Systemdokumentasjon

Monitorering av produksjonsløyper ved Nasjonalbiblioteket

- Project BAKE

Utarbeidet av:

Einar Wågan
Kristian Akerhei

Studium:

Informasjonssystemer

Innlevert:

26.5.2015

Innhold

1 Innledning

Denne rapporten er utarbeidet av Einar Wågan og Kristian Akerhei, som en del av en avsluttende oppgave på et treårig bachelorstudie i Informasjonssystemer ved Høgskolen i Nesna. Oppgaven er skrevet for avdelingen DBU-Drift ved Nasjonalbiblioteket.

Nasjonalbiblioteket er et norsk bibliotek som har mål om å være nasjonens hukommelse, og bevare alle norske publikasjoner i det offentlige rom. Dette inneholder de fleste medier som for eksempel bøker, aviser, radio og tv. Det er et mål for Nasjonalbiblioteket å digitalisere dette for å kunne lettere tilby det til offentligheten.

BAKE er en webapplikasjon som monitorerer produksjonsløypene på Nasjonalbiblioteket. Webapplikasjonen er skrevet i *Spring framework* og kjøres på en *tomcat* server.
I denne applikasjonen kan man søke opp objekter i produksjonsdatabasen og se en tidslinje over når objektet har vært innom de ulike stegene i produksjonsløypen. Man får også mye annen informasjon om objektet, blant annet de ulike IDene som er tilknyttet objektet.
Man kan se hvor mange objekter som ligger i hvert steg i hver produksjonsløype og hvilken status disse objektene har. Det er også mulighet for å se historikk på inntil et år i hver produksjonsløype. Kombinert med den grafiske framstillingen av objekter gjør dette det mulig og lettere analysere og finne flaskehalser i produksjonen.
Fullstendig kildekode for webapplikasjonen pluss imitert REST-tjeneste ligger på den vedlagte CDen (se Vedlegg 4)

2 Krav

I kravinnsamlingen kom det fram både funksjonelle og ikke funksjonelle krav. For de funksjonelle kravene ble det utarbeidet *user stories* fra Nasjonalbiblioteket. Disse ble laget av produkteier i samarbeid med et utvalg brukere på Nasjonalbiblioteket. De ikke funksjonelle kravene kom fram i samarbeid mellom prosjektgruppen og produkteier. Disse ble laget for å utfordre prosjektgruppen til å lage et bedre produkt men samtidig holdes innenfor rimelighetens grenser, slik at de skulle være gjennomførbare.

2.1 Funksjonelle krav:

Som funksjonelle krav er det brukt user stories. Disse viser hva brukerne etterspør av funksjonalitet i applikasjonen. Oppbygningen av en user story er:
Som <Brukerrolle> vil jeg <mål/krav> slik at <nytte>.

2.1.1 User story 1

"Som kontrollør vil jeg ha informasjon om antall objekter fordelt på stegene i produksjonsløypa i sanntid, slik at jeg til enhver tid kan få et øyeblikksbilde av produksjonen."

2.1.2 User story 2

"Som kontrollør vil jeg ha informasjon om hvordan objektene i produksjonsløypa er fordelt på status (ok, failed etc.) i sanntid, slik at jeg kan få et øyeblikksbilde av status for enkelt objekter i produksjon."

2.1.3 User story 3

"Som kontrollør vil jeg ha tilgang til trendkurver for antall objekter fordelt på stegene i produksjonsløypa slik at jeg kan se utviklingen over tid."

2.1.4 User story 4

"Som kontrollør vil jeg ha tilgang til trendkurver for hvordan objektene i produksjonsløypa har vært fordelt på status (ok, failed etc.) slik at jeg kan se utviklingen over tid."

2.2 Ikke funksjonelle krav:

De ikke funksjonelle kravene som ble satt i forprosjektrapporten har prosjektgruppen prøvd å oppnå. For å finne ut om disse var oppnådd er det blitt kjørt tester på kravene der det har latt seg gjennomføre. De kravene som ikke kunne testes men som handler mer om skjønn, har tilbakemeldinger vært avgjørende for om kravene har blitt oppnådd eller ikke.

2.2.1 Brukeropplevelse

Systemet skal ha et godt design, og skal oppleves som et lett og brukervennlig system.

2.2.2 Ytelse

Systemet skal oppleves som raskt og responsivt, det vil si at det ikke skal gå flere minutter å laste siden. Det skal heller ikke bruke unødvendig mye ressurser fra produksjonsmiljøet. Det skal ikke ta mer en 30 sekund å laste webapplikasjonen.

2.2.3 Drift og vedlikehold

Systemet skal være lett for Nasjonalbiblioteket (NB) og vedlikeholde. Det skal ikke brukes teknologier som drift/utviklingsavdelingen hos NB ikke har kjennskap til. Koden skal skrives slik at den kan gjenbrukes til andre prosjekter og applikasjoner hvis man ser behov for det.

2.2.4 Konsistens

I user stories blir begrepet sanntid brukt ofte, med sanntid i disse tilfellene menes det forholdsvis ny data. Dette systemet har ikke behov for data fra sanntid, vi vil derfor senke på kravet om sanntidsdata og heller fokusere på ytelse.

.

3 Teknologier

Alle teknologier som blir brukt i webapplikasjonen er beskrevet kort under.

3.1 Ajax

Står for Asynchronous JavaScript and XML, og er en teknikk innen web-utvikling som gjør at nettleseren og serveren kommuniserer med hverandre i bakgrunnen. Dette gjør at man slipper å laste siden på nytt hver gang man foretar seg noe i nettleseren. En annen fordel med *Ajax* er at nettsiden vil føles mer responsiv. Webapplikasjonen benytter *AJAX* i alle *JavaScriptene.*

3.2 CSS

CSS står for Cascading Style Sheets, og er et språk som definerer hvordan *HTML* elementer blir presentert. Blir brukt for å lage designet på webapplikasjonen. *CSSen* som er brukt i webapplikasjonen er hentet fra www.nb.no og modifisert til å passe denne løsningen.

3.3 HTML

Står for Hyper Text Markup Language og er et standard språk for å lage nettsider. HTML er brukt i webapplikasjonen for å presentere data i nettleseren.

3.4 Java

Java er et open source og veldig utbredt objektorientert programmeringsspråk. *Java* er ikke plattformavhengig, og kan kjøres på alle plattformene som støtter Java Virtual Machine. Det er benyttet *Java* på all kode som ikke kjøres hos brukerne.

3.5 JavaScript

JavaScript er et programmeringsspråk for html og web. Det er en av de mest populære programmeringsspråkene og de fleste nettleserne har støtte for det. Fordelen med *JavaScript* er at de kan gjøre nettsidene interaktiv, og dynamiske. *JavaScript* er benyttet på for å kjøre kode i nettleseren hos brukeren.

3.6 Maven

Apache Maven gjør jobben for utvikleren lettere, siden det er et verktøy som kan administrere og bygge et hvilket som helst J*ava* prosjekt. Det kan ta seg av avhengigheter i eksterne java-bibliotek, og utvikleren skal slippe bryet med dette, og på denne måten bli mer produktiv. Maven er brukt for å hente biblioteker som webapplikasjonen er avhengig av, men også for å bygge webapplikasjonen til et kjørbart format.

3.7 Spring framework

Spring framework er et open source rammeverk for *Java*. Det er veldig omfattende og kan brukes til blant annet enkle web-sider, transaksjonshåndtering, datatilgang, testing, *REST-tjenester* og mye mer.

En av de største fordelene med *Spring framework* er at det har støtte for "dependency-injection", som gjør det lettere å knytte sammen applikasjonen. Webapplikasjonen *BAKE* er skrevet i *Spring framework*

3.8 Tomcat

Apache Tomcat er en open source applikasjonsserver. Det er Apaches implementasjon av Java Servlet og JavaServer Pages, slik at den kan kjøre javabaserte webapplikasjoner.

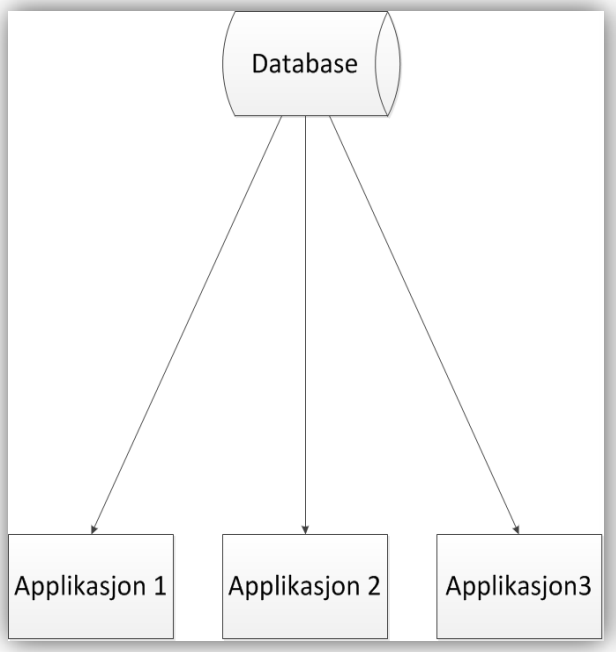
Webapplikasjonen *BAKE* kjøres på denne typen applikasjonsserver, og er kun testet på *Tomcat*.

4 Arkitektur

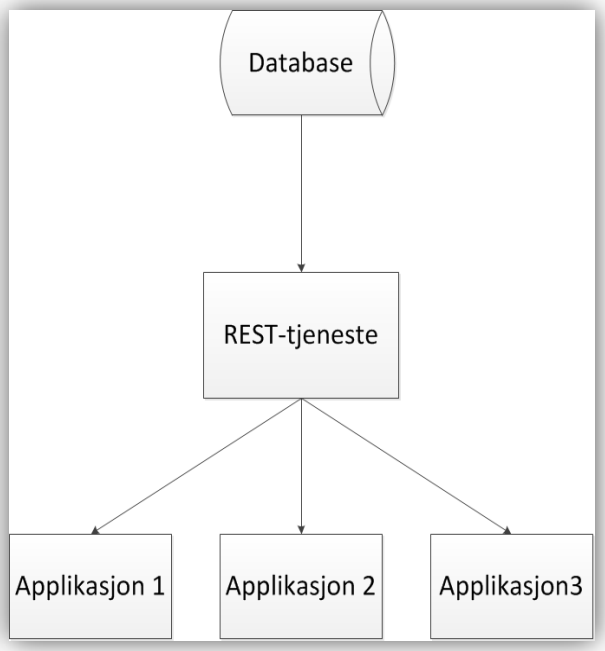
4.1 Overordnet arkitektur

Hvis flere applikasjoner går direkte mot en database(se Figur 6) kan dette gi store utfordringer ved endringer i denne, fordi man må skrive om hver enkelt applikasjon for å håndtere denne endringen.

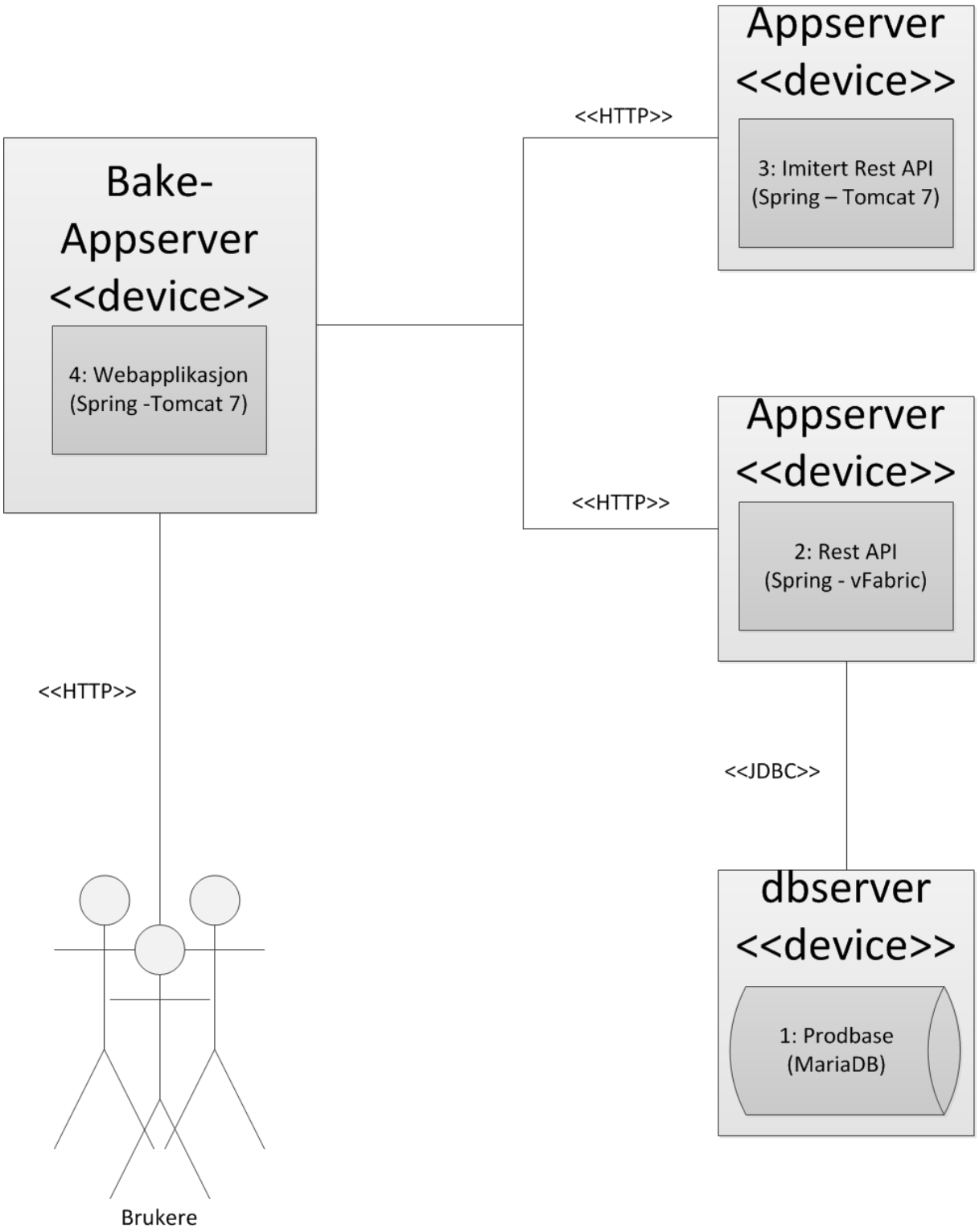
Arkitekturen til denne løsningen bygger på en applikasjon som snakker med *REST-tjenester*. Det finnes flere fordeler med å sette opp slike *REST-tjenester*. Ved å la applikasjonene gå mot denne typen tjenester får man et felles knutepunkt(se Figur 5), og et skille mellom data og applikasjon. På denne måten unngår man å måtte gjøre endringer i alle applikasjonene hvis det gjøres endringer i databasen. Man trenger derfor kun å endre i REST-tjenesten, og dermed slippe unna mye arbeid. Det ble derfor bestemt at applikasjonen skulle benytte den eksisterende REST-tjenesten hos Nasjonalbiblioteket.



Figur 1 - Arkitektur uten felles knutepunkt



Figur 2 - Arkitektur med felles knutepunkt



Figur 3 - Arkitektur

BAKE snakker med to *REST-tjenester* som du kan se i illustrasjonen (se Figur 7), den ene *REST-tjenesten* er Nasjonalbibliotekets applikasjon som snakker med produksjonsdatabasen. Den andre er *REST-tjenesten* er satt opp midlertidig, for å imitere manglende funksjonalitet i Nasjonalbibliotekets *REST-tjeneste*.

Forklaring til Figur 7:

1. Produksjonsdatabasen hos Nasjonalbiblioteket, kjører *MariaDB* som er en avart av *MySQL*. *MariaDB* er open-source og benytter *SQL* som språk.
2. *REST-tjeneste* utviklet hos Nasjonalbiblioteket, noe som betyr at i stedet for å kjøre spørringer direkte til produksjonsdatabasen, kan man bruke denne tjenesten. Den bruker *JDBC* for å kommunisere med *REST-tjenesten*. Spørringer til denne tjenesten går over *HTTP*, noe som gjør at du kan bruke nettleseren til å prøve spørringer, dette gjør testing mye lettere.
3. Imitert *REST-tjeneste*, denne er laget med data som blir tilfeldig generert, for å vise funksjonaliteten på webapplikasjonen.
4. Webapplikasjonen benytter *HTTP* for å snakke med *REST-tjenestene*, og bruker *HTTP* for å sende bilder og data til brukeren i form av vanlige nettsider, med noen *AJAX* kall i *JavaScriptene* i form av *JSON(AJAJ)*.

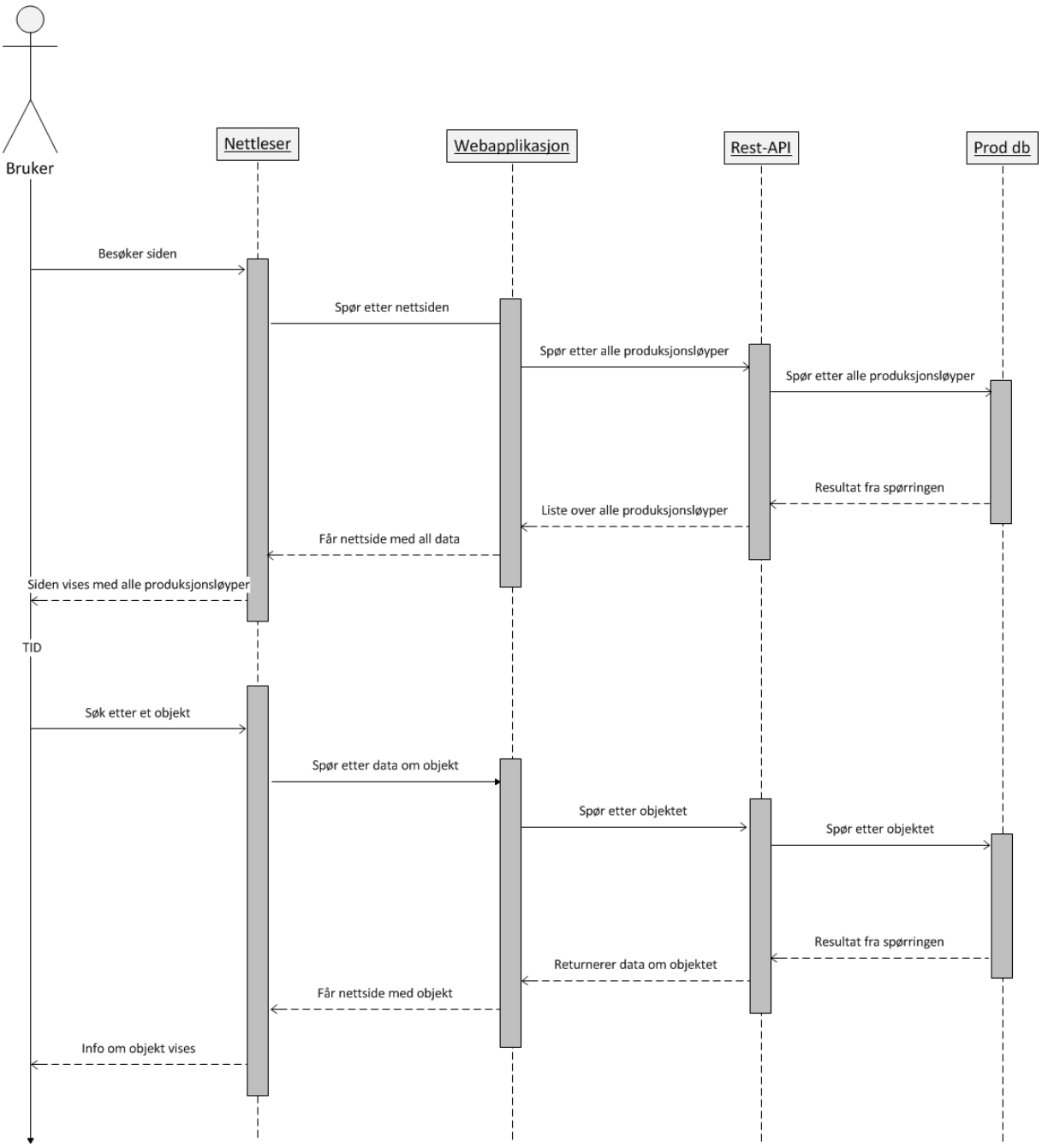
4.2 MVC

Webapplikasjonen er basert på en *MVC* tankegang, både innad i selve webapplikasjonen og i arkitekturen. I webapplikasjonen er det et skille mellom datamodellen, kontrollerne og det visuelle, slik at endringer i det visuelle ikke skal ha noen innvirkning på annen kode som kjører. Arkitekturen er bygd med samme tankegang, man har databasen som fungerer som datamodellen, *REST-tjenesten* som en kontroller, og webapplikasjonen som tar det visuelle.

4.3 Sekvensdiagram

Sekvensdiagrammet(se Figur 8) viser hvordan alle elementene i arkitekturen samhandler med brukeren. Når en bruker besøker siden vil nettleseren sende en *HTTP GET* melding til web applikasjonen, når dette blir mottatt vil applikasjonen hente data fra *REST- tjenesten.* Det blir da sendt en *HTTP GET* melding til *REST-tjenesten*. Når denne meldingen blir mottatt av *REST- tjenesten* spør den databasen med en *SQL* spørring, og resultatet blir sendt til *REST-tjenesten*. *REST- tjenesten* svarer web-applikasjonen, og kan gi data videre til nettleseren, som viser dette til bruker.

Når brukeren bruker noen av de andre funksjonene i applikasjonen, skjer det samme bare i stedet for vanlig *HTTP GET* fra nettleseren til applikasjon skjer et *AJAX* kall av typen *JSON* (*AJAJ*).



Figur 4 - Sekvensdiagram

4.4 Funksjonalitet

Her er en liste over funksjonalitet som er tilgjengelig i webapplikasjonen.

4.4.1 Objektsøk

Med denne funksjonaliteten kan man søke i produksjonsløypa for bøker. Hvis søket ditt returnerer bare et treff, kommer du direkte inn på informasjon om dette objektet. Hvis søket ditt returnerer flere treff vil du få se en liste over de aktuelle objektene. Her kan du velge et objekt, og du vil få informasjon om dette objektet. Når du har kommet inn på informasjonen om et objekt, vil du få frem en tidslinje over når objektene har vært i de ulike stegene. I tillegg vil du få frem alle de ulike IDene som er med lenker til eksterne sider, knyttet til dette objektet. Man kan da for eksempel trykke seg inn på *bibsys* sine sider hvis det er en *bibsys* id tilknyttet objektet.

4.4.2 Produksjonslinje

Med denne funksjonaliteten kan man velge seg en produksjonsløype, man blir da å få oversikt over alle stegene i produksjonsløypa, i tillegg til hvor mange objekter det ligger i hvert steg, fordelt på statusene Ready, Terminated og Failed.

4.4.3 Steg historikk

Med denne funksjonaliteten får du oversikt over historien for steget du har valgt, du får først historien på hvor mange objekter steget har hatt for et år. Du kan trykke deg helt ned til du får oversikt over en uke. I tillegg er det en trendlinje som blir beregnet ut fra tidligere data. Denne viser gjennomsnitt og et estimat på hvor mange objekter det vil være i steget fremover i tid.

4.4.4 Produksjonslinje historikk

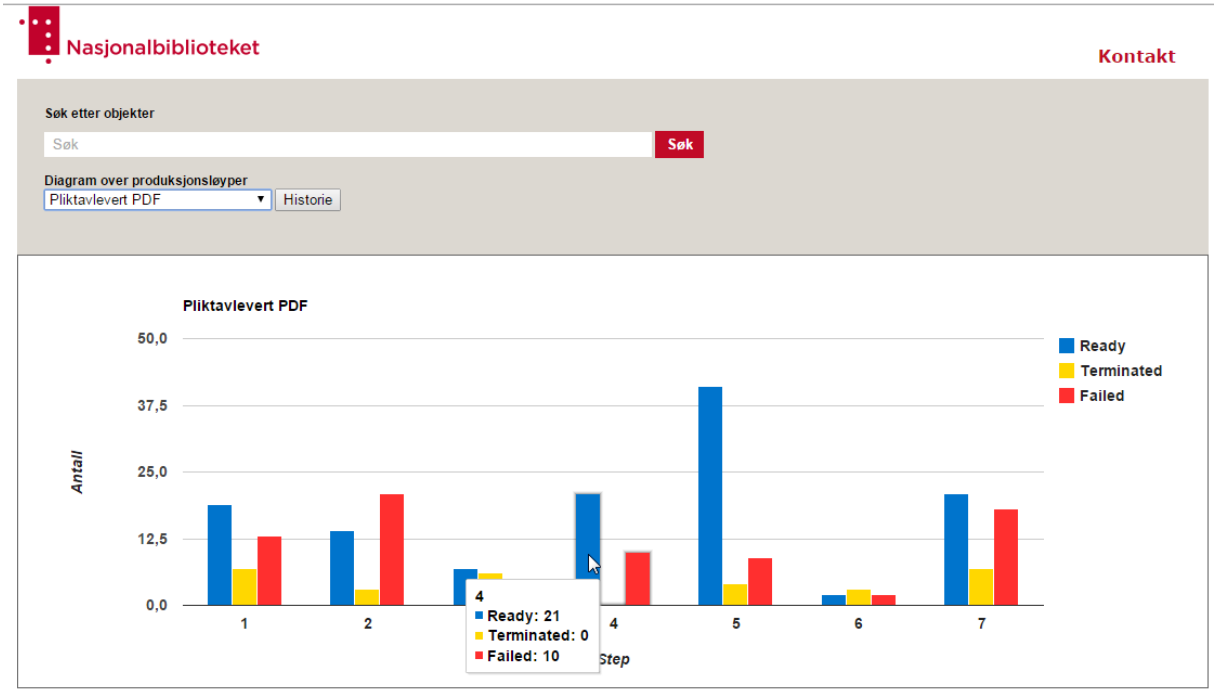
Med denne funksjonaliteten får du oversikt over en hel produksjonsløype, den viser et helt år med data over hvor mange objekter som har vært i produksjonsløypa. Det er i tillegg en trendlinje som viser gjennomsnitt og et estimat på hvor mange objekter det kommer til å være i produksjonsløypen i fremtiden.

5 Kravoppnåelse

5.1 Funksjonelle krav

5.1.1 User story 1

"Som kontrollør vil jeg ha informasjon om antall objekter fordelt på stegene i produksjonsløypa i sanntid, slik at jeg til enhver tid kan få et øyeblikksbilde av produksjonen."

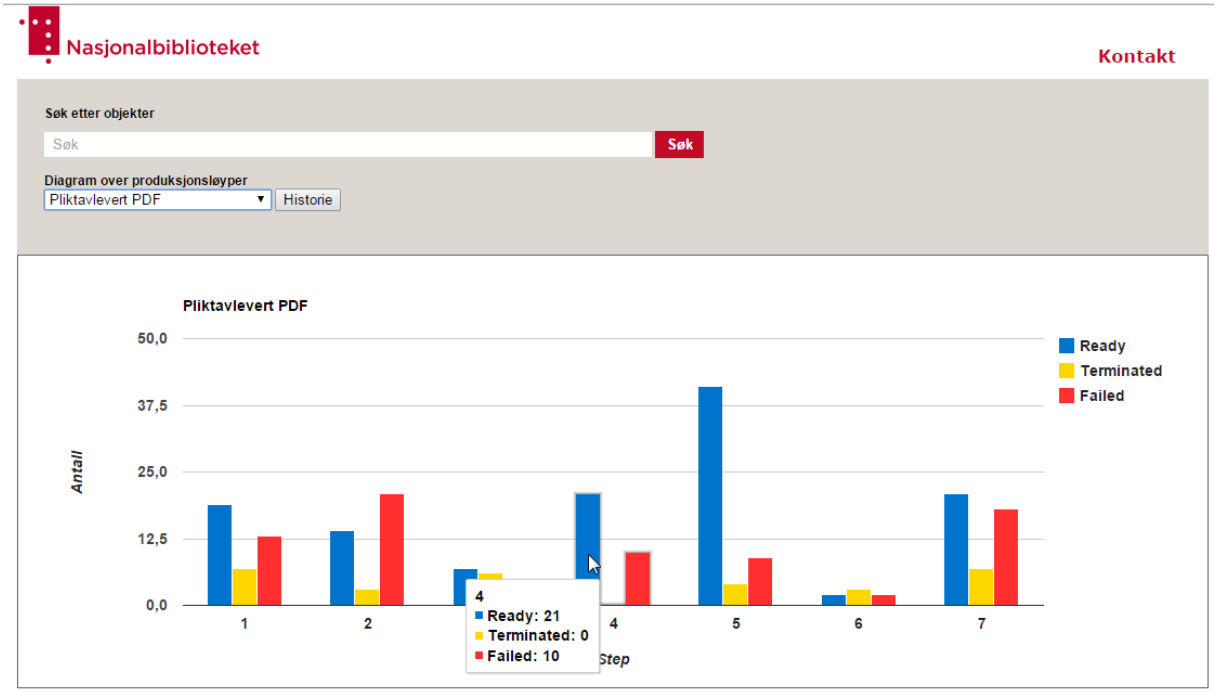


Figur 5 - User story 1

For å oppfylle denne user storyen er det valgt å visualisere dataen ved hjelp av et søylediagram. De er samlet i grupper der hver gruppe er et steg i produksjonsløypen og hver søyle representerer en status (ready, terminated og failed). Som man kan lese ut fra grafen kan man se hvor mange objekter som ligger i hvert steg, i eksemplet (se Figur 1) ser man det ligger 21 objekter med status ready, 0 med status terminated og 10 med status failed i steg 4.

5.1.2 User story 2

"Som kontrollør vil jeg ha informasjon om hvordan objektene i produksjonsløypa er fordelt på status (ok, failed etc.) i sanntid, slik at jeg kan få et øyeblikksbilde av status for enkelt objekter i produksjon."

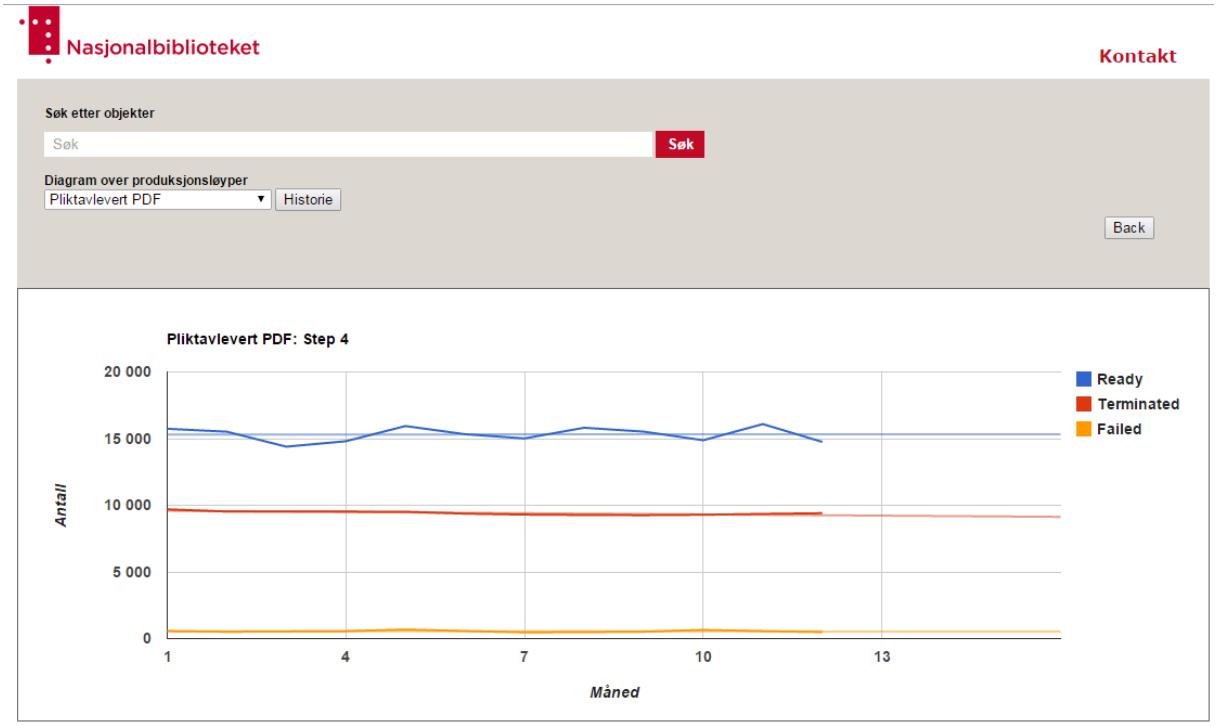


Figur 6 - User story 2

Denne user storyen er blitt innfridd av samme funksjonalitet som i user story 1, og data er organisert på samme vis. Som du kan se i eksemplet (se Figur 2) vises det hvor mange objekter som har hver av de tre statusene.

5.1.3 User story 3

"Som kontrollør vil jeg ha tilgang til trendkurver for antall objekter fordelt på stegene i produksjonsløypa slik at jeg kan se utviklingen over tid."

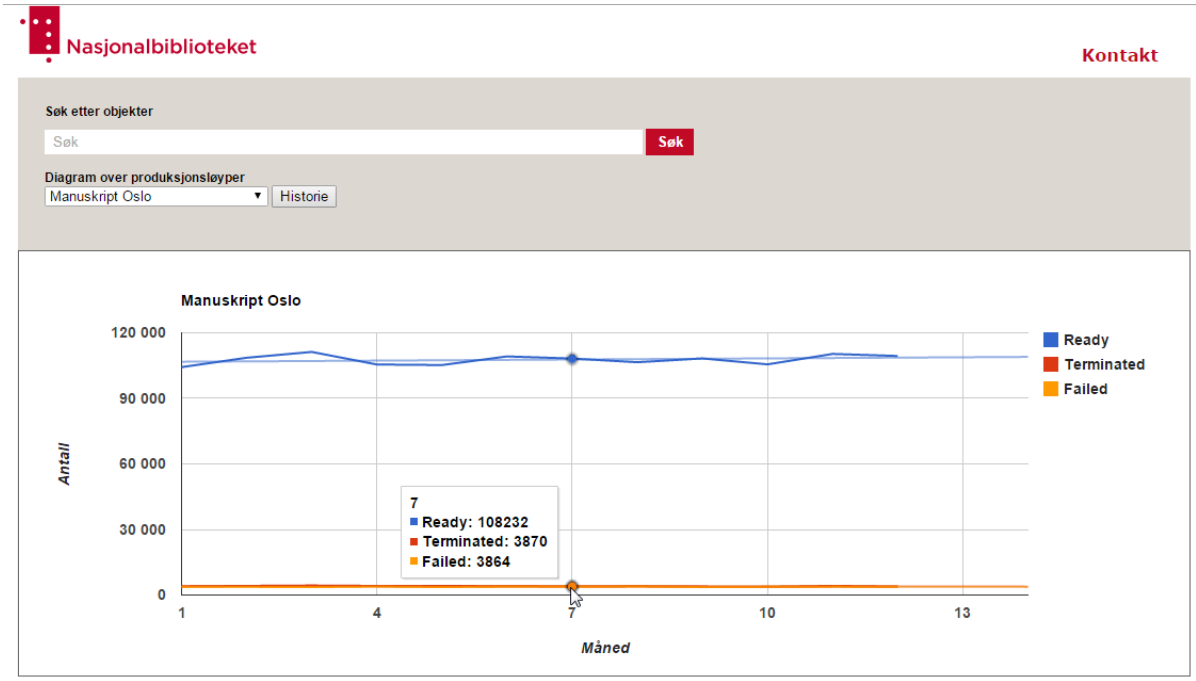


Figur 7 - User story 3

For denne user storyen er det valgt å bruke et linjediagram, fordi andre måter å fremstille historikk på kan ofte bli rotete. Denne grafen gir også veldig bra brukervennlighet da man kan holde musen over grafene for mer detaljert informasjon. På eksempelet (se Figur 3)kan man se flere litt gjennomsiktige streker. Disse strekene er trendlinjer som viser gjennomsnitt, og et estimat på hvor mange objekter det kommer til å være regnet ut fra tidligere data. Denne grafen viser data fra et år på et steg. Hvert datapunkt kan man gå inn på, i dette tilfellet en måned og få informasjonen om akkurat den måneden. Det dypeste man kan gå i grafene er uker, som inneholder data for hver dag.

5.1.4 User story 4

"Som kontrollør vil jeg ha tilgang til trendkurver for hvordan objektene i produksjonsløypa har vært fordelt på status (ok, failed etc.) slik at jeg kan se utviklingen over tid."



Figur 8 - User story 4

I denne grafen har man oversikt over en hel produksjonsløype, der man får oversikt over alle objektene uavhengig av steg. Som du kan se i eksemplet (se Figur 4) kan man se de hvor mange objekter som har hatt de ulike statusene hver måned i et år. Man ser også de litt gjennomsiktige strekene som er en trendlinjene og viser et estimat på hvor mange objekter som kommer til å være i det steget.

5.2 Ikke funksjonelle krav:

De ikke funksjonelle kravene som ble satt i forprosjektrapporten har prosjektgruppen prøvd å oppnå. For å finne ut om disse var oppnådd er det blitt kjørt tester på kravene der det har latt seg gjennomføre. De kravene som ikke kunne testes men som handler mer om skjønn, har tilbakemeldinger vært avgjørende for om kravene har blitt oppnådd eller ikke.

5.2.1 Brukeropplevelse

"Systemet skal ha et godt design, og skal oppleves som lett og brukervennlig"

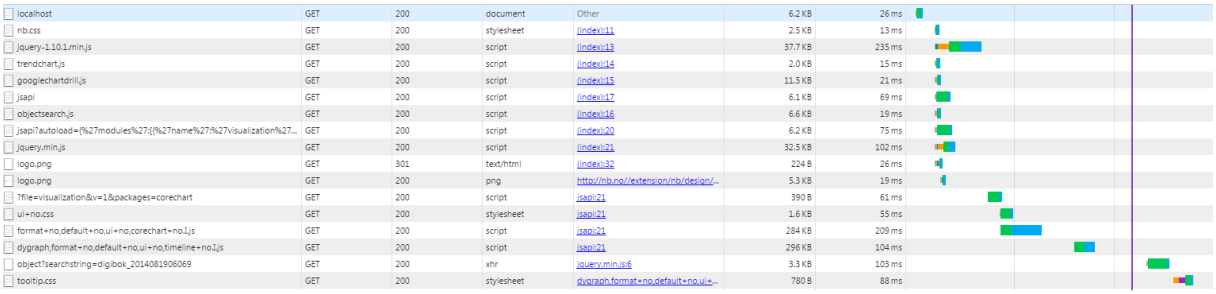
Dette ble løst ved bruk av samme design som www.nb.no og at nettsiden ble laget som en enkeltside. Dette ble mulig ved bruk av *AJAX* som gjør at man ikke trenger å navigere seg mellom flere sider, slik at det bare er en side som oppdateres når man utfører noe. Tilbakemeldingene på utseende og brukervennligheten har vært positive.

En brukermanual som viser eksempler og en kort forklaring av de ulike funksjonene for applikasjonen er lagt ved (se Vedlegg 2).

5.2.2 Ytelse

"Systemet skal oppleves som raskt og responsivt, det skal ikke ta mer en 30 sekund å laste webapplikasjonen"

Siden det meste av data er imitert og ikke reell produksjonsdata, er det per dags dato bare testet på objektsøk funksjonaliteten. Vi har kjørt tester på dette og webapplikasjonen bruker aldri over 30 sekunder, som du kan se på bildet (se Figur 4). I dette eksemplet kjører vi test på objektsøk og webapplikasjonen bruker 1049 ms på å laste forsiden. Informasjon om et objekt blir lastet på 191 ms, dette til sammen tilsvarer 1240ms som er ~1,2 sekunder.



localhost	GET	200	document	Other	6.2 KB	26 ms
nb.css	GET	200	stylesheet	(index):11	2.5 KB	13 ms
jquery-1.10.1.min.js	GET	200	script	(index):13	37.7 KB	235 ms
trendchart.js	GET	200	script	(index):14	2.0 KB	15 ms
googlechartdrill.js	GET	200	script	(index):15	11.5 KB	21 ms
jsapi	GET	200	script	(index):17	6.1 KB	69 ms
objectsearch.js	GET	200	script	(index):16	6.6 KB	19 ms
jsapi?autoload={%27modules%27:[{%27name%27:%27visualization%27...	GET	200	script	(index):20	6.2 KB	75 ms
jquery.min.js	GET	200	script	(index):21	32.5 KB	102 ms
logo.png	GET	301	text/html	(index):32	224 B	26 ms
logo.png	GET	200	png	http://nb.no//extension/nb/design/...	5.3 KB	19 ms
?file=visualization&v=1&packages=corechart	GET	200	script	jsapi:21	390 B	61 ms
ui+no.css	GET	200	stylesheet	jsapi:21	1.6 KB	55 ms
format+no,default+no,ui+no,corechart+no.I.js	GET	200	script	jsapi:21	284 KB	209 ms
dygraph,format+no,default+no,ui+no,timeline+no.I.js	GET	200	script	jsapi:21	296 KB	104 ms
object?searchstring=digibok_201408190 6069	GET	200	xhr	jquery.min.js:6	3.3 KB	103 ms
tooltip.css	GET	200	stylesheet	dygraph,format+no,default+no,ui+...	780 B	88 ms

Figur 9 - Ytelses test på objektsøk

5.2.3 Drift og vedlikehold

"Systemet skal være lett for Nasjonalbiblioteket og vedlikeholde"

Applikasjonen er skrevet i Spring-framework, mens representasjonen er skrevet i html og JavaScript. Disse teknologiene er godt kjent og i bruk på NB. I tillegg er det lagt til en konfigurasjonsfil der man kan endre URIene til *REST-tjenesten*. På denne måten skal applikasjonen være lettere å vedlikeholde om det skulle bli endringer på *REST-tjenesten*, og eventuelt når man skal benytte seg av reell produksjonsdata og ikke den imiterte tjenesten.

En driftsmanual som forklarer de forskjellige klassene og funksjonene er lagt ved (se Vedlegg 3).

5.2.4 Konsistens

"Data som brukes ikke skal være eldre en 20 minutter"

Dette kravet er mindre relevant da det var tidligere tiltenkt en arkitektur med *cache/prefetch* løsning. Den endelige arkitekturen er basert på at applikasjonen benytter *REST-tjenesten* mot produksjonsdatabasen. Hver gang man foretar seg noe i applikasjonen, vil det sendes en forespørsel mot *REST-tjenesten*, som igjen spør databasen. På denne måten får applikasjonen alltid sanntids-data.

6 Fremtidig arbeid:

Det viktigste fremtidige arbeidet blir å legge om applikasjonen til å ta imot produksjonsdata, men om dette blir en stor oppgave eller ikke kommer an på hvordan *REST-tjenesten* blir etter nødvendig funksjonalitet blir lagt til. Det hadde også vært bra å gjøre det mulig å søke på objekter i alle produksjonsløypene og ikke bare i bokløypa.

Brukervennlighet kan også oppgraderes noe, blant annet kunne det vært lagt inn funksjonalitet som gjør at *URLen* oppdateres mens man bruker siden. Slik at man kan kopiere *URL* og dele det man har søkt seg frem til. Man vil også da kunne bruke tilbakeknappen i nettleseren for å navigere på nettsiden.

Det er også andre nyttige elementer og funksjoner som kunne vært lagt til. For eksempel at applikasjonen oppdaterer seg automatisk med jevne mellomrom når man viser grafene, slik at man kunne brukt applikasjonen som en slags overvåkningstjeneste også.

For å få økt ytelse, kunne det vært laget en bedre spørring for objektsøk, der man har fått ut alle events og identifiers med en og samme spørring slik at applikasjonen ikke trenger å kjøre så mange ulike spørringer. Det hadde også vært nyttig å lage en liten side som viser i sanntid hvor mange bøker eller hvor mye data som blir digitalisert hos Nasjonalbiblioteket.

7 Figurliste