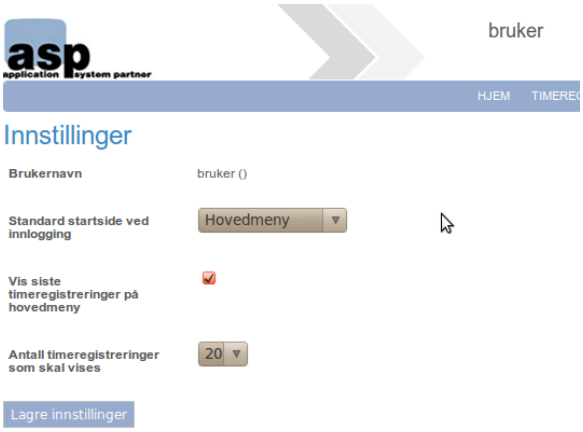
Statistikk



Figur 30: Skjemskudd av statistikk

Her får brukeren en oversikt over totalt registrerte timer over hele året og måneder. Det vises også et diagram over månedestatistikken. Brukeren kan enkelt se hvor mange timer som er ikke-fakturerbare og

Innstillinger



Figur 31: Skjemskudd av innstillinger

Her får brukeren mulighet til å endre startsiden og antall timer som skal vises på timelisten.

Ved å endre på disse innstillingene vil databasen til web-klienten oppdateres med disse endringene for den aktuelle brukeren.

4.4.2 Mobile enheter

Den mobile applikasjonen har lik funksjonalitet som den ordinære applikasjonen, men viser mindre informasjon på skjermen. Dette for å tilpasse det til små skjermer.

Innlogging

Figur 32: Skjemskudd av innlogging for mobil

enne benytter seg det samme innloggingsvinduet som den ordinære applikasjonen. Etter innlogging vil brukeren bli vidresendt til mobilapplikasjonen.

Hovedmeny

Figur 33: Skjemskudd av hovedmeny for mobil

Hovedmenyen viser funksjonalitete for applikasjonen. Størrelsen på dette er tilpasset skjermen på alle mobile enheter.

Timeregistrering

Figur 34: Skjemskudd av timeregistreringssekvens på mobil

Ved å trykke på timeregistrering vil brukeren få vist dette bildet. Her foregår timeregistreringen over flere sider, for å gi brukeren et enkelt og oversiktlig grensesnitt å jobbe mot.

Brukeren blir presentert med de tyve første kundene. Web-klienten spør Web-Service om de første kundene, og disse blir sendt tilbake som objekter, og deretter transformert til HTML.

Hver kunde er trykkbar, og ved trykk lagres kundeID'en i URL'en, og sendes videre til neste bilde.

Timeregistrering sluttføring

Figur 35: Skjemskudd av siste timeregistrering bilde for mobil

Datoen settes automatisk til dags dato, men kan endres på med rullegardinmeny. Varebeskrivelsen blir automatisk satt til standarbeskrivelsen, men dette kan overstyres i feltet. Antall timer skrives inn, og prisen hentes ut på kunde, kundeprosjekt og vare, der vare er den eneste påkrevde i kalkulasjonen.

Timeliste

Timeliste

Siste innføringer i timesystemet. Antall er basert på instillinger.

Dato	Kunde	Kundeprosjekt	Kontaktperson	Beskrivelse	Antall timer
17.02.2010	Bakken, Tordis			FORING NORMAL 27/27-50 PLEWA LECAFIRKANT - UGLASERT	8.0
01.01.2010	navnet er endret			ADAPTER F BITS 1/84 BOSCH3064	12.0
10.03.2010	navnet er endret			ADAPTER F BITS 1/84 BOSCH3064	15.0
14.04.2010	navnet er endret			ADAPTER F BITS 1/84 BOSCH3064	20.0
19.05.2010	Ango A/S			Konsulenttjenester - integrering Nexstep	10.0

Figur 36: Skjemskudd av timeliste for mobil

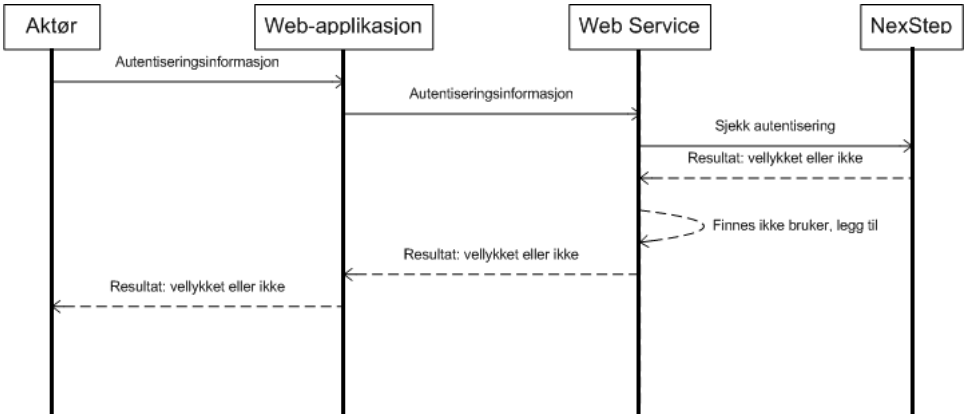
Her får brukeren en oversikt over de siste registrerte innføringene. Antallet innføringer er basert på innstillingene brukeren har konfigurert selv.

Web-klienten spør Web-Service om de første innføringene, og disse blir sendt tilbake som objekter, og deretter transformert til HTML.

4.5 Brukerhåndtering

Det finnes kun ordinære brukere i systemet, ettersom det ikke er behov for høyere tilgang, eller administrative funksjoner. Oppretting av brukere er løst på følgende måte:

- Brukeren skriver inn brukernavn og passord på innloggingsvinduet.
- Systemet sjekker brukernavn og passord opp mot Nexstep.
- Hvis informasjonen stemmer overens, vil systemet opprette denne informasjonen i sin lokale database.

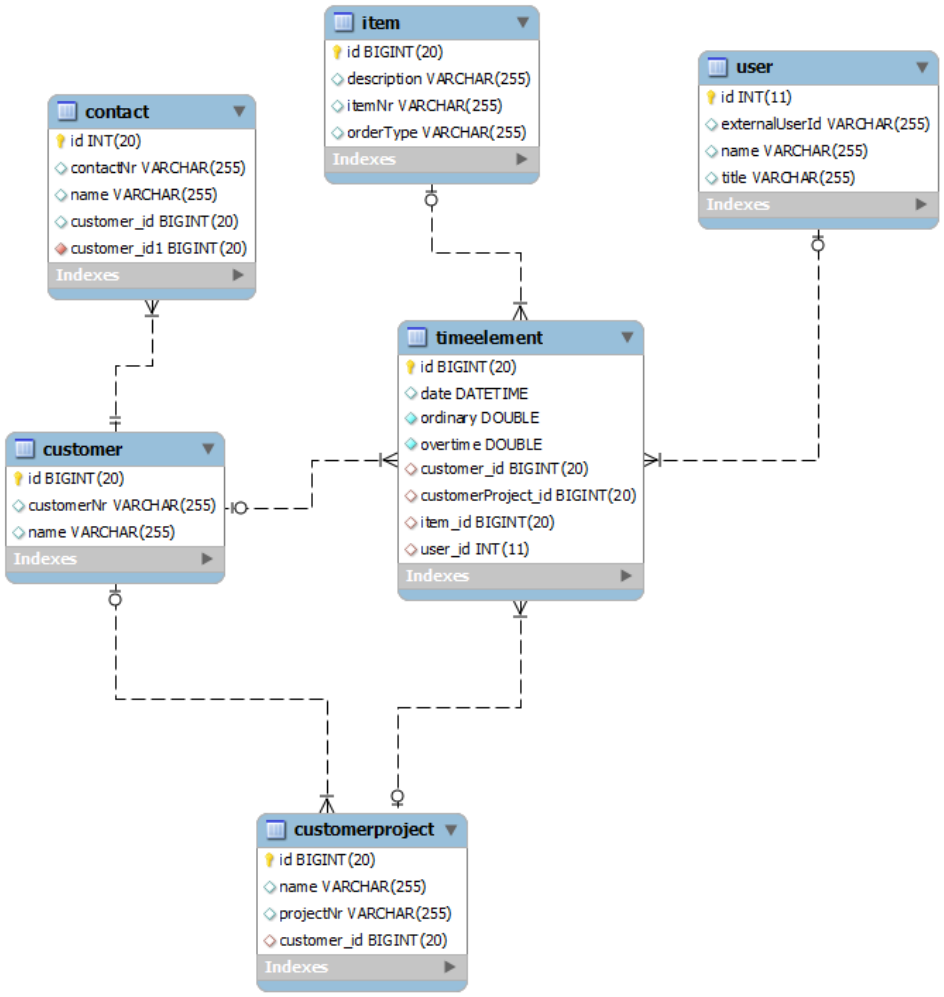


Figur 37: Sekvensdiagram for innlogging

Denne løsningen gjør at det blir mindre kommunikasjon mot Nexstep, og systemet gjør denne operasjonen tilslutt på egenhånd.

Hvis brukeren skifter passord i Nexstep, vil systemet oppdatere denne informasjonen ved innlogging.

4.6 Databasemodell



Figur 38: Databasemodell

Denne figuren viser den faktiske implementeringen av domenemodellen til databasen.

4.7 Logging

I web-klienten loggføres det på to nivåer, debug og error. Debug er for feilsøking av problemer, mens error er når problemet faktisk har oppstått. Error og debug skrives til filer, men kun error vises til brukeren, øverst på siden.

Web Service logger ikke.

4.8 Kommunikasjon med Nexstep

4.8.1 Sending av timer til ordresystemet

Web Service sorterer innføringer som skal på samme ordre, og sender inn til ordresystemet gjennom en Web Service hos oppdragsgiver. Dette sendes på XML.

Web Service sjekker om kunde, kundeprosjekt og varens ordretype er likt, og sorterer det deretter i lister. Den første innføringen blir fjernet fra listen, og sammenlikner denne med alle de andre innføringene for å sjekke om dette er likt, og legger de sammen i en annen liste. Deretter repeteres denne prosessen til alle innføringer er sortert.

Web Service oppretter et ordreobjekt og legger til innføringene som varelinjer, og sender disse til Web Service hos Nexstep. Det blir deretter lagt inn i ordresystemet.

4.8.2 Innhenting av data

Dette foregår automatisk. Hvis oppdragsgiver har lagt til en ny XML-fil på fillokasjonen, vil systemet automatisk sjekke for dette hver natt. Hvis det oppdages en ny fil, vil denne parses, databasen oppdateres med ny data fra denne filen, og deretter blir filen slettet. Dette er implementert i Web Service'en.

For sjekking av nye filer benyttes det en Chron-jobb, som automatisk kjører hver natt. Denne kjører oppdateringen i Web Service'en.

Fillokasjonen er en nettverksmappe, Samba Share.

Listing 4.4: Henter kundeliste

```
1  public static Collection<ParsedCustomer> getCustomerList() {
2    InputStream in = null;
3    ResourceBundle r = ResourceBundle.getBundle("webservices", new Locale("nb", "NO")
         );
4    String loc = r.getString("kunde");
5
6    if(loc.equals(""))
7      in = RemoteDatabaseUtil.class.getClassLoader().getResourceAsStream("kunde.xml")
           ;
8    else {
9      try {
10       in = new FileInputStream(loc);
11     } catch (FileNotFoundException e1) {
12       return new ArrayList<ParsedCustomer>();
13     }
14   }
15
16   try {
```

```
17      CustomerPrintHandler handler = new CustomerPrintHandler();
18      SAXParserFactory factory = SAXParserFactory.newInstance();
19      SAXParser parser = factory.newSAXParser();
20      parser.parse(in, handler);
21      return handler.getCustomerList().values();
22    } catch(Exception e) {
23      return new ArrayList<ParsedCustomer>();
24    } finally {
25      if(in != null)
26        try {
27          in.close();
28        } catch (IOException e) {}
29    }
30  }
```

Dette kodeeksempelet viser hvordan henter inn nye kunder.

Listing 4.5: XML kundeliste fra ASP

```
1  <MESSAGE_RECORD>
2    <InsertChange>
3      <Message>
4        <t:MessageId>1</t:MessageId>
5        <t:FirmaNr>0</t:FirmaNr>
6        <t:Status>NY</t:Status>
7        <t:OpprettetEndretSlettet>
8          <t:OpprettetTidspunkt>1998-11-17</t:OpprettetTidspunkt>
9          <t:EndretTidspunkt>2010-05-03</t:EndretTidspunkt>
10         <t:EndretKl>16:01:24</t:EndretKl>
11         <t:EndretAv>FFF</t:EndretAv>
12       </t:OpprettetEndretSlettet>
13     </Message>
14     <EksterntKundeNr>12625</EksterntKundeNr>
15     <AvdelingNr>0</AvdelingNr>
16     <Kontonummer>0</Kontonummer>
17     <Ehandelskunde>false</Ehandelskunde>
18     <RabattKategorikodeId>1</RabattKategorikodeId>
19     <KundeKategorikodeId>1112</KundeKategorikodeId>
20     <BetalingsbetingelseKodeId>25</BetalingsbetingelseKodeId>
21     <SelgerKodeId>213</SelgerKodeId>
22     <Ordrerabatt>0</Ordrerabatt>
23     <Prisboktype>0</Prisboktype>
24     <MvaBeregning>0</MvaBeregning>
25     <KravTilRekvisisjon>0</KravTilRekvisisjon>
26     <Memosaldo>
27       <t:BelopFelt>0</t:BelopFelt>
28       <t:Valutakode>NOK</t:Valutakode>
29     </Memosaldo>
30     <MemosaldoAlm>
31       <t:BelopFelt>0</t:BelopFelt>
32       <t:Valutakode>NOK</t:Valutakode>
33     </MemosaldoAlm>
34     <Saldo>
35       <t:BelopFelt>-91543.93</t:BelopFelt>
36       <t:Valutakode>NOK</t:Valutakode>
37     </Saldo>
38     <SaldoIfjor>
39       <t:BelopFelt>7940.28</t:BelopFelt>
40       <t:Valutakode>NOK</t:Valutakode>
41     </SaldoIfjor>
42     <KjopIaar>
```

```
      <t:BelopFelt>131292</t:BelopFelt>
      <t:Valutakode>NOK</t:Valutakode>
    </KjopIaar>
    <KjopIfjor>
      <t:BelopFelt>23239.5</t:BelopFelt>
      <t:Valutakode>NOK</t:Valutakode>
    </KjopIfjor>
    <KjopIaarAlm>
      <t:BelopFelt>0</t:BelopFelt>
      <t:Valutakode>NOK</t:Valutakode>
    </KjopIaarAlm>
    <BelopgrenseAlm>
      <t:BelopFelt>0</t:BelopFelt>
      <t:Valutakode>NOK</t:Valutakode>
    </BelopgrenseAlm>
    <Kredittgrense>
      <t:BelopFelt>50</t:BelopFelt>
      <t:Valutakode>NOK</t:Valutakode>
    </Kredittgrense>
    <KredittTid>0</KredittTid>
    <Distrikt>1</Distrikt>
    <Sperrekode></Sperrekode>
    <Kreditsperre>false</Kreditsperre>
    <NavnOgAdresse>
      <t:Navn>ARNE M. OTRA</t:Navn>
      <t:Adresse1>FELTSPADEVEIEN 12</t:Adresse1>
      <t:Postnr>2830</t:Postnr>
      <t:Poststed>OTTA</t:Poststed>
      <t:Telefon></t:Telefon>
      <t:Epostadresser>
        <t:Epostadresse>
          <t:Adresse>test@example.com</t:Adresse>
          <t:Beskrivelse></t:Beskrivelse>
          <t:Standard>true</t:Standard>
        </t:Epostadresse>
      </t:Epostadresser>
    </NavnOgAdresse>
  </InsertChange>
</MESSAGE_RECORD>
```

4.9 Web-sideoptimalisering

I vårt prosjekt har vi gått ett steg nærmere optimalisering av selve web-løsningen. Vi har brukt hjepemiddelet "Page Speed" fra Google, og kodekompressoren fra "YUI Library", utviklet av Yahoo!.

Page Speed er hovedsaklig et diagnostiserings- og analyseverktøy for optimalisering av web-sider. Optimaliseringen er basert på riktig caching, korrekt webserver konfigurasjon, effektiv JavaScript og CSS kode, komprimerte bilder, osv.

YUI Compressor er et verktøy for å optimalisere og komprimere JavaScript og CSS. Vi har her videreutviklet vår egen løsning av dette verktøyet for å imøtekomme kravene fra Page Speed.

"Best Practices" for web-ytelse har Google delt opp i fem kategorier:

- Optimalisere caching
- Minimalisere round-trip time (RTT)
- Minimalisere overlapping av forespørsler (request)
- Minimalisere overflødig svar (response)
- Optimalisere nettleser-rendering

Vi har valgt å fokusere på caching, minimalisering av svar (response) og minimalisering av RTT. Valgene er gjort på grunnlag av bruksmetode av web-applikasjonen.

Optimalisere caching
Mye av innholdet i en web-side er ressurser som ikke endres for hver forespørsel; som CSS, bilder og JavaScript og liknende.

Disse ressursene tar ofte lang tid å last ned over nettverk, og fører ofte til at tiden som brukes på å laste web-siden øker.

Lokal mellomlagring (caching) av disse ressursene; enten i nettleseren eller i en proxy, gjør at ressursene kan hentes lokalt - i steden for å måtte laste dem ned for hver gang.

Vi har implementert caching for de fleste ressurser på siden; CSS, JS, HTML, XML og bilder. Alle ressursene har en expiration-tid på ett år siden dette er noe som skjeldent eller aldri endrer seg (*ett år er anbefalt fra "Page Speed"*).

Dersom det eventuelt skulle endres litt i en JavaScript-fil, og applikasjonen må deployes på nytt - settes "Last-Modified" til korrekt parameter. Browsere eller proxy-servere, som da laster siden, ser at filen er endret og vil laste ned den nye versjonen.

Under er vår implementering av caching; hvor vi setter både “Expires” og “Cache-Control” til ett år i header.

Listing 4.6: Riktig caching av ressurser

```
1 // Setter cache expire
2 GregorianCalendar cal = new GregorianCalendar();
3 cal.add(GregorianCalendar.YEAR, 1);
4
5 // Setter header-cache
6 response.setHeader("Expires", cal.getTime().toString());
7 response.setHeader("Last-Modified", ServletLifetimeUtil.deploymentDate.toString());
8 response.setHeader("Cache-Control", "public, max-age=31536000");
```

Minimalisere overflødig svar (response) Denne kategorien går ut på å minimalisere data som er sendt fra server til klient. Her er det mange feller å gå i og utfallet kan være stort.

Et viktig punkt er å komprimere all data som sendes. Dette skjer ved hjelp av *gzip* eller *deflate* (to metoder å komprimere data på; gzip er basert på deflate).

Men det hjelper ikke bare å komprimere all data som sendes, en må også vite at dataene som mottas kan dekrypteres. Dette er noe de fleste moderne nettlesere i dag støtter, men en kan sjekke om det er støtte for det i resquesten fra klienten.

Vi har valgt å implementere dette ved å sjekke om klientens nettleser støtter gzip-komprimering (“accept-encoding”-header som er i requesten). Dersom nettleseren støtter gzip, komprimeres dataene - hvis ikke sendes dataene som ren tekst.

Listing 4.7: Komprimere med gzip

```
1 String acceptEncoding = request.getHeader("accept-encoding");
2
3 response.setContentType("text/html");
4 response.setCharacterEncoding("utf-8");
5
6 \\ Sjekker om browseren stotter gzip
7 if (acceptEncoding != null && acceptEncoding.indexOf("gzip") != -1) {
8   response.setHeader("Content-Encoding", "gzip");
9   gzOut = new GZIPOutputStream(response.getOutputStream());
10  xmlResult = new StreamResult(gzOut);
11  transformer.transform(xmlSource, xmlResult);
12  gzOut.close();
13 } else {
14  // Browseren stotter ikke gzip-komprimering, print ut pa vanlig mate.
15  xmlResult = new StreamResult(response.getOutputStream());
16  transformer.transform(xmlSource, xmlResult);
17 }
```

Vi har også fokusert mye på minimalisering av både CSS- og JavaScript-kode.

Minimalisering av kode (minify), går ut på å fjerne all unødvendig kode, mellomrom, line-breaks, ol.

Listing 4.8: Eksempel på minimalisering av kode

```
1  var minimum, maksimum, input;
2
3  input = 4;
4  minimum = 1;
5  maksimum = 12;
6
7  if(input <= maksimum && input >= minimum) {
8      document.write("Input er innenfor min/maks");
9  } else {
10     document.write("Input er ikke innenfor min/maks");
11 }
12
13 var minimum,maksimum,input;input=4;minimum=1;maksimum=12;if(input<=maksimum&&input
       >=minimum){document.write("Input er innenfor min/maks")}else{document.write("
       Input er ikke innenfor min/maks")};
```

Minimalisere round-trip time

Dette er tiden det tar for klienten å sende en request, og serveren å sende en response, altså ikke tiden det tar å sende/hente dataene over nettverk. Typisk RTT kan være DNS name resolution, TCP oppkobling, selve HTTP responsen osv.

RTT kan variere fra mindre enn ett millisekund til over ett sekund.

For vår del er dette snakk om å kombinere CSS- og JavaScript-filer.

Til sammen i prosjektet har vi over 25 filer av CSS og JavaScript, og i et "worst-case-scenario" vil disse 25 filene tilsvare nærmere et halvt minutt i round-trip time, og det er uten beregning av tiden klienten bruker på å laste ned filene.

For å forhindre dette "worst-case" scenarioet kombinerer vi alle de JavaScript- og CSS-filene som kan kombineres. Resultatet er en eneste CSS-fil og to-tre JavaScript-filer (varierer fra side til side).

4.10 Webutil

Alle forespørsler som kommer inn til domenet som systemet er satt opp på, blir behandlet på ett sted. På denne måten har vi full kontroll på hva som går hvor og hvordan en url skal se ut. Alle filer (CSS, JavaScript osv.) hentes også gjennom dette systemet. Alt er uavhengig av hvordan filer og mapper er organisert.

4.10.1 HttpPostManager

POST-forespørsler blir behandlet utifra en parameter som blir sendt med i hver enkelt form. En sjekk mot forhåndsdefinerte verdier vil så kjøre en funksjon implementert av en utvidelse av klassen POSTAction.

Listing 4.9: WebUtil PostAction

```
1 public class PostAction {
2
3   public void doAction(FormContainer container) {
4
5   }
6
7   public boolean validate(FormContainer container) {
8     return true;
9   }
10 }
```

For hver HTML-form vil det være en klasse som implementerer POSTAction som behandler forespørselen. Om parameteren mangler vil det det sendes en responskode 404.

Listing 4.10: WebUtil PostManager

```
1 public class HttpPostManager {
2   public void prosess() {
3     PostHandler postHandler = new PostHandler(ServletUtil.getRequest());
4     postHandler.addForm("login", new LoginAction());
5     postHandler.addForm("addTime", new AddTimeAction());
6     postHandler.addForm("updateTime", new UpdateTimeAction());
7     postHandler.addForm("addTimeMobile", new AddTimeMobileAction());
8     postHandler.addForm("sendTime", new SendTimeAction());
9     postHandler.addForm("removeTime", new RemoveTimeAction());
10    postHandler.addForm("settings", new SettingsAction());
11    postHandler.runPostRequest();
12  }
13 }
```

Ved å la SecurePostAction arve PostAction og sjekker om brukeren er logget inn, så har vi hatt muligheten til å legge til dette sikerhetsnivået på alle POST- forespørsler enkelt ved å arve denne klassen.

Listing 4.11: WebUtil SecurePostAction

```
1  public class SecurePostAction extends PostAction {
2    @Override
3    public boolean validate(FormContainer container) {
4      if(SessionUtil.getUserCredentials() != null)
5        return true;
6      else {
7        try {
8          ServletUtil.getResponse().sendRedirect("/");
9        } catch (IOException e) {
10         StackTracePrinter.error(e);
11       }
12       return false;
13     }
14   }
15 }
```

4.10.2 HttpGETManager

Den samme metoden brukes også for GET-foresørsler. GETHandler binder en url opp mot en klasse ved bestemte kodeord og variable tekst- og tallmarkeringer.

Listing 4.12: WebUtil GETManager

```
1  public class HttpGETManager {
2    public static void prosess() throws IOException {
3      GETHandler getHandler = new GETHandler();
4
5      ...
6
7      getHandler.addFilter("/css/<string>", new CSSAction());
8      getHandler.addFilter("/img/<string>", new ImageAction());
9
10     ...
11
12     getHandler.addFilter("/js/<string>", new JavaScriptAction());
13     getHandler.addFilter("/xsl/<string>", new XSLAction());
14     getHandler.addFilter("/ajax/item/<string>", new AjaxItemAction());
15     getHandler.addFilter("/ajax/itemlist", new AjaxItemListAction());
16
17     ...
18   }
19 }
```

Samme som med POST-request, så har vi også her brukt løsningen med “SecureHtml” som arves etter behov for sikkerhetsnivå. Dette gir også en mulighet for å lage mange sikkerhetsnivåer og implementere dette enkelt i den eksisterende strukturen.

Listing 4.13: WebUtil GETHandler

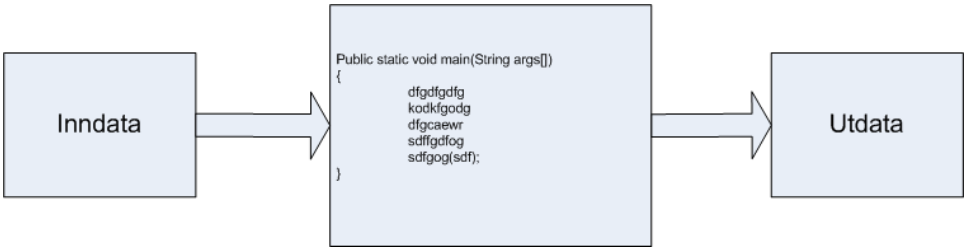
```
1  public class GETHandler {
2    private List<URL> filters;
3
4    public GETHandler() {
5      this.filters = new ArrayList<URL>();
6    }
7
8    public boolean runURL(String urlString) {
9      for (Iterator<URL> iterator = filters.iterator(); iterator.hasNext();) {
10       URL url = iterator.next();
11       if (url.match(urlString)) {
12         if(url.validate())
13           url.doAction();
14         return true;
15       }
16     }
17     return false;
18   }
19
20   public void addFilter(String filter, GETAction action) {
21     List<String> parts = URLParser.parseUrl(filter);
22
23     URL url = new URL(action);
24
25     for (Iterator<String> iterator = parts.iterator(); iterator.hasNext();) {
26       String urlPart = (String) iterator.next();
27
28       if (urlPart.contains("<string>"))
29         url.addURLElement(new URLElement(String.class));
30       else if (urlPart.contains("<int>"))
31         url.addURLElement(new URLElement(Integer.class));
32       else
33         url.addURLElement(new URLElement(urlPart));
34     }
35     filters.add(url);
36   }
37 }
```

5 Testing

Testingen har blitt svært nedprioritert grunnet at oppdragsgiver kom med nye krav i slutten av siste sprint. Dette førte til at den siste test-sprinten falt bort, og at dette ikke ble like mye vektlagt som tidligere planlagt.

Vi har valgt å utføre White-box-testing og Black-box-testing. Dette er kun en oppsummering av testingen som foregikk under hele prosjektet, ettersom vi ikke har tidsressursene til å utføre en egen testrapport.

5.1 White-box-testing



Figur 39: White-box-testing

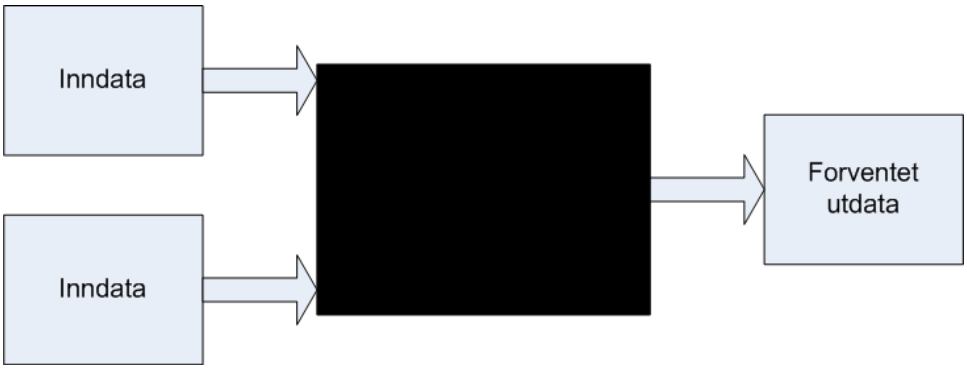
Denne testen går ut på å analysere utskrift fra systemet ved å benytte seg av en person som kjenner systemet internt. Vedkommende velger ut testområder, og analyserer hvordan systemet arbeider seg gjennom inndata, og hvilke stier som blir brukt for å gi ønsket utdata. Denne testingen har vi som gruppe utført underveis.

Testingen har blitt utført gjennom hele prosjektet, like før slutten av hver sprint. Vi har brukt èn dag i slutten av hver sprint for å forsikre oss om at det ikke er mulig å få systemet til å henge, krasje eller gi feil utskrift.

Vi har lært svært mye av denne testingen, og har rettet opp mange feilsituasjoner.

5.2 Black-box-testing

Denne modellen går ut på å få noen som ikke kjenner systemet til å utføre testingen. Her legger man inn ønsket inndata, og analyserer forventet utdata.



Figur 40: Black-box-testing

Som nevnt ovenfor så har testingen vært svært begrenset grunnet mangel på tidsressurser. Vi lot derfor to med-studenter, som ikke hadde noe tidligere kjennskap til systemet, utføre diverse operasjoner med systemet, slik som det kommer til å bli brukt i en normal arbeidsdag. Denne testingen ble utført rett før leveranse av systemet, og forventet utdata på gitte inndata stemte overens.

5.3 Konklusjon

Vi hadde gjort avtale med oppdragsgiver om å la de teste systemet underveis, noe de forøvrig gjorde i en viss grad, men mangel på tid gjorde testingen svært minimal. Systemet fungerte som forventet i de testene som ble gjort hos oppdragsgiver.

Vi la ut systemet på en egen testserver, slik at det skulle være enkelt for andre å prøve systemet.

Testingen gav oss gode idèer om hva som fungerte og hva som ikke fungerte. Dette gav oss god innsikt i hvordan systemet kunne vidreutvikles og forbedres underveis.

6 Avslutning

6.1 Diskusjon av resultater

Vi er svært fornøyd med resultatet på vår oppgave. Vi har overholdt kravene fra oppdragsgiver og tatt det et steg videre. Vi startet med et timeregistreringssystem, men endte opp med et faktureringssystem integrert mot Nexstep. Omfanget på denne oppgaven ble svært stort, men vi klarte alikevel å overholde fristen, ved å jobbe ved høyere intensitet og lagt ned flere arbeidstimer.

Dette er en ferdig og fungerende løsning. Oppdragsgiver ble også svært fornøyd med resultatet.

6.2 Forbedringsmuligheter

Av forbedringsmuligheter vil kunne være å implementere effektuering av timer på mobilapplikasjonen, selv om dette ikke var et krav ifra oppdragsgiver, og derfor ikke med i kravspesifikasjonen.

Vi kunne også ha forbedret JavaScript-metoder, med hensyn på tastetrykkregistrering. Dette med hensyn på kompabilitet mellom nettlesere.

For å forstå hvordan tastatur-event fungerer i JavaScript, valgte vi å skrive all kode selv. Vi kunne derimot ha brukt jQuery, for å spare oss en del arbeid.

Vi hadde ikke muligheten til å være med på den siste testfasen av systemet, den faktiske bruken, grunnet større oppgaveomfang og endret kravspesifikasjon.

6.3 Evaluering av gruppens arbeid

6.3.1 Organisering og arbeidsfordeling

Organiseringen innad i gruppen har vært fellesbestemt, men prosjektlederen har alltid fått siste ord.

Arbeidsfordelingen har vært jevnt fordelt, slik at alle gruppemedlemmer har fått brukt likt antall timer på utvikling og dokumentering. På dokumenteringsdelen har alle gruppemedlemmer sittet sammen.

6.3.2 Loggføring

Loggføringen ble tatt i fellesskap, ettersom gruppen alltid jobbet sammen. Oversikten av loggen ble derfor lik for alle gruppemedlemmer.

6.3.3 Prosjekt som arbeidsform

Vi har lært svært mye av å ha utført et prosjekt i denne skalaen. Dette har bedret vår kommunikasjons- og sammarbeidsevne innad i gruppen, og vi har lært å arbeide strukturert og systematisk. Det å ha en utviklinsmodell å følge har vært nyttig, både med tanke på utbredelsen av denne arbeidsmåten innenfor seriøse utviklingsfirmaer og det å få en bedre struktur på arbeidsmetoder.

6.3.4 Konklusjon

Vi er svært fornøyd med våres arbeid og resultat i bacheloroppgaven. Denne oppgaven har lært oss mye om alt fra systemutvikling til programmering. Underveis i prosjektet har vi hatt god kommunikasjon med oppdragsgiver, og har derfor hatt en god oversikt over oppdragsgivers behov og ønsker. Dette har spart oss for bruk av mye tid på endringer, og vi fikk derfor utført mer arbeid i hver arbeidssyklus i utviklingsmodellen.

Vi hadde en relativt lett oppgave, men den utvidet seg hurtig til å bli til et mer kompleks system. Dette har gitt oss alle en bratt læringskurve, og vi er nå veldig fornøyde med hendelsesgangen i prosjektet. Alt vi har arbeidet med er relevant til arbeidslivet, så vi har hatt god nytte av å gjennomføre dette prosjektet, og lært alt det innebærer.

7 Litteraturliste

Google Page Speed
http://code.google.com/speed/page-speed/docs/rules_intro.html

Yahoo! YUI Library
http://developer.yahoo.com/yui/

JBoss Community - Hibernate
http://www.hibernate.org/

Apache Axis2
http://ws.apache.org/axis2/

WikiBooks - LaTeX documentation
http://en.wikibooks.org/wiki/LaTeX

Applying UML and Patterns, Craig Larman.
Software Engineering"av Ian Sommerville (8. edition)

8 Ordliste

AJAX
Asynchronous JavaScript XML.

Apache
HTTP-server.

ASP
Application System Partner.

AXIS2
Motor for Web Services.

CSS
Cascading Style Sheets.

ERP
Enterprise Resource Planning.

Hibernate
Object-relation mapping.

HTML
HyperText Markup Language.

HTTP
HyperText Transfer Protocol.

HQL
Hibernate Query Language.

ID
Identifikasjon.

JPA3
Java Persistence API, versjon tre.

JQuery
Plattformuavhengig JavaScript-bibliotek.

LaTeX
Dokumentasjonsverktøy for generering av PDF.

Nexstep
Ordresystemet hos ASP.

Servlet container
Lar java kjøre som en web-applikasjon.

SOA
Service Oriented Architecture.

SQL
Structured Query Language.

PDA
Personal Digital Assistant.

PDF
Portable Document Format.

PHP
Hypertext Preprocessor .

Sprint
Arbeidssyklus i utviklingsmodellen Scrum.

Subversion
Versjonshåndteringssystem.

Tomcat
Servlet container.

UML
Unified Modeling Language – objektorientert modellering i systemutvikling.

URL
Uniform Resource Locator

API
Application Programming Interface.

Web Services
En samling av Web-API'er.

WSDL
Web Services Description Language.

XML
Extensible Markup Language .

XSL
Extensible Stylesheet Language.

XSLT
: XSL Transformations.