BACHELOROPPGAVE:

Libellus - Crisis Management Tool

FORFATTERE:
Daniel André Solstad
Ole Johan Rasch

DATO:
19.05.2014

Sammendrag av Bacheloroppgaven

Tittel:	Libellus - Crisis Management Tool	Nr: -
		Dato: 19.05.2014
Deltakere:	Daniel André Solstad	
	Ole Johan Rasch	
Veileder:	Thomas Kemmerich	
Oppdragsgiver:	Conax AS	
Kontaktperson:	Espen Torseth, espen.torseth@conax.com	
Stikkord	Libellus, krisehåndtering, loggeverktøy, journal, portabel, brukergrensesnitt	

Antall sider: 173	Antall vedlegg: 12	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:
Libellus er et journalføringsprogram for kriser og krisehåndteringsopplæring. Programmet er nytenkende på området infrastrukturtilknytning, ved at det kun krever en datamaskin for å kunne tilby all basisfunksjonalitet som loggføring, sortering, datamanipulasjon og oppfølgingsmerknader. Startes flere instanser av Libellus i lokalnettverket, vil all data automatisk synkroniseres mellom disse instansene. Applikasjonen har en lav brukerterskel for effektivt å kunne iverksettes ved en krisesituasjon, noe som er støttet opp med et kjent, minimalistisk og moderne design.

Libellus er primært utviklet i JavaScript og kjører på databaseløsningen CouchDB. I rapporten er det beskrevet hvilke valg og vurderinger som er gjort underveis, sammen med den teoretiske bakgrunnen. I vedleggene ligger en innføring i JavaScript, AJAX og OOP, samt dokumentasjon for grunnleggende bruk, vedlikehold og operativ benyttelse av programmet.

Applikasjonen er åpen kildekode og er tilgjengelig på github.com/libellus og prosjektets hjemmeside libellus.no

Summary of Graduate Project

Title:	Libellus - Crisis Management Tool	Nr: -
		Date: 19.05.2014
Participants:	Daniel André Solstad	
	Ole Johan Rasch	
Supervisor:	Thomas Kemmerich	
Employer:	Conax AS	
Contact person:	Espen Torseth, espen.torseth@conax.com	
Keywords	Libellus, crisis management, journal tool, portability, GUI	
Pages: 173	Appendixes: 12	Availability: Open

Short description of the main project:
Libellus is a journal tool for crisis and crisis management training. The software is innovative in the area of infrastructure, where the only requirement is a computer to offer all the basic functionality such as journaling, sorting, data manipulation and actions. If multiple instances of Libellus is running in a local network, they will automatically synchronize data with each other. The application has a low user barrier to effectively be operative under a crisis, with a familiar, minimalistic and modern design.

Libellus is primarily developed in JavaScript and runs on the CouchDB database. Choices and evaluation are described throughout the report, along with the theoretical background. Basic introduction to JavaScript, AJAX and OOP can be found in the appendices, along with maintenance, usage and operative documentation.

The application is open source and available at github.com/libellus and at the projects homepage libellus.no

libellus /li'bel.lus/
substantiv: en liten bok

Forord

Libellus er utviklet av to bachelorstudenter innen informasjonssikkerhet ved Høgskolen i Gjøvik. Prosjektoppgaven ble utformet i samarbeid med oppdragsgiver Conax AS representert ved Espen Torseth og har bestått i å utvikle et loggeverktøy for bruk under kriser.

En stor takk rettes til følgende for deres bidrag til prosjektet

- Espen for gode innspill, og et godt samarbeid gjennom hele bacheloroppgaven
- Thomas Kemmerich, vår veileder, for råd og vink til selve bachelorrapporten
- Torstein Gleditsch, Jeanne Hammer Espedalen og Erik Abrahamsen hos Conax AS for tilbakemeldinger på programvaren
- Tom Røise (HiG) for tips i den avsluttende fasen av oppgaveskrivingen
- Brukerene på #couchdb@irc.freenode.net for deres kunnskap om Apache CouchDB

Gjøvik, mai 2014

Daniel Solstad

Ole Johan Rasch

Innhold

Figurer

Tabeller

Kodesnutter

1 Innledning

1.1 Organisering av rapporten

Gjennom hele rapporten har vi valgt å benytte norske avløserord og terminologi så langt dette har latt seg gjøre. Vi er tilhengere av et rikt norsk språk og har merket oss at det særlig i IT-verdenen ofte tys til importord selv om norske alternativer finnes. Noen ganger vil alternativene være godt innarbeidet og andre ganger ikke. Listen over forkortelser og ordforklaringer befinner seg i vedlegg A. I de tilfellene vi har benyttet engelskspråklige kilder har vi valgt å bruke indirekte sitater.

All kode er kommentert og beskrevet på engelsk, det samme gjelder for dokumentasjon, da vi anser dette som mest hensiktsmessig for videreutvikling, opplæring og internasjonal bruk av programvaren.

Kursivert tekst benyttes for å utheve fil-, funksjons-, og variabelnavn, samt der vi mener det er naturlig.

Rapporten er delt inn i sju kapitler, i tillegg til vedlegg.

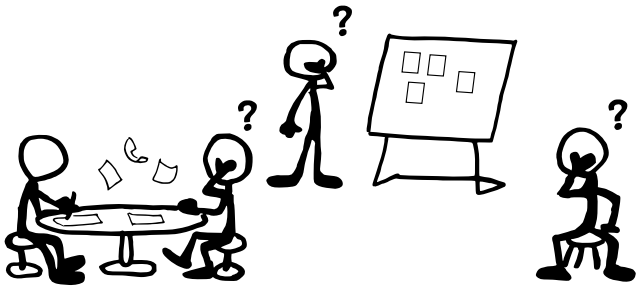
- Kapittel 1 - Innledning
  Omhandler prosjektet og praktiske opplysninger rundt dette.
- Kapittel 2 - Kravspesifikasjon
  Inneholder kravene til applikasjonen, slik de forelå før utviklingen startet.
- Kapittel 3 - Design
  Består av det initielle designet til applikasjonen, slik vi så det for oss før teorigjennomgang og valg av programvare.
- Kapittel 4 - Teoretisk grunnlag
  Inneholder informasjon og teori vi har tilegnet oss underveis i prosjektet.
- Kapittel 5 - Implementasjon
  Her tar vi for oss de viktigste delene av løsningen med kodeeksempler og beskriver implementasjonen nærmere.
- Kapittel 6 - Kvalitetssikring og testing
  En gjennomgang av teststrategier og resultater.
- Kapittel 7 - Avslutning
  Inneholder en drøfting av oppgaven, det ferdige resultatet, videre arbeid og konklusjon.

1.2 Introduksjon

Daglig opplever bedrifter, organisasjoner og offentlige institusjoner ulike typer kriser som må håndteres. Avgjørelsene som blir tatt er basert på kunnskap om virksomhetsenheten, krisehåndtering og den innkommende informasjonen om situasjonen. Feilaktige valg kan føre til store økonomiske tap, og i ytterste konsekvens kan det ende med dødsfall [1]. Det er derfor de fleste har en form for plan over hvilke oppgaver som må løses når en kritisk hendelse inntreffer. Noen har også dokumentasjon som stegvis beskriver hvordan ulike scenario kan løses.

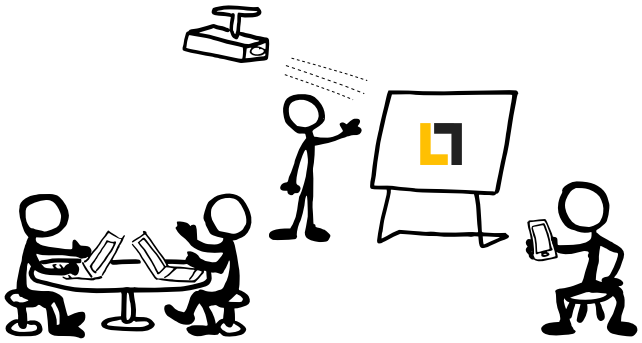
Det er lett å teste slike prosedyrer i et trygt kontormiljø med de kjente støtteverktøyene tilgjengelig, men dessverre fortoner ikke kriser seg slik. Bedriften må være forberedt på at intranettet kan være nede, kontorlokalene utilgjengelige eller et skadevareangrep har satt dataparken ut av spill. Det er denne typen fremtidstenkning som har skapt grunnlaget for loggeverktøyet Libellus, et redskap for hendelseshåndtering som støtter seg på lite eksisterende infrastruktur. Verktøyet skal hjelpe de rammede til å loggføre hendelser og skape situasjonsbevissthet slik at de kan ta de rette avgjørelsene.

Figur 1 viser en situasjon hvor det vil være aktuelt å benytte Libellus. Krisestaben er samlet og har startet arbeidet med å få oversikt over hendelsens omfang. De benytter en tradisjonell struktur hvor en person fungerer som referent. I hektiske perioder vil loggføreren ha mange arbeidsoppgaver: Føre journal, oppdatere liste over tilstedeværende personer og håndtere innkommende opplysninger. For andre i staben kan det være vanskelig å få en total oversikt, uten stadig å måtte spørre loggføreren etter informasjon.



Figur 1: Pre Libellus

Med Libellus vil det bli mer informasjonsflyt, som igjen fører til bedre situasjonsforståelse. Krisestaben kan fremdeles bestemme at en person skal være dedikert referent, eller fordele det slik at den med størst nærhet til informasjonen journalfører og håndterer den. Dette er vist i figur 2.



Figur 2: Post Libellus

1.3 Prosjektets bakgrunn

> A part of the Kudelski Group, Conax is a leading global specialist around the total service protection for digital TV services over broadcast, broadband and connected devices. Through the future-proof Conax Contego™security hub, Conax provides telcos, cable, satellite, IP, mobile and terrestrial and broadband operations with the innovative, flexible and cost-efficient solutions to deliver premium content securely and positioning to capture new market segments. Bundling over 25 years of pioneering experience, its benchmark policy for security-evaluated client devices and strategic partner network, Conax technology enables secure content revenues for operators representing 140 million pay-TV consumers in 85 countries globally. ISO 9001 & 27001 certified, Conax is headquartered in Oslo, Norway, with 24/7 Global Support operations in India.
>
> *- Conax AS*

Oppdragsgiver Conax AS har krypteringsløsninger for TV som sitt hovedprodukt. Mange av deres ansatte innehar dermed kompetanse innen informasjonssikkerhet. Bedriften kjører årlige hendelseshåndteringsøvelser, og i den forbindelse har det kommet opp et ønske om et støtteverktøy for kriseledelse. De har kommet til den konklusjon at dagens verktøy ikke er robuste nok.

1.4 Oppgavebeskrivelse

Formålet med oppgaven har vært å utvikle en programvare som skal være et verktøy for effektiv krisehåndtering. Programvaren har lokal kriselogg som grunnfunksjonalitet, med støtte for vedlegg av både elektronisk og fysisk art. Hele systemet skal kunne kjøres direkte fra en minnepinne og er ikke avhengig av infrastruktur utover den datamaskinen det jobbes fra. Dersom flere instanser av Libellus startes opp i samme lokalnettverk, vil systemet ha full synkronisering av data mellom disse enhetene. All kommunikasjon foregår kryptert, og det er et stort fokus på integritet med tidsregistrering og skrivebeskyttelse av innlagt data.

Det har blitt utviklet skriftlig materiale som støtter opp om programvaren, inkludert brukermanual, vedlikeholdsmanual, og et forslag til hvordan verktøyet kan benyttes under en krise.

Utviklingsprosessen har båret preg av at prosjektgruppen har vært med å definere oppgaven og at vi har fått stor frihet når det gjelder utformingen av programvaren.

Oppsummert har hovedmålene for prosjektet vært å utvikle et system med:

- lokal, tekstbasert krisejournal for notater
- støtte for vedlegg av kommentarer, oppfølgingsmerknader og dokumenter
- minimalt krav til eksisterende infrastruktur
- kryptert datasynkronisering mellom eventuelle andre operative instanser av programvaren
- støtte for datamanipulasjon for effektiv fremvisning av relevant data
- løsninger basert på åpen kildekode
- bruker-, operativ- og vedlikeholdsdokumentasjon

1.5 Målgruppe

Den utviklede programvaren og medfølgende dokumentasjonen er ment for små og mellomstore bedrifter i størrelsesordenen 10 - 1000 ansatte. Loggeverktøyet er tiltenkt brukt under kriser, men også i opplæringsøyemed. Sensor er målgruppen for rapporten, og den anses å være en dokumentasjon over det arbeidet som er innlagt i utviklingsprosjektet. Samtidig belyser rapporten viktige tema for andre som ønsker å utvikle egne loggeverktøy av samme type.

1.6 Avgrensninger

1.6.1 Ekstern kommunikasjon

Programvaren tar ikke høyde for at ulike eksterne kommunikasjonskanaler må være tilgjengelig for å fungere tilfredsstillende. Dette innebærer at forbindelse med omverden via telefonsamtaler, SMS, e-post eller lignende er utenfor dette prosjektets omfang. Kommunikasjon kan kun bli lagt inn i loggeverktøyet som tekstinnlegg eller vedlegg. Synkronisering av journalinnlegg fra ulike lokale nettverk over Internett har vært utenfor omfanget av denne oppgaven.

1.6.2 Portabilitet

Applikasjonen er portabel til maskin- og programvare som var tilgjengelig hos lokale elektronikkforhandlere pr. 01.02.14. Dette ekskluderer eksempelvis eldre versjoner av Microsoft Windows, som XP og Vista. Med portabel programvare mener vi en applikasjon som ikke behøver å bli installert på en datamaskin, men er kjørbar direkte fra et flyttbart lagringsmedium.

1.6.3 Fysisk sikring

Gruppen har ikke fokusert på hvordan det fysiske mediumet, værende minnepinne, harddisk eller ferdig oppsatt datamaskin med programvaren innlagt bør beskyttes. Vi har heller ikke diskutert hvordan organisasjonen som tar i bruk Libellus bør distribuere programvaren, slik at kun krisestaben har denne tilgjengelig.

1.6.4 Mobilapplikasjoner

Det har ikke blitt utviklet native mobilapplikasjoner, men de grafiske elementene har blitt designet med tanke på ulike skjermstørrelser.

1.6.5 Etterbehandling av data

Hvordan data produsert i verktøyet skal etterbehandles, som å generere rapporter i PDF-format eller å kunne redigere vedlegg direkte i programvaren har vært utenfor omfanget av denne oppgaven. Vi har heller ikke sett på analyse av eksportert data.

1.6.6 Selektiv synkronisering

Selektiv synkronisering har ikke blitt prioritert. Eksempelvis at kun enkeltinnlegg replikeres ut til de andre tjenerene i nettverket, eller at innlegg kun blir sendt til bestemte instanser.

1.7 Prosjektmål

1.7.1 Effektmål

Det endelige produktet er ment å være et viktig hendelseshåndteringsverktøy som kan avhjelpe kriserammende bedrifter eller organisasjoner ved å:

- forenkle prosessen med å dokumentere hendelser
- støtte opp om forløpet til å ta de rette avgjørelsene

- i etterkant kunne gjennomgå alle besluttede avgjørelser, basert på informasjonen kjent på det daværende tidspunkt
- lette hendelseshåndteringstreningen
- øke antall personer som deltar i loggføringen og på den måten fordele arbeidsmengden

1.7.2 Resultatmål

- Programvaren lanseres i versjon 1.0 den 19.05.14 med all ønsket funksjonalitet som beskrevet i oppgavebeskrivelsen.
- Den produserte koden og programvaren skal bli publisert og tilgjengeliggjort for hele verden på prosjektets hjemmeside med en åpen kildekode-lisens.
- All ikke-triviell kode er veldokumentert, for å støtte opp om videreutvikling utført av andre.

1.7.3 Faglig bakgrunn

Vi har som studenter ved informasjonssikkerhetslinja lært grunnleggende programmering, operativsystemer, databaseteori og informasjonssikkerhet. Tidligere har vi arbeidet sammen på prosjekter innen programmering, systemutvikling og hendelseshåndtering. Samtidig har vi ulike individuelle interesser og kompetanse som har kommet til nytte i prosjektet. Daniel er dyktig innen vebbutvikling og behersker HTML, CSS og PHP godt. Ole Johan har sin største styrke innen prosjektadministrasjon og utvikling av rutiner.

1.7.4 Prosjektorganisering

- Gruppeleder
  Vår gruppeleder har vært Ole Johan Rasch. Han har vært ansvarlig for administrative oppgaver, som romreservasjon, loggføring, innleveringer og dokumentasjon i tillegg til å være delaktig i programmeringen.
- Teknisk ansvarlig
  Daniel André Solstad har vært teknisk ansvarlig. Hans ansvarsområder har vært innen å følge opp prosjektet når det gjelder programvareutvikling, samt valg av kodebibliotek. Han har også vært vår kontaktperson mot oppdragsgiver.
- Veileder
  Thomas Kemmerich, Førsteamanuensis ved Høgskolen i Gjøvik.
- Oppdragsgiver
  Conax AS, representert ved Espen Torseth (Senior Security Analyst).

2 Kravspesifikasjon

Denne delen av rapporten beskriver krav til programvaren. Disse er utarbeidet med bakgrunn i oppgavebeskrivelsen, samt kommunikasjon med oppdragsgiver. Vi har valgt å benytte teknikker som bruksmønsterdiagram og bruksmønsterbeskrivelser [2] for å fremvise hvordan brukeren interagerer med programvaren. De operasjonelle kravene har vi konkretisert ned i en punktoppstilling [2].

2.1 Funksjonelle krav

Figur 3: Bruksmønsterdiagram - Brukeren i systemet er loggføreren

2.2 Detaljerte bruksmønsterbeskrivelser

I dette avsnittet beskriver vi de mest sentrale bruksmønstrene som krever detaljer, hvor for eksempel hendelsesløpet vil ha ulikt utfall avhengig av de operasjonene brukeren utfører.

Bruksmønster:	Logge inn
Aktør:	Bruker
Mål:	Logge inn for å benytte loggeverktøyet
Prebetingelse:	• Applikasjonen er startet
Postbetingelse:	Brukeren blir innlogget
Foretrukket hendelsesløp:	• Bruker: Starter applikasjonen • System: Innloggingsvindu vises, der brukeren kan skrive inn brukernavn • Bruker: Skriver inn ønsket brukernavn • System: Validerer om brukernavnet ikke er tomt eller ikke
Variasjoner:	• Brukeren forsøker å logge inn med et tomt brukernavn og får beskjed om å prøve på nytt

Tabell 1: Bruksmønster - Logge inn

Bruksmønster:	Loggføre hendelse
Aktør:	Bruker
Mål:	Loggføre en hendelse
Prebetingelse:	• Brukeren er logget inn
Postbetingelse:	Nytt journalinnlegg blir lagt inn
Foretrukket hendelsesløp:	• Bruker: Trykker på en knapp for å få fram et skjema • Bruker: Skriver inn overskrift, innhold, nøkkelord, hendelsestid og klassifisering • Bruker: Trykker på lagre-knappen • System: Eksternt hentet tid blir lagt til innlegget • System: Data blir sendt til databasen • System: Innlegget blir vist for brukeren i tabellform • System: Data blir replikert til alle Libellus-instanser
Variasjoner:	• Eksternt hentet tid er ikke tilgjengelig. Kun lokal tid lagres

Tabell 2: Bruksmønster - Loggføre hendelse

Bruksmønster:	Kommentere på hendelse
Aktør:	Bruker
Mål:	Supplere hovedinnlegg med mer data
Prebetingelse:	• Det finnes et journalinnlegg
Postbetingelse:	En kommentar blir lagt til et hovedinnlegg
Foretrukket hendelsesløp:	• Bruker: Ekspanderer hovedinnlegget • Bruker: Velger kommentar fra en meny eller boks • Bruker: Fyller ut tekst i kommentarfeltet • Bruker: Velger om det skal legges til et vedlegg • Bruker: Trykker på lagre-knappen • System: Nøyaktig tid blir lagt til innlegget • System: Data blir sendt til databasen • System: Innlegget blir vist for brukeren i tabellform • System: Data blir replikert til alle Libellus-instanser
Variasjoner:	• Brukeren kan velge mellom å laste opp en digital fil eller bruke *Filed copy*, som er å referere til et fysisk objekt med en tilfeldig tekststreng.

Tabell 3: Bruksmønster - Kommentere på hendelse

Bruksmønster:	Få oversikt over aktive oppfølgingsmerknader (actions)
Aktør:	Bruker
Mål:	Få oversikt over handlinger som må utføres i ønsket sortert rekkefølge
Prebetingelse:	Det finnes minst en aktiv oppfølgingsmerknad
Postbetingelse:	Ingen
Foretrukket hendelsesløp:	• Bruker: Trykker på Actions i menyen for å få opp alle oppfølgingsmerknader • Bruker: Trykker på overskriftene i tabellene for å sortere på ønskede attributter • System: Sorterer innholdet, enten på tjener- eller klientsiden • Bruker: Kan trykke på en oppfølging for å få opp mer informasjon og flere valg

Tabell 4: Bruksmønster - Få oversikt over aktive oppfølgingsmerknader (actions)

Bruksmønster:	Endre status på oppfølgingsmerkander
Aktør:	Bruker
Mål:	Endre status til løst
Prebetingelse:	• Brukeren er inne på oversikten for oppfølgingsmerknader
Postbetingelse:	Oppdatering av oppfølgingen i databasen som blir replikert
Foretrukket hendelsesløp:	• Bruker: Trykker på en oppfølging for å få opp mer informasjon og flere valg • Bruker: Trykker på en knapp som markerer oppfølgingen som løst • System: Informasjonen blir sendt til databasen • System: Informasjonen blir replikert til alle Libellus-instanser i nettet

Tabell 5: Bruksmønster - Endre status på oppfølgingsmerkand

Bruksmønster:	Kommunisere med andre brukere i sanntid
Aktør:	Bruker
Mål:	Formidle informasjon til alle andre brukere via lynmeldinger
Prebetingelse:	• Brukeren er inne på oversikten for lynmeldinger
Postbetingelse:	Meldingen blir lagt til i databasen, som blir replikert og vist for alle andre Libellus-instanser
Foretrukket hendelsesløp:	• Bruker: Trykker på tekstboksen og skriver en melding • Bruker: Trykker så på sendknappen • System: Informasjonen blir sendt til databasen • System: Informasjonen blir replikert til alle Libellus-instanser i nettet

Tabell 6: Bruksmønster - Kommunisere med andre brukere i sanntid

Bruksmønster:	Lete etter andre instanser
Aktør:	Libellus
Mål:	Sette opp replikering med andre Libellus-instanser
Prebetingelse:	
Postbetingelse:	Replikeringsinformasjon blir lagt inn i replikeringsdatabasen
Foretrukket hendelsesløp:	• Brukeren starter Libellus, som setter i gang en bakgrunnsprosess, der den leter etter andre Libellus-instanser og legger de nyoppdagende klientene i en liste over hvem den skal replikere til
Variasjoner:	• Dersom maskinen ikke har nettverkstilgang, vil bakgrunnsprosessen bli startet når denne blir tilgjengelig

Tabell 7: Bruksmønster - Lete etter andre instanser

2.3 Operasjonelle krav

2.3.1 Brukervennlighetskrav

- Systemet skal ha allment gjenkjennbare elementer og ikoner, som skal være med å bidra til en lav brukerterskel. Eksempelvis kan spørsmålstegnikonet benyttes for å få frem hjelpefunksjonen i programmet.
- I den grad applikasjonen krever tekstlige menyvalg og forklaringer skal disse være forenklet og skrevet på engelsk.
- Ved feilmeldinger skal brukeren få en skriftlig forklaring over hva som kan ha gått galt.
- Programvaren skal automatisk hente og vise journaloppdateringer i nær sanntid, uten at brukeren aktivt må be om oppdateringer.

2.3.2 Ytelseskrav

- Systemet skal støtte synkronisering av data mellom 20 ulike instanser samtidig.
- Fra et tekstinnlegg uten vedlegg er innlagt til det er spredt til andre kjørende instanser av programvaren skal det høyest ta tre sekunder. Dette gjelder der nettverksinfrastrukturen er kjent og det kan garanteres for minimum 10 Mbps. Denne tiden, tre sekunder er hva vi anslår bør være en maksgrense før tregheten blir ansett som et problem og irritasjonsmoment av brukerene.

2.3.3 Sikkerhetsmessige krav

- All kommunikasjon mellom kjørende instanser av programvaren skal være kryptert.
- Innlegg i databasen skal kunne spores med klokkeslett, forfatter, og instansidentifikasjon.
- Systemet må ha integritetssikring av innlagt data.
- Systemet skal være robust og ha pålitelig tidssynkronisering og tidsstempling.

2.3.4 Tilgjengelighet

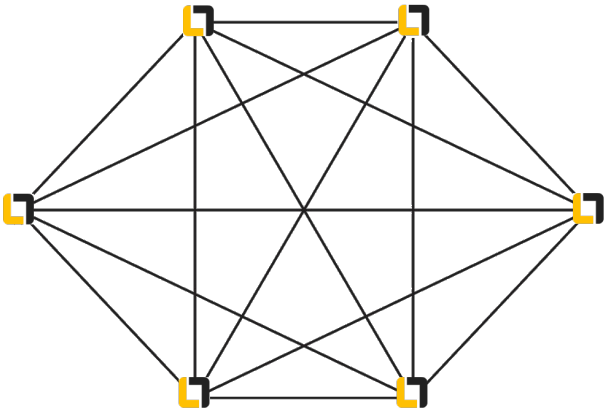
- Oppdragsgiver har satt som krav at programvaren skal være portabel til Microsoft Windows 7 og 8. I tillegg ønsker vi å utvide dette til Apple OS X Mavericks og Ubuntu Linux 12.04 LTS.
- Systemet skal støtte replikering av data til enheter i lokalnettverket.
- Programvaren skal ikke være avhengig av ekstern infrastruktur, som brukerdatabaser eller andre systemer.
- De grafiske elementene skal skalere i forhold til brukerens enhet, om denne er en smarttelefon, projektor, bærbar datamaskin eller et nettbrett.

3 Design

I dette kapittelet beskriver vi de tankene vi hadde rundt utformingen av den endelige løsningen. Aktuelle tema var arkitektur, brukergrensesnitt, og databasestruktur. Det ferdig utviklede produktet har vi valgt å kommentere under implementasjon i kapittel 5. Der kommer også en nærmere skisse over valgt arkitektur og sammensetting av applikasjonen.

3.1 Valg av arkitektur

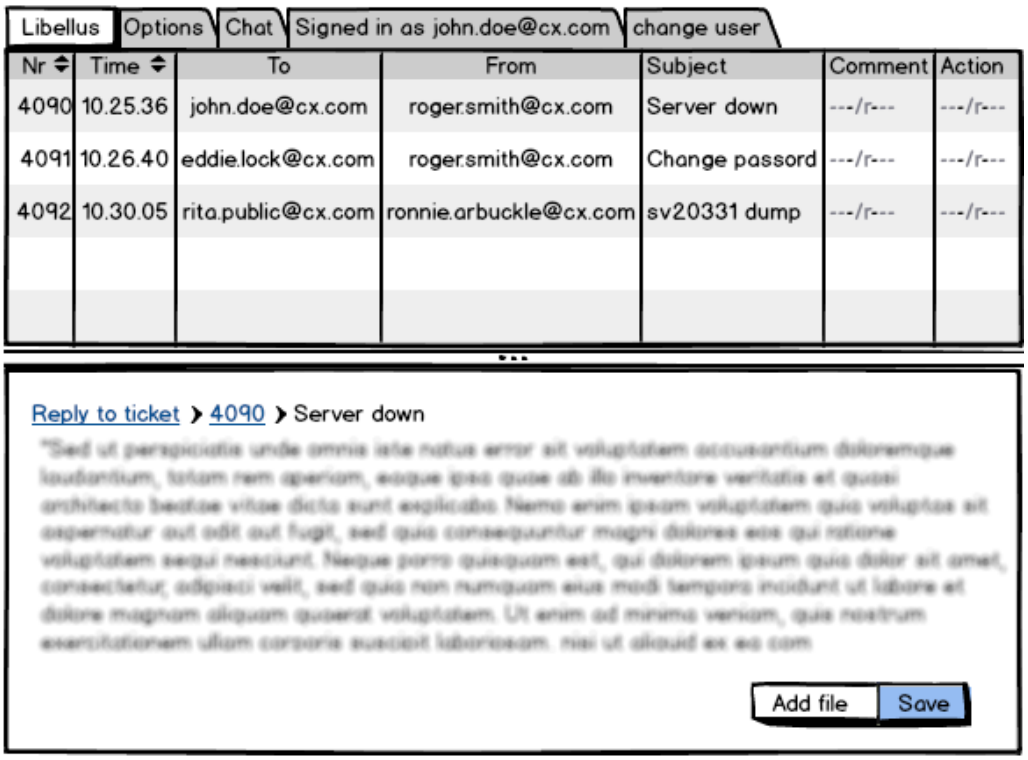
Med én datamaskin tilbyr Libellus loggføring hvor data lagres i en lokal database, og fremvises for brukeren. Dersom flere lokale maskiner starter programvaren, skal disse begynne å replikere til hverandre. Det vil dermed være naturlig å gå for en jevnbyrdsnettsarkitektur, der alle instanser av programmet i seg selv er en egen klient og tjener samtidig. Dette oppsettet vil redusere infrastrukturbehovet og organisasjonen slipper å ha to forskjellige versjoner av programvaren liggende på minnepinnen. Når krisen inntreffer skal det være lett å finne frem Libellus og starte loggingen med en gang, uten å måtte tenke på hvilken maskin som skal være tjener eller klient. I figur 4 vises et komplett maskenettverk med maksimal redundans. Går en node ned, vil ikke dette ha noen innvirkning på de andre.



Figur 4: Komplett maskenettverk

3.2 Brukergrensesnitt

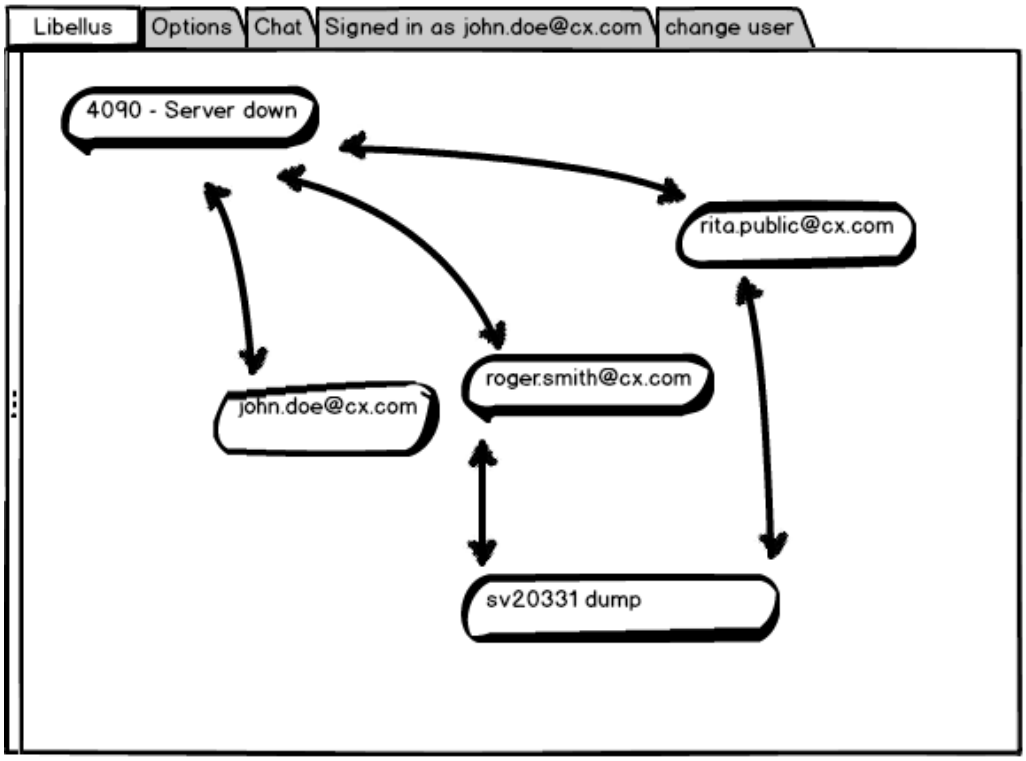
Som beskrevet i kravspesifikasjonen skal brukergrensesnittet på en enkel og oversiktlig måte tilby fremvisning av data i loggeverktøyet, samt effektivt kunne lede brukeren til å legge inn nye innlegg og kommentarer. Figur 5 og 6 viser utkast til hvordan brukergrensesnittet kunne ha sett ut, dette har blitt kontinuerlig formet etter hvert som funksjonalitet har blitt lagt inn i programmet og bruksmessige tilbakemeldinger fra oppdragsgiver og andre testere har kommet inn.



Figur 5: Hovedbildet i applikasjonen

Figur 5 viser et utkast til hvordan hovedbildet i applikasjonen kunne ha sett ut. Den består av følgende komponenter:

- **Menylinje** - Øverst ligger menylinjen som skal gi brukeren en mulighet til å navigere mellom hovedfunksjonaliteten i programmet. Den skal også bestå av en visning av innlogget bruker, samt funksjon for å bytte bruker.
- **Datafremvisning** - Datafeltet skal være i fokus på hovedbildet. Data som innleggsdato og klokkeslett, tittel, hvem som er ansvarlig for eventuell oppgave tilknyttet innlegget, og tellefunksjonalitet for kommentarer og tiltak skal være med. En enkel form for sortering kan plasseres her.
- **Datainnlegging** - Registrering av nye hendelser og kommentarer på hendelser gjøres via et tekstfelt. Her skal funksjonalitet for opplasting av vedlegg også plasseres.



Figur 6: Visuell fremstilling i applikasjonen

Figur 6 viser et utkast til den visuelle fremstillingen av innlegg i applikasjonen. Den består av følgende komponent:

- **Visuelt panel** - Dette skal være en annen fremstilling av data enn rent tekstlig. Samtidig er det viktig at den gir en merverdi til programmet. I illustrasjonen vises det en visualisering over hvordan ulike saker kan henge sammen, men det kan også være aktuelt å gå for en mer tradisjonell tidslinje. Det visuelle panelet skal benytte hele skjermflaten.

3.3 Databasestruktur

I understående liste vises datastrukturen vi så for oss, uavhengig av databasehåndteringssystem. Dersom vi hadde valgt å benytte en strukturert relasjonsdatabase, ville journalinnlegg, kommentarer og oppfølgingsmerknader blitt plassert i ulike tabeller. Med en ustrukturert databasemodell, blir det i tillegg lagt inn en type-attributt, som skiller på hva slags innlegg det er. Dette er fordi NoSQL-databaser vanligvis ikke støtter flere tabeller, slik at alle poster ligger på samme sted.

- **id** - En unik referanse til innlegget
- **forfatter** - En identifikasjon på hvem som har skrevet innlegget
- **overskrift** - En kort oppsummering over hva innlegget handler om
- **tid** - Når hendelsen fant sted
- **innhold** - Hele innholdet
- **lokaltid** - Lokaltiden på maskinen når posten ble lagt inn i databasen
- **eksterntid** - Tid fra eksternkilde som f.eks. NTP, for når posten ble lagt inn i databasen

4 Teoretisk grunnlag

I denne delen av rapporten gjennomgår vi teoretiske emner som er relevante for applikasjonen. Vi starter med omkringliggende teori og går over til konkrete teknologivalg mot slutten.

4.1 Hendelseshåndtering

Den ferdig utviklede programvaren skal passe inn i hendelseshåndteringsprosessen og være et hjelpende verktøy som det er enkelt å integrere i planverket. Det er essensielt at denne ikke blir et forstyrrende element; Her er det viktig å finne frem til kilder og mønsterpraksis på området som kan være formende for programmets funksjonalitet. Det vil også kunne være positivt i markedsføringssammenheng at programvaren er i overensstemmelse med en eller flere standarder. I den forbindelse har vi undersøkt et flertall bøker [3, 4, 5, 6, 7, 8, 9] og veiledninger [10, 11] relatert til krisehåndtering. Etter gjennomlesning har få av disse vært relevante til utvikling av selve programvaren, men de har vært med på å øke vår forståelse for krise- og hendelseshåndtering.

I emnet *IMT3521 Sikkerhetsplanlegging og hendelseshåndtering* benyttet vi læreboken *Principles of Incident Response and Disaster Recovery* [9]. Denne omhandler metoder for å identifisere sårbarheter og finne mottiltak for å redusere risiko i en organisasjon. Den er også utdypende på beredskapsplanlegging og kontinuitetsplaner.

Det er viktig å loggføre alle detaljer relatert til en hendelse, da denne informasjonen kan være verdifull i etterkant, når det skal føres en tilstandsoversikt over hva som har hendt og hvilke avgjørelser som ble tatt. Det å dokumentere umiddelbart etter at informasjonen har kommet inn, sparer tid. For dersom dette ikke gjøres med en gang kan informasjon gå tapt, eller en blir nødt til å kontakte kilden på nytt for å få det repetert [9, s.457-8].

Det bør som et minimum loggføres

- alle hendelser
- alle beslutninger som tas
- all ekstern kommunikasjon, inkludert hvem det ble kommunisert med og et referat av innholdet
- med nøyaktig klokkeslett og hvem som var involvert

Det bør jevnlig tas kopier av journalen. Denne kopien bør lagres på et trygt sted, gjerne hos en tredjepart som kan garantere for integritet [9, s. 457-8].

Den andre kilden vi har tatt for oss er *BSI-Standard 100-4, Business Continuity Management* [11]. BSI (Bundesamt für Sicherheit in der Informationstechnik) er det føderale byrået for informasjonssikkerhet i Tyskland. Disse har utgitt en rekke standarder, veiledninger og anbefalinger innenfor informasjonssikkerhet som igjen er basert på ISO-standarder. *100-4 Business Continuity Management* tar for seg hendeleseshåndtering.

Alle verktøy som kan brukes til å notere på kan benyttes til dokumentasjonsformål. Dette inkluderer internettverktøy og tekstredigeringsprogrammer på bærbare enheter. Verktøy og kriseplaner bør være lett tilgjengelig, eksempelvis liggende på en minnepinne. Dette gjør at de er raskt tilgjengelig under en krise, uten at de må letes opp på en tjener eller hentes ut av en

safe. Investering i opplæring av personell er viktigere enn programvare, da teknologi alene ikke løser kriser [11, s.30].

Viktige punkter det må tas hensyn til ved valg av elektroniske verktøy [11, s.101-2]:

- Programvaren bør være støttet på flere plattformer, eller via vebbteknologi sikre plattformuavhengighet.
- Det bør være tilknytninger til andre verktøy som allerede finnes i organisasjonen, som f.eks. kundeservice-, lagerstatus-, eller alarmsystemer.
- Programvaren bør være brukervennlig. Dette gjelder særlig for dokumentasjonsverktøy.
- Det bør være mulig å skape ulike visuelle fremstillinger av data, avhengig av behov, situasjon eller brukerens rolle.
- Systemet må ha sikkerhetsfunksjonalitet som beskytter innlagt data.
- Verktøyet bør ha høy tilgjengelighet.
- Verktøyet bør være robust, med tanke på at en krise er en stressende situasjon, og kan lede til flere brukerfeil.

Tabell 9 viser hvilke felter det er aktuelt å ha med i en journal [3, s.58].

Nr.	Tid	Fra/Til	Tekst/Innhold	Anmerkning/Tiltak

Tabell 9: Eksempel på loggbok

Ut i fra de kildene vi har brukt kan det tyde på at det ikke finnes noen mønsterpraksis for hva som bør loggføres i en journal, eller hvordan et krisehåndteringsprogram anbefales å bygges opp. Nærmest er BSI [11] med listen over krav til elektroniske verktøy. Det er heller ikke nevnt noe om problematikken rundt manglende infrastruktur. Vi vil ta med oss den informasjonen vi har funnet, og supplere denne med egne erfaringer vi oppnår gjennom utvikling og testing av programvaren.

4.2 Brukervennlighet

Loggeverktøyet skal brukes i situasjoner med stort arbeidstrykk hvor brukeren er stresset og sliten. Samtidig kan det være en stund siden vedkommende sist fikk opplæring i bruk av programvaren. Dette gjør at kravet til et godt brukergrensesnitt er til stede. Vi har erfaring innen bruk av gode og dårlige tekniske løsninger, men lite kjennskap til teorier innenfor ergonomi i digitale medier. Vi ble derfor nødt til å tilegne oss mer kunnskap om brukergrensesnitt og kognisjon.

Anthony Franco er medforfatter på en bok [12] som omhandler utforming av brukervennlig programvare. Denne inneholder blant annet åtte kriterier for brukervennlig programvare. Hovedpunktene i boka oppsummerer han i sin blogg [13]. Vi gjengir disse her sammen med våre valg over hva vi har tenkt til å fokusere på for hvert tema.

4.2.1 Brukervennlig programvare hjelper brukeren med å oppnå mål

En bruker av Libellus vil ha som hovedmål å føre journal, samt å få oversikt over eksisterende hendelser. Disse funksjonene må være intuitivt implementert i grensesnittet, slik at brukeren ikke trenger å bruke tid på å lese gjennom dokumentasjon for å utføre de ønskede handlingene.

4.2.2 Brukervennlig programvare er responsiv

I dag forventes det at datasystemer og programvare oppfører seg smidig. Dersom programmet føles tregt vil brukeren la seg irritere av dette. Denne negative følelsen kan igjen smitte over på brukerens evne til å tenke klart og ta riktige avgjørelser under krisen.

4.2.3 Brukervennlig programvare er konsistent og troverdig

Informasjonen brukeren skal legge inn i journalen vil ofte være av sensitiv art. Dersom systemet i seg selv ikke ser ut til å kunne stoles på, vil brukeren kunne ha motforestillinger til å legge inn data, noe som gjør at det er viktig at programvaren ser profesjonell ut.

4.2.4 Brukervennlig programvare gir brukeren tilbakemeldinger

En bruker som foretar en handling, forventer å få en visuell tilbakemelding på at den er utført. Det samme gjelder dersom noe går galt, i så tilfelle må brukeren bli presentert med en relevant og forståelig feilmelding det kan gjøres noe med.

4.2.5 Brukervennlig programvare oppfører seg konsekvent

Dersom brukeren har lært seg å benytte en funksjon i systemet, vil han eller hun forvente at resten av programvaren oppfører seg konsekvent i liknende operasjoner. All innlegging av informasjon i systemet bør ha tilsvarende like visuelle elementer, dette gjelder også for funksjoner som gir tilbakemeldinger til brukeren.

4.2.6 Brukervennlig programvare føles kjent å bruke

Selv om programvaren er innovativ og ny av sitt slag, skal det ikke kreves at brukeren må sette seg inn i helt nye måter å finne frem eller arbeide på. Dette løses ved at programmet har gjennkjennbare elementer som menylinje øverst, med fanene fil, rediger, hjelp, osv. Ikoner som benyttes bør også være universelle.

4.2.7 Brukervennlig programvare er effektiv

Programmet skal være et løsningsverktøy under en krise. Det er viktig at det benyttes minst mulig tid på å legge inn data, slik at det blir mer tid til å løse krisen. Dette kan oppnås ved å sørge for at plasseringen av de elementene brukeren interagerer med har en naturlig flyt etter hverandre.

4.2.8 Brukervennlig programvare er elegant og engasjerende

Det er lite motiverende å bruke et program med utseende fra 90-tallet. Dersom programvaren ser gammel ut kan brukeren tenke at programmet er gammelt, og dermed mindre bra.

I løpet av denne gjennomgangen har vi fått større innblikk i brukergrensesnitt og teorier. Temaet er stort og vi ser i ettertid at vi kunne ha fokusert mer på hva evangelistene innen emnet sier, som Jakob Nielsen eller Jesse James Garrett. Vi har registrert at det er flere debatter rundt brukergrensesnitt og særlig universell utforming [14]. For prosjektet og applikasjonens del har vi drøftet tanker om mulige strategier for hvordan vi kan gjøre designet mest mulig brukervennlig.

4.3 Eksisterende løsninger og bruken av disse

Innledningsvis i oppgaven antydet vi at verktøyene brukt for krisehåndtering blant bedrifter og organisasjoner i stor grad bærer preg av at de krever infrastruktur. Oppgaven vår var å lage en ny type programvare som er så frigjort som mulig på dette området, men samtidig har vi hatt interesse av å høre hva som eksisterer og benyttes.

Vi har vært i kontakt med bedrifter og organisasjoner hvor vi har spurt om hvilken programvare de bruker i krisehåndteringsarbeidet. Vi har ikke grunnlag for å si at de vi har snakket med er representative for norske bedrifter. De er snarere tilfeldige vi har hatt kontakt med i andre sammenhenger i løpet av bacheloroppgaven. Like fullt har vi valgt å ta det med fordi vi mener det belyser et av hovedpoengene ved oppgaven vår, nemlig infrastrukturtilknytning. Når det kom til konkret planverk, var de fleste nokså restriktive på informasjonsdelingen, så vi valgte å ikke gå noe dypere inn i det.

- UiO-CERT ved daværende leder Margrete Raaum (27.09.2013).
  - Til daglig og under kriser benyttes programvare på lynmeldingsprotokollen XMPP for internkommunikasjon. Når det gjelder sammenføring av notater o.l. i forbindelse med rapporter, brukes det et tekstredigeringsverktøy. De mer aktive verktøyene som er i bruk for å løse eller etterforske hendelser gir fra seg tidsstempler, som igjen legges inn i rapporten. For å håndtere ekstern kommunikasjon og oppfølging av saker nyttes RTIR [15], som er en utvidelse av RequestTracker [16] med Incident Response-funksjonalitet. RequestTracker er for øvrig lisensiert under GPLv2.

- DSS ved Kontorsjef Sven-Erik Egge (21.03.2014).
  - Departementenes Sikkerhets og Serviceorganisasjon bruker RequestTracker. Dersom eventuell infrastruktur er utilgjengelig, går de over til "eldre metoder", uten at Sven-Erik ville utdype dette nærmere.

- RingNett AS, en mindre ISP, ved driftsingeniør Sondre Rabbe (10.03.2014).
  - Fører fortløpende kriselogg på sitt intranett, der de benytter en blogg-løsning. Data fra overvåkningssystemet Big Brother linkes inn i blogginnleggene med tidsinformasjon for når det har skjedd noe i nettet eller på tjenere. Bedriften bruker bloggen som en tekstuell debrifing, og søker opp avsluttede hendelser etter merkelapper.

- Forsvaret benytter en proprietær løsning som heter XOmail [17], utviklet av Thales. Systemet kan kobles opp og driftes uavhengig av resten av forsvarets infrastruktur, men er allikevel avhengig av andre komponenter, som for eksempel brukerdatabase, der den bruker Active Directory.

- Flere norske flyplasser og flyselskap deriblant helikopterselskapet Airlift, bruker OpsCom [18]. Dette systemet er utviklet av OpsCom Systems og er et kombinert hendelsesrapportering og håndteringssystem. Systemet er proprietært, og blir driftet av utvikleren. En aktiv internettilagang er helt nødvendig for å kunne bruke programvaren.

- Kristestøtteenheten, underlagt Justis- og beredskapsdepartementet benytter en løsning som heter *Beslutningsstøtte- og loggføringssystem [19]* (KSE-CIM) utviklet av One Voice. Dette systemet brukes også av andre i offentlig sektor. Programvaren er proprietær, men kan installeres og driftes ute i kundens infrastruktur eller hos One Voice.

Kort oppsummert viser dette oss at ingen av de vi har vært i kontakt med har et loggføringsprogram som ikke har bindinger til intern og eller ekstern infrastruktur. Samtidig er det bra at samtlige har programvare som kan brukes til å føre logg.

4.4 Teknologivalg

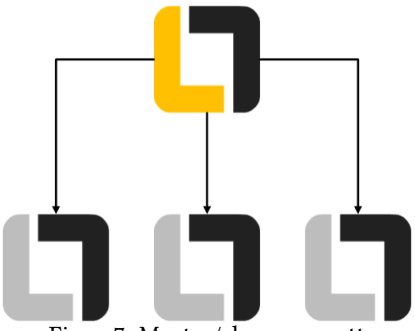
Vi mener at et viktig valg i et utviklingsprosjekt når det gjelder teknologi og programvare, er hvilket databasesystem som skal brukes. I vårt tilfelle skal databasen holde orden på journalinnlegg og vedlegg. Dette er i utgangspunktet en oppgave de fleste systemer takler, derfor har vi også lagt noen andre kriterier til grunn.

Et typisk scenario for bruken av programvaren som skal utvikles, er der det inntreffer en hendelse hvor det er usikkerhet rundt hva som er operativt og kan stoles på av infrastruktur. Med infrastruktur mener vi her klientdatamaskiner, lokalt nettverk, tjenere, og eksterne nettverk som f.eks. Internett. I ytterste konsekvens må nytt datautstyr anskaffes på raskeste og enkleste måte.

I hendelsens tidlige fase, før det har blitt avdekket nok informasjon, startes det en lokal journal på en enhet. Når krisestaben har samlet seg og opplysninger begynner å komme inn, blir det etter hvert nødvendig å koble opp en enda en datamaskin for å effektivisere loggføringen. Allerede innlagte data må da kunne synkroniseres inn i den nye enheten. Det kan også tenkes at logging har blitt startet på to ulike maskiner uavhengig av hverandre, og når nettverkskommunikasjon blir tilgjengelig må innleggene automatisk kunne samordnes.

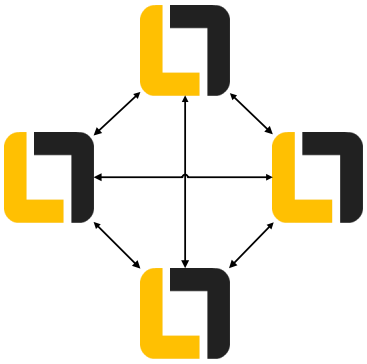
Dette fører til to krav som er dekket i kravspesifikasjonen. Det ene er portabilitet, at databasesystemet er tilgjengelig på ulike operativsystem. Det andre er replikeringsmuligheter mellom ulike databaser. Replikering og synkronisering kan løses av det omliggende programmet, men dette er en omfattende oppgave å få til å virke. I tillegg må det også legges inn funksjonalitet for å sikre informasjonsintegriteten, slik at synkronisert data ikke overskriver hverandre.

I all hovedsak er det to måter å replikere på, disse kalles mester/slave og multimester. I et mester/slave-oppsett som vist i figur 7 er en dedikert node i gruppen utpekt som mester, denne er da den eneste som kan modifisere dataelementene. Data blir så sendt ut til slavene, og dersom mesteren faller ut er det mulig å la en slave overta som mester [20] [21, kap. 4.3].



Figur 7: Mester/slave-oppsett

I et multimester-oppsett som vist i figur 8 er alle noder mestere, dette betyr at alle har anledning til å lese og skrive til databasen. Eventuelle endringer på databasen forplanter seg så ut mellom alle nodene [20] [21, kap. 4.3].



Figur 8: Multimester-oppsett

Muligheten for effektiv samordning av data innlagt på to enheter støttes kun i et multimester-oppsett [22] [21, kap. 4.4]. Et valg av multimester reduserer også infrastrukturbehovet. Totalt fører dette til tabell 10 som viser en oversikt over databasesystemer som er portable til to eller flere operativsystem, hvorav et av dem er Windows, og støtter multimester-replikering. Vi har utelatt proprietære databasesystemer, da oppdragsgiver ønsket at løsningen skulle være basert på åpen kildekode.

	CouchDB	Ingres	MySQL	PostgreSQL
Portabilitet	Android, BSD, GNU/Linux, OS X, Solaris, Windows	32/64-bit GNU/Linux, 32-bit Windows	FreeBSD, GNU/Linux, OS X, Solaris, Windows	BSD, GNU/-Linux, OS X, Solaris, Windows
Replikering	Multimester-replikering, Mester/slave-replikering	Multimester-replikering, Mester/slave-replikering	Multimester-replikering, Mester/slave-replikering, MySQL Cluster	Multimester replikering (med tilleggsverktøy)
API	RESTful (Gjennom HTTP-kommandoer)	Bindinger til programmeringsspråk	Bindinger til programmeringsspråk	Bindinger til programmeringsspråk
Modell	Dokumenter (JSON-objekter)	Relasjonell	Relasjonell	Relasjonell
Offisiell dokumentasjon	God [23, 24]	Dårlig [25]	God [26]	God [27]
Lisens	Apache 2	GPL	GPL 2	PostgreSQL License

Tabell 10: Aktuelle databasesystemer

Det var viktig for oss at teknologien vi bestemte oss for var godt dokumentert, slik at vi kunne slippe å bruke tid på å prøve oss frem på hvordan programvaren virket. Vi satte derfor opp punktet offisiell dokumentasjon, med det mener vi en eller flere offisielle kilder som blir vedlikeholdt og oppdatert. Ingres falt igjennom på dette området, da søkbarheten er dårlig og dokumentasjonen i stor grad består av av flere PDF-dokumenter. Det bæres også et preg av at det er et kommersielt selskap som står bak programvaren. PostgreSQL har egentlig ikke multimester-replikering rett ut av boksen, men det finnes en forgrening som har det [28].

Etter denne gjennomgangen gjensto kun CouchDB og MySQL. Merk at vi ikke har valgt å

ta med MariaDB som er en forgrening av MySQL, da de har tilsvarende funksjonalitet når det gjelder replikering [29].

Selv om databasesystemet støtter replikering, trenger det ikke nødvendigvis å bety at prosessen med å sette det opp er like rett frem. Vi gjengir ikke fremgangsmåten her, men etter å ha prøvd begge sier vi oss enige i følgende sitat:

> Getting replication going with MySQL is like jump-starting an old car with a manual transmission. Getting replication going with CouchDB is like push-button remote starting a modern luxury vehicle.
>
> *- Will Conant [30]*

Sikkerhetsmessig har CouchDB en fordel når det kommer til dataintegritet, da den ved å føre revisjoner for når endringer blir gjort, gjør det enkelt å gå tilbake i tid å spore disse [31]. MySQL har ikke en fullgod løsning på dette, da alternativet er å analysere binærloggen [32] eller implementere egendefinerte triggere [33].

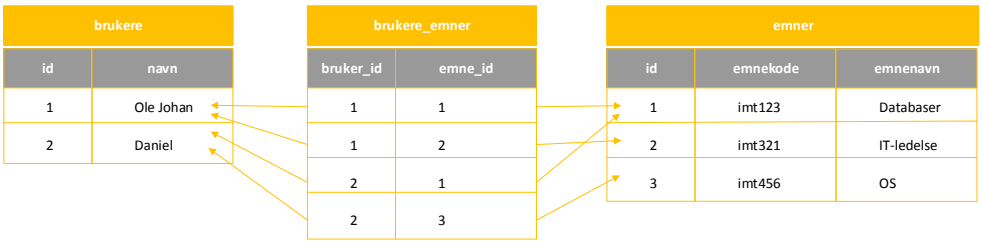
En annen og viktig forskjell på MySQL og CouchDB er datamodellen. MySQL har tradisjonelt alltid vært SQL-basert, selv om den fra versjon 5.6 har fått en mindre støtte for NoSQL [34]. CouchDB har kun en ren NoSQL-tilnærming med JavaScript-objekter [31]. NoSQL var også nytt for oss, og noe vi ville lære mer om. Dette tatt i betraktning bestemte vi oss å gå for CouchDB.

4.5 NoSQL

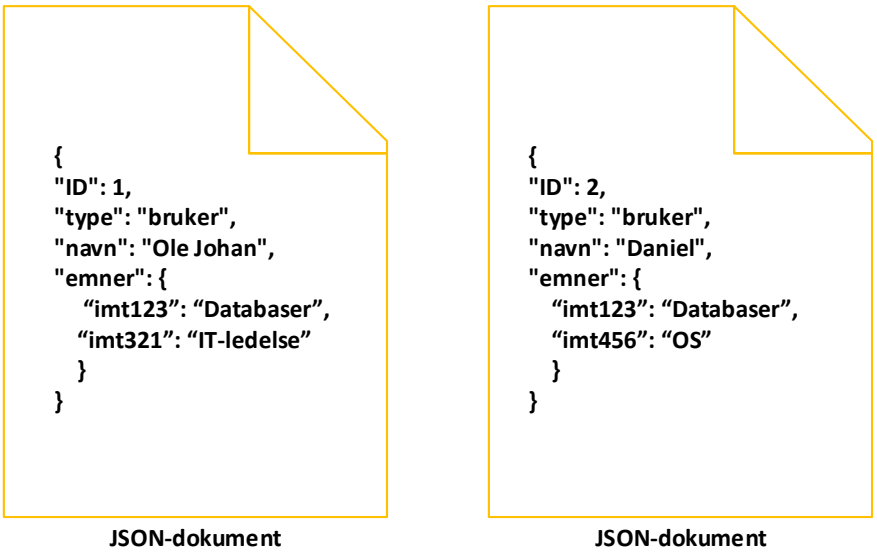
Selv om ordet NoSQL har røtter tilbake til 1998 [35], er det først i de senere årene denne trenden innenfor databaseverdenen virkelig har tatt av. NoSQL går bort fra relasjoner og bytter plass på rader og kolonner fra tradisjonelle relasjonsdatabaser. Det finnes ingen skjemaer for å strukturere data, ei heller normalformer. Dette åpner for mer fleksibilitet, men går på bekostning av at applikasjonen overtar mye av ansvaret, som blant annet dataintegritet og ACID-prinsippene [21].

En NoSQL-database lagrer typisk all data i et dokument som i praksis er et JavaScript-objekt, populært kalt JSON. Det finnes også andre datastrukturer i NoSQL-verdenen, som grafdatabaser, kolonneorienterte og nøkkel-verdi-struktur [21], men vi har valgt å fokusere på dokumentbasert fordi det er dette CouchDB bruker.

I en tradisjonell relasjonsdatabase vil en gjerne dele opp data i så mange tabeller som mulig for å normalisere den, slik som vist i figur 9.



Figur 9: Data i SQL



Figur 10: Samme data representert som JSON

I et NoSQL-dokument som vist i figur 10 er all nødvendig data tilgjengelig på et sted, dette fører til at informasjon ikke må hentes ut og knyttes sammen fra flere tabeller. Noe som gjør uthenting av data enklere og raskere, men fører til duplisering. I dag er ikke lagringskapasitet et utbredt problem lenger, da det nå er ytelse som er viktigst [36].

4.6 CouchDB

CouchDB er en database og vebbtjener skrevet i Erlang, som blir utviklet av Apache Software Foundation [31]. CouchDB sitt slagord er *relax*, og tanken bak programvaren er enkelthet når det gjelder databaser og vebb.

Kort fortalt har CouchDB følgende funksjonalitet:

- JSON dokument-orientert
- RESTful API
- views
- replikering
- revisjonering

Vebbtjenerfunksjonaliteten gjør at det er mulig å laste opp vebbfiler som vedlegg og tilby de gjennom CouchDB. På denne måten kan hele vebbapplikasjonen ligge inne i databasen, istedenfor å ha databasen og vebbfilene separert.

CouchDB sitt administratorgrensesnitt Futon har visse likheter med databaseverktøyet phpMyAdmin. Her er det blant annet mulig å se på og redigere databasen, endre konfigurasjonsfilen og sette opp replikering.

API-et er av typen RESTful, som bruker HTTP-kommandoene GET, PUT, POST og DELETE, for å manipulere databasen med de grunnleggende metodene Create, Read, Update, Delete

(CRUD). Dette gjør at manipulasjon på databasen kan gjøres fra en terminal med cURL [37] eller et socket-bibliotek, som de fleste programmeringsteknologier har innebygd. Det finnes også bibliotek eksplisitt laget for CouchDB for å kjøre spørringer enda enklere. Eksempelvis kommer CouchDB med en egen jQuery-plugin.

Eksempeler på bruk av API-et [38]:

```
curl -X PUT http://localhost:5984/album
```

Kodesnutt 4.1: Oppretter en database med navn "album"

```
curl -X POST -H "Content-Type: application/json" http://localhost:5984/album -d
    '{"title":"The Dark Side of the Moon"}'
```

Kodesnutt 4.2: Lagrer et nytt dokument i databasen

```
curl -X GET http://localhost:5984/albums/_all_docs
```

Kodesnutt 4.3: Henter ned alle dokumenter fra databasen

```
curl -X DELETE http://localhost:5984/album
```

Kodesnutt 4.4: Sletter databasen

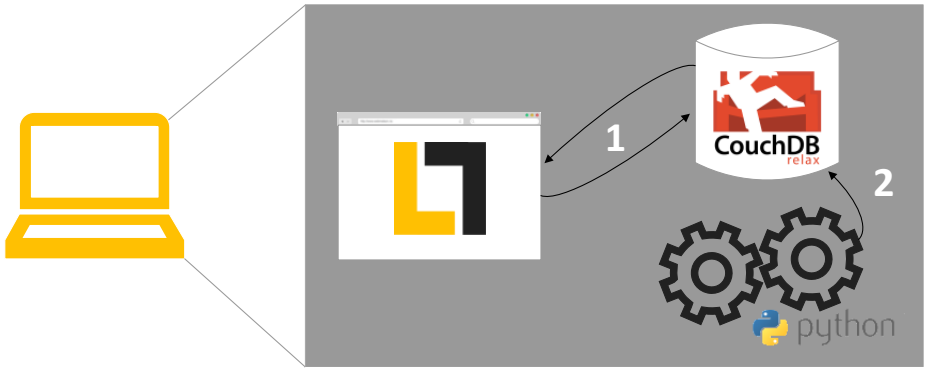
CouchDB og NoSQL-databaser bruker ikke SQL som språk for å foreta spørringer mot databasen. Derfor er det ikke mulig å kjøre *SELECT * FROM users WHERE username = 'olanordmann'*. CouchDB har løst dette med å implementere views, eller utsnitt. Et view er et JavaScript-objekt på lik linje med alle andre dokumenter, og brukes til å manipulere data før den blir sendt til klienten. Her er det vanlig å bruke en MapReduce-funksjon for å filtrere ut ønsket innhold. Det mest typiske bruksområdet er å ha et type-attributt, og sende med et URL-parameter i spørringen der verdien er hva slags type dokument som ønskes.

5 Implementasjon

I dette kapittelet blir det beskrevet eksempler fra implementasjonen. Vi tar for oss ulike valg som ble tatt i utviklingsprosessen med hensyn til kravspesifikasjon og design. Diskusjon med oppdragsgiver rundt funksjonalitet vil også bli utdypet her. Til sist beskrives testing og prosessen rundt denne.

5.1 Systemtegning

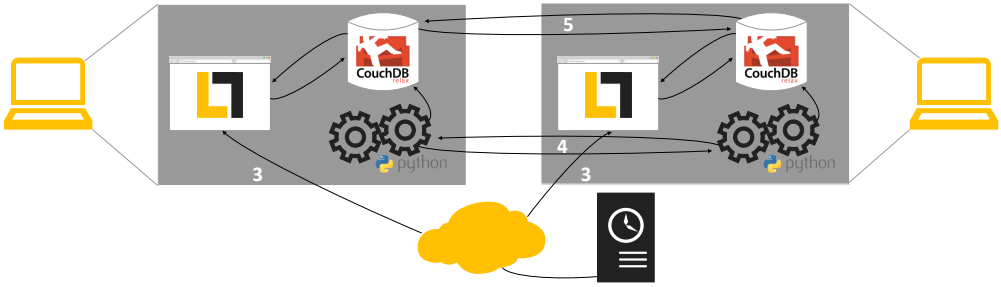
I dette avsnittet viser vi grunnleggende hvordan Libellus er bygget opp med sine hovedkomponenter. Systemtegningen er fordelt over to illustrasjoner, figur 11 og figur 12. Libellus består av CouchDB som database og vebbtjener, en Python-demon for å finne andre instanser og en applikasjon skrevet i JavaScript, HTML og CSS som blir tilgjengeliggjort via vebbtjeneren. Merk at brukeren selv må stille med en datamaskin med et støttet operativsystem og en moderne nettleser.



Figur 11: Figuren viser en enkeltstående datamaskin uten nettverkstilknyttning; Dette er den minste infrastrukturen en Libellus-instans kan kjøre på.

1. Nettleseren på maskinen kommuniserer med CouchDB for å hente ned og laste opp ny data.

2. Python-demonen kjører i bakgrunnen og leter etter andre instanser av Libellus.



Figur 12: Her har nettverksinfrastrukutur i form av en ruter, eller et datamaskin-til-datamaskin-nettverk og Internettilgang blitt tilgjengelig. Det har også blitt koblet opp en ny maskin i nettverket.

3. Nettleserene henter tid fra en kjent tjener på Internett.

4. Python-demonen oppdager en annen Libellus-instans og sender informasjon om denne til CouchDB.

5. CouchDB replikerer data med den andre instansen.

5.2 Verktøy

Selv om medlemmene av gruppen er tilhengere av å anvende programvare med åpen kildekode, og ønsker å frigjøre all vår kode, har vi et pragmatisk forhold til programvarebruk. Vi benytter altså de verktøyene som er mest effektive for oss til den oppgaven som skal løses.

I utviklingsprosessen har vi benyttet Google Chrome [39] med sin JavaScript-konsoll. Vi har også testet løsningen vår i andre nettlesere, som Mozilla Firefox [40] og Microsoft Internet Explorer [41], for å sjekke kompatibilitet. Før integrering av nye moduler har vi konferert med tjenesten caniuse.com [42] som lister opp kompitablitetsstøtte.

De fleste av de grafiske elementene i rapporten er generert med Microsoft Visio [43]. Logo og grafikk på nettsiden og i programmet er laget i Inkscape [44].

Som feilrapporteringssystem har vi satt opp og benyttet Tiny Issue [45].

For å skrive programkode har vi brukt tekstredigeringsverktøyene Vim [46], gEdit [47], Notepad++ [48] og Sublime Text 2 [49].

Google Docs [50] har blitt benyttet til samarbeid på dokumenter, mens den endelige rapporten ble utarbeidet i LaTeX-verktøyet TeXworks [51]. For kode og fildeling har vi brukt Dropbox [52]. Årsaken til at vi har valgt dette kontra Git eller SVN, er automatikken i oppdateringene. Med denne løsningen slipper vi å manuelt hente ned data for å teste på lokalt miljø.

Utvikling av hjelpedokumentasjon har foregått i Sphinx [53].

For opplasting i CouchDB har vi brukt CouchApp [54], som er et tredjeparts-verktøy for å gjøre utvikling av applikasjoner til CouchDB mer triviell. Den setter opp en filstruktur, og gjør det mulig å sende kodeendringer direkte inn i CouchDB, der applikasjonen lever.

5.3 Programmeringsteknologi

Før vi startet på oppgaven så vi for oss å lage en vebbasert løsning. Dette muliggjør kryssplattform, så lenge brukeren benytter en moderne nettleser. CouchDB legger opp til å bli brukt mot et vebbgrensesnitt fremfor en skrivebordsapplikasjon, dette gjør at denne teknologien passer vårt prosjekt. En annen fordel med å lage programmet vebbasert, er at mobile enheter også kan bruke applikasjonen uten at det trengs å lage native utgaver av programmet til hver plattform.

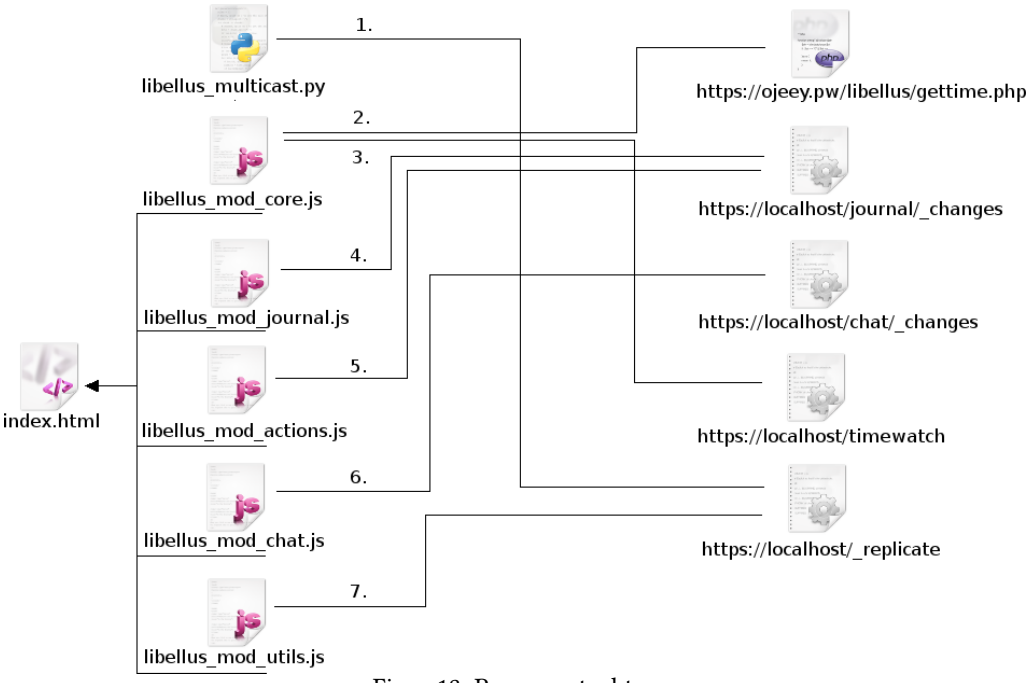
Når det kommer til utvikling for vebb er det HTML, CSS og JavaScript som er de naturlige klientside-språkene. PHP, Python, Ruby eller Perl er de vanligste teknologiene for bruk på tjenersiden for å håndtere innhold, sesjoner og databasekall. Når CouchDB gjør det mulig med databasekall med JavaScript, er det ikke nødvendig med enda en teknologi og avhengighet på tjenersiden.

Med JavaScript, asynkrone bakgrunnskall og ny funksjonalitet i HTML5, kan en vebbside i dag føles som en interaktiv applikasjon. I vedlegg B: *Innføring i JavaScript, AJAX og OOP* ligger den lærdommen vi har tilegnet oss om JavaScript i prosjektet. JavaScript kan ikke utføre nettverksoperasjoner, noe som gjorde at vi trengte støtte fra en annen teknologi ved siden av. For å søke opp aktive instanser fant vi et skript skrevet i Python, som er er beskrevet i avsnitt 5.10. Python er også mye brukt, godt dokumentert og støttet på de plattformene Libellus er aktuell for.

5.4 Kodestandard

I prosjektet har vi valgt å følge kodestandarden til Python [55]. Selv om den tar for seg Python-syntaks, kan kodestilen følges i de fleste språk. Vi valgte denne standarden fordi den er veldig kjent og akseptert som en anerkjent kodestil. I tillegg har begge gruppemedlemmene god erfaring med den.

5.5 Programstruktur



Figur 13: Programstruktur

Figur 13 viser en forenklet oversikt over hovedfilene som er involvert i systemet. Alle spørringer går over HTTPS-protokollen, og data blir sendt som JSON. Alle vebbfilene er lagret og kjøres på CouchDB og som figuren viser, er alle JavaScript-filene inkludert i *index.html*. Python-skriptet er lagret utenfor databasen, men blir påkalt av CouchDB som en bakgrunnsprosess.

1. Python-skriptet leter etter andre Libellus-instanser ved å kontinuerlig sende ut spørringer i nettverket. Dersom noen enheter lytter på samme portnummer, blir informasjon om disse sendt med en POST-spørring til *_replicate*. Deretter vil de oppdagede enhetene synkronisere sine lokale databaser, ved å ta imot og sende fra seg data til den Libellus-instansen som kjørte Python-skriptet.

2. Kjernemodulen *libellus_mod_core.js* sender en GET-spørring til en tidstjener på Internett, hvor svaret er et JSON-objekt med et Unix-tidsstempel.

3. Dersom lokaltiden på den kjørende instansen av Libellus endres, ved at brukeren manuelt stiller klokka, loggføres dette med å sende en POST-spørring til timewatch-databasen med informasjon om hendelsen, inkludert nåværende lokal- og eksterntid, identifikasjonsnummer (UUID), og brukernavn.

4. Journalmodulen sender regelmessig en GET-spørring til journaldatabasen sin *_changes*-fil, som responderer med alle de siste endrede journalinnlegg, kommentarer og oppfølgingsmerknader (actions).

5. Actions-modulen gjør akkuratt det samme som journalmodulen, men ved hjelp av et view på databasesiden, vil kun actions bli sendt tilbake til nettleseren.

6. Chat-modulen sender en GET-spørring til *chat/_changes* for å hente ned alle nye meldinger fra chatten.

7. Verktøymodulen har mulighet til å fjerne innlagte replikeringsforbindelser. Dette gjør den ved å sende en DELETE-spørring med ID-en til det aktuelle replikeringsobjektet som skal fjernes.

5.6 Imperativ eller objektorientert

Tidlig i prosjektet hadde vi veldig lyst til å få til en objekt-orientert løsning, men med den kunnskapen vi hadde om objekt-orientert JavaScript på det tidspunktet, viste det seg å være vanskelig og kronglete. Det endte med at vi gikk for tradisjonell bruk av JavaScript.

Noe av det som var utfordrende med OOP, var eksempelvis feilmeldinger. Siden objekter i JavaScript ikke har en type, kan feilmeldinger som *Uncaught TypeError: Object [object Object] has no method 'setValue'* forekomme. Dette gjør det vanskelig å rette programfeil, for dersom det finnes mange objekter rundt i koden, er det vanskelig å finne frem til det objektet som forårsaker feilmeldingen.

Et annet problem var å skille HTML fra JavaScript, da et objekt-orientert konsept er å ha klasser så rene som mulig slik at de teoretisk sett også kan bli brukt utenfor et nettlesermiljø. Vi kunne ha brukt et malbibliotek som f.eks. Handlebars [56] for å adskille, men dette løser ikke problemet med renhet, da en fortsatt sitter igjen med malkoder. Disse malkodene er heller ikke universelle og vi blir låst til et bestemt bibliotek. Vi valgte derfor å lage egne funksjoner som vist i kodesnutt 5.1 for å generere HTML, istedenfor å sette oss inn i en malmotor.

```
1  function insertJournalTable() {
2      var html = '';
3      html += '<table id="table_journal" class="tablesorter_default">';
4      html +=   '<thead>';
5      html +=     '<tr class="tr_header">';
6      html +=       '<th>Nr.</th>';
7      html +=       '<th><span>Author</span></th>';
8      html +=       '<th><span>Subject</span></th>';
9      html +=       '<th><span>Time</span></th>';
10     html +=       '<th><span>Class</span></th>';
11     html +=       '<th><span>#Com</span></th>';
12     html +=       '<th><span>#Act</span></th>';
13     html +=       '<th nowrap><span>Added (Internet)</span></th>';
14     html +=       '<th nowrap><span>Added (Local)</span></th>';
15     html +=     '</tr>';
16     html +=   '</thead>';
17     html +=   '<tbody>';
18     html +=   '</tbody>';
19     html +=   '<tfoot>';
20     html +=     '<tr class="tr_header">';
21     html +=       '<td colspan="9"> </td>';
22     html +=     '</tr>';
23     html +=   '</tfoot>';
24     html += '</table>';
25     $('#section_journal').append(html);
26 }
```

Kodesnutt 5.1: Eksempel på en funksjon som dynamisk genererer journaltabellen

5.7 Modulbasert

Selv om vi ikke har brukt OOP i bacheloroppgaven, har vi delt opp koden i mange funksjoner som er så generiske som mulig. Hver modul har et sett med funksjoner øverst i koden, mens nederst, inne i en *$(document).ready()* vil funksjonene bli kjørt, informasjonskapsler generert, handlinger ved triggere definert etc. Dette for å tvinge vebbsiden til å laste ferdig før den setter i gang med å generere dynamisk innhold, slik at alle HTML-elementer er på plass før de blir manipulert.

I tillegg til modulene er det laget en kjernefil med generelle verktøy og tidsfunksjoner, kalt *libellus_core.js*. Ved siden av denne finnes det en konfigurasjonsfil, *libellus_config.js*, som gjør det enklere å endre på oppdateringstider og begrensning på opplastingsstørrelse etc.

Hele programmet er satt sammen av separate uavhengige moduler, hvor alle har "mod" i filnavnet, som f.eks. *libellus_mod_journal.js* og *libellus_mod_actions.js*. Om disse studeres, vil en utvikler med mindre kjennskap til prosjektet fort klare å lage og implementere nye moduler på egenhånd. I disse modulfilene genereres alt av HTML tilknyttet moduler dynamisk med JavaScript. Dette gjør at *index.html*, som er rotdokumentet til hele applikasjonen holdes helt ren og opprettholder en ryddig struktur.

Det eneste som må endres i *index.html* for å installere en ny modul er en menyknapp og inkluderingen av selve modulen. Dette er vist i kodesnutt 5.2 og 5.3. Menyknappen kunne også vært generert dynamisk i hver modul, men vi var redde for at rekkefølgen på knappene ville ha blitt tilfeldig innsatt hver gang.

```
<li><a id="_journal" href=""><span class="glyphicon glyphicon-list"></span>
    Journal</a></li>
```

Kodesnutt 5.2: Eksempel på en menyknapp

```
<script src="script/libellus_mod_journal.js"></script>
```

Kodesnutt 5.3: Eksempel på inkludering av modul

I kodesnutt 5.4 vises den endelige koden for kontinuerlig oppdatering av kommentarer, handlinger og journalinnlegg.

updateContent() blir kjørt i et gitt intervall, der standard er på ett sekund. Kort fortalt kaller den på tre andre funksjoner, som hver seg henter ned forskjellige typer data.

```
function updateContent() {
    getChangedJournalDocs(function() {
        getChangedJournalDocsComments(function() {
            getChangedJournalDocsActions(function() {
                console.log("Updated content");
            });
        });
    });
}
```

Kodesnutt 5.4: updateContent()

I kodesnutt 5.5 vises et utdrag fra funksjonen som henter journalinnlegg. Hver operasjon er kommentert i koden. De to andre funksjonene som henter ned kommentarer og oppfølgingsmerknader ser veldig like ut, noe som fører til duplisering av kode. Det kunne vært én funksjon med forskjellige parametere som endret innholdet, men vi har valgt å lage tre veldig like funksjoner. Grunnen til dette er for å få funksjonene kortere og mer lesbare, uten mange if-setninger som endrer oppførselen. Funksjonen sender en GET-spørring til filen *_changes* i CouchDB, som henter alle endringer etter ett gitt sekvensnummer som blir sendt med som parameter. I starten hentet vi alle innlegg i databasen for hvert intervall med en spørring til *_all_docs*, mens

nå blir det kun hentet ut løpende endringer. Dette gjør at programmet blir raskere og sparer båndbredde.

```
// Fetch and insert all journal entry into the journal
function getChangedJournalDocs(callback) {
    var nr = 0;
    // The URL will fetch docs that is changed after last sequence number.
    var url = '/journal/_changes?filter=app/type&type=journal&since=' +
        getCookie('journal_last_seq');

    $.getJSON(url, function(changes) {
        // Now we update the last sequence number, so we don't fetch old ones
            next time.
        document.cookie = 'journal_last_seq=' + changes.last_seq;

        // Remove deleted docs
        function filterDeleted(element) {
            return element.deleted != true;
        }
        // A new array with deleted docs filtered out
        var filtered_changes = changes.results.filter(filterDeleted);
        var i = 0;
        var len = filtered_changes.length;

        // If there is any changes, we continue
        if (len > 0) {
            // Iterates through all changed docs
            $.each(changes.results, function(index, item) {
                // Opens current doc
                $.couch.db('journal').openDoc(item.id, {
                    success: function(doc) {
                        insertJournalEntry(doc);
                        // Initates tablesorter for the inserterd appendix table
                        $('table[name=table_appendix][data-id=' + doc._id + ']')
                            .tablesorter({selectorHeaders: '> thead > tr > th'
                            });

                        // When we have all currently new changes
                        if (++i == len) {
                            // We reinitialize tablesorter with the new data
                            if (typeof $('#table_journal')[0].config != 'object'
                                ) {
                                $('#table_journal').tablesorter({selectorHeaders
                                    : '> thead > tr > th'});
                            }
                            callback();
                        }
                    }
                });
            });
        } else {
            callback();
        }
    });
}
```

Kodesnutt 5.5: getChangedJournalDocs(callback)

Som vist i kodesnutt 5.6 starter hele oppdateringsprosessen, hvor først *updateContent()* blir kalt, og deretter kjørt igjen hvert sekund, eller hva verdien til *cfg_upd_journal_intval* er satt til.

```
$(document).ready(function() {
    updateContent();
    setInterval(updateContent, cfg_upd_journal_intval);
});
```

Kodesnutt 5.6: document.ready()

5.8 Datapresentasjon

I databasen vil et journalinnlegg være representert med et JavaScript-objekt, vist i kodesnutt 5.7.

```
{
   "_id": "9029b5f0788c0693bf5ffadcd72f827a",
   "_rev": "1-07c684eddcaec7858bdb85344c677805",
   "subject": "This is a test subject",
   "content": "This is a test content",
   "author": "dsol",
   "classification": "Releasable to all employees",
   "type": "journal",
   "happening_time": 1398254285000,
   "added_timestamp_local": 1398254287357,
   "keywords": "test",
   "uuid": "aae73d6f73a1836c57d7b251b628360e",
   "order": "01"
}
```

Kodesnutt 5.7: JSON-eksempel

_id og *_rev* er to standardattributter som er inkludert i hvert objekt. Attributtet *_id* er en global unik referanse til objektet, mens *_rev* er revisjonen. Denne endrer seg hver gang objektet blir forandret, derfor begynner revisjonsverdien på 1 i dette eksempelet.

For å hente ut alle journalinnlegg, bruker vi et view som vist i kodesnutt 5.8. Dette henter ned alle dokumenter av en gitt type, der denne blir valgt med et parameter i URL-en. For å hente ned alle kommentarer ser spørringen slik ut:

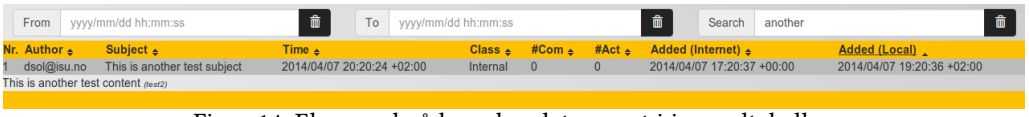
```
http://localhost:5984/journal/_changes?filter=app/type&type=journal
```

```
{
   "type": "function(doc, req){if(doc.type == req.query.type) {return true;}
       return false;}",
   "all": "function(doc){if(doc._deleted != true && (doc.type == 'comment' ||
       doc.type == 'action' || doc.type == 'journal')) {return true;} return
       false;}"
}
```

Kodesnutt 5.8: Kodesnitt - View

Selv om et view er den mest effektive måten å hente ut data på, er det også mulig laste all data fra databasen til nettleseren, for så deretter å filtrere ut ønsket innhold.

Data fra kodesnutt 5.7 vil bli presentert for brukeren som vist i figur 14.

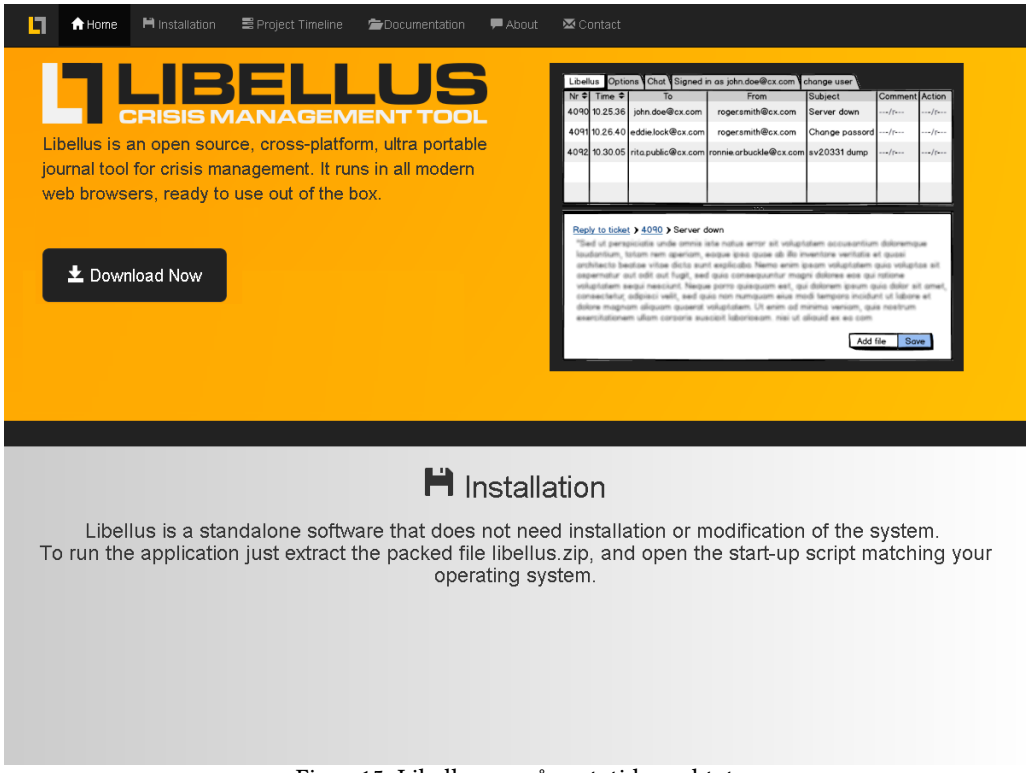


Figur 14: Eksempel på hvordan data ser ut i journaltabellen

5.9 Grafisk brukergrensesnitt

På det første offisielle møtet med oppdragsgiver i Oslo viste vi frem et utkast til brukergrensesnittet. Dette var to foreløpige illustrasjoner av hvordan vi hadde sett for oss programmet, og var laget for å få noen initielle tilbakemeldinger. Illustrasjonene hadde vi publisert på prosjektets nettside libellus.no, vist i figur 15. Deltakerne på møtet virket veldig fornøyd, og lurte på om de kunne få de samme fargene i applikasjonen også. Vi forsto da at vi hadde overkommunisert litt, i og med at bildene var i svarthvitt. Da vi senere satte i gang med utviklingen og hadde bestemt at applikasjonen skulle være vebbasert, gjenbrukte vi deler av designet fra hjemmesiden vår. Figur 16 og 17 viser det endelige designresultat for applikasjonen Libellus. Vi fremviser her

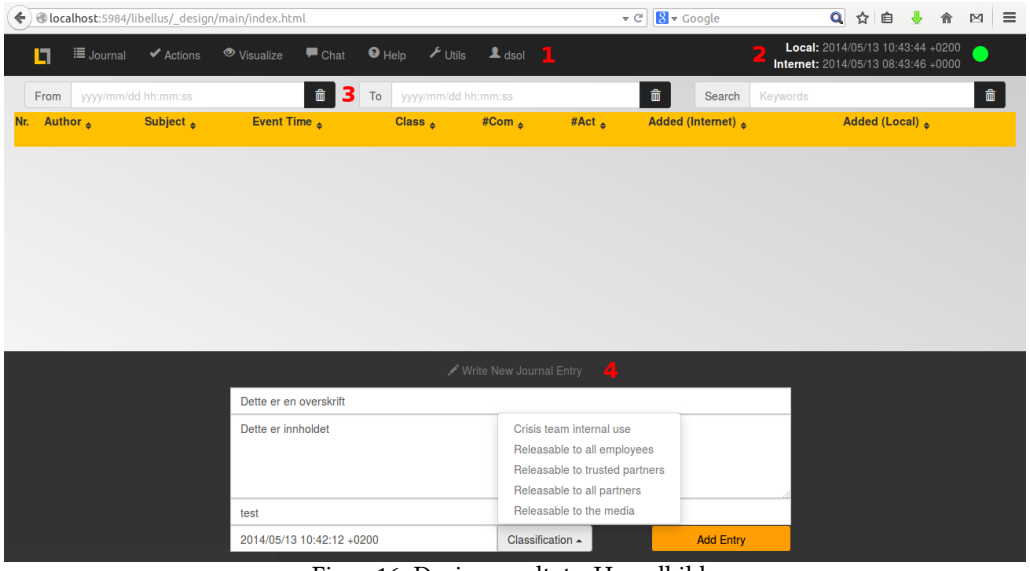
hovedbildet i applikasjonen og omtaler design og funksjonalitet. Flere illustrasjoner av brukergrensesnittet ligger i brukerdokumentasjonen, vedlegg H.



Figur 15: Libellus.no på møtetidspunktet

I oppbygningen av brukergrensesnittet har vi hentet inn igjen de tankene vi gjorde oss om brukervennlighet. Dette er særlig synlig i hvordan funksjonene brukeren skal benytte er plassert etter hverandre. Vi mener også at vi har oppnådd målet med å lage en moderne applikasjon, som er støttet opp med at operasjoner i Libellus vil skje uten at nettleseren må laste siden på nytt, noe som får applikasjonen til å føles mer interaktiv.

Når det gjelder fargevalg har vi benyttet en mørk gul kulør for å skape oppmerksomhet rundt viktige elementer. Gult lys er også stimulerende for hjernen og nervesystemet, fremmer årvåkenhet og aktiverer nervene i musklene. Dette i motsetning til for eksempel blåtoner som symboliserer rolighet og harmoni, eller rød som minner om konstant fare [57]. I tillegg til dette har vi benyttet ulike gråtoner, lyse som mørke for å få frem de andre elementene i applikasjonen.



Figur 16: Designresultat - Hovedbilde

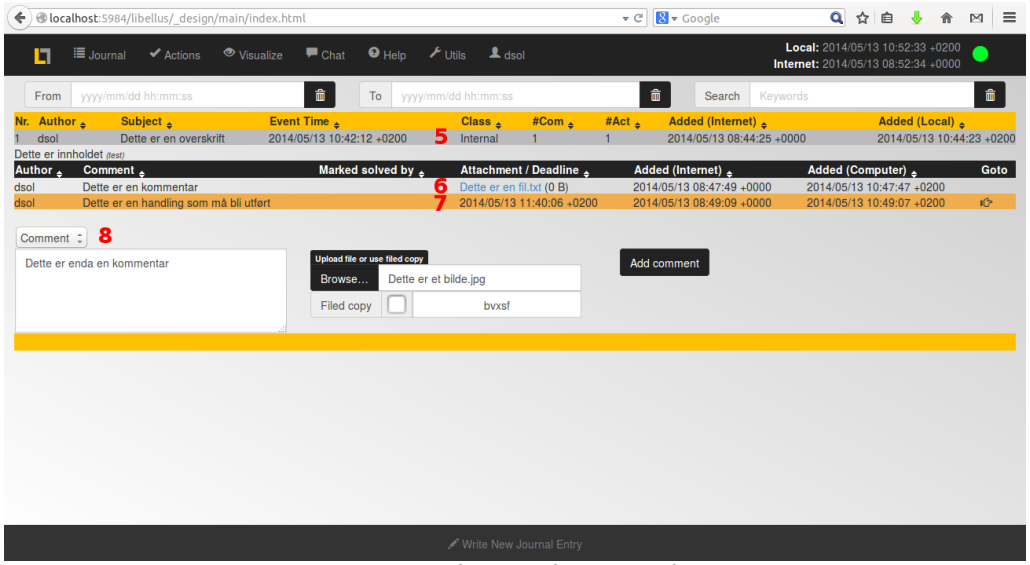
1. Her vises menylinjen, denne brukes for å bytte mellom ulik modulfunksjonalitet i programmet. Ved lav vindusoppløsning vil teksten fjernes slik at ikonene blir mer fremtredende. Dersom oppløsningen er enda lavere, som på smarttelefoner, vil hele menyen gjøres om til en egen menyknapp som innehar alle menyvalgene. Menyen er naturlig plassert øverst for å likne andre applikasjoner og nettsider. Ikonene vi har brukt er valgt med tanke på gjennkjennbarhet, eksempelvis brukes en skiftenøkkel for innstillinger og en snakkeboble for lynmeldinger.

2. Applikasjonen har to klokker, hvor den ene viser datamaskintid med lokal tidssone. Den andre viser ekstern tid hentet fra en tjener på Internett, presentert i UTC. Ved siden av disse er det en indikasjonssirkel. Denne er farget grønn om lokaltiden er likt synkronisert som den eksterne tjeneren. Dersom tidsforskjellen er over tre sekunder, vil fargen endre seg til gul og etter seks sekunder blir den oransje. Ved bortfall av eksterntid vil sirkelen være rød. Dersom brukeren holder musepekeren over sirkelen vil det dukke opp en liten boks som viser hvor mye tidsforskjell det er mellom klokkene.

3. Her vises filtreringsmulighetene, som gjør det mulig å hente ut journalinnlegg, kommentarer og oppfølgingsmerkander basert på innhold og/eller tid. Disse blir ved valg satt til nåværende tid for å effektivisere tidsfiltreringen. Denne tiden kan justeres ved bruk av tastatur eller mus. I tekstfeltet kan brukeren skrive inn forfatter, filnavn, nøkkelord eller setninger som finnes i et journalinnlegg. Filtreringen skjer i sanntid og resultatet endres etter hvert som brukeren skriver. For å fjerne innholdet i tekstboksene benyttes knappene med søppeldunk-ikon. Filtreringsboksene er plassert under menyen for å få en naturlig flyt fra meny, til utvelgese og deretter resultat.

4. Dette er innfyllingsboksen for å legge til et nytt journalinnlegg. Selve tekstboksen kan gjemmes eller hentes frem ved å trykke på *Write New Journal Entry*. Alle feltene er utstyrt med en standardtekst (ikke vist på bildet), som forteller brukeren hvilken informasjon som skal skrives inn. Disse feltene er plassert i en logisk rekkefølge etter hverandre. Feltet for tid vil som standard bli satt til den aktuelle for når innlegget ble påbegynt. Det vil også dukke opp en boks der brukeren kan velge tiden ved å trykke med musepekeren

istedenfor å skrive inn tidsstrengen manuelt. Til sist benyttes *Add Entry* for å sende informasjonen til databasen. Selve innfyllingsboksen er plassert nederst fordi dette gjør at brukeren kan opprette nye og lese på eldre journalinnlegg samtidig.



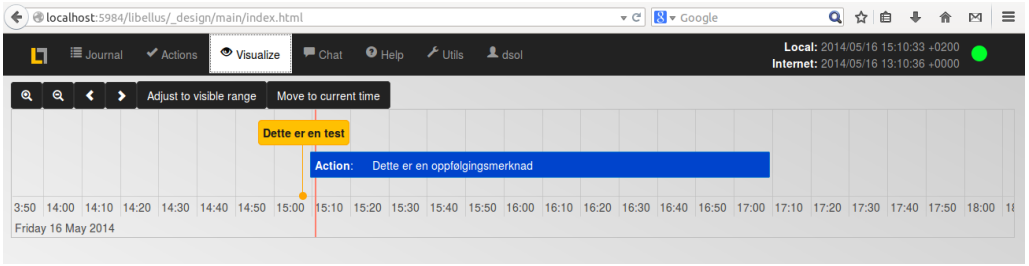
Figur 17: Designresultat - Innlegging av kommentar

5. Her er journaltabellen som standard viser alle journalinnleggene på hver rad nedover. I dette bildet er det ene innlegget ekspandert, og alle kommentarer og oppfølgingsmerknader knyttet til dette innlegget vises. Dersom brukeren trykker et sted på linjen markert med femtallet vil innlegget minimeres og tabellen viser kun denne linjen. For å få tilbake all informasjonen igjen, kan brukeren trykke på linjen på nytt.

6. Dette er en kommentar med et vedlegg knyttet til journalinnlegget. For å laste ned vedlegget kan brukeren trykke på filnavnet *Dette er en fil.txt*. Vedlegget kan også være en *filed copy* som er en referanse til et fysisk objekt representert med en tilfeldig generert tekststreng. Et eksempel på bruk av dette er dersom organisasjonen har en stor fil på en ekstern harddisk eller en perm med dokumenter de ikke får importert inn i Libellus.

7. Her vises en oppfølgingsmerknad (action) som gir informasjon om noe som må bli gjort i henhold til journalinnlegget. Istedenfor et vedlegg vises tidsfristen for når denne handlingen må være utført. Fargen på linjen er oransje fordi det er under én time til tidsfristen og når tidsfristen er overskredet, vil den bytte farge til rød. Dersom det er over én time til tidsfristen er linjen blå og når oppfølgingen er løst vil den være grønn. Disse fargeendringene vil skje i sanntid.

8. Her er et skjema for å legge til nye kommentarer og oppfølginger til journalinnlegget. Først velger brukeren comment eller action fra menyboksen, og deretter fyller inn skjemaet. Brukeren beveger så musepekeren til høyre og laster eventuelt opp et vedlegg eller benytter *filed copy*. Brukeren trykker på *Add comment* for å sende innlegget til databasen.



Figur 18: Designresultat - Tidslinje

Tidslinjen som vist i figur 18, er bygd opp med en menylinje øverst og hoveddelen nederst. I menyen finnes knapper for å zoome inn og ut, bevege tidslinjen forover og bakover i tid, samt å midtstille tidslinjen etter nåværende tid eller få alle objekter innenfor rammen. Det er også mulig å bruke musepekeren, tastaturet og rullehjulet til å flytte på og zoome tidslinjen.

Journallinnlegg og oppfølgingsmerknader dukker opp i tidslinjen som tykke linjer. Journalinnlegget vil alltid ha en fast lengde, mens oppfølgingen vil strekke seg til tidsfristen den er satt til. Fargene samsvarer med fargetemaet i journalmodulen.

5.10 Implementasjon av tjenesteoppdagelse

I en klassisk klient/tjener-situasjon vil klientene koble seg opp til en kjent tjener, deretter kan data distribueres gjennom tjeneren, eller direkte fra klient til klient så lenge kommunikasjonspunktene, eller IP-adressene, er kunngjort for begge. I vårt tilfelle trengs derimot en desentralisert tjenesteoppdagelse. Dette er typisk for f.eks. *peer-to-peer*-systemer eller jevnbyrdsnett, der omgivelsene er dynamiske.

En samlebetegnelse for denne teknologien er Zero-configuration networking (zeroconf) [58]. Denne omhandler adressevalg, navnoppslag og tjenesteoppdagelse. De mest kjente implementasjonene er Avahi [59]og Apple Bonjour [60]. Vi undersøkte disse to og en løsning programmert i Erlang [61], Kriteriene her var krysskompatibilitet og godt dokumentert kildekode. Erlangløsningen ble vi nødt til å se bort i fra, fordi vi ikke klarte å få data fra konsoll og inn i ønsket replikatordatabase.

For effektivt å kunne teste replikeringsfunksjonalitet laget vi først kode i Bash og Powershell, som benyttet Nmap [62] for å søke opp andre instanser av Libellus og cURL for å legge inn data.

Disse skriptene hadde følgende funksjonalitet:

1. Finner alle lokale IPv4-adresser med subnett.
2. Skanne gjennom alle nettene som ble funnet, etter maskiner med ønsket port åpen.
3. Opprette en socket opp mot klienten med cURL, og sjekke om det er en Libellus-instans i den andre enden.
4. Dersom andre Libellus instanser blir funnet, blir disse IP-adressene lagt inn i den lokale replikatordatabasen.
5. Nettleseren med Libellus blir så startet.

Å skanne nettverk med Nmap er en ganske langsom prosess, selv med tidsavbrudd satt så lavt som mulig. Vi forsøkte derfor med verktøyet Masscan, som skal kunne skanne hele Internett på under seks minutter [63]. Dette ga merkbare resultater kontra å bruke Nmap, men

cURL brukte fremdeles for lang tid. Etter en systematisk gjennomgang av skript på GitHub fant vi et UDP Multicast eksempelskript [64], skrevet av Aaron Cohen. Dette var laget i Python, godt dokumentert og lisensiert under CC BY 3.0 [65].

Dette Python-skriptet gjør i korte trekk:

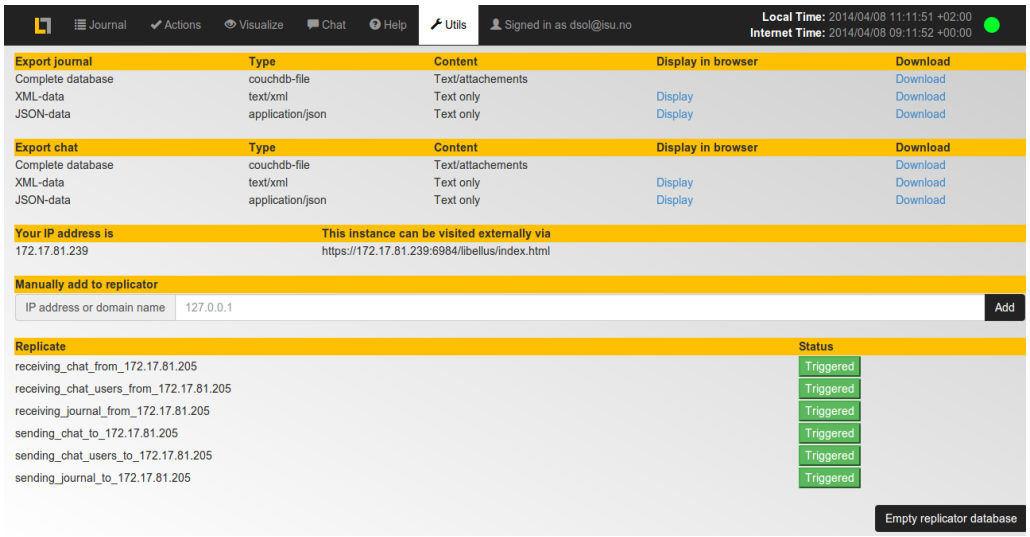
1. finner maskinens første lokale IPv4-adresse
2. lager en ny UDP-socket som abonnerer på en bestemt multicast-gruppe
3. knytter en socket til det nettverkskortet som er i bruk
4. sender ut annonseringsmeldinger
5. mottar annonseringsmeldinger

Dette har vi utvidet slik at koden nå i tillegg har denne funksjonaliteten:

- henter ut alle lokale IPv4-adresser, og starter en lytte og annonseringsinstans for hver av dem
- dersom andre enheter på nettverket oppdages, blir deres informasjon lagt inn som JSON-data og sendt til lokal replikatordatabase

Koden er også gjort portabel til Apple OS X og GNU/Linux, samt at det er lagt inn sjekk for oppsøking av IP-adresser og kjørende CouchDB-instanser. Skriptet avsluttes automatisk dersom det ikke får forbindelse med databasen. Hele det modifiserte skriptet ligger i vedlegg F.

CouchDB har en egen konfigurasjonsinnstilling for oppstart av bakgrunnsprosesser. Denne påkaller Python-skriptet med jevne mellomrom. Hele oppdagelsesprosessen er skjult for brukeren, men inne i Libellus listes det opp instanser det replikeres til og fra som vist i figur 19. Der er det også mulig å fjerne synkronisering.



Figur 19: Skjermskudd fra applikasjonen som viser verktøymodulen med aktiv replikering.

5.11 IPv4 og IPv6

CouchDB støtter simultan bruk av både IPv4 og IPv6 [31]. Tjenesteoppdagelsesskriptet kan også omskrives til å benytte begge, dette er imidlertid ikke noe vi har valgt å fokusere på for versjon 1.0 av Libellus, da vi forutser at det kan oppstå noe problematikk med dobbeltkommunikasjon. Dette innbefatter replikering av databaser til tjenere kjørende med begge protokollene. IPv4 blir heller ikke faset ut av lokalnettverk med det første.

5.12 Portabilitet

I det første punktet hentet fra *BSI 100-4* [11] er det beskrevet at flerplattformstøtte er viktig. Applikasjoner som ikke må installeres på operativsystemet, men som kan kjøres direkte fra et lagringsmedium omtales i denne rapporten som portabel. Programvare som er kryssplattform kan kjøres under ulike operativsystemer. CouchDB er blant annet utviklet for de nyeste utgavene av Microsoft Windows, Apple OS X og Ubuntu Linux, noe som gjør programvaren kryssplattformkompatibel.

En mulighet for å utvikle en portabel løsning av Libellus kunne ha vært å installere programmet i en Ubuntu-installasjon, og generere en live-utgave av denne som kan kjøres rett fra en minnepinne. Dette gikk vi bort i fra allerede før de operasjonelle kravene ble diskutert, grunnet problematikk rundt vedlikeholdsmuligheter og effektiv igangsettelse. Det å vedlikeholde et helt operativsystem med programmet innlagt blir en mye større jobb enn å kun vedlikeholde programmet. For å kunne starte en live versjon av et operativsystem må minnepinnen være prioritert i oppstartsrekkefølgen. Dette krever mer teknisk kompetanse av brukeren enn vi kan forvente. Det er heller ikke garantert at all maskinvare er støttet av operativsystemet, eksempelvis kan drivere for nettverkskort mangle.

I utgangspunktet ønsket vi å lage en applikasjon som skulle være portabel til Microsoft Windows 7, 8, Apple OS X Mavericks og Ubuntu Linux 12.04 LTS. CouchDB er støttet på disse operativsystemene, men installasjonsmåte og oppstart er ulik, noe som fører til at prosessen for å gjøre programvaren portabel blir nokså forskjellig. På GNU/Linux-distribusjoner må CouchDB kjøre under den lokal brukerkontoen *couchdb* som kun kan opprettes med administratortilgang. I Windows må CouchDB startes med et Batch-skript og under OS X er hele programmet pakket i en *.app*-fil.

Det å finne et filsystem som er portabelt til de tre ulike operativsystemene er også en utfordring. FAT32 er riktignok støttet av alle, men takler ikke filer over 4 GB. Databasefilene har potensiale til å bli større enn dette, og det finnes ingen mulighet i CouchDB for å dele opp disse. Med FAT32 er det heller ikke mulig å gjøre filer eksekverbare for GNU/Linux-systemer. Vi kunne ha partisjonert minnepinnen til ulike OS, men vi ønsket å samle alle databasefilene på en partisjon som var lesbar fra alle, for enkelt å kunne kopiere ut filene uten å være avhengig av OS-et benyttet under loggføringen.

Med slike dilemma ble vi nødt til å konferere med oppdragsgiver om hvilket operativsystem vi skulle ha mest fokus på. Det ble da avgjort at operativsystemer med kortest mulig iverksettelsestid ved krise, altså systemer som kan kjøpes i butikk er viktigst. I dag benytter bedriften mange maskiner med Windows som plattform, så full støtte for dette operativsystemet var helt nødvendig. De ønsket også at vi skulle se på muligheten for å gjøre programmet portabelt til Apple OS X, da dette kan fungere som et avlastningssystem dersom bedriften har blitt rammet av skadevare som spesifikt rammer Windows-plattformen.

Det var en vanskelig avgjørelse å skulle gå bort fra Ubuntu i et prosjekt med fokus på åpen kildekode, men her veide praktiske hensyn tyngst. Riktignok kan Libellus fremdeles legges inn

på alle OS som CouchDB støtter, men dette krever noe manuell jobb. Testtjeneren app.libellus.no kjører blant annet på Ubuntu 14.04 LTS.

Som en anbefaling har vi i vedlikeholdsdokumentasjonen beskrevet at Libellus bør legges inn på en minnepinne formatert til det propertiære filsystemet exFAT [66]. Årsaken til dette er at exFAT er det eneste filsystemet som er støttet på de nyeste utgavene av både Windows og OS X, samtidig som begrensingen på filstørrelse er 16 Exbibyte [66]. Hvordan klargjøre Libellus for en minnepinne er også beskrevet i vedlikeholdsdokumentasjonen, vedlegg J.

Libellus er pr. 22.04 portabel til Microsoft Windows 7 og 8. Vi har også lagt ned tid på å få programvaren portabel til Apple OS X Mavericks, men her gjenstår det noe arbeid før vi kan publisere en endelig utgave. Vi kjenner prosessen og vet hvilke endringer vi må utføre, men må sette oss mer inn i applikasjonsutvikling på OS X og bruk av Xcode[67]

I og med at CouchDB er en vebbtjener vil alle enheter som har en nettleser og er tilsluttet samme lokalnettverk, uavhengig av operativsystem, kunne koblet seg opp til Libellus og få tilgang til det samme brukergrensesnittet som en lokal bruker. Dette er også i tråd med *BSI 100-4* [11] og plattformuavhengighet. Merk at all data vil lagres i databasen og eventuelle oppkoblede enheter som andre datamaskiner, smarttelefoner eller nettbrett på denne måten kun vil fungere som tynnklienter.

5.13 Bruk av biblioteker

Vi har brukt en rekke bibliotek for å slippe å finne opp hjulet på nytt, samt for å minimere kodebasen vi selv må vedlikeholde. Ved valg av bibliotek har vi sett etter kode som har en fri lisens, da helst Apache eller MIT [68], fordi dette gir prosjektet friere tøyler. Eksempelvis blir det mulig å kommersialisere deler av programvaren, eller anvendelsen av den, ved å tilby spesialtilpassede moduler eller kurs i Libellus/hendelseshåndtering. Det at koden er kontinuerlig vedlikeholdt er også viktig, da det å bytte bibliotek senere på grunn av feil eller sikkerhetshull som ikke blir rettet kan være en komplisert prosess.

Vi har brukt jQuery [69] og Bootstrap [70] som generelle JavaScript- og CSS-bibliotek for å gjøre utviklingen enklere og mer triviell. Disse er *de facto* standard når det kommer til vebbutvikling og er vidt utbredt, i tillegg til å være veldokumentert og godt vedlikeholdt.

For å sortere data lagde vi først en egen funksjon som gjorde denne jobben på tjenersiden, men dette viste seg å være tungvint da vi måtte hente ned all data fra databasen på nytt hver gang vi ville sortere på noe annet. Derfor var vi på utkikk etter et bibliotek som kunne gjøre denne jobben for oss på klientsiden. Vi falt for Tablesorter.js [71], da den var veldig enkel å ta i bruk, samt godt dokumentert og stadig under utvikling.

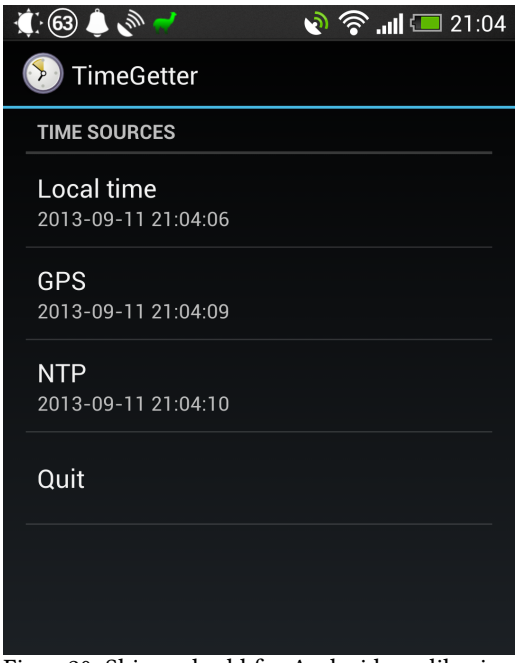
Vi har brukt Almendes Timeline.js [72] på den visuelle fremstillingen av journalen. Vi valgte denne etter å ha prøvd andre alternativer som Knighlabs timeline.js [73], chronoline [74] og SIMILE Widgets Timeline [75], fordi den var best vedlikeholdt og godt dokumentert.

For å sende filer med AJAX opp til CouchDB bruker vi jQuery.form.js [76]. Dette kom vi over etter å ha sett et kodeeksempel på GitHub [77].

Tilbakemeldinger til brukeren ble først gitt med *alert()*-kommandoen i JavaScript, men dette har den ulempen at alle operasjonene i nettleseren blir pauset når dialogboksen vises. Noe som fører til at alle bakgrunnsoppgaver stanser, og klokka vil komme ut av synk. Boksene er heller ikke så visuelt pene å se på. Her inkluderte vi derfor Bootstrap-dialog.js [78], som tilbyr dialogbokser i Bootstrap med lite ekstra kode.

5.14 Tidsfunksjonalitet

Første gang vi hørte om oppgaven ble vi veldig engasjert i å kunne sikre påliteligheten til tiden ved å hente inn ulike kilder for tid. Tanken var å benytte en mobiltelefon som klokke, og at denne igjen hentet inn klokkeslett fra GPS, tidstjenere og mobilnettverket. Telefonen skulle være låst med kode for å sikre mot manipulasjon. Vi laget derfor et eksempelprogram for Android med denne funksjonaliteten som vist i figur 20. Dette fungerte som forventet, men vi innså at dette var litt på siden av oppgaven og ville kreve mer testing og vedlikehold. Oppdragsgiver sa også det ikke var kritisk at tiden stemte hundre prosent, så lenge loggeprogrammet leverte sterkere bevis enn en eventuell motpart. Vi valgte derfor ikke å fokusere noe mer på dette.



Figur 20: Skjermskudd fra Android-applikasjon

Alle innlegg som lagres i databasen blir tilføyet to Unix-tidsstempeler, som er antall sekunder siden klokka 00:00:00 UTC den 01.01.1970. Det ene tidsstempelet kommer fra den lokale klokka på datamaskinen Libellus blir kjørt fra, mens det andre blir hentet fra en tjener på Internett. Dersom Libellus ikke har tilgang til Internett eller tidstjeneren er utilgjengelig vil ikke attributtet for eksterntid lagt til. Hvis internettilgangen blir tilgjengeliggjort senere vil disse innleggene få innlagt tid i etterkant, denne blir i så tilfelle utregnet med forhold til den lokale tiden.

Vi har lagt inn mer redundans når det gjelder å hente ekstern tid fra tjeneren på Internett, slik at om den første tidstjeneren er nede, vil Libellus prøve en annen. Vi har også lagt ved et PHP-skript som vist i kodesnutt 5.9, slik at de som benytter Libellus kan laste dette opp på en egen tjener for å erstatte de nåværende kildene eller å oppnå enda flere reservekilder for eksterntid.

Det å bruke en ekstern tjener på Internett er en infrastrukturtilknyttning, men denne er ikke nødvendig for at programvaren skal fungere. Årsaken til at vi har valgt denne løsningen, er for å få en ekstern tidskilde det kan stoles mer på enn den lokale klokka. Mer om sikkerhetsaspektet rundt tid blir omtalt videre i avsnitt 5.15.2.

```php
<?php

// Requires the php5-json package. Debian/Ubuntu: sudo apt-get install php5-json

// We dont want to send errors.
ini_set('display_errors', 0);

// Tell the browser that we are sending JSON
header('Content-Type: application/json');

// Returns time in milliseconds
function millitime() {
    $microtime = microtime();
    $comps = explode(' ', $microtime);
    return sprintf('%d%03d', $comps[1], $comps[0] * 1000);
}

// Returns a full datatime string
function fulldatetime() {
    return date('c');
}

$datetime = array();

$datetime['unixTimestamp'] = millitime();
$datetime['dateString'] = fulldatetime();

// Pack the JSON inside a callback function to comply with JSONP.
if (isset($_GET['callback'])) {
    echo $_GET['callback'] . "(";
}

echo json_encode($datetime);

if (isset($_GET['callback'])) {
    echo ")";
}

?>
```

Kodesnutt 5.9: libellus_gettime.php

Libellus bruker HTTPS som kommunikasjonsprotokoll, dermed må også tjeneren PHP- skriptet kjøres fra bruke den samme protokollen. Om Libellus prøver å kommunisere med en tjener som kjører vanlig HTTP, vil nettleseren nekte tilknytning, komme med en feilmelding og ikke oppnå eksterntid, som vist i kodesnutt 5.10. Det at kommunikasjonen er kryptert fører også til at det blir vanskeligere for en angriper å interfere og endre på oversendt tidsinformasjon.

```
[blocked] The page at 'https://ojeey.pw:6984/libellus/_design/main/index.html'
    was loaded over HTTPS, but ran insecure content from 'http://dsolstad.com/
    gettime.php?callback=jQuery21005171040231361985_1400068017621&_
    =1400068017625': this content should also be loaded over HTTPS.
```

Kodesnutt 5.10: HTTPS-feilmelding

Initielt ønsket vi å implementere at all tid som ble brukt og fremvist i systemet skulle være i zulu-tid, altså UTC eller GMT+0. Vår tanke bak dette var at det ville være enklere å forholde seg til en lik tid dersom programvaren ble brukt samtidig på ulike steder i verden. Alle de lokale krisestabene kunne da effektivt kommunisere med hverandre uten å måtte tenke på omregning. Etter et møte med oppdragsgiver, kom det frem at de heller ønsket at innleggene og klokka i høyre hjørne skulle bli vist i den lokale tidssonen. Årsaken til dette er at det skulle bli lettere å få med seg hverdagslige oppgaver utenfor krisen, som å hente barn i barnehagen osv.

Før denne endringen var all tidsfunksjonalitet i programmet bygget opp av egenlagde funk-

sjoner. For å få til omregning mellom ulike tidssoner, samt sjekk om det er sommertid eller vintertid, inkluderte vi biblioteket Moment.js [79]. Dette førte med seg noen andre nyttige funksjoner, blant annet muligheter for validering av tid på de stedene brukeren har mulighet til å legge inn denne.

5.15 Sikkerhetsaspekter og funksjonalitet

Libellus kjøres lokalt og brukeren vil dermed ha tilgang til å endre på programvaren og få utført uønskede handlinger. Det enkleste eksemplet på dette er at databasefilene kan slettes fra lagringsmediumet. Imidlertid er hensikten med verktøyet å støtte opp om løsningsprosessen av en krise ved å være lett tilgjengelig og enkel å benytte. Dersom en bruker bevisst går inn for å sabotere programvaren, og med det krisehåndteringsarbeidet, må dette håndteres på organisasjonsnivå. Sikkerhetsfunksjonaliteten i programvaren er i hovedsak fokusert rundt sporbarheten til endringer. Det femte punktet i *BSI 100-4* [11] sier også at data innlagt i systemet må kunne sikres. I dette avsnittet beskriver vi sikkerhetsfunksjoner og diskuterer sikkerhetsaspekter.

5.15.1 Konfidensialitet

Konfidensialitet i et system hvor tilgjengelighet har høy prioritet er en komplisert affære, og disse to prinsippene må da nødvendigvis settes opp mot hverandre. Vi kommer senere tilbake til dette under tilgjengelighet 5.15.3. Et av tiltakene som innbefatter konfidensialitet er sikring av data under overføring fra et system til et annet, noe som i Libellus er løst med TLS 1.2. Implementasjon av TLS er i utgangspunktet ikke en innviklet prosess på CouchDB, men vi hadde allikevel en utfordring. Etter å ha generert selvsignerte sertifikater og fulgt konfigurasjonsveiledningen [80] fikk vi en fullstendig programkrasj av Erlang-konsollen dersom vebbtjeneren ble besøkt med HTTPS.

```
Erlang R16B02 (erts-5.10.3) [source] [smp:4:4] [async-threads:4]

Eshell V5.10.3  (abort with ^G)
1> Apache CouchDB 1.5.1 (LogLevel=info) is starting.
Apache CouchDB has started. Time to relax.
[info] [<0.36.0>] Apache CouchDB has started on https://127.0.0.1:6984/
[error] [<0.367.0>] SSL: certify: tls_connection.erl:2286:Fatal error: unknown
    ca
```

Kodesnutt 5.11: Erlang R16B02 med feil

Dette skjedde i nettleseren Firefox v26 som selv ga feilmeldingen

```
Error code: sec\_error\_invalid\_key
```

Google Chrome v31 svarte med

```
ERR\_SSL\_CLIENT\_AUTH\_SIGNATURE\_FAILED.
```

Imidlertid virket alt som det skulle i Internet Explorer 11.

Feilmeldingene ga oss ikke mye å jobbe med, men i og med at sertifikatene var egengenererte tenkte vi at det kunne være noe galt med dem. Vi verifisere de derfor med OpenSSL-verktøyet [81] uten å finne noen feil. Etter å ha forespurt på IRC-kanalen til CouchDB-prosjektet og lest igjennom lignende rapporterte feil i JIRA [82], fikk vi en mistanke om at feilen kunne være relatert til Erlang-implementasjonen. Versjonen vi hadde forsøkt på dette tidspunktet var CouchDB 1.5.0 på Erlang/OTP R16B02. Dette var den offisielle utgaven som var tilgjengelig på couchdb.org. Etter å ha lest igjennom versjonshistorikk for Erlang/OTP, så vi at det hadde kommet en feilfiks R16B03-1 som rettet tre feil relatert til TLS [83]. Vi sydde deretter sammen en løsning med CouchDB 1.5.0 på Erlang/OTP R16B03-1. Dette fungerte, og vi kunne da påvise at

R16B02 var rammet av feilen. Denne egeninstallasjonen ville imidlertid bli vanskelig å vedlikeholde, noe som førte til at vi byttet til en stabil utgave av CouchDB og Erlang/OTP, hvor feilen ikke var introdusert. Dette var Erlang/OTP R14B04.

```
Erlang R14B04 (erts-5.8.5) [source] [smp:4:4] [rq:4] [async-threads:4]

Eshell V5.8.5  (abort with ^G)
1> Apache CouchDB 1.5.1 (LogLevel=info) is starting.
Apache CouchDB has started. Time to relax.
[info] [<0.36.0>] Apache CouchDB has started on https://127.0.0.1:6984/
[info] [<0.157.0>] 127.0.0.1 - - GET / 200
[info] [<0.157.0>] 127.0.0.1 - - GET /favicon.ico 200
```

Kodesnutt 5.12: Erlang R14B04 uten feil

I Libellus benytter vi selvsignerte sertifikater, som vil si at de er utstedt av oss selv og ikke en autoritet. Hovedårsaken til dette er at vi er nødt til å distribuere den private nøkkelen sammen med programvaren. Dette vil igjen si at alle som bruker Libellus må generere egne sertifikater for å sikre at konfidensialiteten opprettholdes. Dette er beskrevet i vedlikeholdsdokumentasjonenm, vedlegg J.

Under panseret til Libellus ligger det to systemkontoer, dette er kontoer brukeren ikke kan se, eller skal ha noen befatning med, men som er med på å sikre systemet. Fra standard av finnes det ingen brukere i CouchDB, dette kalles *Admin Party* [31]. Alle som har tilgang til databasen vil da være administratorbrukere og har mulighet til å opprette, redigere og slette data fra databasen. For å hindre dette har vi opprettet brukeren *libellusadmin*. Denne brukeren er inaktiv, men sikrer at det ikke kan opprettes nye administratorbrukere. I tillegg til dette finnes det en vanlig bruker som vi har kalt *libellususer*. Denne brukerkontoen blir automatisk innlogget når en instans av Libellus startes, og har kun lese og skrivetilgang til databasen. Det betyr at brukeren ikke kan slette innlagt data. Det er heller ikke mulig å overskrive informasjon, da CouchDB lagrer alle endringer med revisjoner.

5.15.2 Integritet

CouchDB bruker MVCC [84] for å unngå behovet for å låse databasefilen under skriving. I tillegg er dokumenter versjonert, omtrent som i et versjonskontrollsystem som Git. Endres det på en verdi i et dokument, opprettes det en ny versjon av dokumentet, endringen utføres og det nye dokumentet lagres. Databasen vil da inneholde to dokumenter, ett gammelt og ett nytt, dette er med på å sikre at all data i databasen er fullstendig.

Siden Libellus er skrevet i JavaScript og brukergrensesnittet nås gjennom en nettleser, må det gjøres tiltak mot XSS. Som følge av god programmeringspraksis på området gjøres dette når data er på vei ut av databasen [85]. Kort forklart er grunnen til dette at vi ikke nødvendigvis kjenner til alle måtene data har kommet inn i databasen på, men vi vet hvor den skal fremvises i applikasjonen.

```
<script>alert('xss');</script>  -->  &lt;script&gt;alert('xss');&lt;/script&gt;
```

Kodesnutt 5.13: HTML blir uskadeliggjort

I Libellus er det to klokker øverst i høyre hjørne. Den ene viser lokal tid fra klienten som har åpnet applikasjonen, mens den andre viser eksterntid dersom denne er tilgjengelig. På klientsiden vil brukeren alltid ha mulighet til å stille sin lokale klokke. Vi har derfor innført noen tiltak som loggfører alt som har med tid å gjøre, som vist i kodesnutt 5.14. Dersom en bruker endrer klokka på sin lokale klient, blir dette ført i databasen. Om en klient ikke har internettforbindelse, lagres innlegg kun med lokaltiden, og hvis klienten får forbindelse med en tidstjener

på Internett, vil programmet regne ut tidsforskjellen mellom tiden fra denne tiden og den lokale klokka, for så å legge inn eksterntid på alle innleggene som mangler dette. I tillegg lagres alle innlegg med en teller, slik at det er mulig å fremlegge rekkefølge innleggene ble lagt inn i dersom tidspunktene ikke samsvarer.

```
// If the local time changes, we put a note in the database about it
function logTimeChange() {
    // If we know the last timestamp
    if (glob_prev_local_timestamp) {
        // If the current timestamp differs by over 3000 ms since the last
            timestamp
        var diff = glob_prev_local_timestamp - glob_local_timestamp;
        if (Math.abs(diff) > 3000) {
            var obj_info = {};
            obj_info['prev_local_timestamp'] = glob_prev_local_timestamp;
            obj_info['new_local_timestamp'] = glob_local_timestamp;
            obj_info['diff'] = diff;
            obj_info['current_external_timestamp'] = glob_external_timestamp;
            obj_info['username'] = getCookie('username');
            obj_info['uuid'] = glob_uuid;

            addDoc('timewatch', obj_info, function(doc) {
                console.log("Added time change:");
                console.log(doc);
            });
        }
    }
    glob_prev_local_timestamp = glob_local_timestamp;
}
```

Kodesnutt 5.14: logTimeChange()

Kodesnutt 5.15 viser en funksjonen som blir kjørt hvert femte minutt, der den regner ut tidsforskjellen og legger inn eksterntid på de innlegg som ikke har det fra før.

```
// Updates all entries that does not have external timestamp and sets it
    according to the local/external offset.
function setExternalTimestamps() {
    // Iterates through all journal and appendix docs
    $('#table_journal > tbody > tr[name=journal_entry_metadata], table[name=
        table_appendix] > tbody > tr').each(function(k, v) {
       // Opens the current doc
       $.couch.db('journal').openDoc($(this).data('id'), {
          success: function(doc) {
              // Only change docs that doesn't have external time
              if (doc.added_timestamp_external == null) {
                  doc['_id'] = doc._id;
                  doc['_rev'] = doc._rev;
                  // Calculates the offset
                  var offset = getTimeOffset();
                  doc.added_timestamp_external = doc.added_timestamp_local +
                      offset;
                  updateDoc(doc, function(resp) {
                      console.log(resp);
                  });
              }
          }
       });
    });
}
```

Kodesnutt 5.15: setExternalTimestamps()

5.15.3 Tilgjengelighet

Det at verktøyet skal ha høy tilgjengelighet er et av kjernekravene, noe som også er også nevnt i punkt seks i *BSI 100-4* [11]. Programmet skal kunne ligge inaktivt på en minnepinne, for så å bli driftssatt når en krise inntreffer. Da er det viktig at programmet kan startes raskt, uten komplisert brukerinteraksjon eller avhengigheter til infrastruktur. Vi har valgt å ikke ha prekonfigurerte brukernavn og passord for å forenkle vedlikeholdsbehovet ved endring av personell i krisestaben. I tillegg vil det under en krise være rimelig å anta at det er vanskelig å huske et passord som ikke har vært i bruk på en stund. Det er viktig å kunne knytte hvert enkelt skrevet innlegg til en forfatter, da dette gjør det mulig å gå tilbake til loggføreren hvis noe er uklart. Dersom en ny loggfører ikke er blant de prekonfigurerte brukerne, vil kontodeling kunne bli et aktuelt dilemma.

Løsningen vår er å la brukeren selv velge et brukernavn under oppstart. For å sikre at brukernavnet er unikt anbefaler vi å benytte brukerens unike identifikasjon i organisasjonen, som eksempelvis den ansattes e-postadresse. Dersom en ny bruker overtar tastaturet, er det lagt til en knapp i menyen for å logge inn med en annen bruker.

5.15.4 Heartbleed

Den 07.04.14 kom nyheten om at det var funnet en feil OpenSSL. I løpet av tidligere feilsøking på TLS hadde vi fått kjennskap til at OpenSSL var i bruk i Erlang/OTP. Siden vi bruker TLS og hadde testtjeneren vår tilgjengelig via Internett, var vi spente på om hullet kunne utnyttes mot CouchDB.

Etter å ha konferert med brukerene IRC-kanalen, fikk vi bekreftet at Erlang/OPT benytter OpenSSL-biblioteker, men at heartbeat-modulen ikke er implementert [86]. I tillegg er all TLS håndtrykkslogikk skrevet i Erlang, som ikke er sårbar for minneavgrensningsfeil på samme måte som C.

5.16 Dokumentasjon

Vi har laget tre typer dokumentasjon til prosjektet: Brukermanual H, operasjonell dokumentasjon I og vedlikeholdsmanual J. Oppdragsgiver ønsket at brukerdokumentasjonen skulle være innbakt og tilgjengelig inne i selve programmet. Dersom en eventuell feil skulle føre til at Libellus ikke starter eller kan gjøres operativ, blir denne utilgjengelig. Vi måtte derfor ha en løsning både for dokumentasjon innenfor og utenfor programmet. Da det ikke følger med noen PDF-leser for Windows 7, vurderte vi HTML som det mest portable alternativet.

For å generere dokumentasjonen har vi benyttet Sphinx [53]. Dette er programvaren som brukes for å sette sammen blant annet Python og CouchDB sin dokumentasjon. Fordelen med dette verktøyet er at innhold struktureres med reStructuredText, deretter kan det genereres ulike typer filer, blant annet EPub, LaTeX, HTML og Man.

Kildedokumenter og konfigurasjonsfil er vedlagt prosjektets kildekode. Ut fra disse lages LaTeX-filer som omgjøres til PDF. Ren HTML brukes både inne i programmet og for de filene som ligger rett på rota av minnepinnen.

5.17 Unike identifikasjonsnummer

CouchDB bruker globalt unike strenger for å identifisere hvert dokument, dette hindrer duplisering under replikering. Samme teknologi (UUID) er også i bruk for unikt å kunne identifisere hver enkelt tjener som kjører programvaren. Slik CouchDB er ment brukt, vil disse bli generert under installasjon. I vårt tilfelle er bruksscenarioet at programvaren kopieres rundt på minnepinner fra en kilde, eller legges inn på hver enkelt maskin den skal kjøres fra. Dette vil føre til at alle har lik konfigurasjon og ID. Uten en unik identifikasjonsmetode for hver tjener vil ikke replikering fungere som ønsket. Derfor kjøres det et genereringsskript ved oppstart, vist i kodesnutt 5.16, som legger ny UUID-verdi inn i den lokale konfigurasjonsfila (*local.ini* på Windows). Ved utarbeidelse av denne funksjonaliteten fant vi noen uregelmessigheter i brukerdokumentasjonen til CouchDB. Disse ble meldt inn av oss og rettet av de dokumentasjonsansvarlige i CouchDB-prosjektet.

```
1  import uuid, sys
2
3  # Writes new UUID to the configuration file
4  def configWriter(filepath):
5      # Opens the configuration file
6      with open(filepath, "wb") as config_file:
7          # Writes a UUID in hex only, using uuid1
8          # The first 24 bits are random, where the last 8 is the MAC-address
9          #   of eth0 on the local network card. This is later used for
               synchronization
10         #   and tracking
11         config_file.write("[couchdb]\r\nuuid = {}".format(uuid.uuid1().hex))
12
13 if __name__ == '__main__':
14     if sys.argv[1] == "windows":
15         configWriter("libellus/windows_database/CouchDB/etc/couchdb/local.ini")
```

Kodesnutt 5.16: configWriter()

5.18 Konfigurasjon

CouchDB sine konfigurasjonsfiler er lokalisert i */etc/couchdb*. Disse er fra installasjon av *default.ini* og *local.ini*. Førstnevnte inneholder all standardkonfigurasjon, og blir overskrevet ved en oppgradering av CouchDB. *local.ini* er tiltenkt en lokal egendefinert konfigurasjon. Da CouchDB i vårt tilfelle må oppgraderes manuelt, og det i Windows ikke er anledning til å benytte flere enn disse to filene, har vi valgt å legge all fast konfigurasjon i *default.ini*.

I dette avsnittet beskriver vi de endringene som er spesifikke for at Libellus skal fungere som ønsket. Standardinnstillinger vil ikke bli berørt her, men er godt kommentert i selve konfigurasjonsfilen. Utdragene er hentet fra Windows-konfigurasjonen, men er tilsvarende på andre operativsystem. Unntakene er OS-spesifikke elementer, som fremoverskråstrek for katalogseparatør, kontra bakoverskråstrek.

Her vises et egendefinert navn som beskriver installasjonen. Dette er synlig ved et besøk direkte på rota til vebbtjeneren, eksempelvis https://127.0.0.1:6984. Denne informasjonen kan i et klyngeoppsett brukes til å kontrollere at kjørende instanser i nettverket er oppdatert.

```
1  [vendor]
2  name = Libellus - Crisis Management Tool
3  version = 1.0
```

Dersom UUID-skriptet ikke fungerer, eller filen *local.ini* ikke kan avleses av CouchDB, er det definert en UUID til å falle tilbake på. Dette er for å sørge for at Libellus fremdeles kan starte, men med en begrensning i funksjonalitet, nemlig at replikering ikke vil fungere.

```
[couchdb]
uuid = 18789bcd65b863fb25196e57c5ccde9b
```

For å åpne opp for trafikk fra utsiden, slik at CouchDB lytter på enhver tilgjengelig IPv4-adresse settes *bind_address* til 0.0.0.0. Merk at tilsvarende også kan gjøres for IPv6, men i vårt tilfelle er dette kommentert ut.

```
[httpd]
; Binds to all IPv4-adresses
bind_address = 0.0.0.0
; Binds to all IPv6-adresses
;bind_address = ::
```

Ved å legge til faste linker til databasefilene, kan de hentes ut gjennom nedlastningsgrensesnittet på verktøymodulen. Dette er gjort for å sikre at databasefilene enkelt kan eksporteres uten inngående teknisk kompetanse. Eksportering til XML og JSON er samlet i samme modul.

```
[httpd_global_handlers]
journal.couch = {couch_httpd_misc_handlers, handle_utils_dir_req, "../../../
    db_files/journal.couch "}

chat.couch = {couch_httpd_misc_handlers, handle_utils_dir_req, "../../../
    db_files/chat.couch"}
```

Dersom rota til vebbtjeneren besøkes på https://127.0.0.1:6984, vises en melding om hvordan Libellus kan startes. Dette er primært relevant for brukere av nettbrett og smarttelefoner som har fått oppgitt en IP-adresse til en Libellus-tjener. Full sti til Libellus sitt vebbgrensesnitt står også i verktøymodulen. I tillegg er stien til Libellus forkortet fra *127.0.0.1:6984/libellus/_design/main/index.html* til *127.0.0.1:6984/libellus/index.html*.

```
/ = {couch_httpd_misc_handlers, handle_welcome_req, <<"Welcome, please go to
    https://127.0.0.1:6984/libellus/index.html to start Libellus">>}
[vhosts]
127.0.0.1:6984/libellus = /libellus/_design/main/
```

Ved å kommentere ut *_utils* deaktiveres administrasjonsgrensesnittet Futon, dette er med på å hindre manipulasjon av databasen. Merk at det uansett kreves brukernavn og passord for å gjøre endringer her, men med denne konfigurasjonsendringen er det lagt en ekstra sperre for utilsiktet adgang.

```
; _utils = {couch_httpd_misc_handlers, handle_utils_dir_req, "../share/couchdb/
    www"}
```

I Libellus er CouchDB satt til å starte i en skjult konsoll. For å hindre at denne og lagringsmediumet fylles opp med logginformasjon fra databasesystemet, er logging deaktivert. På høyeste nivå der *level* er satt til "debug" kunne loggen ha inneholdt data som er med på å vise endringer på databasen, men da blir den fort uforholdsmessig stor. Derfor er det heller gjort tiltak i Libellus for å sikre integritet og sporbarhet.

```
[log]
level = none
```

CouchDB holder selv styr på at replikatorskriptet og holdes kjørende, ved at dette startes som en bakgrunnsprosess. For å kjøre dette via Python på windows, startes skriptet via en satsvis fil.

```
[os_daemons]
repl = ..\..\..\portable_scripts\start_libellus_multicast_win.bat
```

HTTPS-trafikk har vi satt som standard kommunikasjonsmåte med CouchDB, og vanlig HTTP er deaktivert for å sikre kryptert kommunikasjon.

```
[daemons]
httpsd = {couch_httpd, start_link, [https]}
;httpd={couch_httpd, start_link, []}
```

Stien til nøkkelparene for TLS er definert, det samme er hvilken port trafikken skal overføres på. Det benyttes en port utenfor skalaen 0 - 1023 (velkjente porter) for å unngå at det kreves administratortilgang for å kjøre Libellus.

```
[ssl]
cert_file = ../../../config/cert/server_cert.pem
key_file = ../../../config/cert/server_key.pem
port = 6984
```

6 Kvalitetssikring og testing

Hensikten med dette kapittelet er å beskrive de testene som har blitt utført på programvareløsningen. For de konkrete testene har vi laget en testplan, hvor vi har prioritert sjekk av forskjellig funksjonalitet. Vi har gjennomført systemtester med brukere for å få bruksmessige tilbakemeldinger på design, og for å luke ut feil som åpenbarer seg i praktisk bruk. Til slutt har vi avsluttet med en akseptansetest, hvor oppdragsgiver har testet programvaren under en kriseøvelse.

6.1 Underveis i prosjektet

Under hele utviklingsperioden har vi kontinuerlig testet ny funksjonalitet og konferert med hverandre om kode, design og brukervennlighet. Det å utvikle med JavaScript fører til veldig hyppig testing, og vi har benyttet konsollen til Google Chrome mye og med gode resultater.

I avsnitt 5.2 nevnte vi at Libellus har blitt testet i de mest brukte nettleserne. Vi har da kategorisk gjennomgått all funksjonalitet, og påsett at resultatene har blitt som ønsket og likt visuelt utformet i alle nettleserene. Tjenesteoppdagelsesskriptet er testet på samme måte i Windows 7 og 8, Ubuntu Linux 12.04 og 14.04 samt på Apple OS X Mavericks.

I starten av programmeringsperioden kikket vi innom muligheten for å kontinuerlig modulteste programvaren. Det mest aktuelle alternativet hadde vært å sette seg inn i QUnit [87], som har sitt utspring i jQuery-prosjektet, men vi så på dette som meget ressurskrevende. Det tar lang tid å lage modultester, noe vi ikke fant plass til i Gantt-skjemaet. Vi testet i stedet hver enkelt kodesnutt manuelt, noe som er effektivt med JavaScript-terminalen i Google Chrome.

Bruken av feilrapporteringssystemet Tiny Issue har vært veldig verdifullt. Pr. 22.04 har vi i underkant av 150 avsluttede saker, med 240 kommentarer. For det meste har disse sakene omhandlet programfeil, og forbedringsforslag.

6.2 Nyinstallert miljø

Libellus skal fungere operativt på en nyinnkjøpt datamaskin anskaffet hos nærmeste elektronikkforhandler. For å teste dette valgte vi å kjøre en nyinstallasjon av Windows 8 i et virtuelt miljø. Hovedpunktet som ble testet her var normal operativ oppstart og drift av programvaren. Under denne testingen ble det klart for oss at en ren installasjon av CouchDB medfører at noen DLL-filer fra *Microsoft Visual C++ 2010 Redistributable Package (x86)* [88] blir installert, dersom disse ikke er innlagt fra før. Vi valgte derfor å legge disse inn sammen med binærfilene til CouchDB. Dette løste problemet og all annen funksjonalitet i Libellus fungerte etter planen.

På slutten utviklingsperioden gjennomførte vi en tilsvarende test på Windows 8.1, fordi dette operativsystemet har høyest sannsynlighet for å være ferdiginstallert en nyinnkjøpt datamaskin. Den største endringen fra Windows 8 til 8.1 med mulighet til å påvirke vårt system er overgangen fra en standardinstallasjon av nettleseren Internet Explorer 10 [89] til Internet Explorer 11 [90]. Når Libellus blir startet åpner den brukerens standard nettleser. Testen på Windows 8.1 foreløp uten problemer.

6.3 For oppdragsgiver

På de møtene vi har hatt med oppdragsgiver har vi fremvist applikasjonen og latt Espen ta rollen som bruker av systemet, for å få bruksmessige tilbakemeldinger. Dette har vært utbytterikt da

vi stadig har fått en bekreftelse på at vi har vært på rett vei.

Vi hadde også en møtesesjon i Oslo med flere deltakere, der vi i hovedsak demonstrerte all funksjonalitet, og beskrev de tankene vi hadde hatt bak løsningene. Dette var en spennende og ny situasjon å komme opp i, hvor vi virkelig fikk prøvd å selge produktet vårt.

Responsen vi fikk var god, men de hadde også noen ønsker og forslag til endringer på brukergrensesnitt og funksjonalitet. Som tidligere nevnt i avsnitt 5.14 ønsket oppdragsgiver en fremvisning av klokke med lokal tid. De ønsket også noen andre tekstlige endringer, blant annet andre navn på klassifiseringer, og endringer på teksten i menyen. En tydeligere visuell profil på det aktive menyelementet ble også etterspurt.

I oversikten over oppfølgningsmerknader ville de gjerne ha fire typer status: Avsluttet, forkastet, åpen og ventende. I følge oppdragsgiver kunne den ventende statusen være en oppfølging som ikke rekker tidsfristen den er satt til. Vi påstår at det ikke er noen forskjell på en åpen og ventende sak. Istedenfor mener vi at dersom en sak ikke blir løst innen tidsfristen, kan denne avsluttes med en kommentar før det opprettes en ny sak med samme innhold og ny tidsfrist. Dette har vi hatt en god kommunikasjon med oppdragsgiver om, og de er enige i vårt resonnement.

6.4 Skalatest

Under skalatesten var formålet å teste anvendelsen av programvaren på et flertall ukjente maskiner. Vi tok i bruk en av høgskolens datasaler, og startet programvaren på 16 maskiner med Windows 7 installert. Alle maskinene var tilsluttet skolenettet.

Punkter som ble vektlagt under testingen:

- feilfri oppstart på ukjente maskiner
- responsiv kommunikasjon mellom de ulike enhetene

I første omgang ble Libellus startet på en og en maskin av gangen, noe som foregikk uten problemer. Etter at den siste maskinen hadde startet Libellus var alle instansene knyttet sammen og replikerte til hverandres databaser. I neste test startet vi fire og fire maskiner synkront, som også forløp etter planen. Deretter startet vi å logge fra hver vår maskin, og observerte at data ble sendt ut til de andre kjørende instansene. Dette tok fra ett til to sekunder, uavhengig av datamengde. Kort oppsummert fikk vi testet de to punktene, uten å finne noen feil eller uregelmessigheter.

Som et ytelseskrav hadde vi satt at Libellus skal støtte synkronisering av data mellom 20 ulike instanser samtidig. Med de ressursene vi hadde tilgjengelig lot det seg ikke gjøre å teste dette kravet fult ut. Selv om differansen på fire noder i et maskenett kan virke mye, med tanke på formelen $c = \frac{n(n-1)}{2}$ som tilsier at økningen på antall linker blir fra 120 til 190, betyr ikke dette at trafikkmengden øker tilsvarende. Dette antallet beskriver hvor mange stier trafikken kan gå, og ikke noe om datamengen.

6.5 Brukertest

Under brukertesten fikk vi hjelp av seks studenter med ulik kompetanse innenfor informasjonssikkerhet og drift av data og nettverkssystemer. Alle benyttet sin egen bærbare maskin, hvorav fire Windows-brukere, både versjon 7 og 8, fikk tilgang på programvaren slik den vil være på en minnepinne, mens de med GNU/Linux og OS X koblet seg opp mot vår test-tjener. Testerne hadde ikke fått tilgang på programvaren i forkant, og var dermed nødt til å finne ut av hvor-

dan systemet skulle brukes på egenhånd og ved hjelp av brukermanual. Vi hadde to scenario planlagt som skulle illudere hendelser programvaren er tiltenkt brukt i. Disse ligger i vedlegg C.

Elementer som ble vektlagt under testingen:

- hvorvidt brukergrensesnittet var intuitivt og lett å forstå
- hvilke elementer som var selvforklarende, og hvilke det eventuelt burde stå mer om i brukermanualen
- om scenariene virket realistiske og gode
- kommunikasjon mellom enhetene
- funn av eventuelle feil i programvaren
- forsøk på manipulasjon av systemet, sletting av innlegg og liknende uønsket aktivitet

Her fikk vi konkrete tilbakemeldinger på at noen av de grafiske elementene burde være mer tydelige, dette gjaldt kommentarknappen og hvor tidlig rullefeltet kom på plass. Brukerne hadde ingenting å utsette på brukermanualen, da denne etter deres mening var meget anvendelig.

I denne testen koblet vi også inn våre egne maskiner, slik at det var totalt åtte ulike brukere som benyttet Libellus. All kommunikasjon i systemet foregikk raskt og responsivt. Vi foretok fem målinger av tidsbruk fra datainnleggelse til informasjonen var distribuert mellom alle instansene. Ved alle målingene tok det rundt ett sekund fra data var innlagt til den dukket opp på de ulike skjermene.

Det ble funnet flere feil i programvaren, men ingen graverende eller som påvirket sikkerheten på noen måte. For det meste dreide de seg om overfylte tekstbokser, altså der brukeren med vilje hadde laget en tittel med flere tusen tegn. I lynmeldingsmodulen ble det oppdaget en feil som gjorde at innlegg ikke kom opp automatisk dersom en bruker benyttet et blankt brukernavn. I funksjonen som genererer XML var det en feil, som førte til at om det ble lastet opp ett vedlegg som startet på et tall, så ble all XML-data ugyldig.

Vi fikk også inn et forslag til forbedring når det gjaldt innlegging av dato og klokkeslett i applikasjonen. Brukerne mente at det var noe tungvint å gjøre dette i et tekstfelt, og ønsket seg heller en løsning der det var mulig å bruke musepeker for å legge inn denne informasjonen. Denne funksjonaliteten har blitt implementert i den endelige versjonen av Libellus.

6.6 Test hos oppdragsgiver under en kriseøvelse

Den 09.05.14 gjennomførte oppdragsgiver Conax en kriseøvelse som hadde som mål å teste Libellus som krisehåndteringsverktøy sammen med bedriftens hendelseshåndteringsplaner. Komplett referat fra øvelsen ligger vedlagt i vedlegg G, og vi gjengir her en kort oppsummering av dette sammen med våre kommentarer.

Tilbakemeldingen konkluderer med at Libellus er enkel å starte, lære og benytte. Conax sier også at programmet gir en merverdi når det kommer til dataoversikt og etterevaluering av krisen. De introduserer videre ønsket funksjonalitet skreddersydd deres virksomhet, med predefinerte prosedyrer og sjekklister. Vi ser på dette som en spennende funksjonalitet ha med i programmet og har laget Libellus slik at det skal være enkelt å implementere nye moduler.

Den offisielle åpent tilgjengelige versjonen ønsker vi imidlertid at skal være så generisk som mulig slik at den får en størst mulig utbredelse.

7 Avslutning

7.1 Effektutnyttelse av Libellus

For å få effektutnyttelse av Libellus må verktøyet implementeres i bedriftens hendelseshåndteringsprosess. I den forbindelse er det viktig å være klar over hvilke momenter Libellus introduserer, og hvilke nye aspekter det er avgjørende å være observant på.

Alle kjørende instanser av Libellus vil ha tilgang til den samme innlagte informasjonen. Dette vil resultere i en bedre informasjonsflyt, som igjen gir grunnlag for å ta gode og raske avgjørelser. Samtidig er det en fare for å bli rammet av informasjonsoverbelastning, fordi det med mer informasjon stilles høyere krav til utfiltrering av relevant data. Libellus har filtreringsfunksjoner der det er mulig å hente ut innlegg mellom gitte tidsrom og med spesifikt innhold. Disse bør brukes aktivt for å oppnå informasjonsoverlegenhet.

Mennesker har en begrenset kognitiv kapasitet og belastningen kan fort bli stor dersom en person alene må være ansvarlig for all loggføring, ha oversikt over saker som skal følges opp, og samtidig ha andre stabsmedlemmer rundt seg som forespør informasjon. Dette kan løses ved at flere logger separat på hvert sitt papir eller digitale tekstdokument, men da mistes strukturen over informasjonen. Med Libellus kan flere ha rollen som sekretær samtidig, noe som vil minske belastningen på enkeltpersoner, og sørge for at den som har størst nærhet til informasjonen fører denne inn. Et eksempel på dette kan være at den stabsansatte som er ansvarlig for sikkerhet journalfører sikkerhetsrelaterte hendelser. Med en fordelt loggføringsrolle blir det viktigere å ha en god intern kommunikasjon, slik at lik data ikke blir lagt inn flere ganger.

Data fra Libellus kan fremvises på ulike skjermer etter behov. For å få oversikt kan de aktive oppfølgningsmerknadene bli vist på en storskjerm. En annen utnyttelse er å benytte lynpratfunksjonen til å føre logg over tilstedeværende personell, og når fraværende kommer tilbake.

Elektronisk loggføring har den ulempen at en feil i program- eller maskinvare kan gjøre innlagt data utilgjengelig, og videre logging umulig. Libellus synkroniserer data lokalt mellom eventuelle andre kjørende instanser av programvaren, loggføringen vil ved denne typen feil kunne fortsette på en annen instans. Hvis Libellus kjøres fra en minnepinne, og datamaskinen det logges på går ned, kan minnepinnen med all data forflyttes til en annen maskin der journalføringen kan fortsette. Dette er fordi all data lagres lokalt på minnepinnen.

I etterkant av en hendelse eller krise, både reelle og øvelser, er det viktig å evaluere og diskutere hva som kunne ha vært gjort annerledes. Med Libellus er det mulig å gå tilbake til spesifikke tidspunkt for å få en oversikt over hvilken informasjon som var kjent, og på hvilken bakgrunn avgjørelser ble tatt. Verktøyet støtter også fremvisning av data i en tidslinje.

7.2 Resultater

I oppgavebeskrivelsen beskrev vi hovedmålene for programvaren vil skulle utvikle. I dette avsnittet vil vi hente frem igjen disse punktene, vist i tabell 11, og oppsummere de resultatene vi har oppnådd.

• Lokal, tekstbasert krisejournal for notater	☑ Dette er grunnfunksjonaliteten i Libellus. Å føre logg er hovedmålet til en bruker av programvaren. Vi har derfor lagt mye ressurser inn i å få løsningen til å være rask og sette seg inn i, samt effektiv å benytte under en krise.
• Støtte for vedlegg av kommentarer, oppfølgningsmerknader og dokumenter	☑ Programmet støtter kommentarføring på innlagte saker. For å kunne legge til vedlegg, elektroniske eller fysiske, må det legges inn en kommentar. Oppfølgingsmerknader har en egen oversiktsside, slik at det er lett for krisestaben å få en oversikt over fullførte og utestående oppgaver.
• Minimalt krav til eksisterende infrastruktur	☑ Minimumskravet for å ta i bruk Libellus er en datamaskin som kjører Windows 7 eller 8. Det kreves ingen dedikerte tjenere eller internettilgang for å gjøre journalføringssystemet operasjonelt.
• Kryptert datasynkronisering mellom eventuelle andre operative instanser av programvaren	☑ Datasynkronisering er støttet fullt ut i et lokalnettverk med en minimumsinfrastruktur bestående av en ruter eller et datamaskin-til-datamaskin-nettverk.
• Støtte for datamanipulasjon for effektiv fremvisning av relevant data	☑ Vi har implementert filtrerings og søkefunksjoner for å effektivt kunne vise frem relevant data.
• Løsninger basert på åpen kildekode	☑ All kode og programvare brukt i og produsert til systemet er åpen.
• Bruker-, operativ- og vedlikeholdsdokumentasjon	☑ Vi har utviklet ønsket dokumentasjon i tillegg til et eget avsnitt om effektutnyttelse av Libellus.

Tabell 11: Resultater

Vi er fornøyd med å ha nådd alle målene på en meget tilfredsstillende måte. På områder som ikke er definert i oppgavebeskrivelsen er det imidlertid enkelte ting vi skulle ha implementert bedre. Det ene er tidslinja som ligger under den visuelle modulen til programvaren. Vi definerte allerede i forprosjektet, vedlegg K, at en visuell fremstilling av innlagt data med grafer og statistikk ville kreve mye ressurser. Dette resulterte i at denne modulen ble lagt i kategorien "fint å ha", og dermed ikke har fått en høy prioritet. Vi har ikke skreddersydd vår egen løsning, men istedenfor prøvde vi mange ulike, ferdige bibliotek for dette, frem til vi til slutt fant et vi likte. Denne tidslinja har vi så modifisert med fargetema og datainnhenting slik at den fungerer i Libellus. Bruksnytten er likevel noe redusert, da det ikke er mulig å hoppe mellom tidslinje og innlagt data. En eventuell forbedring her må lanseres i en senere versjon av programvaren.

Det andre vi gjerne skulle ha fått til å fungere er full portabilitet til Apple OS X Mavericks, hvor det her gjenstår å få kompilert en pakket utgave av programvaren hvor linker til databasefiler er egendefinerte. Innen tidsfristen for rapportinnlevering 19.05.14, har vi ikke fått dette til å fungere.

7.3 Videre arbeid

Programvaren er levert i versjon 1.0 og fungerer operativt etter de kravene som ble stilt fra oppdragsgiver. I løpet av utviklingsperioden har vi allikevel kommet over flere spennende temaer og dertil funksjonalitet vi gjerne skulle ha implementert, men som vi har innsett at vi ikke har hatt tid til.

Stedsangivelse - Vi ser absolutt nytten i å kunne plassere en hendelse på et kart, men dette fører imidlertid til en rekke problemstillinger. Disse går ut på hvor lenge en hendelse skal være gyldig i kartet, og om enkelte områder utelates for å få bedre detaljer, da kartdata tar stor plass. I tillegg er kartdata ferskvare, og det stiller høyere krav til hvor ofte programvaren skal oppdateres.

Meldingskø - Innlegg i nettpraten distribueres i dag via CouchDB sin replikator. Denne har ikke noen form for garantert levering innen en gitt tid. Det er heller ikke sikkert at innleggene blir sendt i den rekkefølgen de ble lagt inn. Under testing har vi ikke oppdaget noen problemer, men med et økt antall klienter og datamengde, vil en meldingskø være å anbefale. Alternativer her er for eksempel ActiveMQ, eller RabbitMQ. Sistnevnte er skrevet i Erlang, og vil trolig være lettere å implementere sammen med CouchDB.

Innhenting av eksterntid utføres i dag med et AJAX-kall opp mot et PHP-skript, kontra en løsning ved bruk av NTP. Dette er valgt fordi det ikke er mulig å hente inn NTP-trafikk via nettleseren. Dette må eventuelt løses av en demon, men er noe vi i denne omgang ikke har drøftet fordeler eller ulemper ved.

Vi valgte under avgrensningen til oppgaven å se bort i fra selektiv synkronisering, fordi vi antok at det ville kreve mye tid å få det anvendelig nok. Etter å ha havnet på CouchDB som databaseløsning og lest igjennom dokumentasjonen for replikering, så vi at denne funksjonaliteten er støttet igjennom filtrert replikering. Her sto vi ved et veiskille, men vurderte opp mot Gantt-skjemaet at å inkludere dette kom til å kreve mer tid enn vi hadde til rådighet.

Under hele arbeidsperioden har vi diskutert ulike måter å få til flerlokalisert logging. Problematikken her er knyttet til NAT, brannmurer og trafikk som ikke slipper igjennom mellom to punkter som er tilknyttet Internett. I tillegg er IP-adresser ofte dynamiske, som gjør at det er vanskelig å garantere at kommunikasjonen vil opprettholdes under hele krisen. En metode er å sette opp en Libellus-tjener på Internett og la de andre instansene synkronisere opp mot denne, men dette vil igjen kreve en form for infrastruktur. Vi har uansett lagt til muligheten for å manuelt synkronisere opp mot en valgt klient i programvaren, slik at dette eventuelt kan settes opp. En annen metode vil være å benytte VPN-tunelleringsteknologi mellom lokasjonene det logges på, dette blir helt utenfor programvaren, men er en mulig løsning på problemet. Det siste alternativet vi har tenkt på er å benytte Tor-nettverket for kommunikasjon. Her eksisterer det allerede teknologi som TorChat, en nettpratklient basert på jevnbyrdsnett-teknologi. Torprosjektet jobber også med Tor Instant Messaging Bundle [91], så her er det to alternativer for meldingsutveksling.

I dag vil alle på lokalnettverket ha mulighet til å legge inn og se meldinger i systemet, så lenge de har korrekt URL. Dette er gjort slik for at brukere av nettbrett, smarttelefoner, eller andre enheter uten kjørende programvare kan bruke systemet. Et sikkerhetstiltak som kan gjøres for å få til samme funksjonalitet uten mye konfigurasjon fra brukerens side, er å ta i bruk en en mellomleddstjener [92]. CouchDB kan da settes til kun å akseptere innkommende trafikk på 127.0.0.1. Det opprettes en database hvor det fra Libellus kan spesifiseres hvilke brukere som har anledning til å logge seg på. Denne tilgjengeliggjøres fra verktøymodulen. Så kjøres f.eks. Nginx i forkant som fra oppstart kun tillater at replikering slipper igjennom. Denne henter

brukere fra databasen. Dette fører til at programmet er opererbart innenfra uten brukernavn og passord, men at det fra utsiden kreves innlogging.

7.4 Bidrag til åpen kildekode

Underveis i prosjektet har vi kommet med innspill til CouchDB på IRC-kanalen, opprettet feilrapporter, og kommet med forslag til utvidelser i deres feilrapporteringssystem JIRA.

All egenutviklet kode som utgjør kjernefunksjonaliteten i Libellus, altså all JavaScript, CSS og HTML er lisensiert under Apache 2 [93] og delt på prosjektets GitHub-side [94]. I tillegg er konfigurasjonsfilene, brukermanualen og skriptene med et unntak delt under samme lisens. Unntaket er filen *libellus_multicast_.py*, der vi bruker lisensen Creative Commons Attribution 3.0 Unported [65].

Logoen vår, med de derivasjonene vi selv har laget, er lisensiert under CC BY-NC-ND 4.0 [95].

Skriftlig materiale vi har produsert i forbindelse med denne bacheloroppgaven, inkludert scenario, operativ-, vedlikeholdsdokumentasjon og selve rapporten, er lisensiert under CC BY-SA 4.0 [96].

7.5 Evaluering av prosjektarbeidet

Ettersom vi som gruppemedlemmer kjenner hverandre godt både når det gjelder kompetanse, arbeidsvilje, humor og har en meget god kommunikasjon, visste vi fra prosjektstart hvilke egenskaper det var viktig å spille på og videreutvikle. Som en gruppe på to vil ikke lærdommen innenfor gruppedynamikk nødvendigvis bli størst, og vi har heller ikke forsøkt å fremme dette, men heller diskutert rundt måten vi har arbeidet på nå, kontra tidligere prosjekter med flere deltakere. Dette betyr ikke at vi ikke har vært flinke til å gi hverandre konkrete tilbakemeldinger, men den største kritiker til arbeidet har nok vært den som har utført det. Dermed har vi heller brukt mer tid på å fremsnakke den andres arbeid og komme med justeringer, snarere enn nedsablinger.

Vi har hovedsakelig arbeidet sammen på høgskolen, eller hatt løpende IP-telefonsamtaler for å koordinere og få tilbakemeldinger på arbeid. Så nær som alle avgjørelser som har blitt foretatt i prosjektet har kommet på bakgrunn av gruppemedlemmenes individuelle, konkrete forslag. Vi mener at dette er den mest effektive måten å sikre et prosjekts fremdrift på, uten at for mye tid går til spille. Et annet element som har sørget for å få prosjektet tidsmessig i havn er Gantt-skjemaet. I utgangspunktet hadde vi satt opp dette noe flytende, men her fikk vi innspill på at vi burde være flinke til å bruke konkrete milepæler. I vedlegg D vises Gantt-skjemaet med reell tidsbruk for prosjektet. Vårt estimat ligger i forprosjektrapporten K og som det fremgår av disse, har anslått tidsbruk stemt noenlunde overens med det som ble resultatet. Differansen utgjør hovedsaklig de sløyfede punktene risikoanalyse og databasemodellering. Sistnevne ble mindre omfattende enn antydet på forhånd, fordi vi valgte å gå for en ustrukturert database. Risikoanalysen ble omgjort til en iterativ prosess for hver funksjonalitet vi utviklet, eksempelvis har vi sett på hvor mye rettigheter en bruker må ha for å utføre de nødvendige handlingene. Vi har ikke vært helt lojale mot timeføringen i dette prosjektet, men er sikre på at vi begge minst har lagt inn 30 timer hver i prosjektet hver uke. Dette begrunnes med tre faste arbeidsdager på skolen ukentlig, i tillegg til egenmotivert kvelds og helgejobbing.

Vi har hatt noen utfordringer som har krevd ekstra innsats. Det å gjøre CouchDB portabel har krevd mye prøving og feiling, for selv om databasen er installerbar på flere plattformer er den ikke like flyttbar. Vi tror allikevel ikke at vi kunne ha gjort noe annerledes på dette området, da denne databaseløsningen absolutt har vært den beste for vårt prosjekt.

Vi ser i ettertid at vi kunne brukt mer tid på å undersøke andre applikasjoner av åpen kildekode basert på CouchDB. Dette hadde gitt oss et bedre bilde over hvordan fullstendige applikasjoner ser ut, og ville satt oss raskere i riktig tankegang når det gjelder bruk av views og API-et.

I dette prosjektet valgte vi å gå for en minimalistisk metode for systemutvikling med inkrementelle og iterative prosesser. Dette har ført til at vi har fått stadige tilbakemeldinger fra oppdragsgiver som har vært med på å forme produktet. Samtidig har vi også gitt hverandre tilsvarende tilbakemeldinger på utført arbeid. De forskjellige arbeidsoppgavene i prosjektet har vi forsøkt å fordele etter personlig interesse og kompetanse, men vi har også lagt vekt på at begge skal få prøve seg på områder de er mindre sikre på. Vi har i dag lik oversikt over programvarens funksjonalitet og oppbygning, således som teorien bak.

Det har vært mange spennende temaer å ta tak i under bacheloroppgaven, og vi innser at vi har brukt tid på å ta avstikkere hvor vi har sett nærmere på interessante tema som ikke nødvendigvis har havnet i oppgaven. Allikevel føler vi at dette har vært tid vel brukt, og som har vært med på å gi oss en større helhetsforståelse innenfor hendelseshåndtering, brukergrensesnitt og programmering.

7.6 Konklusjon

Etter et halvt års arbeid med bacheloroppgaven, er vi svært stolte og fornøyde med å kunne presentere det ferdige produktet vårt, Libellus 1.0. Programvaren lever opp til alle kravene satt av oppdragsgiver, og vi har bevist at det er mulig å lage et portabelt infrastrukturuavhengig journalføringsprogram med støtte for replikering.

Det har vært spennende og utfordrende å lage en moderne og interaktiv vebbapplikasjon med tidsriktige bibliotek som Bootstrap og jQuery. Vi har også fått kjenne på hvordan det er å utvikle i et miljø der kode blir eksekvert i tilfeldig rekkefølge. Inn i prosjektet hadde vi med oss grunnleggende kunnskap om tradisjonelle relasjonsdatabaser, og igjennom utviklingsperioden har vi lært oss å kjenne den ustrukturerte måten å tenke på, nærmere bestemt NoSQL. Dette blir av mange sett på som databasene for fremtiden, så kunnskap innenfor denne teknologien er bra å ha i erfaringslista.

Vi har under hele perioden gjort avveininger mellom faktorene i CIA-triaden; Formålet og bruksområdet til Libellus fører til at tilgjengelighet og integritet har fått høyest prioritet, tatt i betraktning at verktøyet skal benyttes i et lukket rom av en krisestab som sitter der fordi de har tillit i organisasjonen.

Vi har dokumentert at det er behov for den typen programvare vi har utviklet, og vi håper at Libellus blir tatt i bruk, distribuert og videreutviklet. Verktøy alene løser ingen kriser. For det trengs det et kontinuerlig arbeid med planlegging, trening og opplæring i hendelseshåndtering.

Bibliografi

[1] Gjørv, A.B. *Rapport fra 22. juli-kommisjonen: Oppnevnt ved kongelig resolusjon 12. august 2011 for å gjennomgå og trekke lærdom fra angrepene på regjeringskvartalet og Utøya 22. juli 2011 : Avgitt til statsministeren 13. august 2012.* Norges offentlige utredninger. Univ.-Forl.

[2] Sommerville, I. 2010. *Software Engineering.* Pearson.

[3] Kjeserud, R. & Weisæth, L. 2007. *Ledelse ved kriser: - en praktisk veileder.* Gyldendal.

[4] Politidirektoratet. 2007. *Politiets beredskapssystem del I, Håndbok i krisehåndtering.* Politidirektoratet.

[5] Bråten, O. A. 2013. *Håndbok i krisehåndtering.* Cappelen Damm akademisk.

[6] Jon Gangdal, G. A. 2014. *Krise : forebygging, beredskap, håndtering, kommunikasjon.* Fagbokforlaget.

[7] Eriksen, J. 2011. *Krise- og beredskapsledelse : teamtrening.* Cappelen Damm akademisk.

[8] Aarset, M. 2010. *Kriseledelse.* Fagbokforlaget.

[9] Michael E. Whitman, H. J. M. 2006. *Principles of Incident Response and Disaster Recovery.* Cengage Learning, Inc.

[10] U.S. Department of the Army. 1997. Operational terms and graphics. field manuel no. fm 101-5-1/mcrp 5-2a, the military decision-making process. http://www.au.af.mil/au/awc/awcgate/army/fm101-5_mdmp.pdf. [Hentet 05.02.2014].

[11] German Federal Office for Information Security (BSI). 2009. Bsi-standard 100-4, business continuity management. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/BSIStandards/standard_100-4_e_pdf.pdf.

[12] Anderson, J., McRee, J., Wilson, R., & Team, T. 2010. *Effective UI: The Art of Building Great User Experience in Software.* O'Reilly Media.

[13] Franco, A. 2009. The 8 criteria for usable software. http://anthonyfranco.wordpress.com/2009/02/14/the-8-criteria-for-usable-software/. [Hentet 05.05.2014].

[14] Norges Blindeforbund. 2014. Debatt om universell utforming av nettsteder. https://www.blindeforbundet.no/internett/nyheter/debatt-om-universell-utforming-av-nettsteder. [Hentet 12.04.2014].

[15] Best Practical Solutions LLC. Rtir: Rt for incident response. https://www.bestpractical.com/rtir/. [Hentet 11.05.2014].

[16] Best Practical Solutions LLC. Rt: Request tracker. http://www.bestpractical.com/rt/. [Hentet 11.05.2014].

[17] Thales. Xomail. http://www.xomail.com/. [Hentet 11.05.2014].

[18] OpsCom Systems. Opscom. http://opscomsystems.com/. [Hentet 11.05.2014].

[19] One Voice. Kse-cim. https://www.onevoice.no. [Hentet 11.05.2014].

[20] Wikipedia. 2014. Replication (computing) — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Replication_(computing)&oldid=602400128. [Hentet 03.05.2014].

[21] Sadalage, P. & Fowler, M. 2012. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education.

[22] Wikipedia. 2014. Multi-master replication — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Multi-master_replication&oldid=604438445. [Hentet 03.05.2014].

[23] The Apache Software Foundation. 2014. Apache couchdb 1.6 documentation. http://docs.couchdb.org/. [Hentet 20.01.2014].

[24] J. Chris Anderson, J. L. & Slater, N. 2014. Couchdb the definitive guide. http://guide.couchdb.org/. [Hentet 20.01.2014].

[25] Actian Corporation. 2014. Actian online documentation portal. http://docs.actian.com/. [Hentet 20.01.2014].

[26] Oracle Corporation. 2014. Mysql documentation: Mysql reference manuals. http://dev.mysql.com/doc/. [Hentet 20.01.2014].

[27] The PostgreSQL Global Development Group. 2014. Documentation. http://www.postgresql.org/docs/. [Hentet 20.01.2014].

[28] The PostgreSQL Global Development Group. 2014. Postgresql 9.0.17 documentation, chapter 25. high availability, load balancing, and replication. http://www.postgresql.org/docs/9.0/static/different-replication-solutions.html. [Hentet 09.05.2014].

[29] SkySQL Corporation Ab. 2013. Mariadb versus mysql - features. https://mariadb.com/kb/en/mariadb-versus-mysql-features/. [Hentet 09.05.2014].

[30] Conant, W. 2013. 4 good things about couchdb. http://willconant.com/posts/2013-06-02/4-good-things-about-couchdb. [Hentet 12.04.2014].

[31] Anderson, J. C., Lehnardt, J., & Slater, N. 2010. *CouchDB: The Definitive Guide Time to Relax*. O'Reilly Media, Inc., 1st edition.

[32] Oracle Corporation. The binary log. http://dev.mysql.com/doc/refman/5.0/en/binary-log.html. [Hentet 11.05.2014].

[33] Oracle Corporation. Using triggers. http://dev.mysql.com/doc/refman/5.0/en/triggers.html. [Hentet 11.05.2014].

[34] Oracle Corporation. 2013. What's new in mysql 5.6. http://dev.mysql.com/tech-resources/articles/whats-new-in-mysql-5.6.html. [Hentet 09.05.2014].

[35] Haugen, K. 2010. A brief history of nosql. http://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html. [Hentet 03.05.2014].

[36] Couchbase. 2014. Why nosql? http://www.couchbase.com/why-nosql/nosql-database. [Hentet 03.05.2014].

[37] Stenberg, D. curl. http://curl.haxx.se/. [Hentet 11.05.2014].

[38] CouchDB. 2014. The core api, couchdb. http://guide.couchdb.org/draft/api.html. [Hentet 06.05.2014].

[39] Google. 2014. Chrome browser. https://www.google.com/intl/en/chrome/browser/. [Hentet 11.05.2014].

[40] Mozilla. 2014. Firefox web browser. http://www.mozilla.org/en-US/firefox/desktop/. [Hentet 11.05.2014].

[41] Microsoft. 2014. Internet explorer web browser. http://windows.microsoft.com/en-us/internet-explorer/download-ie. [Hentet 11.05.2014].

[42] Deveria, A. 2014. Can i use... http://caniuse.com/. [Hentet 11.05.2014].

[43] Microsoft. 2014. Microsoft visio. http://office.microsoft.com/en-us/visio/. [Hentet 11.05.2014].

[44] Software Freedom Conservancy, Inc. 2014. Inkscape. http://www.inkscape.org/. [Hentet 11.05.2014].

[45] Hasselbring, M. 2014. Tiny issue. https://github.com/mikelbring/tinyissue. [Hentet 11.05.2014].

[46] vimonline development. 2014. Vim. http://www.vim.org/. [Hentet 11.05.2014].

[47] The GNOME Project. 2014. gedit. https://wiki.gnome.org/Apps/Gedit. [Hentet 11.05.2014].

[48] Ho, D. 2014. Notepad++. http://notepad-plus-plus.org/. [Hentet 11.05.2014].

[49] Sublime Text. 2014. Sublime text 2. http://www.sublimetext.com/2. [Hentet 11.05.2014].

[50] Google Inc. 2014. Google docs. https://docs.google.com/. [Hentet 11.05.2014].

[51] Jonathan Kew, Stefan Löffler, C. S. 2014. Texworks. https://www.tug.org/texworks/. [Hentet 11.05.2014].

[52] Dropbox, Inc. 2014. Dropbox. https://www.dropbox.com. [Hentet 11.05.2014].

[53] Brandl, G. & the Sphinx team. 2014. Sphinx - python documentation generator. http://sphinx-doc.org/. [Hentet 11.05.2014].

[54] CouchApp. 2014. Couchapp: Standalone couchdb application development made simple. https://github.com/couchapp/couchapp. [Hentet 11.05.2014].

[55] Python Software Foundation. 2001. Style guide for python code. http://legacy.python.org/dev/peps/pep-0008/. [Hentet 03.05.2014].

[56] Handlebars. 2014. Handlebars. http://handlebarsjs.com/. [Hentet 09.05.2014].

[57] Zelanski, P. & Fisher, M. 2003. *Color*. Prentice Hall.

[58] Guttman, E. May 2001. Autoconfiguration for ip networking: enabling local communication. *Internet Computing, IEEE*, 5(3), 81–86.

[59] The Avahi Team. 2012. Avahi. http://avahi.org/. [Hentet 09.05.2014].

[60] Apple Inc. 2014. Bonjour. http://www.apple.com/support/bonjour/. [Hentet 09.05.2014].

[61] Davies, J. emdns - bonjour/zeroconf/mdns in erlang. https://github.com/jasondavies/emdns. [Hentet 11.05.2014].

[62] Lyon, G. Nmap. http://nmap.org/. [Hentet 11.05.2014].

[63] Graham, R. D. 2014. Masscan: Mass ip port scanner. https://github.com/robertdavidgraham/masscan. [Hentet 05.02.2014].

[64] Cohen, A. 2013. Udp multicast in python and bittorrent live. http://engineering.bittorrent.com/2013/02/04/udp-multicast-in-python-and-bittorrent-live/. [Hentet 05.02.2014].

[65] Creative Commons. Creative commons attribution 3.0 unported. http://creativecommons.org/licenses/by/3.0/legalcode. [Hentet 11.05.2014].

[66] Microsoft. 2014. File system functionality comparison. http://msdn.microsoft.com/en-us/library/ee681827(VS.85).aspx. [Hentet 11.05.2014].

[67] Apple Inc. 2014. Xcode. https://developer.apple.com/xcode/. [Hentet 22.04.2014].

[68] Free Software Foundation, Inc. 2014. Various licenses and comments about them. https://www.gnu.org/licenses/license-list.html. [Hentet 11.05.2014].

[69] The jQuery Foundation. 2014. jquery: The write less, do more, javascript library. http://jquery.com/. [Hentet 11.05.2014].

[70] Bootstrap team. 2014. Bootstrap. http://getbootstrap.com/. [Hentet 11.05.2014].

[71] Bach, C. 2014. tablesorter. http://tablesorter.com/docs/. [Hentet 11.05.2014].

[72] Almende. 2014. Timeline. http://almende.github.io/chap-links-library/timeline.html. [Hentet 11.05.2014].

[73] Knightlab. timeline.knightlab. http://timeline.knightlab.com/. [Hentet 11.05.2014].

[74] Leung, K. chronoline.js. http://stoicloofah.github.io/chronoline.js/. [Hentet 11.05.2014].

[75] Massachusetts Institute of Technology and Contributors. Simile widgets timeline. http://www.simile-widgets.org/timeline/. [Hentet 11.05.2014].

[76] Alsup, M. 2014. jquery.form.js. https://github.com/malsup/form. [Hentet 11.05.2014].

[77] Steinert, R. 2014. Simple-file-attachment-for-couchdb.html. https://gist.github.com/rjsteinert/4074427. [Hentet 05.02.2014].

[78] nakupanda. 2014. bootstrap-dialog. https://github.com/nakupanda/bootstrap3-dialog. [Hentet 11.05.2014].

[79] Isaac Cambron, Matt Johnson, T. W. 2014. Moment.js. http://momentjs.com/. [Hentet 11.05.2014].

[80] The Apache Software Foundation. 2014. Apache couchdb 1.6 documentation, configuring couchdb, secure socket level options. http://couchdb.readthedocs.org/en/latest/config/http.html#secure-socket-level-options. [Hentet 08.05.2014].

[81] The OpenSSL Project. 2014. Openssl, verify. http://www.openssl.org/docs/apps/verify.html. [Hentet 08.05.2014].

[82] Apache Software Foundation. 2014. Jira: Couchdb. https://issues.apache.org/jira/browse/COUCHDB. [Hentet 11.05.2014].

[83] Open-source Erlang. 2014. Erlang otp r16b03-1 has been released. http://www.erlang.org/news/66. [Hentet 11.05.2014].

[84] CouchDB The Definitive Guide. 2014. Eventual consistencyy. http://guide.couchdb.org/draft/consistency.html. [Hentet 08.05.2014].

[85] The OWASP Foundation. 2014. Xss (cross site scripting) prevention cheat sheet. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet. [Hentet 08.05.2014].

[86] The Apache Software Foundation. 2014. Couchdb and the heartbleed ssl/tls vulnerability. https://blogs.apache.org/couchdb/entry/couchdb_and_the_heartbleed_ssl. [Hentet 08.05.2014].

[87] The jQuery Foundation. Qunit. http://qunitjs.com/. [Hentet 11.05.2014].

[88] Microsoft. Microsoft visual c++ 2010 redistributable package (x86). http://www.microsoft.com/en-us/download/details.aspx?id=5555. [Hentet 11.05.2014].

[89] Microsoft. Internet explorer 10 faq for it pros. http://technet.microsoft.com/en-us/library/hh846773.aspx. [Hentet 11.05.2014].

[90] Microsoft. Internet explorer 11 - faq for it pros. http://msdn.microsoft.com/en-us/library/dn268945.aspx. [Hentet 11.05.2014].

[91] Ulvestad, L. L. 2014. De vil gi anonym chat til folk flest. http://www.tu.no/it/2014/03/05/de-vil-gi-anonym-chat-til-folk-flest. [Hentet 03.05.2014].

[92] Newson, R. 2012. Nginx as a reverse proxy. http://wiki.apache.org/couchdb/Nginx_As_a_Reverse_Proxy. [Hentet 03.05.2014].

[93] The Apache Software Foundation. 2004. Apache license, version 2.0. http://www.apache.org/licenses/LICENSE-2.0.html. [Hentet 11.05.2014].

[94] Daniel Solstad, Ole Johan Rasch. Libellus. https://github.com/Libellus. [Hentet 11.05.2014].

[95] Creative Commons. Creative commons attribution-noncommercial-noderivatives 4.0. http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode. [Hentet 11.05.2014].

[96] Creative Commons. Creative commons attribution-sharealike 4.0 international. http://creativecommons.org/licenses/by-sa/4.0/legalcode. [Hentet 11.05.2014].

[97] Mozilla Developer Network. 2014. Inheritance and the prototype chain. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Inheritance_and_the_prototype_chain. [Hentet 09.05.2014].

A Forkortelser og ordforklaringer

Her følger en forklaring av forkortelser og ord benyttet i rapporten.

Forkortelser:

- **AJAX** - Asynchronous JavaScript and XML
- **API** - Application Programming Interface
- **CLI** - Command-line Interface
- **DLL** - Dynamic-link Library
- **DOM** - Document Object Model
- **GMT** - Greenwich Mean Time
- **GPS** - Global Positioning System
- **HTTPS** - Hypertext Transfer Protocol Secure
- **ISP** - Internet Service Provider
- **JSON** - JavaScript Object Notation
- **LTS** - Long-term Support
- **NAT** - Network Address Translation
- **NTP** - Network Time Protocol
- **OOP** - Objektorientert Programmering
- **REST** - Representational State Transfer
- **SQL** - Structured Query Language
- **SVN** - Apache Subversion
- **TLS** - Transport Layer Security
- **UTC** - Coordinated Universal Time
- **UUID** - Universally Unique Identifier
- **VPN** - Virtual Private Network
- **XMPP** - Extensible Messaging and Presence Protocol
- **XSS** - Cross-site Scripting

Ordforklaringer:

- **Avluse** - Prosessen med å finne og korrigere feil i et dataprogram (debug)
- **Datamaskin-til-datamaskin-nettverk** - En midlertidig tilkobling mellom datamaskiner, løst via direkte trådløs nettverkskommunikasjon mellom enhetene (ad hoc network)
- **Demon** - Bakgrunnsprosess (daemon)
- **Informasjonskapsel** - En liten tekst delt av nettsiden og klienten (cookie)
- **Informasjonsoverlegenhet** - Evnen til å utnytte informasjon bedre enn en eventuell motpart
- **Infrastruktur** - Eksisterende faste anlegg herunder nettverk, internettilknytning, brukerdatabaser og tjenester
- **Jevnbyrdsnett** - Desentralisert og distribuert nettverksarkitektur (peer-to-peer)
- **Klynge** - Sammenstilling av datamaskiner som samarbeider (cluster)
- **Krise** - En hendelse med potensiale til å true viktige verdier og svekke en organisasjons evne til å utføre vesentlige funksjoner
- **Libellus** - Ordet libellus er latinsk og betyr liten bok. Vi valgte dette navnet på programmet vårt fordi det norske domenenavnet var ledig og fordi vi mener det klinger bra.
- **Mellomleddstjener** - En tjener som står mellom en klient og en annen tjener (proxy)
- **Modultest** - En måte å teste enkelte moduler / kodesnutter på, ved å verifisere at de fungerer som de skal (unit test)
- **Mønsterpraksis** - Framgangsmåter som i praksis har vist seg å fungere bedre enn andre (best practice)
- **Programlus** - En feil i et dataprogram (bug)
- **Replikere** - Dele informasjon og sikre konsistent data mellom redundante kilder
- **Rullefelt** - Et felt som ofte er plassert på høyre side av skjermen og som brukes til å navigere oppover og nedover på lange nettsider (scroll bar)
- **Satsvis fil** - Tekstfil som inneholder kommandoer kjørt av en CLI-klient (batch file)
- **Unix-tidsstempel** - Antall sekunder siden 1. januar 1970
- **Vebb** - Verdensomspennende informasjonsnett basert på et system av tjenermaskiner på Internett. Også kjent som verdensveven (web)

B Innføring i JavaScript, AJAX og OOP

Da vi bestemte oss for kjerneteknologi forsto vi raskt at det var viktig å sette seg godt inn applikasjonsutvikling med JavaScript. Vår tidligere erfaring med språket begrenset seg til å gjøre brukergrensesnittet på ulike vebbsider mer dynamisk. Det er en ganske stor overgang fra dette til å lage en hel applikasjon. Vi samlet derfor opp tips og forklaringer i et eget dokument, hovedsakelig basert på egne erfaringer. Dette ble etter hvert av en viss størrelse, så vi bestemte å innlemme det i hovedrapporten vår. I følgende avsnitt kommer det derfor en mer utfyllende guide til viktig funksjonalitet i JavaScript.

AJAX - Asynkron JavaScript and XML er et konsept for utveksling av data med en tjener. Spørringene skjer parallelt i bakgrunnen i nettleseren, slik at nettsiden ikke trenger å lastes på nytt for å hente ny data. Dette får nettsiden til å virke mer som en applikasjon.

Callbacks - En anonym funksjon som blir kalt når en hendelse er oppnådd. Dette blir mye brukt i AJAX for å avklare når en spørring er ferdig. Et eksempel på en funksjon med callback blir vist i B.1.

```
1 function foo(arg1, arg2, callback) {
2     var result = arg1 + arg2;
3     callback(result);
4 }
```

Kodesnutt B.1: Funksjon med callback

For å dra nytte av denne funksjonen kan den kalles på som vist i kodesnutt B.2.

```
1 foo(1, 2, function(data) {
2     console.log(data);
3 });
```

Kodesnutt B.2: Eksempel på bruk av callback

Den anonyme callback-funksjonen blir kalt når funksjonen er ferdig eksekvert. Akkurat i dette tilfellet vil nok en return på result i *foo()* i stedet for callback vært mer naturlig.

Her er det ikke mulig å kjøre return på variabelen data, fordi funksjonen da vil returnere data inne i callback-funksjonen og ikke i *foo()*.

AJAX & Callbacks - I AJAX brukes callback-funksjoner aktivt, da kode blir kjørt asynkront og det må kunne signaliseres når en spørring er ferdig.

Dette er vist i følgende eksempel B.3.

```
1 $.ajax('get_username.php?user_id=123', function(username) {
2         $('#element').text('brukernavnet til bruker med id 123 er ' + username);
3 });
```

Kodesnutt B.3: Henter ned brukernavn og viser resultatet i et HTML-element

Koden i B.3 vil åpne filen *get_username.php* og sende med et parameter i URL-en. Vebbtjeneren vil sende tilbake brukernavnet til den aktuelle brukeren. Når spørringen er ferdig blir callback-funksjonen kalt og variabelen *username* vil inneholde brukernavnet. Denne verdien kan ikke returneres, men for eksempel settes rett inn i et HTML-element.

Siden AJAX er asynkront, vil dette fremheve uønsket oppførsel av programmet, fordi kode ikke trenger å bli eksekvert i den rekkefølgen det er skrevet. I eksempel B.4 kan det antas at

innholdet til HTML-elementet skal bli "Brukernavnet er olanor og fornavnet er Ola", men det kan ikke garanteres at den første spørringen blir ferdig før den andre, og resultatet kan dermed bli "og fornavnet er Ola Brukernavnet er olanor".

```
$.ajax('get_username.php?user_id=123', function(username) {
       $('#element').text('Brukernavnet er ' + username);
});

$.ajax('get_firstname.php?user_id=123', function(firstname) {
       $('#element').append(' og fornavnet er ' + firstname);
});
```

Kodesnutt B.4: Ikke stol på rekkefølgen i koden

En løsningen er å nøste disse AJAX-kallene, slik at den første må være ferdig før den andre starter.

```
$.ajax('get_username.php?user_id=123', function(username) {
    $.ajax('get_firstname.php?user_id=123', function(firstname) {
        $('#element').text('Brukernavnet er ' + username +
           ' og fornavnet er ' + firstname);
    });
});
```

Kodesnutt B.5: Nøsting av AJAX

OOP - JavaScript blir ikke sett på som et fullverdig objekt-orientert språk, da det på langt nær har all funksjonalitet som andre OOP-språk har, som blant annet klasser. Det er mulig å simulere klasser med funksjoner, da disse er objekter i seg selv. Eksempelet i kodesnutt B.6 viser en funksjon med én offentlig attributt og to metoder inne i seg [97]. Det er også en konstruktør for å initiere navnet i det objektet lages.

```
var Mother = function(str) {
    this.name = str;

    this.getName = function() {
        return this.name;
    }

    this.setName = function(val) {
        this.name = val;
    }
}

var mor = new Mother('Synne');
console.log(mor.getName());
```

Kodesnutt B.6: Denne koden vil produsere teksten "Synne"

For å få til arv og polymorfisme kan det løses på måten vist i B.7 . Her lages det en ny objekt, eller en funksjon som det egentlig er, og det kalles på *Mother* sin konstruktør for å få satt navnet[97]. Neste operasjon er selve arvebiten, der *Daughter* arver prototypen til *Mother*. Det vil si at *Daughter* vil nå ha alle de samme attributtene og metodene som *Mother*.

```
var Daughter = function(name) {
    Mother.call(this, name);
}

Daughter.prototype = Object.create(Mother.prototype);

var datter = new Daughter('Beate');
console.log(datter.getName());
```

Kodesnutt B.7: Denne koden vil produsere teksten "Beate"

C Scenario

C.1 Scenario - Kontorene brenner

Det er tirsdag 22. april og første virkedag etter påskeferien. Klokka er 12:30 og dere er på vei tilbake til kontorene fra lunsj når brannalarmen går. I det dere runder hjørnet ser dere at det strømmer folk ut fra bygningen. Det er ingen visuelle tegn til brann, og i folkemengden begynner det å snakkes om øvelse.

12:35. Det høres sirener to kvartaler unna. Samtidig er det litt usikkerhet på om alle er ute av bygget, eller om de bare har gått ut en annen utgang. Det har blant annet vært noen møter i tredje etasje.

- Diskuter ledelsesstruktur for hvordan denne hendelsen bør håndteres
- Hva er prosessen for opptelling av antall personer i bygningen?
- Hvilke tiltak bør nå gjøres?

12:40. Brannvesenet og politi er på plass. Gata har blitt stengt og politiet anmoder alle om å flytte seg bak sperringene. Røykdykkere tar seg inn i bygget.

12:50. Brannvesenet bekrefter at bygningen er tom, og at overrislingsanlegget har stanset et branntilløp i fjerde etasje. Det har vært noe røykutvikling og kontorene må gjennomluftes før noen av de ansatte kan ta seg inn i bygget. Dette kan tidligst skje påfølgende dag. Alle kontorene til bedriften er vannskadet.

- Hva fortelles til de ansatte?
- Hvordan informeres de ansatte?

13:00. Bedriften er i avslutningsfasen med et prosjekt til en ny stor kunde. Leveringsfristen er 24. april 12:00. Det hersker usikkerhet rundt hva det er tatt sikkerhetskopier av, og om tilstrekkelig med prosjektdata befinner seg utenfor de nå utilgjengelige kontorene.

14:00. Krisestab er samlet, og de nødvendige ressursene som trengs er innhentet. Dokumentasjonsprosessen starter. Hva har hendt? Hva er utført av oppgaver? Hva er planlagt videre?

Onsdag 08:00. Bygningen er ikke frigjort enda, det avventes ingeniørtekniskpersonell for å avgjøre om det har vært noen strukturell påvirkning.

11:35. Politiet har sterke indisier på at brannen var forårsaket av varmgang i et elektrisk apparat. De anslår at bygget vil bli frigitt i løpet av torsdag.

14:00. Adm. dir lurer på hva som skjer med den viktige nye leveransen. Han har kunden på hold på en annen linje.

Torsdag 09.00. Bygget er frigitt og nødvendig personell kan ta seg inn i bygningen. Alt teknisk utstyr er vannskadet.

- Hvordan skal bedriften komme seg fra nåtilstand, tilbake til normaltilstand?
- Hvordan holdes informasjonsflyten i gang? Hvor skal de ansatte møte på jobb?

C.2 Scenario - Skadevare på internnettverket

Det er torsdag 5. juni, og i kontorlandskapet jobbes det iherdig. Tidsfrister må holdes, så lyden av raske, usynkroniserte tastetrykk brer seg iherdig ut over hele rommet. En ansatt i HR-avdelingen merker at datamaskinen hans blir usedvanlig treg og slutter å respondere. Han holder av-knappen inne til maskinen starter på nytt. Datamaskinen starter igjen, men den er like treg. Han klør seg i hodet, det er jo ikke så lenge siden maskinen var ny. En kort samtale til IT-avdelingen senere, fører til at han kan reise hjem for dagen, de har ikke mulighet til å se på det nå. Han blir lovet at en erstatningsmaskin skal være på plass påfølgende dag.

Mandag 9. juni 10:00. To i salgsavdelingen skal holde en glødende presentasjon, men sliter med å få den bærbare maskinen til å spille på lag. De blir nødt til å improvisere på tusjtavla. Etterpå hos IT, blir de avfeid med at det er Windows 8, menyene er ikke helt som før og kommer nok til å kreve litt tilvenning.

Det er onsdag 11 juni 10.35. To på sikkerhetsavdelingen oppdager uavhengig av hverandre ukjent trafikk på internnettverket. De melder umiddelbart i fra til IT, som skjønner alvoret og stenger ned all ekstern kommunikasjon.

11:00. Krisestab er samlet. Hva har hendt? Hva er utført av oppgaver? Hva er planlagt videre?

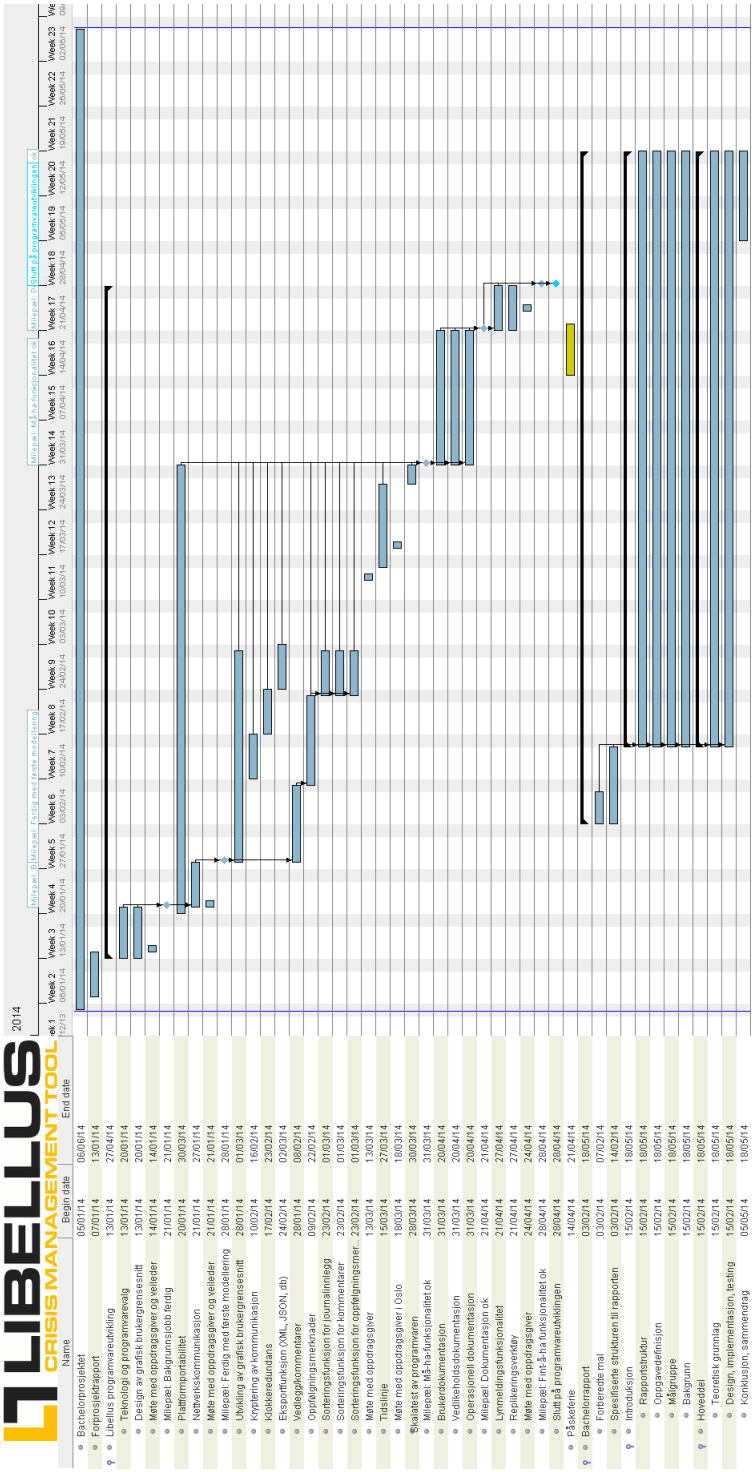
11:15. Det har ikke lykkes i å komme i kontakt med leverandør av sikkerhetsproduktet bedriften benytter.

12:05. Pressekontakten har blitt ringt opp av *Verdens Gang*, som har fått snusen i bedriftens problemer. De er veldig på hugget etter de siste dagenes oppslag om storbanken XBanks problemer med alvorlig skadeprogramvare på internnettverket. Deres bedrift har så langt hverken bekreftet eller avkreftet noen ting. Ryktene har trolig spredd seg gjennom sosiale medier.

Hva skal pressekontakten fortelle media? Hva informeres egne ansatte om?

14:15. VG og Dagbladet publiserer omtrent synkront overskriftene "Monstervirus rammer norske bedrifter"og Norske bedrifter sprer virus!Deres bedrift er ikke nevnt med navn i artiklene.

D Gantt-skjema



E Statusmøter

Vi har for det meste holdt statusmøter med oppdragsgiver v/ Espen Torseth. På disse møtene har vi diskutert fremdrift for utviklingen, utfordringer og endelige avgjørelser.

Når: **11/11-13 12:00-13:00**

- Hvor: HiG
- Hvem: Espen Torseth, Daniel Solstad, Ole Johan Rasch

Først snakket vi litt løst og fast om oppgaven. Deretter kom vi frem til disse hovedpunktene:

- Viktig å definere hva oppgaven ikke omhandler.
- Tidstyv, men også mulig med betraktninger rundt: problematikk med tid, synkronisere tid mellom enheter som er på nett og ikke.
- Hvordan løser andre instanser, som forsvar og politi problematikken. Disse institusjonene har allerede fungerende infrastruktur på plass, gjerne også med redundans.

Vi diskuterte også et møte med Conax i Oslo i løpet av desember, der agendaen vil bli:

- Innspill fra de andre i Conax som driver med sikkerhet.
- Tanker rundt funksjonaliteten til et journalføringsverktøy.

Når: **17/12-13 11:00-12:00**

- Hvor: Oslo
- Hvem: Torstein Gleditsch, Erik Abrahamsen, Jeanne Hammer Espedalen, Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Vi viste frem nettsiden og skisser på hvordan vi så for oss det grafiske grensesnittet til programmet.
- Følgende funksjonalitet ble diskutert: todoliste med tidsfrister, predefinerte valg / case, Innlinking, kilder, vedlegg, rettigheter/deling som blir satt etter oppstart.
- Vi snakket om hvor vesentlig det er å få loggført hvem som starter og avslutter en sak eller et journalinnlegg.
- Lokal deling mellom enheter er viktigere enn synkronisering mot andre lokasjoner. Kan ta med tekniske vurderinger, eksisterende programvare på dette (VPN).
- Ferdige maler for tiltak som måtte iverksettes for ulike kriser.
- Mulighet for å eksportere og importere data i ulike format, blant annet PDF.

Når: **14/1-14 1400-1600**

- Hvor: HiG
- Hvem: Thomas Kemmerich, Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Ble kjent, og snakket litt løst og fast om oppgaven
- Diskuterte tidsfrister og videre oppfølging.
- Thomas sa at han ville ha mest å bidra på i rapporten og ikke det tekniske.
- Vi skriver forprosjektet på engelsk og rapporten på norsk.

Når: **21/1-14 1400-1600**

- Hvor: HiG
- Hvem: Thomas Kemmerich, Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Vi gjennomgikk forprosjektrapporten.
- Gannt-skjemaet trengte forbedring i form av milepæler underveis i prosjektet.

Når: **13/3-14 1200-1400**

- Hvor: HiG
- Hvem: Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Kontraktene var på vei
- Vi gikk gjennom hele Gantt-skjemaet.
- Ble enige om å ikke eksportere til PDF, men istedenfor XML, for så å bruke andre verktøy til å generere PDF eller andre formater. Dette er utenfor oppgaven.
- Oppdragsgiver ville ha snarveier i programmet for å gi mer flyt.
- Det var ønskelig med nummerering av innlegg.
- Brukerdokumentasjon må være så enkel at en som har brukt en pc før og som kan lese er i stand til å få nytte av den.
- Operasjonell dokumentasjon må være en guide til hvordan organisasjonen må forme sine operasjonelle rutiner, slik at disse passer sammen med systemet.
- Finne opp scenario som kan brukes i testing.
- Skrive hvordan stabsarbeidet bør organiseres rundt systemet.

Når: **18/3-14 1200-1400**

- Hvor: Oslo
- Hvem: Torstein Gleditsch, Erik Abrahamsen, Espen Torseth, Daniel Solstad, Ole Johan Rasch

Her viste vi frem en demo av hele programmet mens vi forklarte rundt. Oppdragsgiver ønsket seg videre:

- Mulighet for å legge til en status for oppfølgningsmerknader (actions), som tilsa at saken var på vent.
- Egen dedikert funksjon for å loggføre tilstedeværende personell.
- Snarvei fra journalen til actions og vice versa.
- Det ble bestemt å ikke satse på Ubuntu og OS X med tanke på portabilitet.
- To klokker, der den ene er i den lokale tidsonen.

Når: **4/4-14 1300-1500**

- Hvor: HiG
- Hvem: Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Enighet om at utviklingen snart skulle avsluttes for å prioritere rapporten.
- Oppdragsgiver er veldig fornøyd med programvaren.

Når: **24/4-14 1200-1400**

- Hvor: HiG
- Hvem: Espen Torseth, Daniel Solstad, Ole Johan Rasch
- Få mer frem hva som er nytt og orginalt med løsnigen.
- Snakket om kriseøvelsen Conax skulle ha, og at de trengte en versjon med programvaren.

F Tjenesteoppdagelsesskript

```
# Code by: Libellus
# http://libellus.no
# Based on multicast_example.py
# https://gist.github.com/aaroncohen/4630685
# Written by Aaron Cohen -- 1/14/2013
#
# This work is licensed under a Creative Commons Attribution 3.0 Unported License.
# See: http://creativecommons.org/licenses/by/3.0/

import sys
import re
import socket
import time
import json
import urllib2
import httplib
from threading import Thread

def extract_ips(ip_data):
    # Extract IP addresses from getaddrinfo using regex, while skipping loopback
    regex = "^(?!127)(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
    return re.match(regex, ip_data) is not None

def get_local_ips():
    # socket.getaddrinfo returns much information, we only need the IPs
    local_ips = [ x[4][0] for x in socket.getaddrinfo(socket.gethostname(), 80)
                  if extract_ips(x[4][0]) ]

    # If no addresses are found, try another way by opening a connection to a known server
    if not local_ips:
        # create a standard UDP socket ( SOCK_DGRAM is UDP, SOCK_STREAM is TCP )
        temp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            # Open a connection to one of Googles DNS servers
            temp_socket.connect(('8.8.8.8', 9))
            # Get the interface used by the socket
            local_ips = [temp_socket.getsockname()[0]]
        except socket.error:
            pass
        finally:
            # Close socket when done
            temp_socket.close()

    # If no IP addresses are found, wait for 60 seconds and try again
    if not local_ips:
        time.sleep(60)
        get_local_ips()
    return local_ips


def create_socket(multicast_ip, port, local_ip):
    # create a UDP socket
    my_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # allow reuse of addresses
    my_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
57
58      # set multicast interface to local_ip
59      my_socket.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_IF, socket.
            inet_aton(local_ip))
60
61      # Set multicast time-to-live to 1
62      # This is to stop data from escaping the local network
63      my_socket.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 1)
64
65      # Construct a membership request...tells router what multicast group we want
            to subscribe to
66      membership_request = socket.inet_aton(multicast_ip) + socket.inet_aton(
            local_ip)
67
68      # Send add membership request to socket
69      # See http://www.tldp.org/HOWTO/Multicast-HOWTO-6.html for explanation of
            sockopts
70      my_socket.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP,
            membership_request)
71
72      # Bind the socket to an interface.
73      # If you bind to a specific interface on the Mac or Linux, no multicast data
            will arrive
74      # If you try to bind to all interfaces on Windows, no multicast data will
            arrive
75      if sys.platform.startswith("darwin") or sys.platform.startswith("linux"):
76          my_socket.bind(('0.0.0.0', port))
77      else:
78          my_socket.bind((local_ip, port))
79      return my_socket
80
81  def listen_loop(multicast_ip, port):
82      # Get all IP addresses assigned to the local network interfaces
83      local_ips = get_local_ips()
84
85      # Start a new listen thread for each IP
86      for local_ip in local_ips:
87          my_socket = create_socket(multicast_ip, port, local_ip)
88          t = Thread(target=listen_content, args=(local_ip, my_socket,))
89          t.start()
90
91  def current_milli_time():
92      return int(round(time.time() * 1000))
93
94  def listen_content(local_ip, my_socket):
95
96      # Set a timestamp for startup
97      timer = current_milli_time()
98      # Adds the local IP address to the array
99      # This is done to prevent communication with yourself
100     #   in case of several network cards
101     added_array = [local_ip]
102
103     while True:
104         data, address = my_socket.recvfrom(4096)
105
106         # If the found IP address is not in the array
107         if data not in added_array:
108             # Add to array
109             added_array.extend([data])
110             # Send replication data (IP) to the local CouchDB instance
111             json_send(data)
112
113         # Empty the array after some time
114         #   to make it possible to use the "clean replication"
115         #   button in Libellus Utils and getting new data
116         if (current_milli_time() - 120000) < timer:
```

```
117             try_connect()
118             del added_array[:]
119             added_array = [local_ip]
120             timer = current_milli_time()
121 
122 def json_send(target):
123     # URL for the local CouchDB instance
124     url = 'https://127.0.0.1:6984/_replicator'
125     # Databases to replicate
126     databases = ['journal','chat','chat_users']
127 
128 
129     # JSON data
130     for database in databases:
131         receiving_data = {
132         "_id":"receiving_"+database+"_from_"+target+"",
133         "source":"https://libellususer:libellususer@"+target+":6984/"+database+"
                ",
134         "target":"https://libellususer:libellususer@127.0.0.1:6984/"+database+""
                ,
135         "continuous": True,
136         "create_target": True,
137         "user_ctx":{
138             "name":"libellususer",
139             "roles":[]
140             }
141         }
142 
143         sending_data = {
144         "_id":"sending_"+database+"_to_"+target+"",
145         "source": "https://libellususer:libellususer@127.0.0.1:6984/"+database+"
                ",
146         "target": "https://libellususer:libellususer@"+target+":6984/"+database+
                "",
147         "continuous": True,
148         "create_target": True,
149         "user_ctx": {
150             "name": "libellususer",
151             "roles": []
152             }
153         }
154 
155         # Send data for each database both Tx/Rx
156         json_sender(receiving_data, url)
157         json_sender(sending_data, url)
158 
159 
160 def json_sender(data_type, url):
161     # Create a request and send the JSON data
162     req = urllib2.Request(url)
163     req.add_header('Content-Type', 'application/json')
164     req.get_method = lambda: 'POST'
165 
166     try:
167         response = urllib2.urlopen(req, json.dumps(data_type))
168         response.close()
169     # Handle error if entry already exists in database
170     except urllib2.HTTPError as e:
171         pass
172     # Handle connection error, exit script
173     # This will occur if CouchDB is not running
174     except IOError:
175         sys.exit(0)
176 
177 def try_connect():
178     # Check if local Libellus is running, else close the script
179     try:
```

```
180         connection = httplib.HTTPSConnection('127.0.0.1', 6984, timeout=2)
181         connection.connect()
182         connection.close()
183     except:
184         sys.exit(0)
185
186 def announce_loop(multicast_ip, port):
187     # Offset the port by one so that we can send and receive on the same machine
188     port+1
189
190     # Get all IP addresses assigned to the local network interfaces
191     local_ips = get_local_ips()
192
193     # Start a new announce thread for each IP
194     for local_ip in local_ips:
195         my_socket = create_socket(multicast_ip, port, local_ip)
196         t = Thread(target=announce_content, args=(local_ip, my_socket,
                multicast_ip, port,))
197         t.start()
198
199 def announce_content(local_ip, my_socket, multicast_ip, port):
200     while True:
201         # Send data. Destination must be a tuple containing the IP and port.
202         my_socket.sendto(local_ip, (multicast_ip, port))
203         time.sleep(5)
204         try_connect()
205
206 if __name__ == '__main__':
207     # Choose an arbitrary multicast IP and port.
208     # 239.255.0.0 - 239.255.255.255 are for local network multicast use.
209     # Remember, you subscribe to a multicast IP, not a port. All data from all
            ports
210     # sent to that multicast IP will be echoed to any subscribed machine.
211     multicast_address = "239.255.133.7"
212     multicast_port = 6980
213
214     # Starts a thread for the listening interface
215     listen = Thread(target=listen_loop, args=(multicast_address, multicast_port
            ,))
216     listen.start()
217
218     # Starts a thread for the announce interface
219     announce = Thread(target=announce_loop, args=(multicast_address,
            multicast_port,))
220     announce.start()
```

Kodesnutt F.1: libellus_multicast.py

Libellus Lessons Learned – Crisis Exercise May 2014

Friday May 9th 2014 Conax ran a crisis management exercise. This exercise had two goals:

- Test the Libellus crisis management tool
- Identify improvement areas for Conax business continuity plans

This report will focus on Conax' lessons learned with regards to the Libellus tool. The exercise was organized as a table top exercise since the goals didn't demand a full scale execution. Our exercise was run by the crisis manager with two secretaries using Libellus. In addition representatives from IT, Operations, Support, Security, Development, Human Resources and Facility Management participated. Finally there was an exercise leader to ensure that the exercise itself ran smoothly.

Lessons learned about Libellus were in the areas:

- Learning the tool
- Running the tool
- Using the tool
- Presentation with projector
- Post exercise analysis

Learning the tool

Three days before the crisis management exercise on May 9th we had a one hour training session on Libellus. The goal was to simulate an approximation of the expected familiarity with Libellus prior to a crisis.

There were no major obstacles identified in teaching the tool to people with normal IT competence.

Running the tool

The only issue we had running the tool was that the menus didn't appear in Internet Explorer 8 on a freshly installed Windows 7 machine. Since Libellus uses jQuery 2.x which doesn't support Internet Explorer 8, this should be documented.

Another option is that organizations using Libellus can be recommended to install a portable browser on the same stick as Libellus. That way they'll always have a compatible browser at hand in a crisis situation.

Using the tool

The tool itself was easy to use. We tested primarily the "Journal" and "Action" parts. "Chat" was not applicable for this exercise and time didn't permit testing "Visualize".

No major usability obstacles were identified for the crisis secretaries. Their feedback mainly focused on that they wanted better visibility of the event subject and descriptions, action descriptions, and better default values where applicable. In addition the crisis secretaries wanted a way to mark that a journal entry should be disregarded, e.g. because it contains wrong information or other errors.

Most of the crisis secretaries concerns where similarly described by the other participants. We didn't have the time to let all participants test the tools themselves, so their feedback is based on what they

were shown on the projector. They also wanted more visibility on the event subject and content and action descriptions.

With regards to visibility of the suggestions can be summarized as:

- Less emphasis on the time fields. Ideally just one or a maximum of two
- Other fields than “Subject” and the actual content should be either in a narrower font, grey font, or a combination of these
- Fields should be vertically adjusted to the top of the cell/line rather than the middle

The crisis secretaries suggested that a note that was made in error should be easy to mark as such, e.g. by a ~~strikethrough~~ or ~~strikethrough~~ combined with grey font. This way the integrity of the journal is intact and all members of the crisis team easily can see what entries to disregard.

Presentation with projector

We informally tested presentation of the journal and action with a projector (1366x768). The crisis team gave in essence the same feedback as the crisis secretaries. Mainly that the metadata, i.e. dates and ids, take focus away from the main information elements.

They also noted that it was somewhat distracting when they viewed the same screen as a crisis secretary was working on. Ideally they wanted a specific “presentation” view that was optimized for team use on a projector.

Post exercise analysis

We’ve not finished the post exercise analysis yet, but have had no issues with the export functionality. The crisis manager and secretaries all noted that ideally it should be possible to export the data in a readable and printable form as PDF files.

Conclusion

Libellus is in its current form:

- Easy to learn
- Easy to use
- Easy to get running
    - Provided that the documentation is updated
- Gives better tracking of events and information in a crisis than the typical methods based on Word / Excel templates etc.
- Enables better post crisis analysis since it provides a better timeline than the typical Word / Excel templates

Libellus will be improved by:

- Some adjustments in the user interface to emphasize information over metadata
- Default values were applicable, e.g. event time equal to current time
- The possibility to mark entries that should be disregarded
- A “view” optimized for team use on projectors

Future features wanted by Conax:

- A possibility to store business continuity plans and crisis management plans
- A possibility to store predefined procedures for specified events
- A possibility to store check lists
- A possibility to create prioritized lists of critical information requirements
- Better graphical representation of events, timelines, etc.

Since Libellus is implemented as a CouchApp on CoucDB there are no platform specific limitations that prohibit such features, with the possible exception of graphical representations. The challenge is to make them fit seamlessly with the core Libellus features. This project has also shown that there is need for more research and development of good graphical representation of events, actions, etc.

Libellus, as any other tool, does not remove the need for crisis management training. But it improves the potential for more systematic analysis of the results. Especially by automating logging both the event time and the time the event was recorded. Automated logging of event and recording time will rarely be done with typical Word / Excel templates. This enables a better understanding of what information was available at which time regardless of when the event happened. Such analysis will depend on the quality of journal notes etc. This can only be achieved by training, but this applies equally to any tool.

In short; Libellus delivers all the basic features needed for crisis management with minimal infrastructure requirements as discussed in the original project proposal.

Libellus user documentation

Release 1.0

Libellus.no

May 07, 2014

CHAPTER

ONE

INTRODUCTION

This user documentation shows you the basics about how to use Libellus.

CHAPTER

TWO

USER MANUAL

2.1 Start Libellus

To start Libellus, find the file on the USB-drive with the name matching your operating system. If you are running Microsoft Windows 7 or 8, the name of the file will be *windows.bat*.

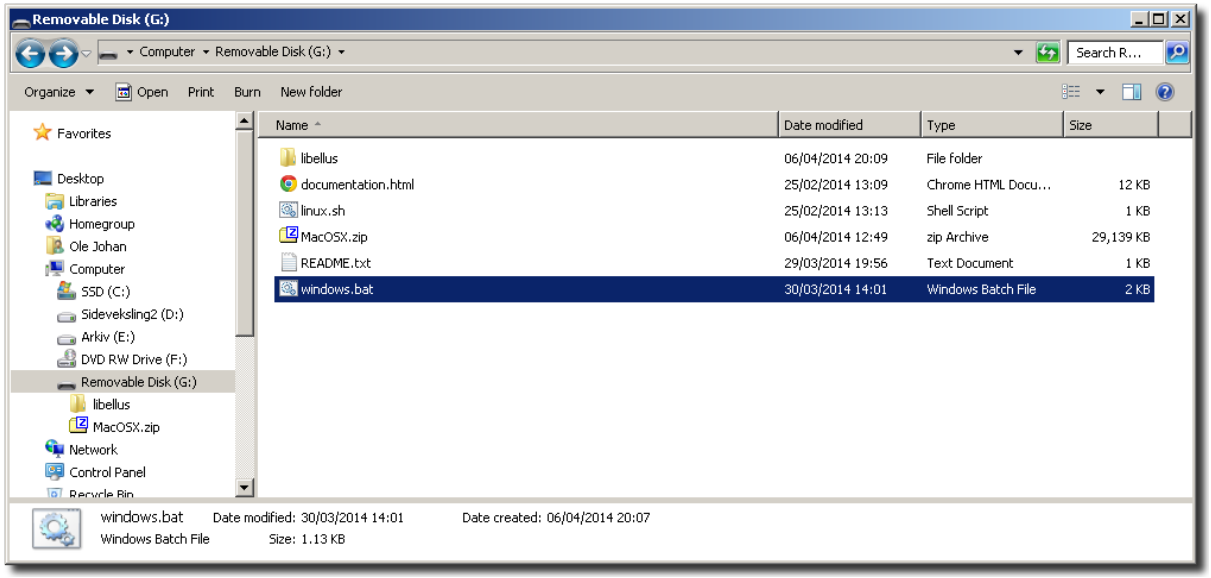


Figure 2.1: This window shows the folder structure on a windows-machine.

This file is a startup script that will perform some processes in the background to get Libellus up and running. All the actions will be run from a console window.

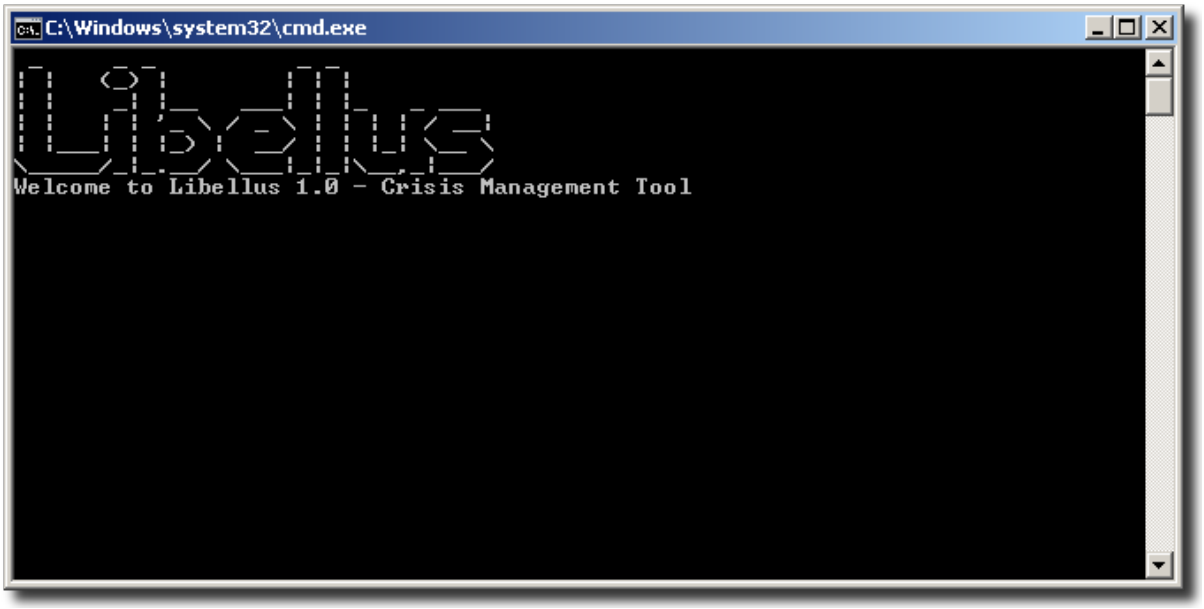
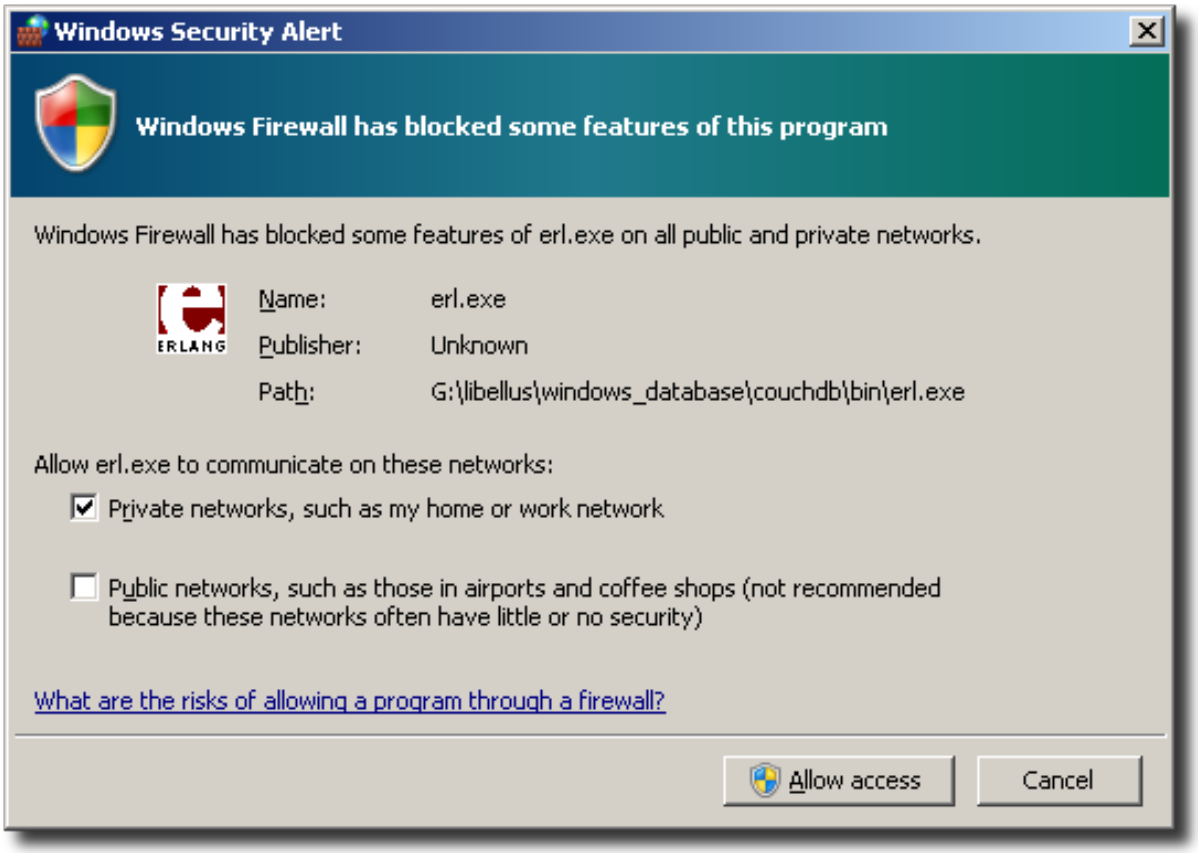
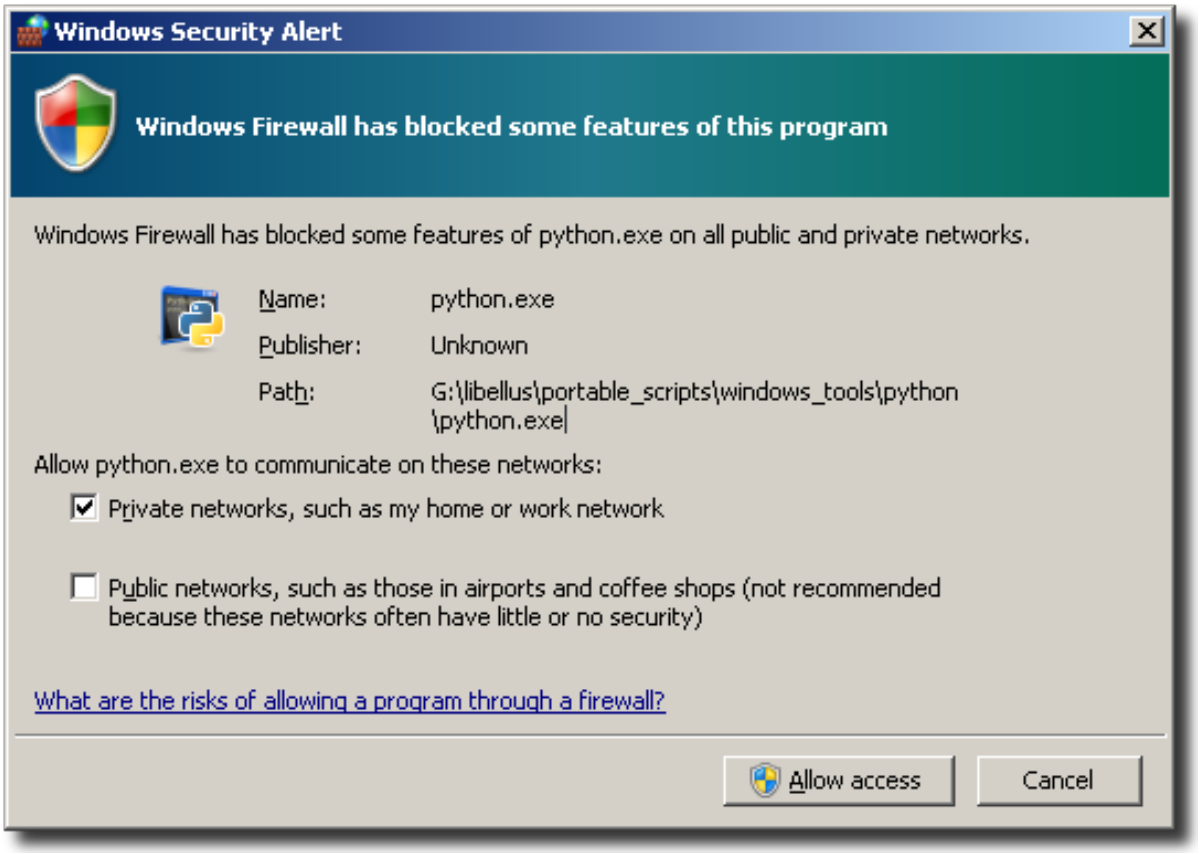


Figure 2.2: This is the typical console-window. Here shown on a computer running Microsoft Windows.

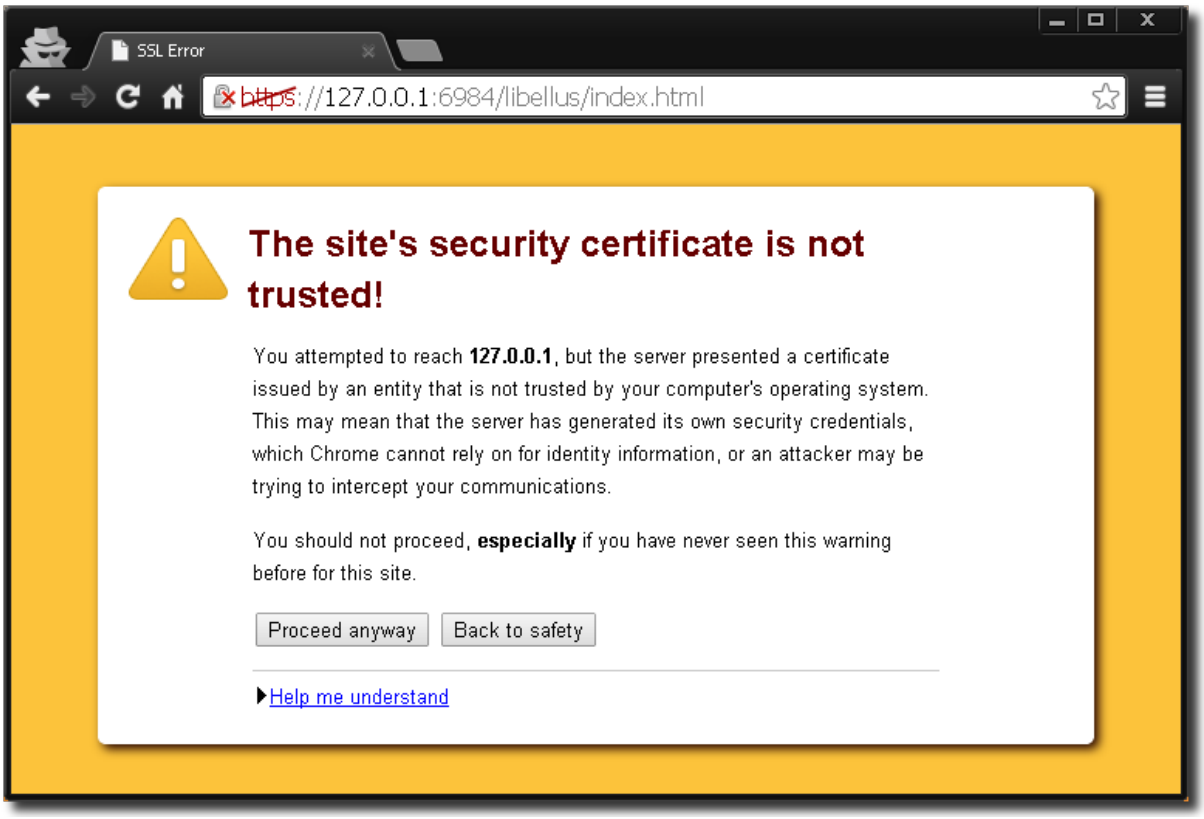
Normally all user accounts can run this file, but if the script senses that it needs elevation, you will be prompted with a message telling you how to do this.

Libellus will be needing access to the network for communication with other running instances of the software. To enable this, click allow access in the two dialog boxes asking for permissions.


Windows Security Alert
Windows Firewall has blocked some features of this program
Windows Firewall has blocked some features of erl.exe on all public and private networks.
Name: erl.exe
Publisher: Unknown
Path: G:\libellus\windows_database\couchdb\bin\erl.exe
Allow erl.exe to communicate on these networks:
Private networks, such as my home or work network
Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)
What are the risks of allowing a program through a firewall?
Allow access
Cancel

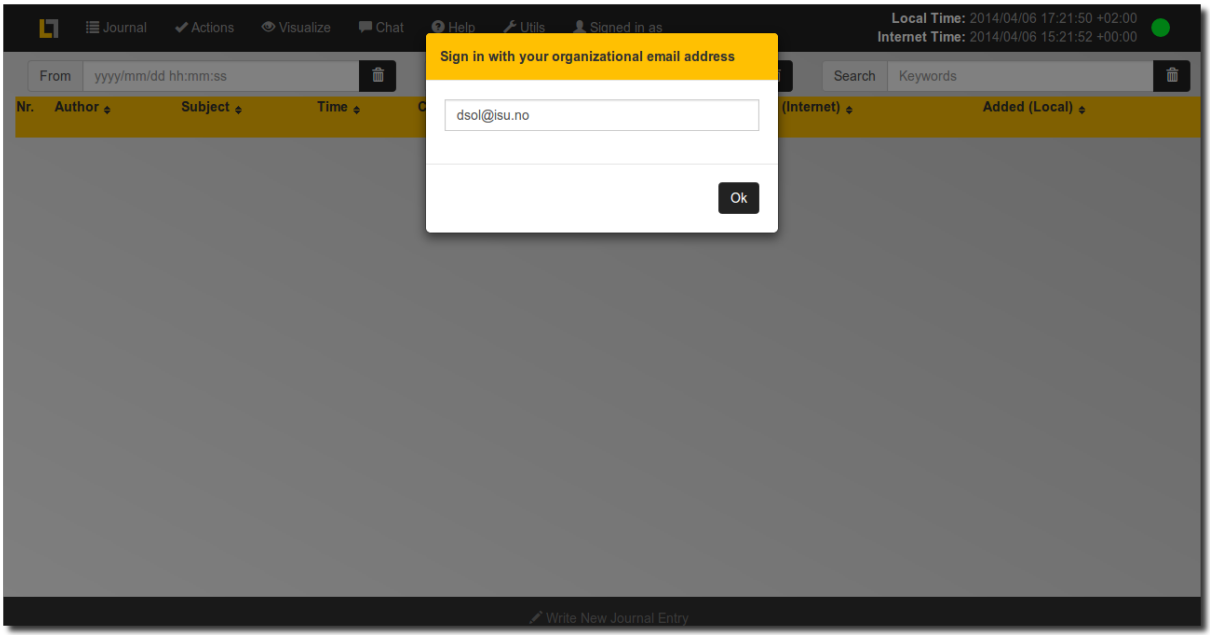


After some seconds, your default browser will open and be directed at the localhost address (https://127.0.0.1:6984/libellus/index.html) All communication in Libellus is transferred encrypted using the TLS technology. To enable this to run on a local computer, Libellus uses a self-signed certificate. This will make your browser display a warning message like the one below. Click on *Proceed anyway* to continue.



2.2 Login

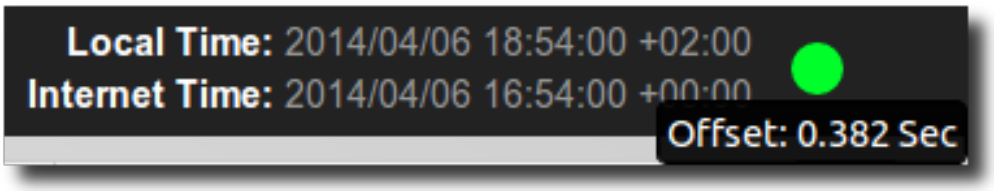
After the startup process, the following screen is the first you will meet.



The only thing you need to enter before you can use the program is a username. There is no users registered beforehand, which makes this the registration. The username can be whatever you want it to be, but we suggest to use something that identifies you, as well to be unique. Your first name will not be a good idea, because there are others with the same first name. We recommend your organizational email address, because it's globally unique and will usually identify you by the screen name. If you need to change the username, this can be done by clicking on the *Signed in as <username>* on the menu.

2.3 Clock

In the upper right corner, you will see two datetimes. Local Time is the time taken from your computer, which is presented in the local timezone the computer is set to. Internet Time is a time taken from an external source on the Internet. This will always be presented in UTC plus 0 hours. If the Local Time and Internet Time is more or less in sync, the circle on the right side will be green. If both clocks are more than 3 seconds apart, the circle will be orange. If Libellus doesn't have access to the Internet, the green circle will be red. You can also hold your mouse over the circle and it will show you the offset between both times.

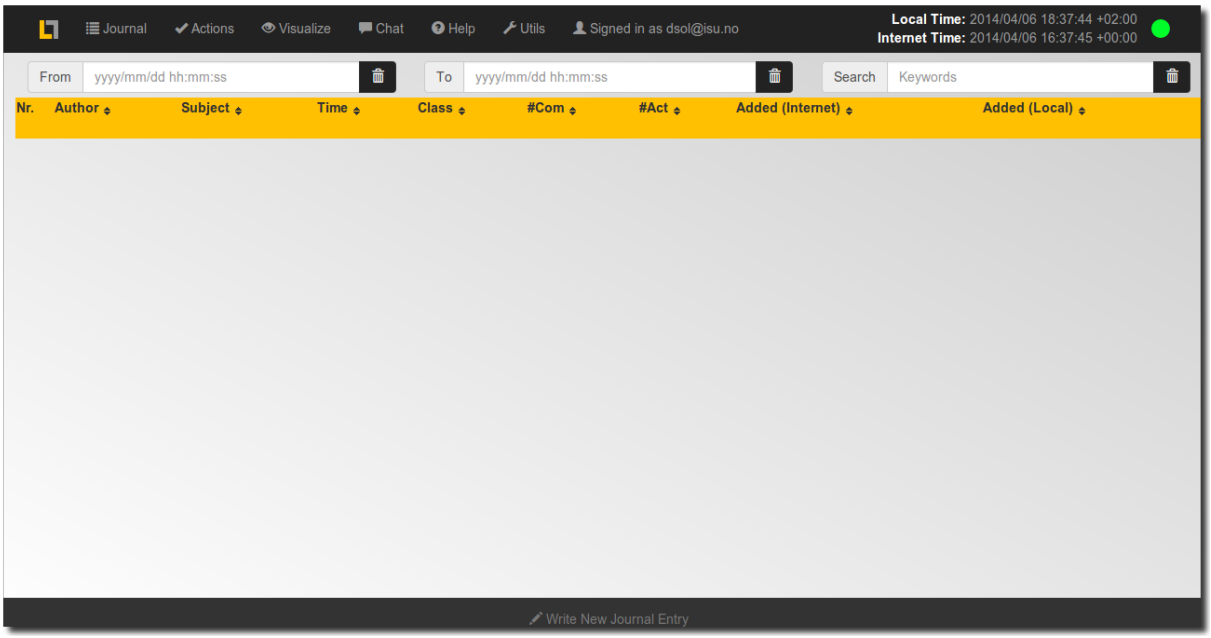


If the clocks are not in sync, we recommend that you change your computer time manually, or that you update it automatically. You can't do this from Libellus, but most operating systems support it. We suggest that all clients on the network have the same local time.

Every entry will be stored with local time, and if Libellus has access to Internet time, it will also store this. Even if you start to log without Internet time, but get access later, Libellus will calculate the offset and update every entry and add Internet time. As another method to ensure integrity, Libellus will log every time the client change the computer time.

2.4 The Journal View

The default screen is the journal view, which is probably where you will spend the most time. Right under the main menu, you can see some filtering options, which we will cover later, when we got some data to apply filters to.



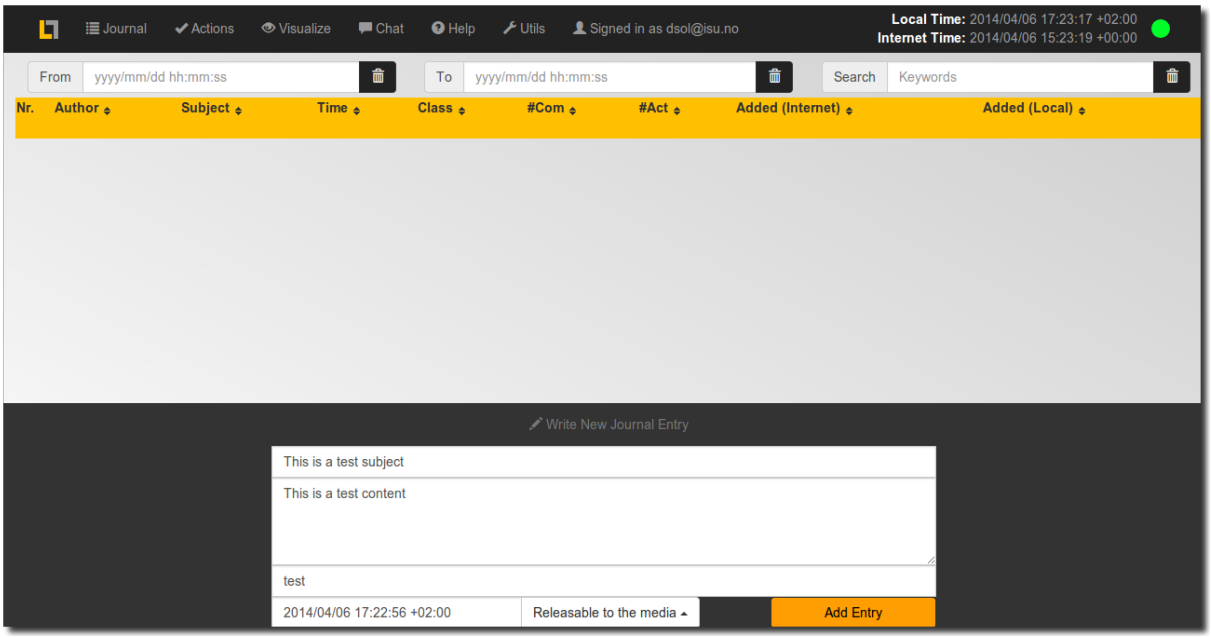
Under the filtering options, you see some headers for the table. Here is the explanations for all the columns

- **Nr.** - This is a running number to have something to refer to.
- **Author** - This is the user that made the journal entry
- **Subject** - The subject of the entry.
- **Time** - This is the happening time, when the event took place
- **Class** - This is the classification of the entry. Hold your pointer over the text to show the full content.
- **#Com** - This is number of comments to the journal entry
- **#Act** - This is the number of actions to the journal entry
- **Added (Internet)** - This is the time taken from an external source on the Internet, when the entry was stored into the Libellus client. This may be displaying N/A if the client didn't have access to the Internet when the entry was stored.
- **Added (Local)** - This is the time taken from the clients computer when the entry was stored in the the database

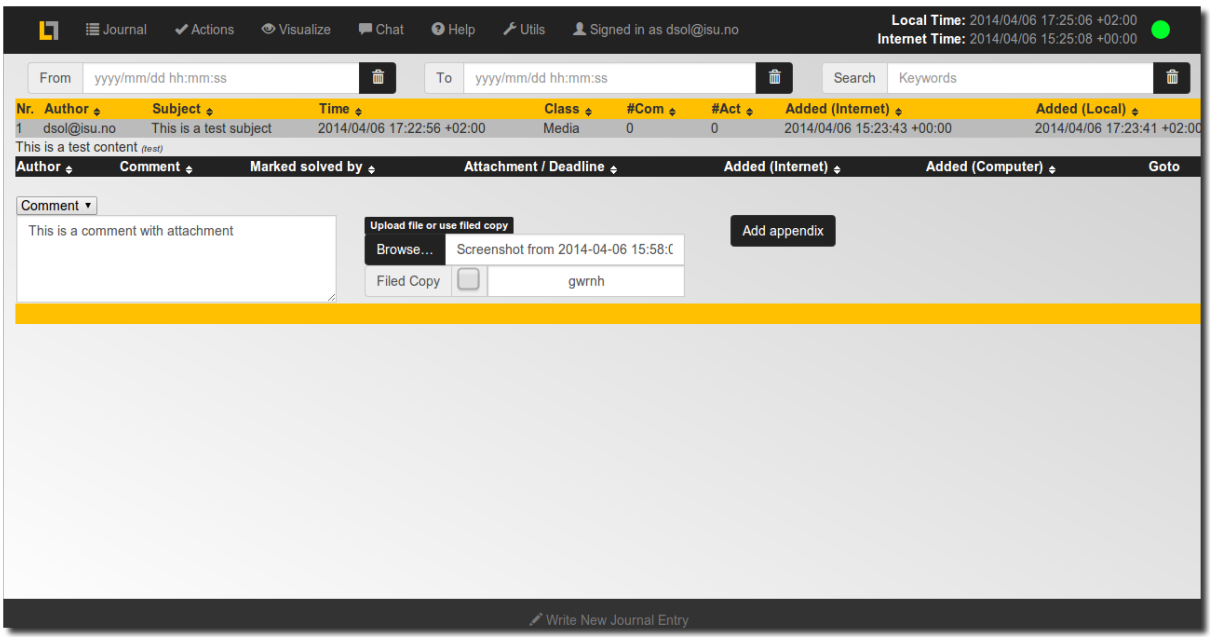
The last thing you may have noticed is the footer at the bottom with the caption Write New Journal Entry, which will be covered next.

2.5 Write New Journal Entry

When you have clicked on the *Write New Journal Entry* bar, a form will appear. The subject should explain in short what the entry is about and content should contain the full story of the event. Keywords is a free text input field, where you can write words to associate the entry with. This is a feature to make it easier to search for entries later. *Happening time* should be in the past, when the event you are writing about took place. This should not be confused with *Added (Internet)* or *Added (Local)*, which is when the journal entry was added. The drop down menu with predefined classification values should describe who can know about the information you are writing about. Libellus does not handle classification different, but the organization that use Libellus might.



When you have filled in the values and clicked on *Add Entry*, you can close the form by clicking on it again. You can now see that the entry is displayed in the journal, and you can expand it by simply clicking on it. Now your screen should look something like the following.



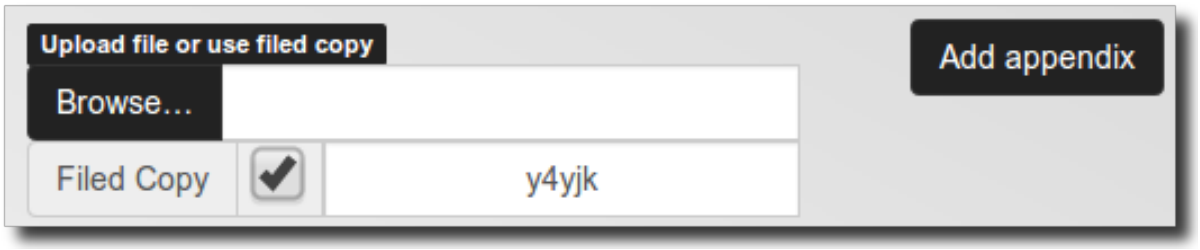
Notice that you can find the content and keywords under the row with metadata.

2.6 Appendices

As you can see in the previous image, there is a form inside the journal table. This is where you can add appendices, i.e. comments and actions. The first drop down menu you see is where you choose whether to create a comment or action. The difference of these alternatives, is that a comment is to add information to the journal entry and actions is anything that needs to be done in relations to the journal entry.

2.6.1 Comments

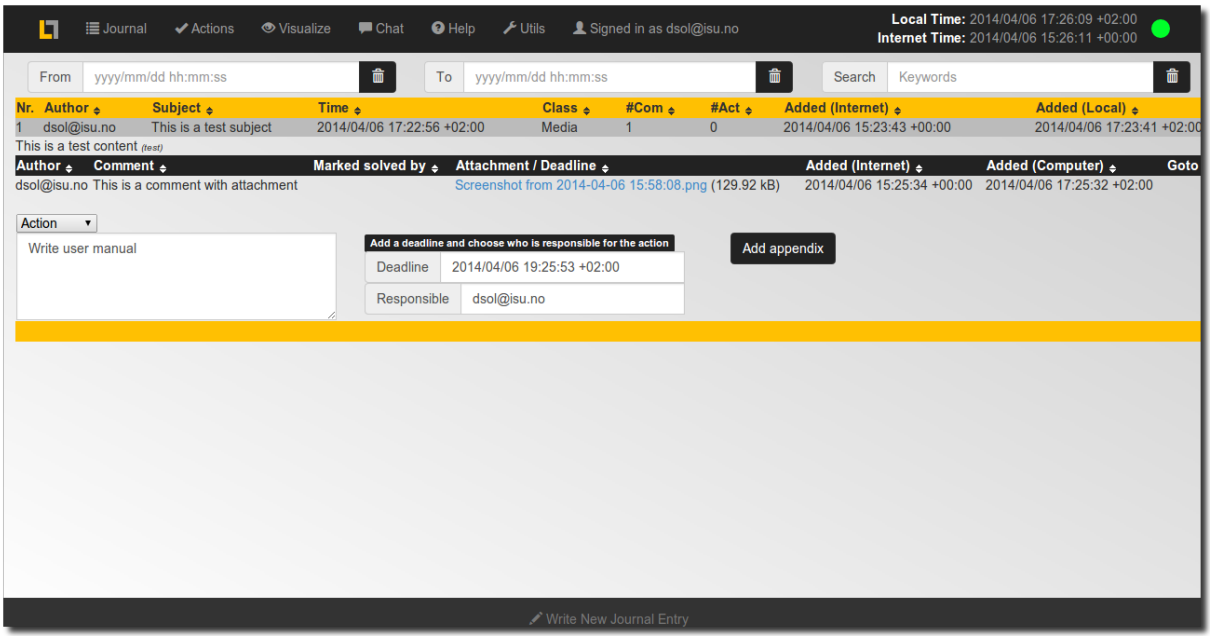
When creating a comment, you have two choices of the type of attachment. The first is to upload a digital file. Just hit the browse button and a file manager will appear, where you can find your file, but be careful to not upload large files. This is because the file may be replicated amongst many clients and take up much bandwidth. The default max file size is set to 10 megabytes. The second alternative is filed copy.



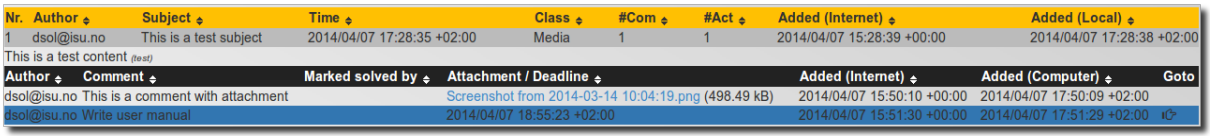
Let's say you have a physical paper, hard drive, CD or basically something that you can't upload, or is too big to be uploaded, but you want the journal to keep a record of. Filed copy makes you do this, by a random generated reference string. Check the filed copy checkbox and write the string on the physical object, or maybe use a post-it note on it, before you hit the submit button.

2.6.2 Actions

To add an action instead of a comment, just change the value on the drop down box. Now you will see some other input fields. Here you can write a deadline to when the action needs to be complete. When clicking in the field, it will get a default value, which is the current date and time, but should be set to a time in the future. The second new input field is who is responsible for the task. This could be his screen name on Libellus or his real name, but it's up to the organization to choose what to use.



When the action is added, you will see that the entry like in the image below. The row is highlighted in blue, because the deadline is over an hour due in the future. The color will change to orange when the deadline is less than an hour and turn red when the time limit is in the past. The row will be green when the action is solved, and get a strike through when rejected, which we will get to later.



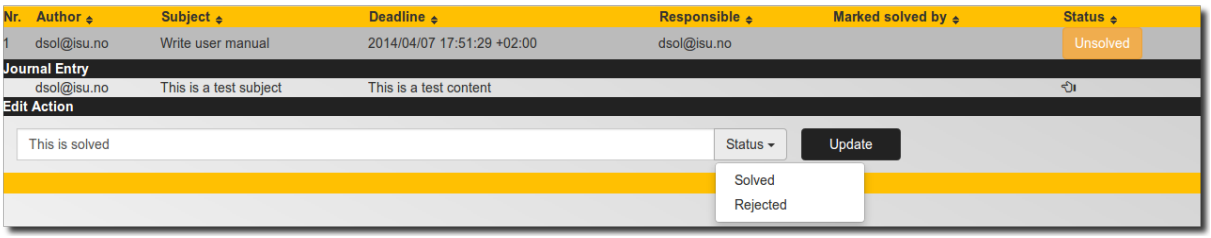
There is another feature that comments doesn't have, which is a goto link on the right, depicted with a pointing finger. When clicking on it, you will be directed to the *Actions View* and the target action entry will be expanded.

2.7 Sorting

You can sort every table by clicking on the headers. Click once for descending and twice for ascending sorting. You can also sort by multiple values by holding down the shift key while clicking.

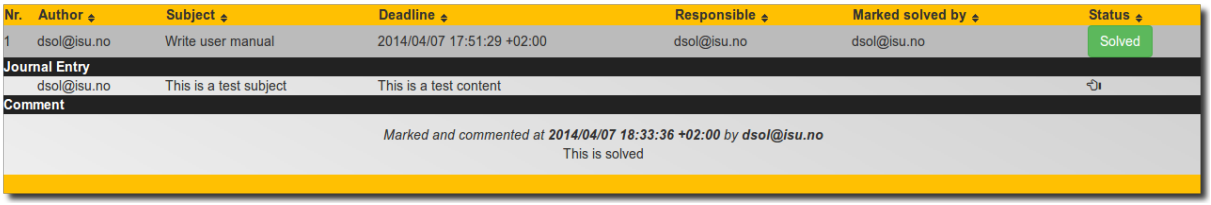
2.8 The Actions View

You can get to the action page by either clicking on Actions in the main menu, or click goto on the inline action entry. The first thing you can notice, is the button on the right, which has changed color to orange in the time between. This button will like the row on the journal page, change color according to the time left for the deadline. The caption on the button now displays *Unsolved*, naturally because no one has solved it yet. The metadata is just like before, but with one exception, that is *Marked solved by*, which is who updated the status of the entry.



We can also see the complete journal entry that the action are referring to. This is because you can read it and instantly know which journal entry the action are referring to. You also have a goto button here on the right side, which will take you back to the target entry in the journal view.

If an action is solved, just write a comment and select *Solved* in the drop down menu. Now this is what you will see. The status button now displays *Solved* with a green background. The comment and who marked it also are shown.

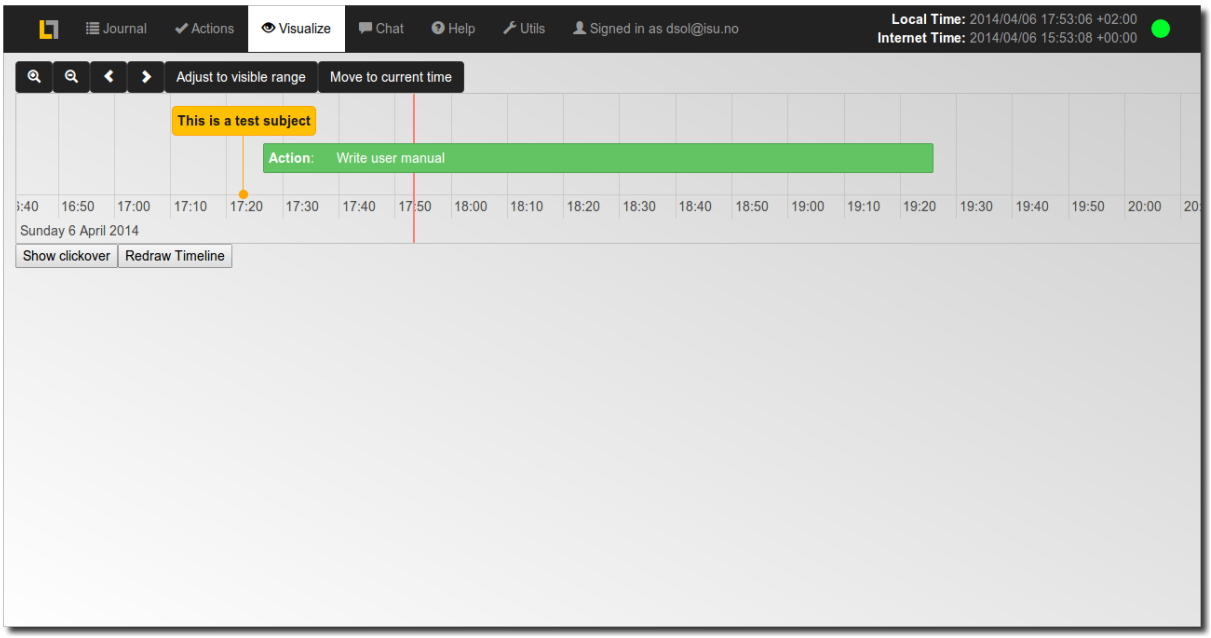


The other alternative to an action is to reject it. This can be useful if the action is not possible to complete or that the deadline is too strict. The image below illustrates how this looks like.

Note that once an action is marked it can not be edited later.

2.9 The Visualize View

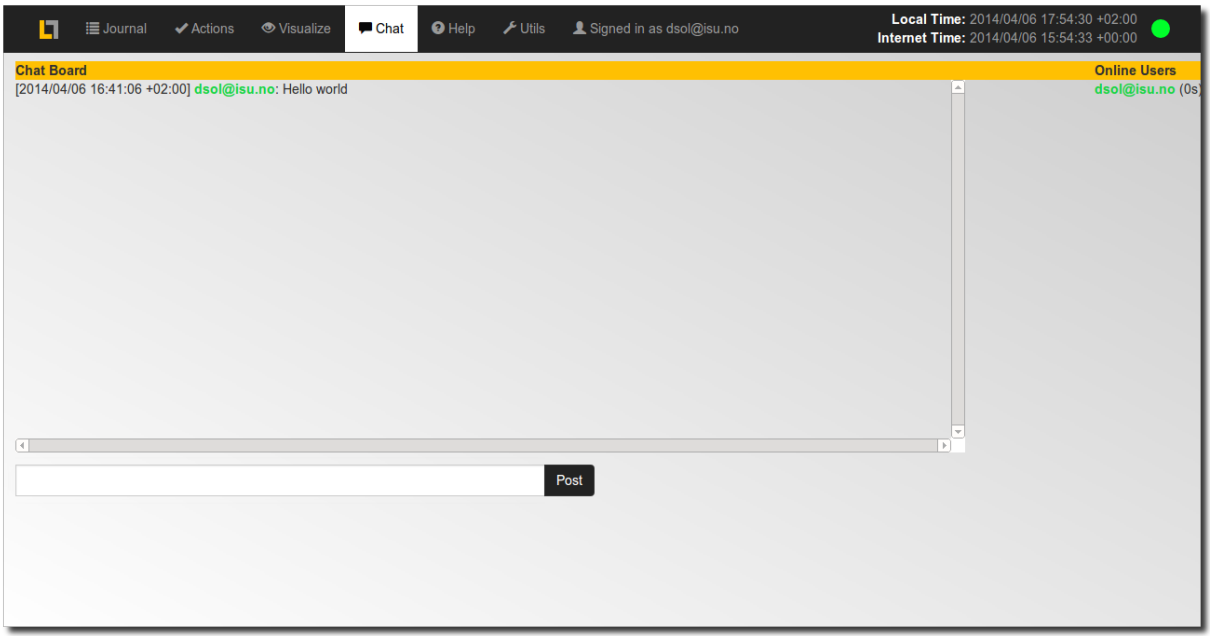
The visualize view shows the journal information on a different perspective. In the example below, you can see a journal entry and an action. Actions will follow the same coloring as they do on the other views, and stretch in length according to the time frame given by the deadline. Use the buttons to navigate in the timeline, or hold and move your mouse in the direction you want to expand the timeline to left or right. You can also zoom in and out with the scroll wheel on your mouse. As you can see, there is a red vertical line in the timeline. This is to mark the current time, which you can always move instantly to by clicking on the button with the caption *Move to current time*.



The visualization view is very much under development, and will most likely get an upgrade in the future.

2.10 The Chat View

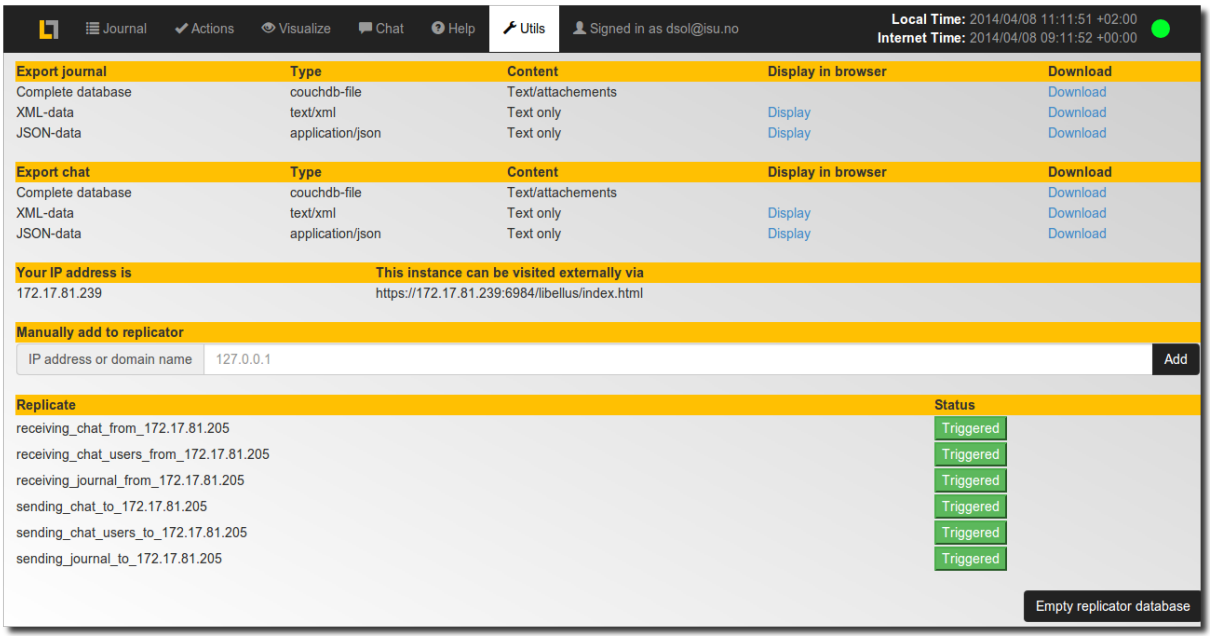
The chat is very simple in design and features, by which the only feature is to write to all. On the right side you have a list that displays all online users, with a number in parentheses to indicate how much latency they got. i.e. how much time since they were online. If a user have zero in latency, that doesn't need to mean they are online in the chat room, because the latency will update with a frequent interval as long as they have Libellus running.



The chat is meant to be used for anything that is not important enough to become an independant journal entry, but can really be used to anything. For instance a bulletin board to keep track of who is available or not. Keep in mind that everything you write in the chat will be logged and synchronized with other Libellus instances on the network. The last thing to know is that when you click on the input field, or hit the submit button, you will automaticly be scrolled down to the bottom of the chat history.

2.11 The Utils View

This is a view where you can do multiple tasks, like exporting data and view replication status.



2.11.1 Export

The first section you see is for exporting. Here you can extract the journal and the chat database. You can download the complete and raw database file, which you can save and add back into Libellus later. View the maintenance guide for an explanation for how to do this. If you export as XML or JSON, you don't get attachments and it's not possible to import these formats into the database again after exporting. This can be used with import to other software.

2.11.2 Visit externally

The section in the middle of the screen is where you can see your own IP address and find the URL for your running instance of Libellus. You can use this to visit your instance externally on other devices, like smartphones or tablets.

2.11.3 Add replication

At startup Libellus will detect other running instances of the software and add them to the replication automatically. But in some cases it may be useful to manually set up replication to other instance. This can for example be an instance on the Internet which may connect multiple local networks.

2.11.4 View status and remove replications

On the bottom you can see the status for all the replications. If the status shows a green box with the caption *Triggered*, like in the image above, everything is as it should. If you see a red box with the word *Error* then something is wrong. This will most likely be the cause of other clients going offline.

Finally you can remove all replications by clicking on the button at the bottom right corner.

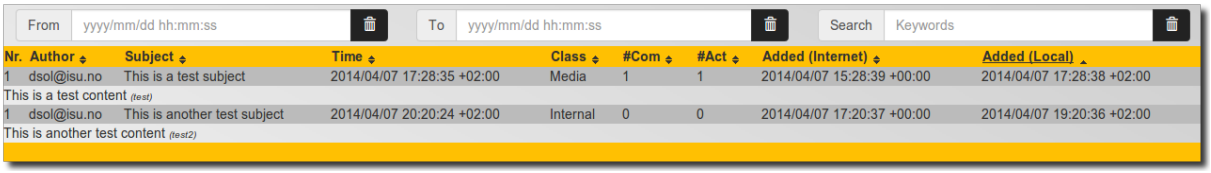
2.12 Filters

The journal view has filters that can be applied to remove everything you don't want to see. There is two possible ways to filter out data, by keyword and datetime. Filtering on both does work, but the last option chosen will be prioritized. The solution is to click on the input field to which you want top priority.

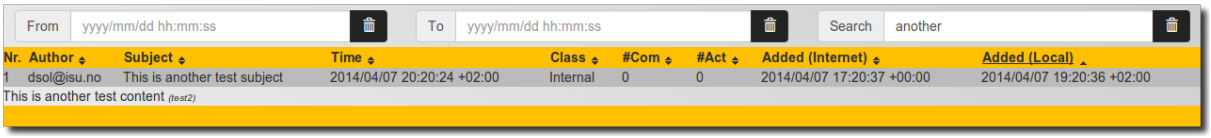
2.12.1 Filter by keywords

To search for a word, you can enter it in the input field on the right. All search words will be matched against everything in the journal entry or an appendix related to an entry. You can for example search for a keyword, person, filename or a filed copy. The search is case insensitive and will with each keystroke adjust the filters to create a real time feeling. The button with the trash can, will as implied empty the input field for text.

Here you can see the journal table without any filters. We got two journal entries, which looks almost the same, but with one difference.

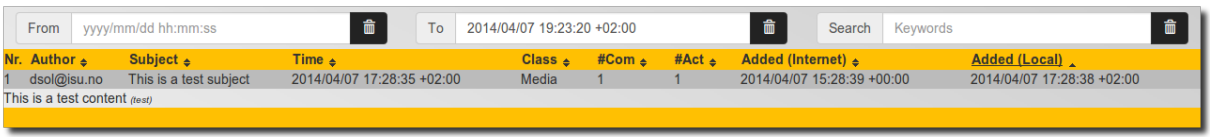


Here is the same table, but here we only show entries that has the word *another* in it.

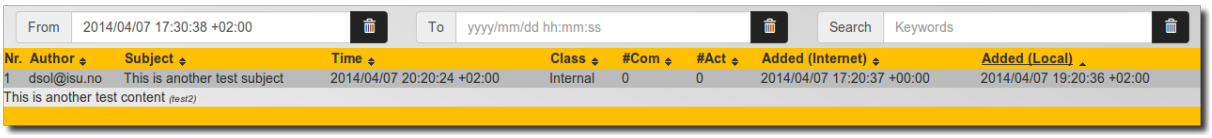


2.12.2 Filter by datetime

When filtering on time you have three alternatives. You can use the *From* input field to set a start time, where you will only see entries that has local added time since the date time you choose. This can for instance be used to get all updates after a status meeting.



You can also use the *To* option to only show entries up to the given date time. The last alternative is to use both, to only get entries that is within the time frame from both *To* and *From*.



CHAPTER

THREE

FAQ

The time sync / offset button is red

This either means that you don't have Internet connection or that your computer time wrong. If the Internet time displays *N/A*, then you don't Internet access most likely, or that the external source is down. If your computer time is incorrect, then you must change it according to your operating system.

Please verify your Internet connectivity by visiting a external webpage, eg http://www.libellus.no Then check that nothing is blocking your access to the Internet time server, by opening the following link manually:

Note: timeapi.herokuapp.com/utc/now.json?callback=jQuery18309364777216687799_1396946053197&_=1396946053268

If you need to accept a non trusted web page, or are presented with a SSL error, review the sites certificate and see if you can accept it.

My local time is in the wrong time zone. How can I change it?

Open the time and date settings on your operating systems and change it there.

How can I sort on several items at the same time?

Hold down the shift key as you click on the table headers.

CHAPTER

FOUR

CHANGELOG

1.0

- Initial release.

CHAPTER

FIVE

CONTRIBUTORS

https://github.com/dsolstad
https://github.com/Ojeey

CHAPTER

SIX

ATTRIBUTION

The development of this software was made possible using the following components:

- CouchDB
- Bootstrap
- jQuery
- Python
- sphinx-doc
- bootstrapx-clickover.js
- bootstrap-dialog.js
- jquery.tablesorter.js
- jquery.browser.js
- jquery.form.js
- timeline.js
- moment.js

CHAPTER

SEVEN

LICENSE

Note: Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at: apache.org/licenses/LICENSE-2.0

I Libellus Operational Dockumentation

Libellus Operational Documentation

Release 1.0

Libellus.no

May 07, 2014

CHAPTER

ONE

PREPERATIONS

USB sticks - The Libellus software should already be installed on USB sticks beforehand. To begin with the installation when the crisis occurs, creates unnecessary stress and delay. The amount of pre installed USB sticks is up to the organization to decide. They need to think about how many people are going to run the software, and then add a couple of backup sticks. If the organization think they will have two instances of Libellus running, they should then have at least four USB sticks ready.

USB encryption - There exist USB sticks with hardware encryption, where you have input buttons on the device to enter the pin code or password to open the USB stick. If anyone get the hold of one of the USB sticks, they can see everything that is going on in Libellus. The upside is that they can only see the traffic if they are in the same local network, and if the RSA keys are updated regularly, the USB sticks they are in hold of are useless. The downside of using USB encryption is that someone needs to remember the password or be written somewhere. It can be devastating if the person that knows the password is not at work that day.

RSA keys - Libellus ships with a pair of default keys. If the organization is only planning to use the software locally with no synchronization over the Internet, the updating of the keys is not that important. The exception is that if anyone internally gets a hold of an USB stick and plug it into the local network, they will be a part of Libellus and see all the content if he has the same key pair. We still recommend a routine for the updating the keys in a given interval.

Define staff - In a military or big organization, there usually exist a staff beforehand. The military uses the special staff structure where you have officers for personnel (S1), intelligence and security (S2), operations (S3), logistics (S4), plans (S5), signals (S6), training (S7), finance (S8) and CIMIC (relations to civil affairs) (S9). A big civil concern normally uses the basic corporate structure of Chief Executive Officer (CEO), Chief Operations Officer (COO), Chief Financial Officer (CFO), Chief Legal Officer (CLO), Chief Security Officer (CSO) and so on. In smaller organizations, these roles may not be defined as concretely and some might not exist at all, but under a crisis everyone should have a defined role. Who leads the crisis event. Who should write journal notes. Who is the contact person to different actors etc.

Training - When a crisis occurs, even the simplest task becomes more complicated and difficult under stress. We strongly recommend to perform test scenarios as often as possible, to remove the factor where everyone needs to learn a new software while the building is burning, and meet real events with one more confidence.

CHAPTER

TWO

INITIAL PHASE

Intern first statement - Agree upon what type of crisis this is, and how much information that can be shared. This statement should be entered as one of the first journal entries in Libellus. Release the statement to other employees to inform them what is going on. If the crisis is of a such nature that media should be informed, use your public relations manager and go public with a press release. Do this to control the information sent to the media.

Set the time - This might not be the first thing that comes to mind when a crisis occurs, but It is important that everyone has the same time. Make sure everybody has the same time on their wristwatches and cell phones. It is also very important that all have the same local time on their computer, so that Libellus logs every event in the correct time.

Initiate Libellus - Start Libellus.

Check who is online - In the chat view in Libellus, you can see who is online. Check for screen names that should not be there. This is in the relations to the USB sections in the initial phase, that there might be other employees that are running the software that should not.

Clear the room - Remove everyone in the room that doesn't need to be there. This is to keep a more calm environment when taking important decisions. Another reason for this is that the more heads that want a say in the decision making, more delay will occur when going back and forth on actions to be taken. To make a lesser good decision is better to not make a decision at all, but one person should have the role as The Devil's advocate when he can.

Define terms - "Happening Time" in Libellus can be ambiguous. It can either be the start time of the event or the end time. The default classifications has some captions, for instance releasable to partners and releasable to trusted partners. It's important that everyone knows the difference. Not that the Libellus software handles the data different, but other staff members might.

CHAPTER

THREE

OPERATING PHASE

Absence log - Keep a log on who is in and out. Use the chat or create a new journal entry to keep track of this. This functionality can be compared to using a whiteboard, but remember that Libellus will preserve this information for the posterity.

Big screens - There should be one or two projectors or big screens, that displays either the journal or the actions view, but preferably both. This is to get a better overview for everyone.

The Devil's advocate - When working in a group, especially in a group where people haven't worked as much with each other before, the concept of being completely up in the air may occur. This is when everyone in the group are in bliss and peace, and every suggestion is met with only positive energy. This is when one person in the group should have the role as The Devils advocate. His role should be to argue in every step of the decision making. This way, the most obvious and brilliant plan, may not be that good after all.

Point of no return - Sometimes a decision must be taken before a given time. This is defined as the point of no return. Libellus supports this by the action functionality.

Reliability - Every incoming information should be verified. Consider how reliable the information should be before it is entered into Libellus. If you enter unverified information, write that the data can not be trusted as for now. Then write a comment when the information is validated.

CHAPTER

FOUR

ENDING PHASE

Backup - Take a backup if the database file and keep its integrity. These backups might be transferred back into Libellus and viewed as before. The crisis or event might end up in court, then it is very important that you have evidence for what was done, in what time, based upon what you knew. The integrity must be intact if used as evidence. To do this, you can burn and archive the database files on a CD, and only allow access to trusted employees.

Evaluation - Walk through the journal log. Ask critical questions, like why did the one in charge make the decision he or she did and what could be done better. In the military, especially in Norway, they do this regularly. This is good practise, but must be performed right. None should feel that they are under the loupe and be afraid, or then this session works against its purpose.

Reinstall - When you have taken backup of the database files, the USB sticks should be formated and a new installation should be downloaded from the Libellus website and copied to the USB sticks, ready to use for a new crisis.

CHAPTER

FIVE

CHANGELOG

1.0

- Initial release.

Libellus Maintenance Documentation

Release 1.0

Libellus.no

May 07, 2014

CHAPTER

ONE

INTRODUCTION

This maintenance documentation show you how to prepare the surrounding system and how to support further development. We recommend that a Unix-like operating system is used for development, since the necessary tools are easily available there.

CHAPTER

TWO

TECHNOLOGIES

Libellus uses the following core technology to work.

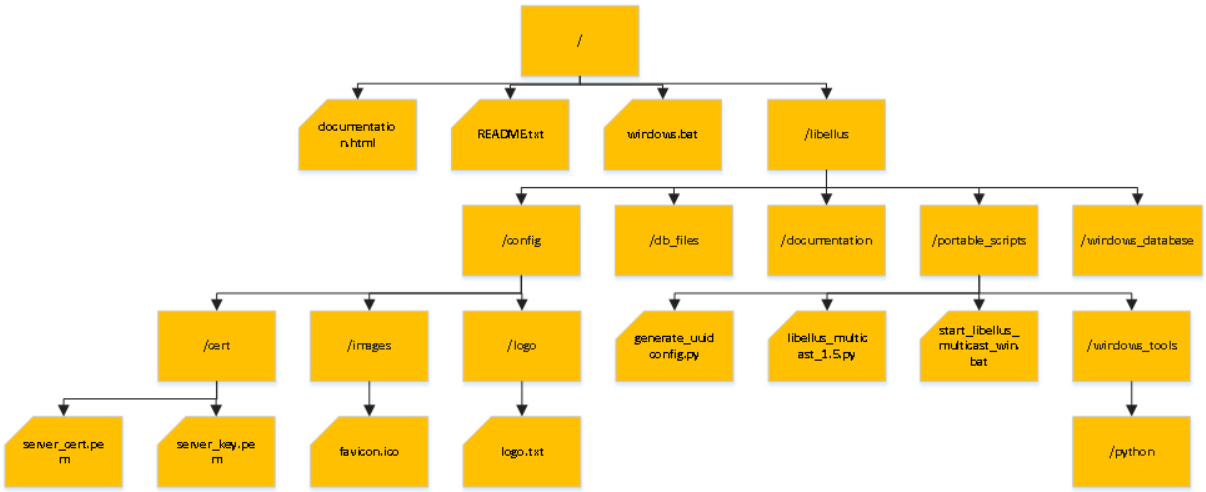
Technology	Version	Homepage	Licence	Description
CouchDB	couchdb-1.5.1_R14B04	couchdb.apache.org	Apache 2.0	Database and web server
Portable Python (win)	2.7.5.1	portablepython.com	Python Licence	Pre-conf Python install

CHAPTER

THREE

FOLDER STRUCTURE

All database files are stored in the db_files folder. The TLS key pair can be found in */libellus/config/cert*



CHAPTER

FOUR

HOW TO EXTRACT THE DATABASE FILES FROM LIBELLUS

The raw database files for Libellus can be extracted in two ways from the system.

On a running system:

Open the *Utils View* from the menu. Download the two databases *journal* and *chat* by clicking on the download links. This can be performed on a live system without data loss.

Extract it from the folder structure:

Enter the root directory of the USB stick. Open *libellus/db_files* and copy the two files named *journal.couch* and *chat.couch*.

Store the files in a safe place. We recommend that you compute hash values for the two files, and store them in a secure separate location. Then before you are going to import the database files again, compute a hash value of the files again, and compare them to the previously generated hash values.

CHAPTER

FIVE

HOW TO PREPARE AND INSTALL LIBELLUS TO A USB STICK

Use a USB stick with at least 8 GB of storage space. Libellus itself uses approximately 300 MB of space, and the rest will be used for the logging data.

Format the device to the exFAT file system. This filesystem is used for cross compatibility between Microsoft Windows and Apple Macintosh systems. It also has support for files larger than 4 GB.

Download the newest portable version of Libellus from libellus.no. Extract the whole package into the root directory of the USB stick

CHAPTER

SIX

HOW TO IMPORT THE DATABASE FILES TO LIBELLUS

Start with a clean install of Libellus as described in “How to prepare and install Libellus to a USB stick.”

Put the two files named journal.couch and chat.couch into libellus/db_files Mark that this must be done before startup of Libellus.

CHAPTER

SEVEN

UPDATE DOCUMENTATION

All the documentation for Libellus is generated using Sphinx. (http://sphinx-doc.org/) Sphinx is written in Python and can be installed with this command:

```
$ sudo apt-get install python-sphinx
```

Clone the wanted documentation from https://github.com/Libellus/Libellus-docs and edit the txt-files containing the documentation text.

After you are finished run:

```
$ make
```

The option singlehtml is used for the documentation in the software.

CHAPTER

EIGHT

UPDATE PRIVATE/PUBLIC KEY PAIR

Libellus uses a self-signed certificate. We strongly recommend that every company that uses Libellus generate their own key pair. This can be done in the following way on an UNIX like operating system:

```
$ openssl genrsa > server_cert.pem
$ openssl req -new -x509 -key server_cert.pem -out server_key.pem -days 1095
```

Place the two files in */libellus/config/cert*

CHAPTER

NINE

COUCHDB CONFIGURATION FILE

Each operating system (Windows and OS X) has its own binary-files for CouchDB, the same goes for configuration files.

For Windows, the configuration files are located in *libellus/portable_scripts/windows_database/CouchDB/etc/couchdb*

The file local.ini contains an uuid that is generated on startup of Libellus by the script *generate_uuidconfig.py*. This is to uniquely identify each running instance of Libellus.

default.ini contains all the specific configuration that is in use. All the lines of the configuration is commented, and will not be repeated here.

CHAPTER

TEN

MAKING COUCHDB WINDOWS PORTABLE

You might want to upgrade the installation if a new release of CouchDB is available. Notice: This step requires some degree of technical knowledge and is not a complete how-to.

- Download the official *CouchDB* Windows installer from couchdb.org
- Install the application to a known location, eg your desktop.
- You will also need the DLLs from The Microsoft Visual C++ 2010 Redistributable Package.
- Edit the file erl.ini located in CouchDB/bin
- Replace the old config with:

```
[erlang]
Bindir=..\\erts-5.8.5\\bin
Progname=erl
Rootdir=..
```

- In this case the erts version is 5.8.5, edit accordingly.
- Do the same with the file erl.ini located in CouchDBerts-5.8.5bin
- Paste the DLLs from Redistributable Package into the /bin and erts-5.8.5/bin folder.
- You should then test the application in a complete clean environment, like a virtual machine with no extra installed DLLs.

CHAPTER

ELEVEN

LIBRARIES

JavaScript and CSS libraries - These works with the current version of Libellus. Upgrading might not work by default and this action is taken at own risk.

Library	Version	Homepage	Licence	Description
bootstrap.css	3.0.2	getbootstrap.com	Apache v2.0	General purpose CSS library
bootstrapx-clickover.js	3.1.1	github.com/lecar-red/bootstrapx-clickover	Apache v2.0	Creates boxes on clickover
bootstrap-dialog.js	N/A	github.com/nakupanda/bootstrap3-dialog	MIT	Creates dialog boxes
jquery.js	2.1.0	jquery.com	MIT	General purpose JS library
jquery.tablesorter.js	2.0.5b	tablesorter.com	MIT or GPL	Sort table data
jquery.form.js	20131228	github.com/malsup/form	MIT or GPL	Submit form data with ajax
jquery.couch.js	N/A	couchdb.apache.org	Apache v2.0	Ajax to CouchDB
jquery.browser.js	0.0.6	github.com/gabceb/jquery-browser-plugin	MIT	Needed for CouchDB
timeline.js	2.6.1	almende.github.io/chap-links-library/timeline.html	Apache v2.0	Visual timeline
moment.js	2.5.1	momentjs.com	MIT	A datetime library
bootstrap-datetimepicker.js	3.0.0	github.com/Eonasdan/bootstrap-datetimepicker	Apache v2.0	A GUI to select datetime

CHAPTER

TWELVE

LIBELLUS CONFIGURATION FILE

The configuration file for Libellus can be found in */_attachments/script/libellus_config.js* The file is well commented and should be straightforward to edit. Here you can for the most part edit time intervals, but also max upload size and the classifications values.

CHAPTER

THIRTEEN

HOW TO INSTALL LIBELLUS ON UBUNTU LINUX

Since there are no official portable Linux release, installation on Ubuntu Linux must be done manually.

Install CouchDB, Git and Curl:

```
$ sudo apt-get install couchdb curl git
```

Install CouchApp:

```
$ sudo apt-get install python-dev
$ curl -O http://python-distribute.org/distribute_setup.py
$ sudo python distribute_setup.py
$ sudo easy_install pip
$ sudo pip install -U couchapp
$ sudo pip install --upgrade couchapp
```

Start CouchDB:

```
$ sudo service couchdb start
```

Clone the Libellus Git repository:

```
$ git clone git@github.com:Libellus/Libellus-src.git
```

Change directory into the Libellus-src folder:

```
$ cd Libellus-src
```

Push Libellus to CouchDB:

```
$ couchapp push main http://localhost:5984/libellus
```

Run the setup_database.sh file to add databases and views:

```
$ chmod +x ./tools/libellus_setup_database.sh
$ ./tools/libellus_setup_database.sh
```

Now view Libellus in your web browser at http://localhost:5984/libellus/_design/main/index.html.

To get a more features of the software you need to do the following:

Use the script in *tools/libellus_multicast.py* to find other Libellus instances and add these to re replication.

Generate private/public key pair add these to the CouchDB configuration file as described below.

The default Libellus/CouchDB configuration file is *default.ini*. This can be found at https://github.com/Libellus/Libellus-portable/tree/master/1.0/libellus/windows_database/CouchDB/etc/couchdb

```
[httpd_global_handlers]
journal.couch = {couch_httpd_misc_handlers, handle_utils_dir_req, "../var/lib/couchdb/journal.couch

chat.couch = {couch_httpd_misc_handlers, handle_utils_dir_req, "../var/lib/couchdb/chat.couch"}
```

By hard coding the URLs to the database files, they can be downloaded inside Libellus in the Utils View.:

```
; _utils = {couch_httpd_misc_handlers, handle_utils_dir_req, "../share/couchdb/www"}
This disables the administration panel futon.

[os_daemons]
repl = your_path_here/libellus_multicast_1.5.py
```

Make the replication script runs as a daemon inside CouchDB.:

```
[daemons]
httpsd = {couch_httpd, start_link, [https]}
```

Enables TLS support.:

```
;httpd={couch_httpd, start_link, []}
```

Disables access over HTTP.:

```
[ssl]
cert_file = your_path_here/server_cert.pem
key_file =  your_path_here/server_key.pem
port = 6984
```

The paths to the key pairs and port number for TLS. Uses a port number below the well known ports to avoid running CouchDB with administrator priveleges.:

```
[vhosts]
127.0.0.1:6984/libellus = /libellus/_design/main/
```

The rewrite pattern which makes 127.0.0.1:6984/libellus/_design/main/index.html point to 127.0.0.1:6984/libellus/index.html.

CHAPTER

FOURTEEN

HOW TO INSTALL LIBELLUS ON APPLE OS X

Just like the guide on how to install on Ubuntu, but download CouchDB for OS X at http://couchdb.apache.org.

CHAPTER

FIFTEEN

ADD ANOTHER TIME SOURCE

Libellus relies on two external time sources on the Internet. One is https://timeapi.herokuapp.com/utc/now.json and the other is https://ojeey.pw/libellus/gettime.php which is as for now supported by Libellus.

To add another time source you can use the following PHP script:

```
<?php

// Requires the php5-json package. Debian/Ubuntu: sudo apt-get install php5-json

ini_set('display_errors', 0);

header('Content-Type: application/json');

// Returns time in millitime
function millitime() {
    $microtime = microtime();
    $comps = explode(' ', $microtime);
    return sprintf('%d%03d', $comps[1], $comps[0] * 1000);
}

// Returns a full datatime string
function fulldatetime() {
    return date('c');
}

$datetime = array();

$datetime['unixTimestamp'] = millitime();
$datetime['dateString'] = fulldatetime();

if (isset($_GET['callback'])) {
    echo $_GET['callback'] . "(";
}

echo json_encode($datetime);

if (isset($_GET['callback'])) {
    echo ")";
}

?>
```

Call the script with a callback function name to get the time in JSONP.

Requested URL:

```
https://ojeey.pw/libellus/gettime.php?callback=foo
```

Result:

```
foo({"unixTimestamp":"1399373804852","dateString":"2014-05-06T12:56:44+02:00"})
```

Note that the time server needs to run https when running Libellus under https.

CHAPTER

SIXTEEN

CHANGELOG

1.0

- Initial release.

K Forprosjekt

Libellus - Crisis management tool

Pre-project report

Daniel André Solstad 110063

Ole Johan Rasch 110240

Contents

1 Goals and constraints

1.1 Background

Most companies have some sort of idea for what to do when a crisis occur. Some of them might also have documentation that describes how different events should be handled. It is very easy to test such plans when sitting in a safe office environment with all the familiar support tools available.

But when everything seems to go wrong, the intranet is down, and you are locked out of your offices? That's when you need a journaling tool that relies on as little infrastructure as possible, and can help you log events and make the right choices.

Libellus aims to be this type of software. Portable and ready to run directly from a memory stick on easily available hardware, as well as providing a graphical overview of the situation with different views and direct message functionality.

1.1.1 Effect goals

It is expected that the finished program will improve crisis management and help a company during a crisis by:

- Simplifying the process of documenting events.
- Taking the right important decisions during a crisis.
- After a crisis being able to review and analyze all actions taken, given the known information available at a specific time.
- Ease the crisis management training.

1.1.2 Result goals

- The software will be released in version 1.0, 19.05.13 with "must-have" functionality described in the task description.
- The produced code will be published on the project web site with a given open source license.
- All non-trivial code should be well documented to support further development by others.

1.2 Constraints

1.2.1 Timeframe

The timeframe for the project is from 07.01.2014 to 05.06.2014. Each student will work at least 30 hours a week on the project.

1.2.2 Juridical

The group will have to comply with the regulations set by Gjøvik University College regarding the bachelor theses.[1] This also includes a "Publishing Contract for student assignments".[2]

When external code and libraries are used, will we comply with the given licenses, and state this in a separate licensing file.

2 Scenario

We have created an evolving scenario which to describe how the software is intended to be used. It is scaled after complexity and addresses a crisis in a single company.

Imagine an ISP that are experiencing a massive fiber cut. Their Internet access and VoIP is down, and the same goes for the cellular connection in the area. They gather together some of the staff to manage the crisis in one of their meeting rooms. The person that is responsible for the logging boots up his computer and attach a USB drive with the journal tool. The only communication they have at the moment is purely physical. i.e. someone needs to go out and gather information and then return to inform the staff. They will then take actions based on the collected information. At the moment, the extent of the case is unknown, but it seems likely that a main cable trench have been affected in some way, since the mobile network also is down. However, it might be that there is some form of attack that is going on, but nothing can be ruled out. They need to think about their customers and at the moment the switchboard is down, and their homepage cannot be reached.

Libellus can help them in structuring the data that comes in, when the secretary can enter all events and attach each event with gathered media, like images, videos or documents. One usage can be to collect events and create a task list with prioritization, and with the graphical view they will be able to sort out and make connections between the recorded events.

The cell network comes back online and they now have voice communicatio and is capable of organizing people in the field, which makes more data comes in. It is needed to divide the crisis handling into two parts. The first is to restore normal state and the second is to keep communication with customers.

Now two more computers is put into the local network, which consist of a router that one of the tecs acquired, and three notebooks which is running Libellus. One of them is connected to a projector where the graphical view is displayed, which enables everyone to get an overview. The new computers will soon have all the already acquired data synchronized from the initial computer.

Normal data communication to their offices in another part of the city comes back online and it is now possible to synchronize documents over the Internet. But the crisis is not resolved yet and some systems are still offline. The other party have their own journal entries entered in Libellus, which means that multi site synchronization will now start. Due to the time functionality in the system, logs will be placed in the correct order, even if there are time-zone differences.

After the crisis is resolved, the logs can be a valuable to help in liability claims, or just explaining why the different actions was taken at the given time. This can also support the process of improving procedures.

3 Extent of task

3.1 Field of study

As support material and background for the bachelor's thesis we will focus on different topics that are relevant to the software we are developing as well as the written procedures that will accompany the program.

Crossplatform - How to make a program that can be run on different operating systems, from Microsoft Windows to Linux, without installing anything on the machine.

Robustness - How different instances of the software running on multiple computers can make the uptime for the "system" itself more or less robust against downtime.

Synchronization - Making replication of data throughout the system without defects in form of integrity errors, and at the same time ensure that all data is delivered.

It will be some focus on security versus usability. For instance that the software should be ready to use without any knowledge of usernames and passwords, but open of access restrictions after the software are put to use.

Crisis management and how the tool can be implemented into existing routines, with usage during the crisis and after for review of actions. This might also imply use as evidence in court.

Different types of databases, traditional relational databases versus NoSQL.

3.2 Task description

The task is to develop a system with the following functionality:

- Local crisis journaling.
- Solutions for attachments, both electronic file types and real world objects.
- Full synchronization of data between locally connected computers.
- Strong focus on integrity, with time registration and read only data.
- Encrypted communication.
- Being runnable from a USB drive.

The solution will make use of already open source tools and libraries, and thus be released into the world of open source software.

In addition there will be written material that support the software, like user guide and maintenance manual. A proposal for how the tool can be used during a crisis and support documentation for crisis management processes will also be developed.

In summary, specific functionality for the software should be:

Must have:

- Be portable to Microsoft Windows 7 and 8.
- Be portable to Ubuntu Linux 12.04.
- Data entry function, with subject, and text field.
- Replication of data, to all other running instances of the software in the LAN.
- View internet time and local time.
- Encrypted communication between clients.
- Export and import of database, to db-file and XML.
- Function for adding comments and actions.
- Function for uploading attachments.
- Function for code generation for real life objects.
- Sorting of data entries, comments and actions.
- Table view of entries.

Nice to have:

- Be portable to Apple OSX Mavericks.
- Extended graphical view of entries.
- Chat functionality.
- Direct message functionality (message queues).
- Site2site communication, eg, outside of LAN communication.

3.3 Limitations

It is not implied that some form of communication must be running to use the software, thus all form of outside communication during a crisis, phone-calls, sms, email is outside of what this project addresses. The communication can only be added as text entries or attachments.

We will discuss possible solutions for a multi-site environment, but it is not given that this will be implemented.

The application should be portable to hardware and software that is available in local stores as of 01.02.14. This excludes older operating systems, such as Windows XP and Vista.

The project group will not focus on how the physical media (memory stick) should be protected.

The aimed crises scale is in the small business market. This will limit the number of local computer clients to around 20.

It will not be developed native mobile applications due to time constraints and complexity. However the GUI elements will be designed with different screen sizes in mind.

Handling the data produced in the software, like generating PDF-documents is out of scope. The same goes for editing uploaded attachments.

Selective synchronization, e.g, that only some journal entries should be replicated, will not be prioritized.

Since the bachelor's project has a strong focus on self-development, and is not considered as consultancy work, the project group states that it might be that not all requests for functionality from the principal will be met.

4 Project organization

4.1 Roles and responsibilities

Group leader
Our group leader will be Ole Johan Rasch, who will be responsible for administrative tasks, room reservation, logging, mandatory hand-ins and documentation, as well as being a member of the programming team.

Technical manager
Daniel André Solstad is the technical manager, lead programmer and has the responsibility to follow the progress of the project when it comes development and usage of code libraries. He is also in charge of the communication with Conax.

Supervisor
Thomas Kemmerich (Associate professor, GUC) is our supervisor during the bachelor's thesis.

Contracting Authority
Conax AS, represented by Espen Torseth (Senior Security Analyst).

5 Planning and reporting

5.1 Choice of methodology

The software we are going to develop will consist of different modules, as well as the written documentation. The modules will interact with each other in such a way that they cannot be written once and then locked down for the rest of the time. We will also ask the principal for feedback every other week. This implies an incremental and iterative development approach. Some of the agile methods like Scrum and eXtreme programming have a lot of principles and techniques that does not fit to a group of two, like pair programming or the scrum master. The same goes for the spiral model, where there is too much management. We will therefore go for a clean incremental and iterative process where the Gantt chart will be leading us towards the finished program and bachelor thesis report.

5.2 Risk management

The table shows the identified threats for the project, with an individual analysis. There are introduced measures against all threats that have medium or higher risk.

Risk	Probability	Consequence	Measure
Data is lost during development	Low	High	We are using an online backup service that support local synchronization.
Sickness in the group	Medium	Medium	Meeting with the principal and supervisor, where it is decided which elements in the milestones that are less relevant.
Lack of completion in parts of the solution	Medium	Low	We can reduce the quality on the modules if it is detected early. Otherwise we must document the reason for the incompletion in the written report.
The time frames is exceeded	Medium	Low	There are some slack space in the Gantt chart, but this will reduce what we develop in the “Nice to have” category.
Disagreements in the group	Low	Medium	Group rules.
Portability for the wanted platforms is too difficult to implement	Medium	Low	As a precaution we have asked the principal which OS that are most important, which is Windows.
Multisite logging is too difficult to implement	Medium	Low	This functionality is put into the “nice to have category.”
Communication failure with the principal	Low	Medium	Our communication so far with the principal have been exceptional, but we have created a role with responsibility for communications as an extra measure.

6 Development plan

We have decided upon five milestones for our development project, these are:

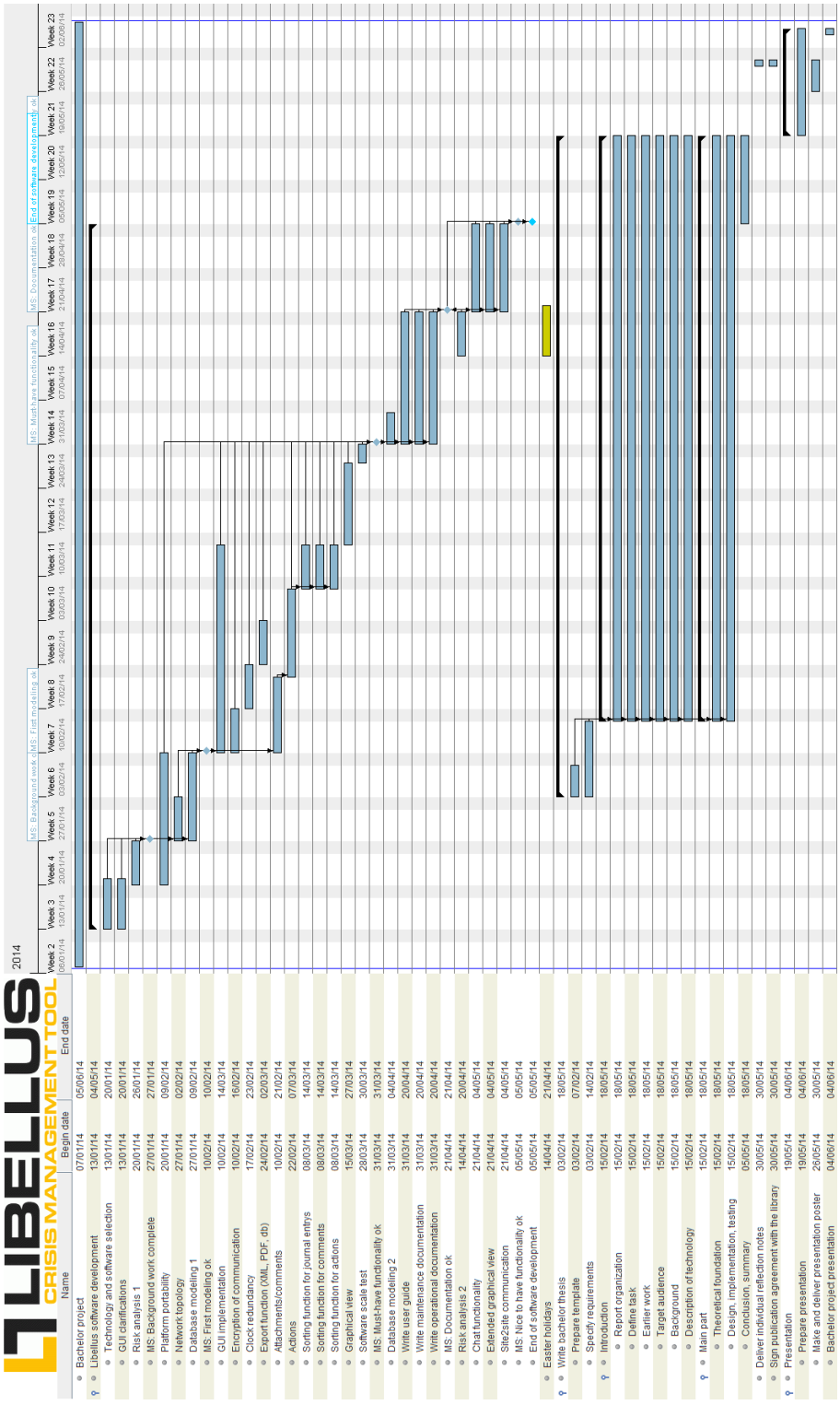
Milestone: Background work complete.
To reach this state, three initial tasks must be complete. Note that work with platform portability are independent of this milestone and will be worked with simultaneously.

Milestone: First modeling ok.
We expect that as development progress, more data needs to be handled in the database. We have therefore focused some time on this.

Milestone: Must-have functionality ok.
Before any documentation is written, all the must-have functionality must be in place. One reason for this is that it will be hard to take screenshots and write concrete steps for unfinished modules. A software scale test on several connected machines will also be performed before this milestone is reached.

Milestone: Documentation ok.
The development part of the project can after this milestone have reached its completeness, depending on remaining time.

Milestone: Nice to have functionality ok.
This is the last milestone in the development project, and must be completed minimum two weeks before the thesis deadline. If the project is delayed it may be that the modules that are part of this milestone is dropped.


LIBELLUS
CRISIS MANAGEMENT TOOL
2014
Name
Begin date
End date
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
Week 17
Week 18
Week 19
Week 20
Week 21
Week 22
Week 23
MS: Background work ok
MS: First modeling ok
MS: Must-have functionality ok
MS: Documentation ok
End of software development ok
Bachelor project 07/01/14 05/06/14
Libellus software development 13/01/14 04/05/14
Technology and software selection 13/01/14 20/01/14
GUI clarifications 13/01/14 20/01/14
Risk analysis 1 20/01/14 26/01/14
MS: Background work complete 27/01/14 27/01/14
Platform portability 20/01/14 09/02/14
Network topology 27/01/14 02/02/14
Database modeling 1 27/01/14 09/02/14
MS: First modeling ok 10/02/14 10/02/14
GUI implementation 10/02/14 14/03/14
Encryption of communication 10/02/14 16/02/14
Clock redundancy 17/02/14 23/02/14
Export function (XML, PDF, db) 24/02/14 02/03/14
Attachments/comments 10/02/14 21/02/14
Actions 22/02/14 07/03/14
Sorting function for journal entrys 08/03/14 14/03/14
Sorting function for comments 08/03/14 14/03/14
Sorting function for actions 08/03/14 14/03/14
Graphical view 15/03/14 27/03/14
Software scale test 28/03/14 30/03/14
MS: Must-have functionality ok 31/03/14 31/03/14
Database modeling 2 31/03/14 04/04/14
Write user guide 31/03/14 20/04/14
Write maintenance documentation 31/03/14 20/04/14
Write operational documentation 31/03/14 20/04/14
MS: Documentation ok 21/04/14 21/04/14
Risk analysis 2 14/04/14 20/04/14
Chat functionality 21/04/14 04/05/14
Extended graphical view 21/04/14 04/05/14
Site2site communication 21/04/14 04/05/14
MS: Nice to have functionality ok 05/05/14 05/05/14
End of software development 05/05/14 05/05/14
Easter holidays 14/04/14 21/04/14
Write bachelor thesis 03/02/14 18/05/14
Prepare template 03/02/14 07/02/14
Specify requirements 03/02/14 14/02/14
Introduction 15/02/14 18/05/14
Report organization 15/02/14 18/05/14
Define task 15/02/14 18/05/14
Earlier work 15/02/14 18/05/14
Target audience 15/02/14 18/05/14
Background 15/02/14 18/05/14
Description of technology 15/02/14 18/05/14
Main part 15/02/14 18/05/14
Theoretical foundation 15/02/14 18/05/14
Design, implementation, testing 15/02/14 18/05/14
Conclusion, summary 05/05/14 18/05/14
Deliver individual reflection notes 30/05/14 30/05/14
Sign publication agreement with the library 30/05/14 30/05/14
Presentation 19/05/14 04/06/14
Prepare presentation 19/05/14 04/06/14
Make and deliver presentation poster 26/05/14 30/05/14
Bachelor project presentation 04/06/14 04/06/14

Bibliography

[1] Høgskolen i Gjøvik. Reglement og kontrakt for bacheloroppgaver, 2014. [Online; visited 17-March-2013] http://www.hig.no/imt/bacheloroppgaver/informasjon.

[2] Høgskolen i Gjøvik. Publiseringskontrakt studentoppgaver, 2014. [Online; visited 17-March-2013] http://hig.no/biblioteket/oppgaveskriving.

A Appendix:

A.1 Group ground rules for Libellus

Group members

Name	Role	Email	Phone
Ole Johan Rasch	Group leader	ole.rasch@hig.no	90789598
Daniel André Solstad	Technical manager	daniel.solstad@hig.no	41572521

§1 The groups goal, ambitions and work effort

The group aims to deliver a product of high professional quality.

§2 Democratic decisions

All decisions and any disputes shall be decided by the group at a show of hands. Group members should try to argue to achieve complete consensus or compromise.

§3 Meetings

The group members are obliged to attend at the agreed location at the agreed time. In case of illness or other valid reason of absence, contact one another as quickly as possible.

The group arranges meeting times that and should preferably be in the period between at 08.00 to 17.00 on weekdays.

§4 Expenses Any expenses will be divided equally among the members. The accounting records and receipts must be retained and made available in the document management system.

§5 Work

§5.1 Distribution of tasks

All tasks shall be divided as equally and fairly as possible.

§5.2 Workload

Each member of the group will put down at least 30 hours of work every week. Group members will deliver the agreed work on time. If you are stuck, it is important to notify the other member. The same applies if you fail to deliver assigned tasks on time.

§5.3 Specification of tasks

Before the group members part after each meeting, both should have well defined tasks to complete for the next meeting.

§6 Logging

After work sessions, a short log is written, containing completed tasks and overall progress. Submissions to the version control system should always have well written change descriptions.

§8 Signing Both group members are pursuant to sign on behalf of the group.

§9 Sanctions

At three repeated violations of group rules, the following sanctions will be held:

- Conversation between group members on the issue.
- Conversation between the whole group and the supervisor.

Signatures:

L Kontrakt

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

Conax AS

(oppdragsgiver), og

Daniel André Solstad

Ole Johan Rasch

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 07.01.2014 til 05.06.2014.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker egne utgifter til gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser samt nødvendige, forhåndsgodkjente, reisekostnader for studentene fra Gjøvik til Oslo.
   - Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke benyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik

offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan vederlagsfritt anvendes av oppdragsgiver i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser gjort av arbeidstakere av 17. april 1970, §§ 4-10.

   Der besvarelsen bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av oppdragsgiver eies prosjektbakgrunnen fortsatt av oppdragsgiver. Eventuell utnyttelse av videreutviklingen, som inkluderer prosjektbakgrunnen, forutsetter at det inngås egen skriftlig avtale om dette mellom student og oppdragsgiver.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan/prodekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten kan bringes inn for de alminnelige domstoler. Partene vedtar Oslo tingrett som eksklusivt verneting

11. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): Thomas Kemmerich

Oppdragsgivers kontaktperson (navn): Espen Torseth

Student(er) (signatur): _____ dato 17/03-14

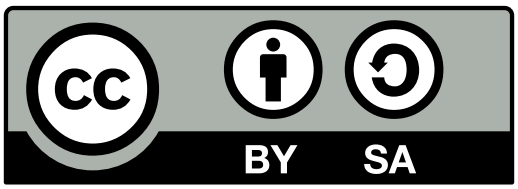
_____ dato 13/03/14

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur): _____ dato _____

IMT Dekan/prodekan (signatur): _____ dato _____


BY
SA