Design og spesifikasjoner for Mobil Læringsapp Bygger (MLAB)

Vedlegg til Konkurransegrunnlag 10/2013

Konsulentoppdrag – Videreutvikling av rammeverk for generering og distribusjon av mobile applikasjoner

For levering til Forsvarets forskningsinstitutt (FFI)

Sammendrag

Mobile plattformer som smarttelefoner og nettbrett gir mange nye muligheter for Forsvaret. Forsvaret benytter i dag, i veldig stor grad, PowerPoint for distribusjon av informasjon, også i undervisningssituasjoner. Med en stadig større satsning på mobile læringsplattformer blir PowerPoint et lite egnet format, spesielt med hensyn til interaksjon og distribusjon. Det er derfor ønskelig å utvikle ett rammeverk som vil la instruktører og andre uten spesiell erfaring med smarttelefoner eller programmering utvikle, vedlikeholde og distribuere mobile applikasjoner.

Applikasjonene som produseres i rammeverket som nå ønskes utviklet vil være HTML5 baserte applikasjoner. Tatt i betraktning at det nå eksisterer en rekke forskjellige mobile operativsystemer og enhetstyper så vil appene lages med HTML5 standarden for best mulig kompatibilitet. HTML5 kan sies å være sidebasert, ikke ulikt PowerPoint, så de som lager en app vil fylle inn informasjonen fortløpende og i de aller fleste tilfeller vil brukere gå igjennom appen fra begynnelse til ende, selv om det kan være linker mellom forskjellige sider som tillater hopping frem og tilbake. Via programvaren Cordova vil HTML5 sidene som er laget kompileres til frittstående applikasjoner som så kan brukes av sluttbrukere med støttede mobilenheter.

Dette rammeverket har fem moduler som vil arbeide sammen.

1. Et relativt ukomplisert verktøy for å lage en app: En HTML5 basert AppEditor. Verktøyet skal brukes i en nettleser av brukere som ikke har forutgående kunnskap om hvordan en app lages.
2. Maler for forskjellige typer apper og komponenter med funksjonalitet som brukerne kan velge som et utgangspunkt for en ny app eller funksjonalitet i en app. Malen vil A) gi appen et standard utseende og B) kan ha innebygd funksjonalitet som for eksempel indeksside eller navigasjon. Komponentene benyttes for å gi ytterligere funksjonalitet.
3. En administrativ modul hvor administrative brukere kan legge til eller fjerne brukere, laste opp nye maler, slette gamle app-prosjekter etc.
4. En kompileringsmodul for hver plattform som støttes. Denne modulen vil bruke Cordova + utviklingsverktøy for den relevante plattformen for å lage den ferdige appen. For den administrative brukeren vil denne prosessen være «usynlig».
5. Ett app marked (à la Google Play Store for Android og Apple's App Store for iPhone) som støtter flere operativsystemer. Her kan sluttbrukere bla igjennom og søke etter apper og lese andre brukeres evalueringer av apper. Det vil også være en administrasjonsmodul til app markedet hvor apper kan gå igjennom en godkjenningsprosess hvis dette er ønskelig.

Det er allerede utviklet et demonstratorsystem som skal illustrere konseptet. Dette dokumentet beskriver en videreutvikling av dette systemet som ønskes gjennomført.

Innholdsfortegnelse

1. Innledning

MLAB systemet skal sette personer i forsvaret, uten spesiell opplæring, i stand til å komponere, kompilere, distribuere og holde oppdatert, forholdsvis enkle mobile applikasjoner. Dette gjøres gjennom en HTML5 kompatibel nettleser og de bakenforliggende tjenestene som utgjør MLAB systemet som for eksempel et app-redigeringsverktøy, app-maler, kompileringsmoduler, administrasjonsmodul og et app marked.

Målgruppen for systemet er av generell karakter, men i denne fasen av utviklingen rettes ressursene mot instruktører og enheter/personer med faglig rådgivende funksjon i Forsvaret. De mobile applikasjonene som brukerne vil lage i MLAB rammeverket vil i stor grad være basert på behovet for å distribuere oppdatert informasjon som leksjoner, anbefalte rutiner og prosedyrer. I tillegg ønskes noe funksjonalitet for kommunikasjon og bruk av 3.parts APIer i de mobile applikasjonene. De ferdige applikasjonene vil typisk fremstå som sidebaserte, hvor en navigerer mellom sider med forskjellig innhold og funksjonalitet.

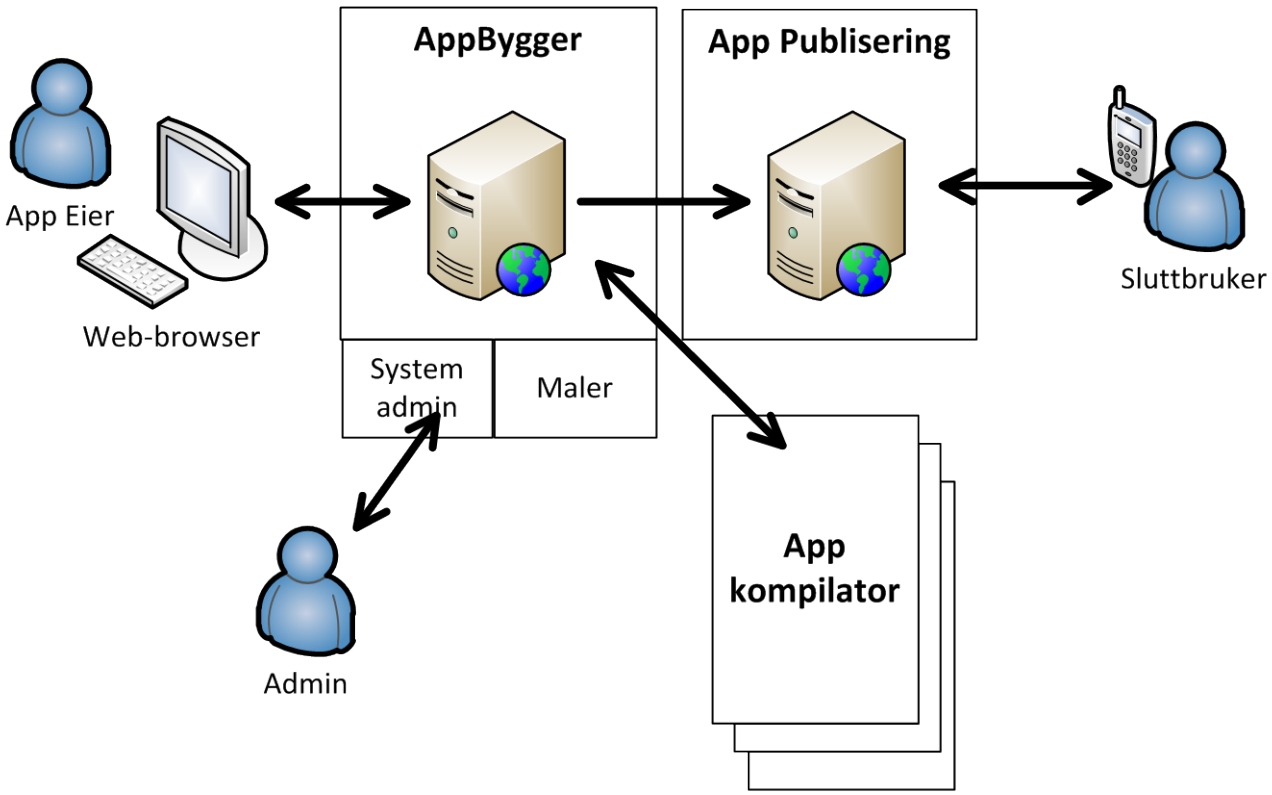
Systemet skal lett kunne videreutvikles til å lage mer avanserte applikasjoner. Dette vil skje gjennom utvikling av nye maler og komponenter. En demonstrator (proof-of-concept) av systemet er allerede tatt frem, og utvikling av systemet må ta utgangspunkt i funksjonalitet som finnes der.

Videre i dette dokumentet presenteres først høynivådesign av systemet. Deretter vil modulene i systemet og utvikling av relevante grensesnitt mellom de forskjellige modulene diskuteres. I seksjon 4 finnes en detaljert spesifikasjon av hva som må utvikles bli presentert. I seksjon 5 finnes generelle krav til utviklingen av systemet.

2. Høynivå design

2.1. Generelt

Systemet gir brukeren (app eieren), gjennom en HTML5/JS/CSS3 basert klient, en ramme for å opprette, editere og distribuere enkle mobile applikasjoner. Systemet består i hovedsak av en nettleser-basert klient og flere servermoduler som er nødvendig for til slutt å kunne produsere HTML5 baserte mobile apps som frittstående apps til flere operativsystemer.



Figur 1 MLAB systemet overordnet

Brukeren (App eieren) forholder seg til systemet gjennom den nettleser-baserte klienten. Klienten lar brukeren administrere applikasjonsprosjekter som innebærer oppgaver som å opprette nye app-prosjekter, kopiere app-prosjekter, kompilere app-prosjekter og distribuere ferdige applikasjoner (resultatene av et kompilert app-prosjekt) eller oppdatere allerede publiserte applikasjoner. I tillegg har klienten funksjonalitet for å redigere applikasjonsprosjekter.

MLAB systemet skal kunne kjøre over internett, men også på lukkede LAN slik at avhengighet til 3.parts systemer ikke kan være avgjørende for MLAB systemets grunnleggende funksjoner. Der ikke annet er nevnt er nettrafikken over https.

App eieren redigerer et app-prosjekt i web-klienten, i en drag-and-drop baserte editor hvor applikasjonsbyggingen foregår på et høyt abstraksjonsnivå (ikke ulikt PowerPoint). Applikasjonene som produseres er "komponentbaserte", slik at app eieren drar inn aktuelle *komponenter* med ønsket funksjonalitet/logikk og fyller inn innhold og gjør nødvendig konfigurering av sidene[1]. Programflyt og

[1] Ikke ulik tilnærmingen som finnes i en lignende editor: Maslo Authoring Tool http://www.academiccolab.org/maslo/. MIT App Inventor for Android har også lignende funksjonalitet http://www.appinventor.mit.edu/ men på et lavere abstraksjonsnivå.

kode er i utgangspunktet skjult for app eieren; i bakgrunnen produseres en html5/js/css applikasjon som kompileres sammen med rammeverket Cordova[2] for å bli frittstående applikasjoner til Android, iOS mfl.

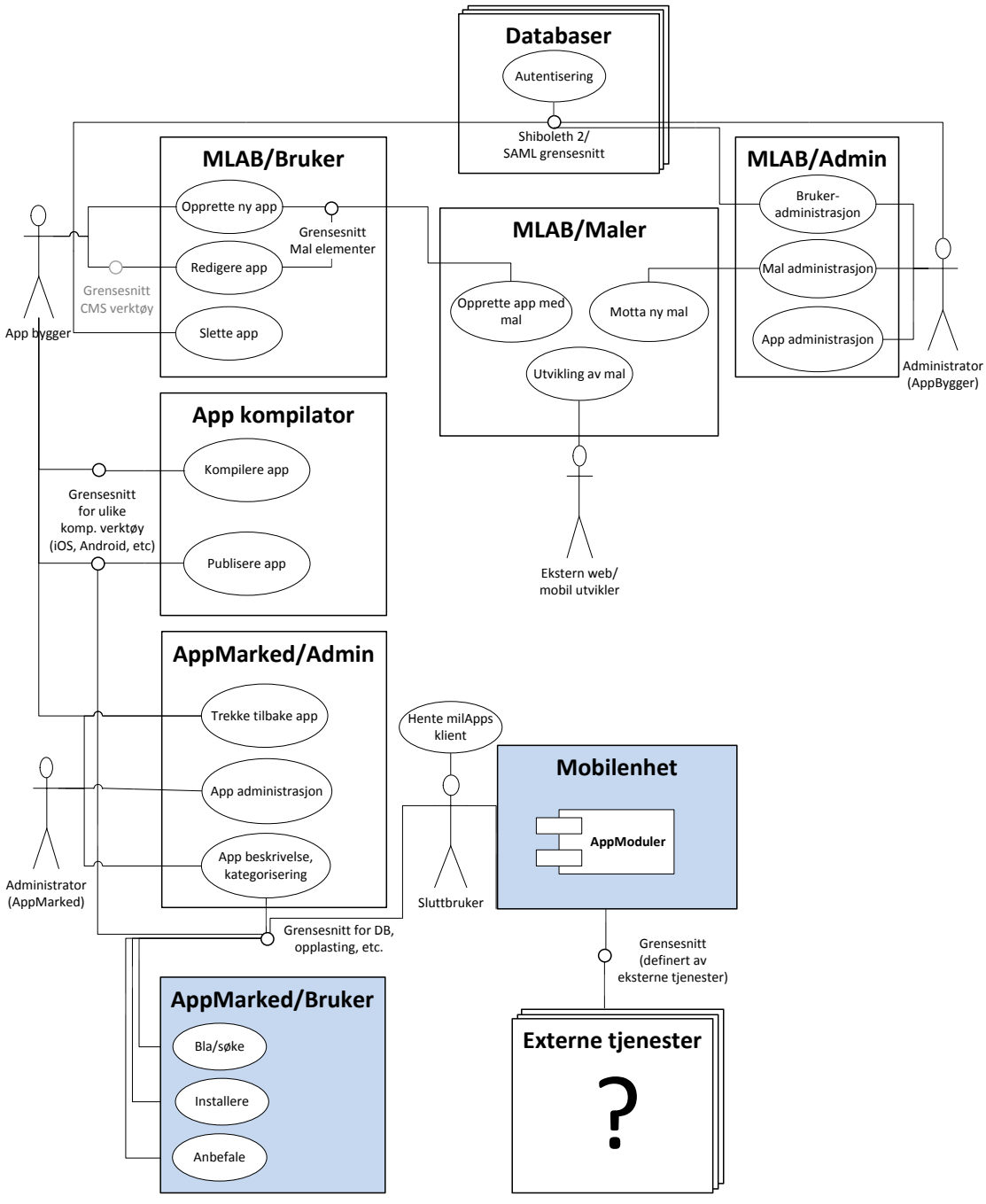
App eierne er i hovedsak instruktører, fagansvarlige og andre i Forsvaret med spesielt ansvar for å produsere og distribuere informasjon som kursmateriell, regler, prosedyrer og best-praksis. Funksjonalitet som er tilgjengelig for bruk i appene implementeres som *maler* eller *komponenter* i MLAB-systemet og systemet skal lett kunne utvides ved å legge til nye maler eller komponenter. Det første settet med maler og komponenter skal reflektere denne første brukergruppens behov.

App-prosjektene skal som minimum kunne kompileres til Android og iOS apps, men MLAB-systemet må lett kunne utvides til også å kunne kompilere apper til andre operativsystemer. Distribusjon av de ferdige applikasjonspakkene fra systemet til aktuelle applikasjons-markeder (ala Google Play for Android og iOS App Store) kan initieres av app eieren fra web-klienten. App eieren kan også laste ned ferdige filer (applikasjonspakker) lokalt for testing på egne mobilenheter. Det skal utvikles ett eget app-marked (med utgangspunkt i F-Droid) som kan distribuere pakker for flere operativsystemer.

[2] http://cordova.apache.org/

2.2.Bruksmønstre: samspill mellom brukerbehov og modulære funksjoner

Det er viktig at systemet er modulært for å sikre at fremtidige brukerbehov kan bli oppfylt uten at man må re-utvikle store deler av systemet. Det betyr at de forskjellige modulene må ha et tydelig definert grensesnitt, samtidig som det er veldig klart hvilke funksjoner hver modul oppfyller i det overordnede bildet. I denne seksjonen vil disse sammenhengene diskuteres, en komplett oversikt kan sees i Figur 2. Detaljerte spesifikasjoner finnes i seksjonen «Detaljert design, teknologier og verktøy».



Figur 2 Moduler og deres funksjoner

De blå modulene er det som sluttbrukeren ser/samhandler med, dette er hovedsakelig App markedet og selve appene som har blitt laget. De resterende modulene utgjør til sammen MLAB systemet som tilrettelegger for app utvikling og vil kort gjennomgås her.

2.3.AppBygger

Dette er hovedmodulen, her vil nye apper bli bygget basert på en valgt mal og komponenter som app eieren har inkludert og konfigurert under byggingen av appen (se nedenfor). Funksjonaliteten her deles mellom nettjeneren (her lagres maler, HTML dataen som utgjør en side i appen lagres, bilder lastes opp, etc) og nettleseren (her påbegynnes nye apper og individuelle sider designes).

Funksjonaliteten som er tilgjengelig for app eierne under bygging av apper i MLAB-systemet kan utvides vha *maler* og *komponenter*. En mal velges for hele applikasjonen, mens komponenter er mindre biter med funksjonalitet som app eieren kan velge å inkludere.

2.3.1. Maler

En *mal* er en kombinasjon av HTML5, Javascript og CSS filer som inneholder de visuelle elementene som sluttbrukerne vil se på sine mobile enheter (f. eks. topp- og bunntekst), og de logiske elementene som naturlig vil være felles for en applikasjon (ett eksempel kan være en spørreskjemafunksjon som sjekker svar umiddelbart på enheten). Disse malene lastes opp til en sentral tjener og kan benyttes av forskjellige app eiere til å starte en ny app. Formatet på malene utgjør ett grensesnitt som må følges nøye, se 3.4 Maler: filer og grensesnitt.

2.3.2. Komponenter

En *komponent* består også av noe HTML kode (minimum et <Div> element) med tilhørende JS og CSS. Etter at app eieren har valgt mal for applikasjonen er komponentene app eierens byggeklosser når han bygger applikasjonen.

2.4.App kompilator

Appene som bygges av app eieren består av én mal og komponentene som app eieren har tatt med. På bakgrunn av dette er det generert en rekke individuelle HTML sider, med koblinger til diverse støttefiler (bilder, videoer, Javascript/CSS filer, etc). Disse må «kompileres» for å utgjøre en selvstendig app som kan kjøres på en mobil enhet. Dette gjøres via Cordova, som kombinerer informasjonen laget av app eieren, mal informasjonen og egne bibliotek til en app. Etter at dette er gjort bør appen testes før den publiseres på app markedet hvor sluttbrukerne kan installere de på sine egne enheter. Det vil være egne grensesnitt her som gjør at brukerne kan kompilere for flere plattformer via den samme nettsiden.

2.5.App marked

Prinsippet her er velkjent i fra Apples AppStore og Google Play, man har et bibliotek av apper som en bruker kan bla eller søke igjennom, og når en relevant app er funnet kan den installeres med ett par trykk på skjermen. Når app eieren publiserer en app så består dette i hovedsak av å kopiere en enkelt app fil inn i biblioteket (som er en egen tjeneste) samt å oppdatere en database med informasjon om appen, f. eks. nøkkelord, skjermbilder, etc. Det er en egen administrator her, de kan overstyre opplastinger, fjerne apper, etc.

Markedet må være tilgjengelig både som en Android app og en nettside. Dette er pga. at iOS enheter kan ikke ha en egen app for alternative App Stores, men kan kun laste inn fra nettsider. Nettsiden skal også kunne benyttes av Androidbrukere som vil finne apper på en PC. Her er det nødvendig med grensesnitt imellom MLAB nettjeneren, app kompilatoren og app markedet, samt mellom app biblioteksdatabasen og app nettsidene/app markeds appen.

Ferdig kompilerte apper skal kunne publiseres på det felles app markedet. For å muliggjøre bruk av andre mobile operativsystemer i fremtiden så skal et grensesnitt utvikles for opplasting av apper uansett hvilket OS det er skrevet for. App-pakker for flere systemer vil således være tilgjengelig fra samme sted når det i prinsippet er den samme appen. Det må implementeres et nettgrensesnitt som lar sluttbrukerne gå igjennom beskrivelse av appene, gi tilbakemeldinger/kommentarer på appene, og lese andre sluttbrukeres vurderinger av appen. En app vil ha en enkelt side som fronter appen, uavhengig av operativsystem, og som gir brukeren tilgang på appen pakket for alle tilgjengelige operativsystemer som MLAB systemet støtter.

Markedet skal kunne fungere uavhengig av AppByggeren og kan sees som et eget system. Markedet skal også kunne distribuere applikasjoner som ikke er generert av AppByggeren i MLAB systemet.

2.6.Brukermodeller og autentisering

Det er i hovedsak tre typer brukere av MLAB systemet:

- *App Eier* – setter sammen apper gjennom bruk av systemets web-grensesnitt.
- *Administrator* – kan gi brukere tilgang, legge til/aktivere/deaktivere funksjonalitet og utseendemaler i MLAB systemet.
- *App Marked Administrator* – kan sette opp brukere i markedet, legge til/aktivere/deaktivere apper.
- *Sluttbruker* – har anledning til å se ferdige apps i app markedet, laste ned ferdige apps og kommenterer på apps.

I tillegg kommer “systembrukere” som modulene benytter for å identifisere hverandre.

Der ikke annet er nevnt aksesserer brukerne systemet over internett eller LAN gjennom en nettleser.

Systemet vil ha to selvstendige brukermodeller: En for marked-modulen og en for “resten”, i hovedsak AppByggeren. Disse skal implementeres som egne selvstendige brukermodeller. De interne brukermodellene skal implementere normal brukerhåndtering (ny bruker, slett bruker, brukergrupper, nytt passord osv.) samt autentisering. I tillegg til de interne brukermodellene skal det være mulig å konfigurere systemet slik at brukerne autentiseres fra en LDAP-server og deres gruppetilhørighet sjekkes mot LDAP-serveren, men dette er en opsjon og systemene skal fungere uten LDAP men med en lokal database.

Det skal være mulig for anonyme sluttbrukere å benytte app markedsmodulen med rettigheter som følger anonyme brukere.

3. Detaljert design, teknologier og verktøy

Av både kostnads- og integreringsårsaker så benyttes åpne standarder. Vi bygger videre på eksisterende åpen kildekodeprodukter for utviklingen Det er viktig å benytte utviklingsverktøy som er mye brukt generelt i IT industrien for å forsikre seg om at det er lett å få nye utviklere involvert og at nye funksjoner kan legges til når det trengs, dette gjelder både for sluttproduktet (appene) og produksjonsverktøyene (i fra data input til app markedet).

3.1.Teknologivalg

3.1.1. MLAB nettleser brukergrensesnitt

Dette vil benytte relativt ny teknologi, den skal derfor testes mot nettleserne Firefox, Chromium og Google Chrome (og ikke Internet Explorer, selv om det i teorien kan brukes under IE når IE tillegget *Chrome Frame* er installert).

- HTML(5)
- CSS 2.0
- Javascript (spesielt jQuery bibliotekene)
- SiteCake (en CMS basert på de ovennevnte teknologiene + PHP, skal utvikles videre)

3.1.2. MLAB tjenere

MLAB nettjeneste (AppBygger)

- Linux (trengs for scripting/kompilering)
- Apache 2 (eller kompatibel) / MySQL / PHP (deler kan også kodes i Python, PERL, BASH)

MLAB kompilator for Android

- Linux
- Cordova
- Bash/PHP/PERL programsnutter
- SFTP klient

MLAB kompilator for iOS

- Apple Mac/OSX
- Cordova
- Bash/PHP/PERL programsnutter
- SFTP/SSH server

App-marked

- Linux
- Apache (for søk og nedlasting av apper) / MySQL / PHP eller Python
- F-Droid app marked tjener for Android
- Web side som iOS enheter bruker for å laste ned apper.[3]

[3] Se http://help.apple.com/iosdeployment-apps/mac/1.1/#app43ad6a6a for mer informasjon

- SFTP/SSH server

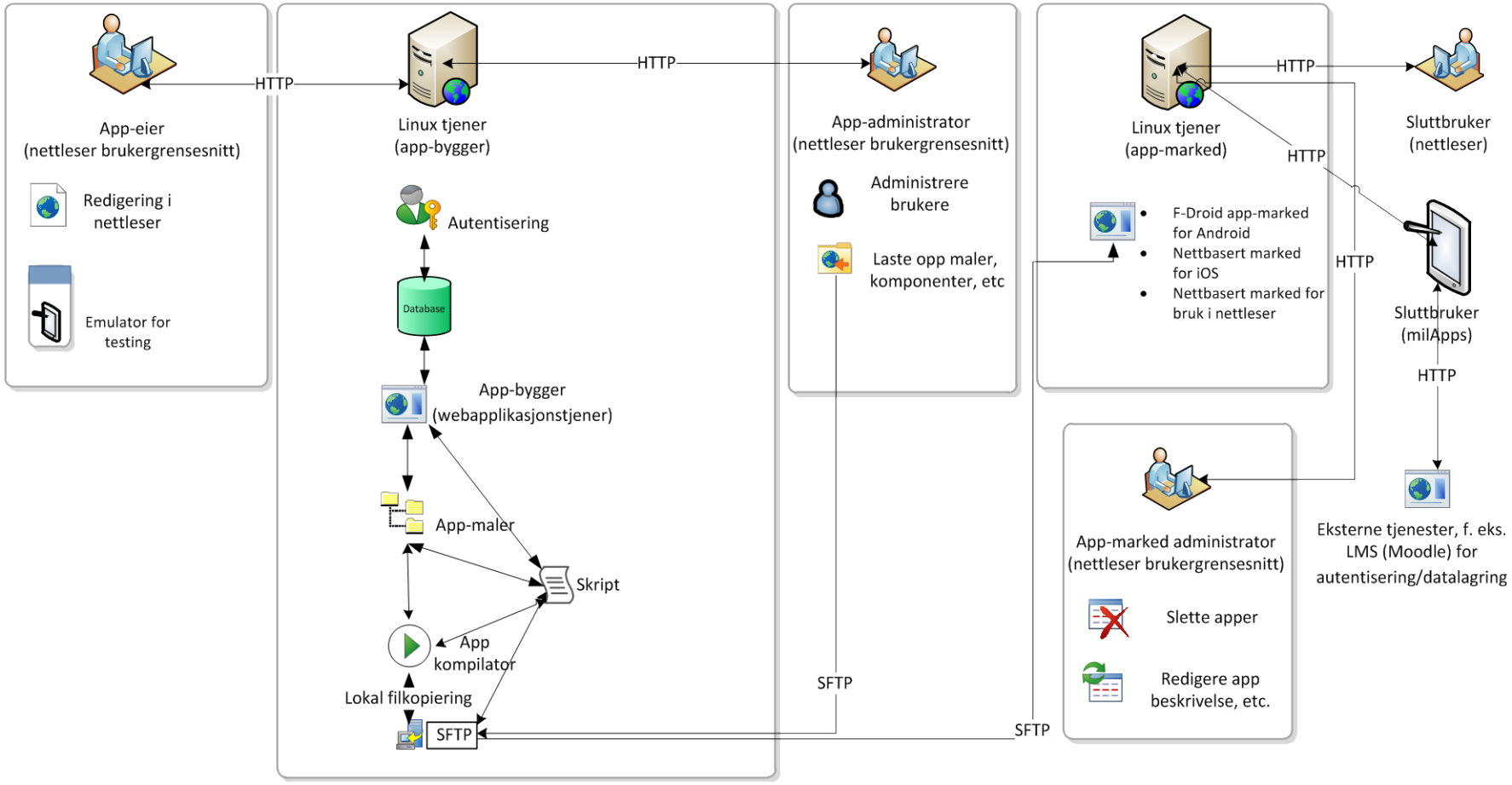
3.1.3. Apper produsert i MLAB systemet

For dette prosjektet vil man ikke lage såkalte «native apps» direkte, men benytte HTML5 apper som kompileres til den respektive plattformen. En rekke app-maler vil bli ferdigstilt, disse vil bestå av visuelle komponenter så vel som logiske komponenter. Man kan for eksempel ha en app som oppfører seg som en ebok, med innebygd søk og automatisk indeks, eller en app som har en interaktiv quiz. Disse appene vil bruke disse teknologiene:

- HTML5 for visning av informasjon
- CSS for formattering
- Javascript (og jQuery Mobile) for programmering av interaktivitet (logg inn, videovisning, etc)
- Cordova funksjonalitet for å kompilere appene og for tilgang til smarttelefon spesifikke funksjoner som GPS, kalender, etc.

3.2.Arbeidsflyt: Fra individuell sideredigering til ferdig app

Modulene som er diskutert ovenfor skal samhandle på klart definerte måter og en del av disse er allerede utviklet til demonstrator stadiet. Figur 3 har en oversikt på hvordan de skal brukes og hvordan de knyttes sammen, dette diskuteres i detalj nedenfor.



Figur 3 Oversikt over arbeidsflyt

3.3.Redigeringsverktøy og komponenter: filer og grensesnitt

SiteCake som skal brukes for redigering støtter dra og slipp redigering av individuelle HTML sider, SiteCake generer den nødvendige HTML koden når man legger til nye elementer på en side. For å oppnå full fleksibilitet så skal SiteCake skrives om så all funksjonalitet er komponentbasert. Dette vil først og fremst berøre menyen som på det nåværende tidspunkt er hardkodet til å støtte de følgende HTML kodene: H1, H2, H3, OL, UL, IMG, samt rene HTML blokker. I tillegg støtter den en del vanlige komponenter: Fancybox lightbox, embedded YouTube video, embedded Google Maps og opplasting av/linking til alle filtyper. Menyen ser slik ut:

Figur 4 Menyen i SiteCake

For å gjøre dette om til en komponentbasert arkitektur så skal hver komponent (f. eks. H1, eller YouTube video) utvikles som en filbasert plug-in som følger disse reglene (eksempelet er for en H1 komponent):

- Alle filer for en komponent er i en mappe som er navngitt etter komponenten i små bokstaver, i dette eksempelet blir dette h1. Filer markert med en asterisk er påkrevd.

Mappe	Filnavn	Forklaring
h1	conf.txt	Konfigureringsinformasjon, for eksempel Google API nøkkel for kart. Dette skal bruke vanlig INI format: [Header] Key=Value
	html.txt*	HTML som skal legges inn når denne komponenten velges, for eksempel: Legg in overskrift her *Hvis komponenten ikke skal redigeres av app eieren så skal det være en plassholder her som bare synes i design modus (bruk jQuery mobile funksjoner for å sjekke om man er på en mobil enhet).* *HTML koden må inkludere all JavaScript kode som trengs for å kjøre appen på den mobile enheten!*
	icon.png*	24x24 piksel ikon som vises på verktøylinja
	exec.js	JavaScript kode som kjøres a) første gangen komponenten legges til siden, og b) når komponenten dobbeltklikkes. Dette brukes for å innhente opplysninger fra brukeren, som for eksempel en URL for en videokobling. I tillegg skal den inneholde kode som manipulerer DOM treet ved å legge inn
	exec.php	PHP kode som kjøres på tjeneren nå denne komponenten legges

	til. Den kan f. eks. kopiere nye JS filer (se under) til appens /js/ folder (se Appendiks 1 for informasjon om mappestruktur) eller oppdatere Cordovas konfigureringsfiler.
rights.xml	Definerer Cordova app tilgangsrettigheter som trengs for denne komponenten, for eksempel GPS rettigheter for en kart komponent.
h1.js	Javascript bibliotekskode som er større enn at det kan legges til i html.txt, eller repeteres flere ganger. *Denne koden kan også ha en funksjon som heter onComponentInitialise(), den kalles opp av kode høyere opp i app.js (onAppReady) som er en del av alle maler, se neste avsnitt for mer informasjon om dette.*

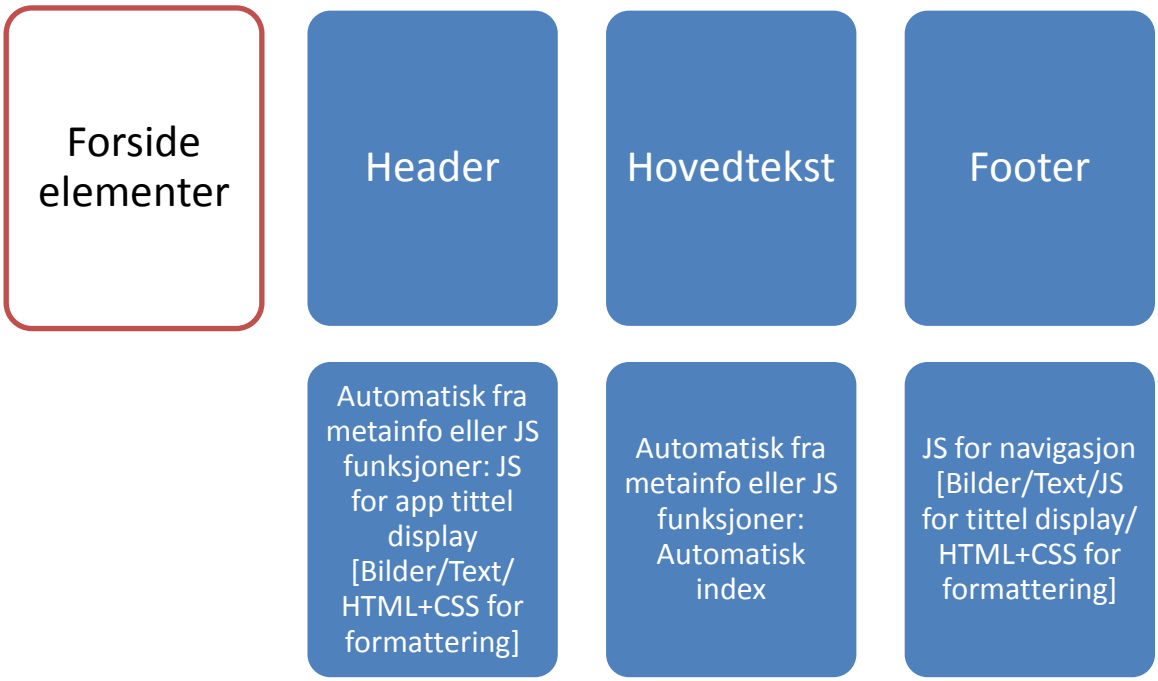
- I rotmappa for komponentene skal det være en conf.txt fil som har en liste over komponenter som skal legges til verktøylinja, m.a.o. man kan ha komponenter i en mappe som ikke er vist i verktøylinja hvis den ikke er listet i denne filen. Denne funksjonaliteten bruker navnet på de forskjellige komponentmappene, «blank» brukes for å indikere et mellomrom i menyen.
- Når en side åpnes for redigering så vil menyen genereres fra denne listen, med innlasting av ikoner i fra de forskjellige undermappene.
- Komponenter pakkes inn i en ZIP fil med rot mappen inkludert i filstien, dette kan så lastes opp av administrator og pakkes ut av et skript på tjeneren.

Systemet skal senere lett kunne utvides ved å legge til nye komponenter.

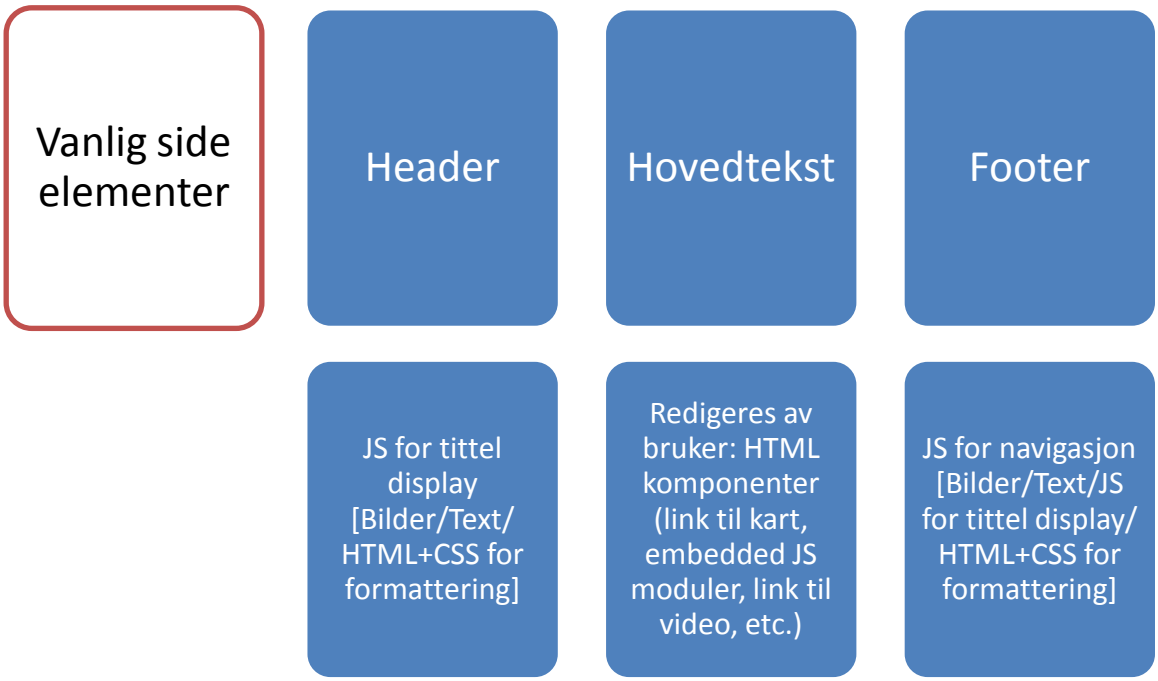
Alle komponenter må virke selv uten nettverkskobling, selv om det er med begrenset funksjonalitet, såkalt «graceful degradation».

3.4.Maler: filer og grensesnitt

MLAB benytter maler som automatiserer en del vanlige operasjoner i appen. En typisk mal vil bestå av design og logikk komponenter samt et område hvor brukeren legger til egen informasjon, dette lages med HTML5/CSS og Javascript/jQuery mobile. Se Figur 5 og Figur 6 for eksempler. Maler vil lages av utviklere og/eller designere som må følge visse regler så malene fungerer innenfor MLABs grensesnitt, dette diskuteres i detalj i neste avsnitt.



Figur 5 Eksempel på forside mal



Figur 6 Eksempel på vanlig side mal

- Alle filer for en mal er i en mappe som er navngitt etter komponenten i små bokstaver, i dette eksempelet blir dette h1.

Mappe	Filnavn	Forklaring
malnavn	forside.html	Brukes for den første siden ("landing page") i en app. Kan for eksempel inneholde hjelpetekst eller en liste over all sidene i appen.
	side.html	Brukes for alle andre sider i en app. Kan for eksempel inneholde topp-og bunntekst.
	.mlabinfo	Brukes for konfigureringsdata, i første omgang en liste over brukergrupper som har tilgang til denne malen.
malnavn/bilder	*	Bilder som inngår i malen
malnavn/css	*	Må minst ha index.css (generell Cordova CSS fil), app.js (CSS for denne appen), mlab-1.0.css (hjelpefil for å fikse formatteringsproblemer i MLAB, m.a.o brukes ikke i appen når den kjøres på den mobile enheten). Andre CSS filer kan legges til ettersom det er nødvendig.
malnavn/js	*	Må minst ha index.js (generell Cordova CSS fil), api.js (se neste avsnitt), app.js (JavaScript spesifikk for denne malen), jquery-1.8.3.min.js og jquery-mobile-1.2.min.js (jQuery /mobile basis filer). *Kan ha undermapper for bilder og biblioteker som brukes i malen.*

- En fil som heter api.js skal utvikles, den er felles for alle maler og vil inkludere de følgende API basis funksjonene, disse vil bruke Cordova funksjoner/SQLite database funksjoner: [4]
  - getName (returnerer navnet på sluttbrukeren via Cordova API)
  - setResult (lagrer data fra et associative array i et namespace som defineres av appen, brukes av spørreskjema, diskutert nedenfor)
  - getResult (returnerer et associative array som matcher det som var lagret)
  - setState (lagrer data fra et associative array i et namespace som defineres av appen, brukes for metainformasjon som posisjon i appen, etc)
  - getState (tilsvarende forrige punkt, men returnerer informasjonen)
  - getAllStates(tilsvarende forrige punkt, men returnerer ALLE informasjonen)
  - setConfig(lagrer data fra et associative array i et namespace som defineres av appen, brukes for konfigurasjon informasjon som font størrelse, etc)
  - getConfig (tilsvarende forrige punkt, men returnerer spesifisert konfigurasjonssetting)
  - getAllConfig (tilsvarende forrige punkt, men returnerer ALLE konfigurasjonssettinger)
  - loginRemotely / logoffRemotely (logger på/av en tjener)
  - loginToken (returner token, eller false hvis ikke logget inn)

[4] Se http://docs.Cordova.com/en/2.3.0/cordova_storage_storage.md.html#Storage

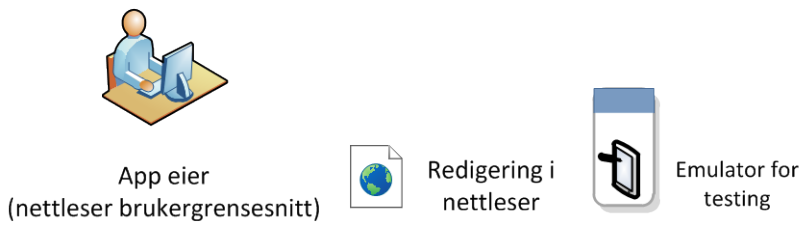
- sendRemoteData (sende data i et associative array til en kilde definert i en setState funksjon, for eksempel http://server.com/do_something.php?action=save_data eller mysql://user:password@dbserver.com/db/table)
- getRemoteData (tilsvarende forrige punkt, men returnerer informasjonen)
- onAppReady (bruker jQuery funksjoner til å kjøre denne funksjonen når appen er ferdiglastet, den skal så se etter om de følgende funksjonene finnes og kjøre disse når appen er ferdig lastet: initialiseApp, initialiseComponent. Med denne funksjonaliteten kan komponenter som f. eks. må logge seg inn vite at koden kjøres automatisk så lenge de har en funksjon som heter initialiseComponent i koden)

- I tillegg kan det være at man må legge til forbehandlingstrinn når appen kompileres. Disse må skrives i et skriptspråk som kan kjøres på tjeneren før programmet kompileres og må kalles ifra en fil som ligger i roten av mal mappen og heter *preprocessing.php* (filen kan gjerne kalle opp andre skripter i andre språk).
- **Alle maler må virke selv uten nettverkskobling, selv om det er med begrenset funksjonalitet, såkalt «graceful degradation».**

4. Hva er gjort og hvilke nye funksjoner må utvikles

Som tidligere nevnt, så eksisterer det allerede et demonstratorsystem. Det er dette systemet som videreutvikles. Videre er en oversikt over hva som er gjort og hva som må videreutvikles. Videre i dette dokumentet er det beskrevet mer konkrete utviklingspunkt.

A MLAB, app eier (AppBygger) funksjonalitet



A.1 **Ny funksjonalitet/forbedringer**: Grunnleggende sikkerhet må være implementert.
Når noen skal lage en app så må de gå igjennom en standard autoriseringsprosess når de kommer til forsiden på nettapplikasjonen (det samme skjer hvis de forsøker å gå til en annen side uten å være logget inn). Denne skal bruke både individuelle og grupperettigheter for senere å avgjøre hvilke apper brukeren har tilgang til, og hva de kan gjøre med disse appene. En brukermodell må implementeres (se 2.6 Brukermodeller og autentisering), en alternativ LDAP autentisering må også implementeres**.**

Etter at dette er gjort vil de ha tilgang til en meny hvor de kan gjøre det følgende:

A.2 Starte en ny app basert på en mal

Implementert: Cordova brukes fra kommandolinjen[5] til å generere ett nytt prosjekt som lagres i en mappe med samme navn som appen (dette er angitt av brukeren). I denne mappen er det en undermappe (/www/), her kopieres to sett med filer. Det første er alle filene i malen som utgjør skjelettet av det som vil bli HTML/CSS/JS filene i appen, her er det en forside mal som kopieres til index.html og en generell mal som kopieres til 001.html (se **Maler: filer og grensesnitt** for en detaljert oversikt over malenes oppbygning). Det andre er SiteCake filene som tillater redigering av filene (dette må kopieres for hver app, grunnet sikkerhetshensyn så vil Linux hard linker ikke virke).

Ny funksjonalitet/forbedringer: Legge til informasjon om hvilke grupper som har tilgang til appen, dette gjøres via en liste hvor individuelle grupper kan velges og lagres i .mlabinfo filen i roten av app mappen).

Ny funksjonalitet/forbedringer: Sjekke om brukeren (basert på gruppetilhørighet som fåes i fra logg-inn detaljene) har adgang til denne malen iht. informasjonen i .mlabinfo file i rotmappa for malen. Bare maler som de kan bruke skal vises i menyen.

A.3 Kopier en app

Implementert: Kopierer en app mappe med alt innhold til en en mappe med samme navn som den nye appen (dette er angitt av brukeren), oppdater <title> tag i HTML header.

Ny funksjonalitet/forbedringer: Sjekke om de (basert på logg inn detaljer) har adgang til denne appen. Bare apper som de kan redigere skal vises i menyen. Sjekke at de ikke bruker eksisterende navn, advare mot overskriving av eksisterende app.

A.4 Åpne en app

Implementert: Legger bare til en cookie som sier oss hvilke app som er den nåværende appen. Åpner så index.html filen (forsiden) for redigering i SiteCake.

Ny funksjonalitet/forbedringer: Sjekke om bruker (basert på logg inn detaljer) har adgang til denne appen. Bare apper som de kan redigere skal vises i menyen.
Låse appen så bare en person kan redigere den på samme tidspunkt, dette gjøres via en enkel låsefil som lagres i appens rot mappe. Låste apper skal markeres med ett ikon i app menyen.

A.5 Ny Side

Implementert: Kopierer mal siden fra mal mappen og gir den et fortløpende navn (002.html, 003.html, osv.), setter en cookie som sier hvilke side som er den nåværende siden, åpner så filen for redigering i SiteCake, lagring av endringer gjøres av SiteCake og er implementert.

[5] Se http://docs.phonegap.com/en/2.3.0/guide_command-line_index.md.html for oversikt over kommandoer

A.6 Åpne en side

Implementert: Legger til en cookie som sier hvilke side som er den nåværende siden, åpner så filen for redigering i SiteCake, lagring av endringer gjøres av SiteCake og er implementert.

A.7 Slette en side

Implementert: Sletting av side (resterende sider er **ikke** renummerert).

Ny funksjonalitet/forbedringer: Legg til sjekk for koblinger til slettet side, advar app eier om koblinger finnes.

A.8 Redigere prosjekt egenskaper

Ny funksjonalitet/forbedringer: Cordova har en tekstkonfigurasjonsfil som forteller kompilatoren hvilke rettigheter en app trenger. Standardinnstillingen er at alle rettigheter er inkludert. Disse rettighetene må kunne redigeres av brukeren via en side som bruker avkrysningsruter for å velge relevante rettigheter, for så å lagre detaljene på tjeneren. I tillegg bør den settes til ingen rettigheter når ett app prosjekt først settes opp.

Som en del av disse egenskapene så kan de også legge til nøkkelord, velge en eller flere kategorier som appen passer inn i, samt skrive en kort (150 ord) beskrivelse av appen. Denne informasjonen lagres i .mlabinfo filen i appens rot mappe. Denne informasjonen vil så bli brukt når appen publiseres på app markedet (se C.5 Grensesnitt for app publisering).

A.9 App versjoner

Ny funksjonalitet/forbedringer: En app må kunne ha nye versjoner (og versjonsnummer) men med den gamle kodebasen intakt. I første omgang gjøres dette ved å kopiere en app mappe til en ny mappe med «#n.n» etter app navnet, hvor n.n er versjonsnummeret. Brukeren vil bli spurt om dette er en stor (major) eller mindre (minor) endring, og nummeret vil bli generert automatisk.

A.10 App testing

Ny funksjonalitet/forbedringer: For å forenkle testing av apper så skal en link gjøres tilgjengelig som laster appen inn i en Ripple emulator[6] som skal installeres på MLAB tjeneren. Ripple er en online emulator hvor skjermstørrelsen til forskjellige mobiltelefoner emuleres og HTML5 apper kan testes, den kjøres inne i en nettleser. Pga. «Cross Origin» blokkeringer av de fleste nettlesere så *kan* dette bety at innholdet i /www mappen kopieres til en annen mappe under Ripple mappen for sjekking og slettes etterpå, dette må sjekkes først og relevant kode må utvikles.

[6] Se https://github.com/blackberry/Ripple-UI og https://chrome.google.com/webstore/detail/ripple-emulator-beta/geelfhphabnejjhdalkjhgipohgpdnoc

Ny funksjonalitet/forbedringer, alle videre utviklingspunkter er nye

A.11 

Grensesnitt for kompilering av apper

Kompileringen vil skje via skripter på de kompilatorserver som startes eksternt fra webapplikasjonsserveren. For all kompileringsfunksjonalitet må ett grensesnitt utvikles så fremtidige mobile plattformer kan legges til uten videre programmering i MLAB. Dette grensesnittet må tilby følgende funksjoner til MLAB via HTTPS:

- Last opp ZIP pakket app kode (for kompilatorer som ikke virker på Linux)
- Kompiler debug versjon
- Kompiler endelig (ikke debug) versjon
- Last ned debug versjon
- Last ned endelig (ikke debug) versjon
- Relevante feilmeldinger/koder må også utvikles.

I første omgang skal dette grensesnittet implementeres på Linux (for Android) og Mac OSX (for iOS) En enkel implementasjon (uten grensesnitt!) for å kompilere Android versjonen av appen via Cordova er utviklet. Koden her kan brukes som start på et grensesnitt.

A.11.1. Cordova for Android SDK må settes opp så endelige APK filer (ikke debug) kan kompileres. Dette er en lite dokumentert prosedyre, her er komplette detaljer: http://chris-allen-lane.com/2012/12/Cordova-compiling-a-release-apk-without-using-Cordova-build/ (Dette trengs fordi Cordova/Android SDK som standard bare lager debug versjoner, den endelige APK filen for distribusjon lages når man laster appen opp til Google Play/Android Market).

A.11.2. Cordova er et rammeverk som virker på de fleste smarttelefoner. For å kompilere apper for iOS enheter så må imidlertid en Apple Mac OSX enhet brukes.[7] Dette betyr at app mappen må lastes opp til OSX enheten (som vil kjøres i «headless» modus, m.a.o. ingen bruker intervensjon) for så å kompileres via en lokal Cordova installasjon. Dette må skje via SFTP/SSH og relevante OSX kompatible skripter må utvikles for å kompilere appen og laste den ned til MLAB tjeneren.

A.12 Online hjelp: Legg til en Hjelp meny som vil åpne en ny URL i en flytende/flyttbar jQuery vindu via AJAX, innholdet vil bli skrevet av FFI.

A.13 

Skrive om SiteCake koden så den støtter et komponentbasert grensesnitt som beskrevet i avsnittet om 3.4 Maler: filer og grensesnitt. I tillegg skal en del komponenter forbedres eller utvikles fra grunnen av:

[7] http://docs.Cordova.com/en/2.3.0/guide_getting-started_ios_index.md.html

A.13.1. Når legger til kart/video/fil, vis dialogboks:

Sitecake støtter bruken av Google kart, linker til Youtube videoer samt opplasting av filer. I den nåværende versjonen så må URLen limes inn i en tekst boks, men det er ingen indikasjon på at dette må gjøres, en tom tekstboks er lagt til. Dette skal erstattes med en relevant dialog boks (enter for en string eller for en filvelger boks) som vises umiddelbart.

A.13.2. Støtt integrert video

Sitecake støtter bare linker til YouTube. Det er ønskelig å legge til støtte for å bruke et standard rammeverk for visning av video som er kompilert inn i appen. Dette må være kompatibelt med både iOS og Android. See https://github.com/macdonst/VideoPlayer for en plugin for dette.

A.13.3. Støtt oppsett av spørreskjemaer via en komponent[8]

Et spørreskjema vil bestå av en eller flere disse HTML elementene: «input checkbox» (for Ja/Nei svar), «select nedtrekksmeny» (for flervalgsspørsmål), «select multiple» (for flere svar til ett spørsmål), «input text» (hovedsakelig for nummer, også for fri form svar med bare et mulig svar, for eksempel «hvem var vikingenes tordengud») og en «submit» knapp per skjema som vil kjøre en Javascript funksjon som heter getCorrectResults(). Hvert HTML element vil ha en «data-response» attributt som har svaret på spørsmålet, denne må kunne oppdateres via en dialogboks når skjemaet designes. I tillegg må det være støtte for jQuery faner (tabs), disse fanene vil være synlige i MLAB modus, men vil gjemmes og Forrige/Neste knapper skal vises i stedet i app modus.

A.14 Integrere SiteCake (redigering) meny med hovedmeny:

Sitecake har sin egen «flytende» meny som må integreres med MLAB's meny (som er festet til toppen av siden) for å gjøre den enklere å bruke. Dette dreier seg stort sett om å redigere HTML og CSS så den alltid er synlig på toppen av siden.

A.15 

Grensesnitt for konvertering av opplastede filer

Mye av informasjonen som brukes til opplæring finnes i forskjellige filformater. Et enkelt grensesnitt som tillater bruk av plugins for konvertering av filer basert på MIME typen skal utvikles. Dette skal tilby følgende funksjoner

- Last opp fil (med mulighet for ZIP pakking)
- Konverter fil til en side
- Konverter fil til mange sider (del opp etter hver side i filen som lastes opp)
- Angre konvertering (sletter sider som er lagt til)

I første omgang skal en implementasjon for PowerPoint 97-2003 formatet utvikles. For å gi mulighet til gjenbruk av dette så skal en importfunksjon som baseres på det åpne kildekode

[8] Se http://mobile.tutsplus.com/tutorials/mobile-web-apps/build-a-jquery-mobile-survey-app-app-logic-interface/ for en nyttig start på dette arbeidet.

pptHTML programmet brukes. I hovedsak så vil dette tillate opplasting av en fil i PPT 97-2003 formatet som så konverteres med pptHTML, og de resulterende filene blir kopiert inn i /www mappen for den nåværende appen og gis nye navn for å passe inn med index.html/001.html/002.html formatet.

B MLAB administrasjon (alle punkter her er nye funksjoner)

App-administrator
(nettleser brukergrensesnitt)

Administrere
brukere

Laste opp maler,
komponenter, etc

Figur 7 MLAB administrasjon

Det er ingen nåværende administrasjonsfunksjonalitet i MLAB, funksjonene beskrevet nedenfor må derfor alle utvikles fra grunnen av. Dette vil i hovedsak bestå av kode som samhandler med forskjellige tjenester via definerte grensesnitt.

B.1 Det må kunne opprettes brukere og grupper med forskjellige rettigheter (se 2.6 Brukermodeller og autentisering). Til å begynne med så vil dette kun være to nivåer, vanlige brukere (app eiere) og administratorer. I tillegg kommer forskjellige, vilkårlige (dvs. gruppenavn og medlemmer har ingen implisitt mening, det er bare for å skille de fra hverandre) app eier grupper som vil bestemme hvilke apper og maler en bruker kan jobbe med.

B.2 Via komponentgrensesnittet skal individuelle komponenter kunne lastes opp og aktiverer/deaktiveres for forskjellige grupper (eller for alle brukere). Komponentene er lagret i ZIP arkiver og pakkes ut i komponent rot mappen, se 3.3 Redigeringsverktøy og komponenter: filer og grensesnitt for mer informasjon.

B.3 Via malgrensesnittet skal individuelle maler kunne lastes opp og aktiverer/deaktiveres for forskjellige grupper (eller for alle brukere). Maler er lagret i ZIP arkiver og pakkes ut i mal rot mappen, se 3.3 Maler: filer og grensesnitt for mer informasjon.

B.4 Sletting av maler/prosjekter: Relevante advarsler må legges inn, filene som tilhører de slettede prosjektene skal ikke slettes fysisk men flyttes inn i en undermappe som heter «.slettet». Det må også være mulig å gjenopprette «slettede» elementer.

C App marked

App markedet i demonstratorsystemet støtter kun Android og er i hovedsak en installasjon av F-Droid[9] som benytter F-Droid sin native Android klient og deres samling av Python skript for å oppdatere et bibliotek basert på Android app-pakker og metadata på serveren.

Det må utvikles et marked som integrerer F-Droid teknologien. F-Droid har ikke noe nettgrensesnitt mot sluttbrukeren som kan benyttes, bare et maskinlesbart grensesnitt som klienten benytter. Det må implementeres et nettgrensesnitt som lar sluttbrukerne se igjennom beskrivelse av appene, gi tilbakemeldinger/kommentarer på appene, og lese andre sluttbrukeres vurderinger av appen. En app vil ha en enkelt side som fronter appen, uavhengig av operativsystem, og som gir brukeren tilgang på appen pakket for alle tilgjengelige operativsystemer som MLAB systemet støtter.

C.1 Implementere en brukermodell med grupper (se også administrasjon). App markedet er en modul som skal kunne fungere uavhengig av resten av MLAB systemet og modulen implementerer sin egen brukermodul. Denne modulen skal også kunne konfigureres til å benytte LDAP for autentisering av brukere og gruppetilhørighet, men dette er kun en opsjon. Anonyme brukere må være tillatt, men da kun med tilgang til apper som er tilgjengelig for den gruppen.

C.2 En forside som gir sluttbrukerne en oversikt over tilgjengelige apper for sluttbrukerens grupper. Det skal være mulig å søke etter apper. Siden skal ha en oversikt over nye apper, apper i nye versjoner, og populære apper (ift nedlasting). Det skal være mulig å sortere apper etter kategorier og nøkkelord (tags). Det skal også være en oversikt over anbefalte apper, som gjelder for brukergruppen, og som administrator har mulighet for å justere.

C.3 Det skal være individuelle sider for appene. En app vil ha en enkelt side som samler informasjon om appen, har lenker til nedlasting av appen (for alle tilgjengelige operativsystemer) over http(s). Denne siden har også en aggregert vurdering (1-5 stjerner) av appen gjort av andre brukere og kommentarer/tilbakemeldinger. Det skal utvikles muligheter for å gi en tilbakemelding og en vurdering av individuelle apper.

C.4 Den tilgjengelige F-Droid klienten skal kunne benyttes mot app markedet for Android applikasjoner. App markedet som utvikles skal kunne serve informasjon og app-pakker til denne klienten. For iOS er ingen klient tilgjengelig, så appene må lastes ned med web-linker[10].

C.5 

Grensesnitt for opplasting av app til app marked

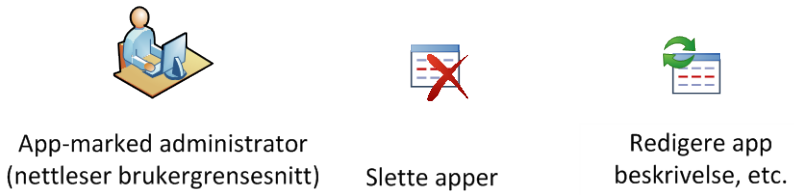
De følgende funksjonene må være tilgjengelig som API:

- Last opp app fil med XML beskrivelse og skjermbilder (returnerer en unik ID som kan brukes til å referere appen senere), dette kan gjøres i en ZIP pakket fil.

[9] Se http://f-droid.org/manual/fdroid.html for mer informasjon
[10] Se http://help.apple.com/iosdeployment-apps/mac/1.1/#app43ad6a6a for mer informasjon

- Få unik ID basert på app navn og versjonsnummer, hvis versjonsnummeret ikke er spesifisert returneres IDen for den siste versjonen.
- Last opp ny versjon av eksisterende app (bruker den unike IDen som referanse).
- Last opp ny XML beskrivelse for eksisterende app (bruker den unike IDen som referanse).
- Last opp skjermbilder for eksisterende app (bruker den unike IDen som referanse).
- Sperr app: sletter ikke appen men den merkes som ikke synlig. Allerede installerte apper beholdes på de mobile enhetene.
- Slett app: appen merkes som ikke synlig (sperres) og allerede installerte apper SLETTES på de mobile enhetene neste gang app markedet kontaktes.

D App marked administrasjon (alle punkter her er nye funksjoner)



Figur 8 App marked administrasjon

D.1 Redigere beskrivelse, nøkkelord og kategorier: Dette vil bruke den samme formen som app-eieren, men vil redigeres direkte på app markedstjeneren.

D.2 Apper kan også sperres eller slettes fra app markedet, hvis dette skjer må en epost sendes til app eieren via epost om hvorfor en app har blitt sperret/slettet. Administratoren vil alltid bli spurt om bekreftelse

D.3 Testfunksjon: En link til filen vil være tilgjengelig så administratoren kan laste ned filen og teste den på en mobil enhet.

E Basismal

Over tid vil en rekke maler bli utviklet for forskjellige avdelinger i Forsvaret. Først skal en basismaler utvikles (basert på FFIs demonstrator) som skal brukes som «mal for maler» (se **3.4 Maler: filer og**

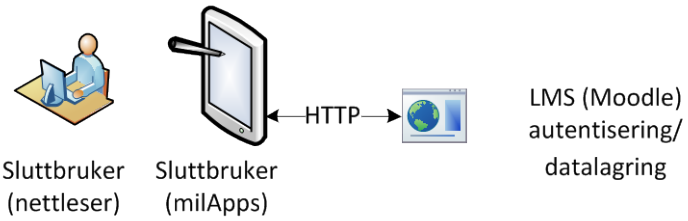
grensesnitt for mer informasjon). Forsvarets visuelle profil skal være utgangspunkt for den malen[11] som utvikles. Malen skal inneholde følgende elementer:

E.1 En forside som inneholder en automatisk generert indeks som lister tittelen på hver side og som åpner denne siden når brukeren trykker på tittelen.

E.2 JavaScript funksjon som lagrer sist leste side og går til denne siden når man åpner appen neste gang.

E.3 Navigering mellom undersidene i et skjema: Skjemaene vil lages med jQuery faner (tabs), disse skal gjemmes (av design og plasshensyn) og istedenfor skal faste navigasjonsknappene i bunnteksten brukes til å bla igjennom hver side i skjemaet.

F Komponenter og Integrering med eksterne tjenester: Experience API



Over tid vil komponenter bli utviklet som kobler seg til eksterne tjenester for å tilby sluttbrukerne nye muligheter. Dette kan være ting som å sende meldinger via Twitter eller søke kart via Google Maps. I tillegg til de komponentene som er nevnt i 3.3 Redigeringsverktøy og komponenter: filer og grensesnitt, skal en komponent utvikles i dette prosjektet som foretar en enkel kobling mot en Learning Record Store (LRS) via Experience APIet (også kalt Tin Can API). Denne komponenten har ikke noe visuelt grensesnitt, men vil kjøres automatisk når appen først startes, brukeren av appen når siste side eller for eksempel har svart riktig på et sett med spørsmål.

Experience APIet[12] er spesielt utviklet for å logge læringsaktiviteter i en LRS. API et er fortsatt under utvikling, og endelig versjon det skal utvikles for er ennå ikke bestemt. I all hovedsak så muliggjør dette APIet kobling mellom 3. parts systemer (appene som lages i MLAB) og et sentralt LRS. I MLABs henseende så skal dette først og fremst benyttes for logging av gjennomførte aktiviteter og lagring av svar på spørreskjema. For dette prosjektet så er målet interaksjon med Moodle, en åpen kildekode Learning Management System (LMS) [13] som har en LRS komponent og mulighet til å motta Experience API meldinger som en plug-in. Disse funksjonene skal lages så generisk som mulig så koden kan gjenbrukes for fremtidige større prosjekter og skal skrives i Javascript/jQuery.

F.1 Autentisering

Enkelte apper vil benytte lagringsmuligheter i Moodle, før dette kan gjøres så må brukeren

[11] Se http://forsvaret.no/om-forsvaret/fakta-om-forsvaret/forsvarets-visuelle-profil/Sider/Forsvarets-visuelle-profil.aspx
[12] Se http://tincanapi.com/overview/
[13] Se https://moodle.org/

logge inn. Når dette er gjort lagres eventuelle «tokens» inntil brukeren logger ut manuelt men sjekkes med Moodle tjeneren når appen startes på nytt. App brukeren skal m.a.o. ikke trenge å logge inn hver gang hun åpner appen, men en sjekk skal foretas så en en mobilenhet på avveie kan blokkeres.

Dette vil bruke APIet definer i api.js (se relevant paragraf under **3.4 Maler: filer og grensesnitt** for mer informasjon) samt vanlige Moodle funksjoner på tjeneren.

F.2 Lagring og lesing av sporingsdata (hvor langt har de lest/hva har de sett, f. eks.)
Dette lagres som en «assosiative array» som apper kan benytte uten begrensninger og uten at appen vet noe om database strukturen i Moodle, men hver app må ha et separat lagringsområde innenfor hver brukers Moodle konto.

Dette vil bruke APIet definer i api.js (se relevant paragraf under **3.4 Maler: filer og grensesnitt** for mer informasjon) samt en ny Moodle modul som må utvikles som en del av dette prosjektet og kjøres på tjeneren.

F.3 Lagring av svar på spørreskjema
Dette vil virke på samme måte som lagring av sporingsdata, men per bruker/app/spørreskjema, m.a.o. ett ekstra namespace nivå må benyttes.

5. Generelle betingelser angående kompetanse, kode og koding

- Leverandører som vurderer å utvikle dette prosjektet må ha bevisbar kompetanse relatert til teknologiene som brukes her (se 3.1.1 MLAB nettleser brukergrensesnitt og 3.1.2 MLAB tjenere) og ha erfaring i gjenbruk av åpen kildekode i fra CMS utvikling/utvidelse av CMS verktøy, samt tidligere bruk av APIer. Eksempler på tidligere arbeid er påkrevet.
- FFI og ADL/FHS skal eie all kode som utvikles i prosjektet og resulterende systemer, det er koders ansvar å verifisere at ingen kode eies av andre og/eller er åpen kildekode. Hvis dette ikke er tilfelle så holdes oppdragsgiver fri for ansvar av koder. Bruk av tredjeparts biblioteker eller annen funksjonalitet skal benyttes kun etter avklaring med oppdragsgiver.
- Kode og grensesnittene skal være godt dokumentert og strukturert for både PHP og Javascript. Førstnevnte skal bruke standard Zend konvensjoner for koding og ha phpDoc type kommentarer som kan brukes av Doxygen eller phpDocumentor for å generere dokumentasjon. Javascript må likeledes dokumenteres med JSDoc kommentarer og kodes etter jQuery standarden .
- Det skal ikke være begrensninger på koden som hindrer oppdragsgiver å legge den ut som åpen kildekode.
- All kode må fungere uten forringelse i funksjonalitet både på internett, lukket lokalnett .
- Alle moduler må være i stand til å tjene 200 samtidige brukere mens de kjøres i en virtuell maskin, et unntak er app markedet som må betjene 1,000 brukere av gangen.

Appendiks A: Ordforklaring

App:	Et selvstendig dataprogram for mobile enheter (f. eks. Samsung Galaxy S smarttelefon eller iPad).
App eier:	Bruker av MLAB systemet som lager en app for mobile enheter via et dra og slipp grensesnitt. Bruker som vedlikeholder og distribuerer sine applikasjoner.
Applikasjonsprosjekt:	Eller *app-prosjekt* er et prosjekt i MLAB systemet som representerer en mobil applikasjon (app). "Appens kildekode".
(MLAB) *administrator*:	Bruker som er ansvarlig for å administrere brukere, maler og komponenter i MLAB systemet.
App administrator:	En administrator i markedsmodulen. Kan f. eks være ansvarlig for å godkjenne og tilgjengeliggjøre apper til app markedet.
App marked:	Betegner en tjeneste tilsvarende iOSs App Store og Androids Play Store, den nåværende tjenesten er kalt milApps.
Komponent:	Element som kan brukes i en app av app eieren, kan være en enkel HTML type som Header 1 (<h1>) eller en komplett og kompleks kartfunksjon.
MLAB:	Foreløpig navn på applikasjonen/rammeverket som bygges her.
Mobilenhet:	Nettbrett, smarttelefon eller lignende enhet.
Modul:	En distinkt og separat enhet i MLAB kjeden som samvirker med andre enheter, og kan byttes ut med andre enheter uten at de andre modulene slutter å virke.
Sluttbruker:	Brukere som benytter de ferdige appene på en mobilenhet.

Appendiks B: Mappestruktur for MLAB systemet

MLAB bruker en rekke åpen kildekode program som er «limt» sammen med egen kode. Det er i stor grad basert på et fil-og-mappe system, databaser brukes hovedsakelig for bruker autentisering. Denne strukturen gjennomgås nedenunder. Hovedmappene kan lagres på forskjellige fysiske eller virtuelle tjenere så lenge disse er tilgjengelig via SFTP eller Rsync eller som aktiverte volum.

mlab	**Webapplikasjon**
├── bilder	bilder brukt i webapplikasjonen
├── css	CSS brukt i webapplikasjonen, vær forsiktig så navn her ikke er samme som er brukt i app maler
├── js	javascript brukt i webapplikasjonen
├── *.php	PHP filer med applikasjonslogikk/design
mlab-apper	**alle appene som bygges har en undermappe her**
├── Myapp	rot mappe for en app, vanlig layout m/ekstra Cordova assets/www mappe
│ ├── AndroidManifest.xml	«build configuration» for Android app, redigeres for rettigheter etc.
│ ├── ant.properties	
│ ├── assets	
│ │ └── www	Cordova rot mappe for HTML5 innhold, når en ny app lages så kopieres en filer hit fra den relevante mal mappen
│ │ ├── 001.html	denne og følgende sider kopieres fra /mlab-maler/mal/side.html
│ │ ├── bilder	Bilder som er del av malen (opplastede bilder er i image mappen)
│ │ ├── cordova-2.2.0.js	Standard fil som må kopieres fra /opt/Cordova/lib/OS_NAVN
│ │ ├── css	App CSS, kopieres fra mal folder med minimum følgende filene:
│ │ │ ├── app.css	CSS spesifikk for denne malen
│ │ │ ├── images	Hvis noen av CSS filene bruker bilder plasseres de her
│ │ │ ├── index.css	Generell Cordova CSS fil
│ │ │ ├── jquery.mobile-1.2.0.css	jQuery Mobile formatting
│ │ │ └── mlab-1.0.css	Hjelpefil for å fikse enkelte formatteringsproblemer i MLAB
│ │ ├── files	Opplastede filer, f. eks. videoer
│ │ ├── images	Opplastede bilder
│ │ ├── img	Bilder som er del av Cordova basis funksjonaliteten, f. eks. logo
│ │ ├── index.html	Start fil, Cordova virker ikke uten denne, kopieres fra /mlab-maler/app-mal/forside.html
│ │ ├── js	App Javascript, kopieres from mal mappe med minimum disse filer:
│ │ │ ├── app.js	Javascript spesifikk for denne malen
│ │ │ ├── index.js	Generell Cordova CSS fil
│ │ │ ├── jquery-1.8.3.min.js	jQuery hovedfil må innkluderes
│ │ │ ├── jquery-mobile-1.2.0.js	jQuery mobile hovedfil, maler burker kanskje andre filer også

│ │ ├── main.js	Cordova CSS, ignorert, deler av innholdet her må være i app.js
│ │ ├── master.css	Cordova CSS, ignorert, deler av innholdet her må være i app.css
│ │ ├── res	Ressurser som ikoner, etc for appen, legg inn nye ikoner i undermapper her
│ │ ├── sc-admin.php	SiteCake redigeringsverktøy, laster inn server kode
│ │ ├── sitecake	SiteCake kode, «client» koden brukes for å redigere sider, «server» koden brukes for å lagre forandringer, laste opp filer etc.
│ │ │ ├── 2.0.0-SNAPSHOT	
│ │ │ │ ├── client	
│ │ │ │ └── server	
│ │ │ ├── credential.php	
│ │ │ └── editor.cfg	
│ │ ├── sitecake-content	SiteCake innhold, dette er data som lastes opp etc.
│ │ ├── spec	Standard Android filer
│ │ └── spec.html	
│ ├── bin	Binære verktøy/kompilerte filer
│ │ ├── AndroidManifest.xml	
│ │ ├── AndroidManifest.xml.d	
│ │ ├── build.prop	
│ │ ├── classes	Inneholder standard Cordova/Android filer
│ │ │ └no	Java
│ │ │ └ffi	Path
│ │ │ └sinett	Defined
│ │ │ └mlab	For
│ │ │ └myapp	App
│ │ ├── Myapp-debug-unaligned.apk.dbg	Kompilerte program
│ │ └── res	
│ ├── build.xml	Build info
│ ├── cordova	Cordova kjørbare program/kompileringsverktøy/logger
│ ├── gen	Generert Java kode
│ ├── libs	
│ │ └── cordova-2.2.0.jar	Standard Cordova bibliotek, må kopieres hit fra
│ ├── res	Standard ressurser
│ └── src	Java kildekode, generert av Cordova
│ └── no	
│ └── ffi	
│ └── sinett	
│ └── mlab	
│ └── myapp	
│ └── Myapp.java	
mlab-maler	**Maler**

├── Land	Individuell mal rot mappe
│ ├── bilder	Bilder som inngår i malen
│ ├── css	CSS filer, trenger minimum disse filene:
│ │ ├── app.css	CSS spesifikk for denne malen
│ │ ├── images	Hvis noen av CSS filene bruker bilder plasseres de her
│ │ ├── index.css	Generell Cordova CSS fil
│ │ ├── jquery.mobile-1.2.0.css	jQuery Mobile formatting
│ │ └── mlab-1.0.css	Hjelpefil for å fikse enkelte formatteringsproblemer i MLAB (m.a.o ikk for bruk i appen når den kompileres)
│ ├── forside.html	Kopieres en gang, som index.html
│ ├── js	App Javascript, trenger minimum disse filene:
│ │ ├── app.js	Javascript spesifikk for denne malen
│ │ ├── index.js	Generell Cordova CSS fil
│ │ ├── jquery-1.8.3.min.js	jQuery basis file
│ │ ├── jquery-mobile-1.2.min.js	jQuery basis file, flere må muligens bli brukt for widgets
│ └── side.html	Kopieres for all sider etter forsiden, disse nummereres 001.html, 002.html, osv.
mlab-redigering	**SiteCake installasjon.**
├── sc-admin.php	Grunnet CrossDomain problemer må innholdet i denne kopieres inn i hver ny app mappe (under /assets/www).
└── sitecake	
├── 2.0.0-SNAPSHOT	
│ ├── client ├── component	Komponent folder, ikke del av nåværende SiteCake
│ └── server	
├── credential.php	
├── credential.php.default	
└── editor.cfg	

---DOKUMENT SLUTT---