

AVDELING FOR INGENIØRUTDANNING
HØGSKOLEN I OSLO

PROSJEKTRAPPORT

SYSTEMUTVIKLING (LO138A)

HØST 2010



SPOTIFY LIVE

GRUPPE 20

FORFATTERE:

LITOVCHENCO, EVY	s161868
ENDSJØ, SIGRID	s136110
ZEYBEK, DENIZ	s161883
KYRKJEBØ NESJE, HELGE	s161932
HENRIKSEN, TONJE	s156049

DATO: 26.11.2010

SAMMENDRAG

Vårt prosjekt dreier seg om å utvikle en løsning for vår oppdragsgiver Spotify som er en svensk musikkstreamingstjeneste. Vi skal utvikle en tilleggsapplikasjon til betalingstjenesten Spotify Premium. Applikasjonen benytter brukerhistorikk for å gi løpende informasjon om relevante konserter i det landet brukeren befinner seg i (dette bestemmes av brukerens IP-adresse).

Målet med Spotify Live er å gjøre Spotify Premium mer attraktivt og øke antall betalende kunder. Selskapet er avhengig av denne inntekten for å kunne honorere artistene bedre. I tillegg er artistene avhengig av økt oppslutning på konserter ettersom de ikke kan overleve kun på CD-salg. Økningen av streaming/ nedlasting av musikk er et problem for artistene. Vi antar at Spotify Live vil gi brukeren en bedre grunn til å velge å betale for musikkopplevelsen.

INNHold

1	INTRODUKSJON	4
1.1	Bakgrunn.....	4
1.2	Mål for prosjektet.....	5
1.3	Omfang av løsning.....	5
1.4	Suksesskriteria	6
1.5	Antagelser	6
2	TILPASNING AV UTVIKLINGSMODELL	8
2.1	Beskrivelse av utviklingsmodell	8
2.2	Begrunnelse for tilpasninger	11
3	ANALYSE	13
3.1	Kravspesifikasjon.....	13
3.2	Use case modell	15
3.2.1	Use case diagram	15
3.2.2	Detaljerte Use case beskrivelser	16
3.3	Domenemodell.....	19
3.4	Aktivitetsdiagram.....	20
4	DESIGN	20
4.1	Klassediagram.....	20
4.2	Sekvensdiagram	23
4.3	Logisk Arkitektur.....	24
4.4	Bruker grensesnitt	24
5	VURDERING.....	27
5.1	Vurdering av foreslått løsning	27
5.2	Vurdering av valg utviklingsmodell	28
5.3	Vurdering av eget prosjektarbeid.....	28
6	KONKLUSJON	29
7	LITTERATURLISTE	29

VERSJONSLOG

Versjon	Dato	Forfattere	Beskrivelse av versjon
0.3	7.10.2010	Evy, Deniz, Helge, Tonje, Sigrid	Første versjon av rapporten, beskrivelse og lett arkitektur av systemet.
0.6	7.11.2010	Evy, Deniz, Helge, Tonje, Sigrid	Andre versjon av rapporten, use case, diagram og arkitektur.
1.0	26.11.2010	Evy, Deniz, Helge, Tonje, Sigrid	Siste versjon av rapport, evaluering og konklusjon

1 INTRODUKSJON

1.1 BAKGRUNN

Vi ønsker å utvikle et system som er en videreutvikling av et allerede eksisterende system. Dette vil gi oss litt strammere tøyler i prosjektet, og vi ser for oss at dette er et mer realistisk scenario enn å finne opp hjulet på nytt. Vi har valgt Spotify som oppdragsgivere.

Spotify¹ er et musikkstreamingsprogram hvor man kan lytte til musikk fra et lydbibliotek som stadig øker. Det er et svensk selskap og programmet har to betalingstjenester (Premium og Unlimited) og to gratistjenester (Free og Open). Per i dag er det 10 millioner brukere og 500 000 betalende medlemmer, og hver dag økes denne gruppen med 1000. Hvert medlem betaler ca 100 kroner i måneden². Gratistjenesten finansieres av reklameinntekter og betalingstjenesten finansieres av brukerne.

Spotify ønsker nå å kunne tilby mer musikk og høyere artisthonorarer. For å greie dette er de avhengige av flere betalende brukere. De satser nå på å videreutvikle Spotify Premium og gjøre denne mer attraktiv enn gratistjenesten.

Selskapet har ansatt oss for å utvikle en applikasjon som skal ta musikkstreamingen ett skritt videre. Deres idé er en tjeneste som tilbyr konsertoppdateringer basert på musikksmak og lokasjon. Dette skal være hovedfunksjonen, men selskapet er åpne for innspill og ideer. Det er få liknende tjenester på markedet i dag, og selskapet har bare lagt inn begrensede ressurser på dette. De ønsker å introdusere konseptet gradvis for å sjekke etterspørselen. Derfor er rapportering og gjennomsiktighet særdeles viktig i vårt prosjekt, om det skulle vise seg at

applikasjonen trenger å utvides på et senere stadium. Det skal være mulig for utenforstående å kunne ta opp tråden og skalere systemet i fremtiden.

1.2 MÅL FOR PROSJEKTET

Prosjektet har følgende målsetning:

- Utvikle en tilleggsapplikasjon for programmet Spotify Premium, som skal
 - gi informasjon om konserter skreddersydd den individuelle sluttbruker
 - bygges rundt eksisterende databaser samt det skal også opprettes en ny database
 - ha et design som skal være i tråd med eksisterende design, for å ikke miste gjenkjennelsesverdi
 - gi brukeren mulighet til å personliggjøre sin egen konsertinformliste
 - bidra til å øke Spotify Premiums brukerandel fra 5% til 10%
- Løsningskonseptet skal være klart til 26. november 2010.
 - Rapport skal leveres 4 ganger (14 sep, 11 okt, 8 nov, 26 nov)

1.3 OMFANG AV LØSNING

Vår løsning skal gi kunden mulighet til å motta konsertinformasjon automatisk, uten å måtte stille inn noe ved første gangs bruk. Kunden logger inn som normalt og Spotify Live starter som en ny menyknapp. Denne knappen lyser opp helt i starten for å informere kunden om at applikasjonen er tilgjengelig.

Man kan lese om applikasjonen første gang ved å klikke på menyknappen, og applikasjonen setter i gang. Spotify Live vil nå opprette en konsertinformliste i bakgrunnen, og menyknappen lyser opp så snart det er kommet en oppføring i konsertdatabasen som samsvarer med en av artistene i brukerens konsertinformliste. Ett klikk på menyknappen fører til en ny side med konsertinformasjon, artist, tid og lokasjon. Fortsetter man å spille musikk vil knappen endre farge til normalt.

Kunden kan se en oversikt over konsertene i en konsertkalender. Disse er uavhengige av om kunden skal delta på konserten eller ikke, og Spotify Live mottar ikke tilbakemeldinger fra

kunden angående konsertene. Tjenesten tilbyr kun informasjon, og muligheten til å redigere hvilken informasjon man vil motta og hvor mye (man kan utvide antall oppføringer i sin egen liste). Redigerer man ikke, vil informasjonen baseres på kundens spillehistorikk.

Det skal også være mulig å deaktivere funksjonen, ettersom man vil motta løpende varsel om konserter. Dette er et valg for dem som syns oppdateringer er stressende. Funksjonen er for å imøtekomme kravet om at systemet ikke skal oppleves som "masete".

Artistenes oppgave er å legge inn informasjon i databasen, og deres brukergrensesnitt har vi nedprioritert. Dette kan løses ved en enkel html-side med link til konsertdatabasen. De logger inn under sitt eget artistnavn, og legger inn informasjon om tid for konsert, lokasjon, og en generell beskrivelse av konserten.

Billettbestilling blir ikke en del av scope, og brukeren må selv sørge for å kjøpe billetter direkte fra arrangøren.

1.4 SUKSESSKRITERIA

Kriterier for suksess er:

- At alle er motiverte og tar oppgaven seriøst.
- At ideen er gjennomførbar og ikke altfor svevende ettersom vi ikke (enda) er profesjonelle systemutviklere.
- At alle leser nødvendig fagstoff slik at vi blir konsekvente med tanke på hva som forventes og hvordan ting skal leveres og presenteres.
- At vi holder oss til faste møter og ikke lar ting skli ut.
- At vi vet hvor vi er på vei.

1.5 ANTAGELSER

Det var tenkt at Spotify Live skulle gi oppdateringer ut i fra brukerens lokasjon. Dette har vi tilpasset til i stedet å gjelde hele Norge. Grunnen til dette er at Spotify er tilgjengelig kun i en håndfull land, og allerede registrerer om man er på reisefot ut i fra IP-adressen til brukeren.

I tillegg er sannsynligheten stor for at man ønsker å dra til feks Bergen om ens favorittartist kun spiller der. Å velge konserter kun fra nærmiljø gir derfor færre valg. Dette gjelder spesielt de som bor i distriktene.

Det ble også vurdert om Spotify Live skal søke nettet kontinuerlig etter relevante konserter og begrense byrden på selve systemet (med tanke på datalagring). Men dette gir igjen veldig press på nettet, og vi har konkludert med at en intern konsert-database er det beste. Ulempen med egen database er at man ikke får info fra artister som ikke har lagt ut informasjonen hos Spotify. Vi tenker likevel at det vil være i artistenes interesse å holde databasen oppdatert for å øke billettsalget. (Særlig siden det eksisterende honoraret til artister ikke er spesielt høyt).

Spotify Live er derfor avhengig av to hovedfaktorer:

- Applikasjonen må linkes opp mot Spotifys brukerdatabase med personlige preferanser. Her må vi få tilgang til gamle strukturer for å tilpasse vårt system opp mot dette.
- I tillegg må den linkes eksternt til en database hvor artister legger ut konsertinformasjon selv hos Spotify.

Prosjektet begrenses på følgende punkter:

- Design vil ikke brukes unødvendig tid på ettersom Spotify allerede har brukt mye penger på sin gamle logo/design.
- Spotify selv er ansvarlige for å håndtere tilbakemeldinger fra brukerne på om systemet fungerer.
- Vi vil ikke ta for oss problemer med innloggingsdelen. Alle usecases baserer seg på at innlogging har gått problemfritt. Dette gjelder både kunde og artist.
- For at Spotify Live skal fungere er det en forutsetning at artistene oppdaterer konsertdatabasen. Vi ser det derfor nødvendig å liste artistene som eksterne aktører. Denne delen er det derimot opp til Spotify å håndtere, ettersom de har kontakten med artistene, og vi fokuserer dermed ikke på artistenes opplevelse av systemet (brukergrensesnitt o.l).
- Det faktum at enkelte artister ikke eksisterer lenger (feks the Beatles) vil vi ikke konsentrere oss om. Nåværende løsning blir at alt vises og det er opp til brukeren å slette de som ikke er aktuelle. Spotify får selv velge om de kun vil tilby oppdateringer fra artister som ligger i konsertdatabasen, eller at de deaktiverer dem som ikke har mulighet for å holde konsert. Denne løsningen kan resultere i at man har en

konserterinformasjon med kun uaktuelle artister, og dermed ikke får noen konsertoppdateringer.

2 TILPASNING AV UTVIKLINGSMODELL

2.1 BESKRIVELSE AV UTVIKLINGSMODELL

I dette prosjektet skal vi benytte Unified Process som prosjektmodell. Modellen er en iterativ og inkrementell modell for utvikling av software og består av følgende fire faser ²:

Idéfase

- Idéfasen dreier seg om å redegjøre prosjektet, lage kravspesifikasjoner, rammer og avgrensninger, lage risikoplan og kostnadsoverslag.
- Denne fasen bør ideelt sett være den korteste fasen i prosjektet.
- Idéfasen avgjør om prosjektet videre er gjennomførbart eller ei.

Utdypningsfase

- Dette er analysefasen med hovedfokus på planlegging, risiko, detaljerte krav, arkitektur og prototyping.
- Systemarkitekturen blir her etablert og man lager grundige modeller på systemnivå.
- Alle modeller som innvirker på systemet som en helhet lages i denne fasen, blant annet use case diagram.
- Slutten av denne fasen skal munne ut i en plan for konstruksjonsfasen, som blant annet skal inneholde en ny lønnsomhetsvurdering.

Konstruksjonsfase

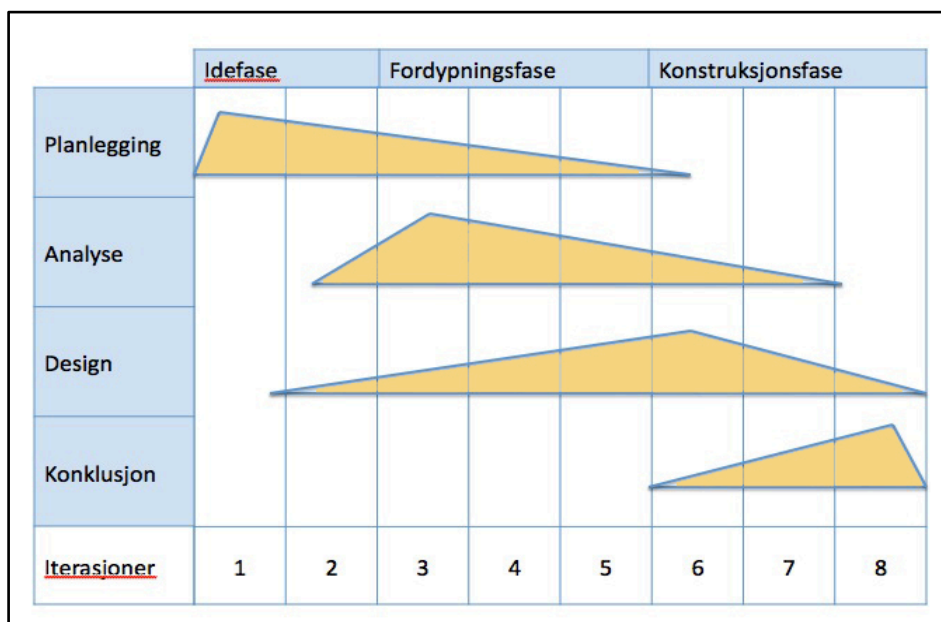
- Dette er den største fasen i prosjektet og består av mange, korte "timeboxed" iterasjoner hvor hver iterasjon inneholder analyse, konstruksjon(design), programmering og testing.
- Delproduktene tilfredsstiller en delmengde av prosjektets kravspesifikasjon: man plukker en funksjonalitet og bygger den ferdig før man starter på neste.
- Jobbene er fordelt, det jobbes individuelt med hvert objekt.
- Det er viktig å realisere use-cases med høy prioritet og risiko først.

Overgangsfase

- Overgangsfasen er den siste fasen og finpussingen. Her retter man seg mot brukerne og får tilbakemelding.
- I denne fasen har man vanligvis introdusert systemet og hjelper brukerne i gang.
- Den siste fasen ender med at produktet lanseres.

I dette prosjektet vil vi kun nå til halvveis uti konstruksjonsfasen, og gi oss på implementering, testing og kvalitetssikring ettersom dette ikke er et programmeringsfag. Prosjektet stopper ved design og konstruksjon.

Prosjektet skal ende opp med en modell som skal være god nok til at programmerere kan sette i gang med programmeringen umiddelbart. Underveis skal vi holde et øye med kostnader, ressurser og tidsfrister, og følge utviklingsmodellen.



Vi har valgt å dele prosjektet opp i 8 iterasjoner. Annen hver iterasjon avsluttes med en innlevering, dette syntes vi var enklest og mest oversiktlig, og også lett å dele opp med tanke på våre ukentlige prosjektmøter. Her presenterer vi en oversikt over alle iterasjonene:

- 1. iterasjon: Det var viktig å få ordnet en solid idé veldig fort ettersom første innlevering var såpass tidlig. Dette ble bare en veldig løs idé med løse mål da vi umiddelbart så mange problemstillinger med ideen som måtte bearbeides. Vi tok litt tid på å bli kjent og kunne etter hvert gi passende roller til gruppe medlemmene.
- 2. iterasjon (innlevering): Prosjektplanmal ble satt opp med de viktigste punktene. Vi ble enige om mål og utviklingsmetode, signerte en kontrakt og skrev risikoplan. Gruppen diskuterte mer på ideen og skisserte løst hvordan systemet burde være, men det var fortsatt ting vi måtte ta stilling til, så vi var litt forsiktige med å skrive for konkrete krav.
- 3. iterasjon: Vi fikk tilbakemeldinger fra første innlevering om at det manglet en del stoff på de siste punktene i prosjektplanen. I tillegg var ikke målene strukturerte nok. Vi jobbet videre med mål og krav og fikk strammet dem opp med mer kvantifiserbare data. Vi ble også enige om hvilke som var gjennomførbare, og hvilke som var helt utenfor scope. Det ble også inkludert noen som muligens kunne innlemmes i systemet, men som vi var usikre på om vi fikk tid til. Disse satte vi opp som ekstra-funksjoner (som en bonus). Vi lagde flytdiagram og den første use case på papir for å se om alle gruppe medlemmene var på nett i forhold til hva vi mente systemet burde håndtere.
- 4. iterasjon (innlevering): Prosjektplanmal ble fullført med tydelige mål og krav, og vi startet på selve prosjektrapporten og kikket på systemets arkitektur. Tidlige use case ble skissert og programmets viktigste funksjoner bestemt. Vi valgte å inkludere ett av ekstra-funksjonene, nemlig kalenderfunksjon, noe som i starten virket litt unødvendig. Vi følte nå at vi hadde tid til å inkludere dette, og at det var en fin funksjon. Vi var usikre på om vi greide å få med en konsert-tilbakemeldingsfunksjon, da dette i høyere grad inkluderer artisten som ekstern aktør. Ettersom systemet er designet med brukeren i fokus, konsentrerte vi oss om den delen som kun inkluderer brukeren først og holdt artisten litt i bakgrunnen. Prosjektrapporten ble utvidet med ferdig use case for hele systemet, og vi utdypet de to av use casene som var de mest grunnleggende.
- 5. iterasjon: Tilbakemelding på andre innlevering var at use casene hadde noen problemer, og vi forstod at vi måtte forklare alle avvik på siden slik at det var lettere å forstå hvordan vi hadde tenkt. Bl.a. hvorfor vi hadde Spotify som en ekstern aktør, når de egentlig er en del av systemet. Vi lagde et nytt use case hvor vi fokuserte kun på brukerens muligheter, og fjernet Spotify fra denne delen. Vi forklarte også nøyere hvordan vi har tenkt for å unngå misforståelser ved neste innlevering. Her fikk vi hjelp i

fra stud-assistent for å sette opp et gunstigere diagram. Vi satte også i gang med aktivitetsdiagram og domenemodell og prøvde finne logiske klasser for systemet. Vi prøvde oss på å lage noen CRC kort for å kunne gruppere klassene. Tilbakemelding fra siste innlevering var også å utdype mer om hver iterasjon, så dette har vi fått på plass.

- 6. iterasjon (innlevering): Prosjektrapporten er utvidet med ferdige use case, brukergrensesnitt og design. Det er laget aktivitetsdiagram, domenemodell, klassesdiagram og sekvensdiagram med normalflyt og avvik. Vi har også beskrevet systemets logiske arkitektur.
- 7. iterasjon: I denne iterasjonen har vi sett på tilbakemelding fra forrige innlevering og endret litt på sekvensdiagrammet. Vi rundet av prosjektet med noen betraktninger på hva vi hadde lært og vurderinger av løsningen.
- 8. iterasjon (innlevering): Prosjektrapport ble ferdigstilt. Alle leste igjennom rapporten og finkjemmet den for selvmotsigelser og skrivefeil.

2.2 BEGRUNNELSE FOR TILPASNINGER

I prosjektet har vi allerede fått rammene for hva oppdragsgiveren er ute etter, og utfordringen vår blir dermed hvordan dette skal henge i hop. Dette betyr at idéfasen vil være relativt kort og utdypings- og konstruksjonsfasene noe mer langvarige. UP gir likevel muligheten til å gå tilbake og endre ting som underveis ikke lenger er aktuelle, men det viktigste er at man i minst mulig grad endrer kravspesifikasjonene. For store endringer her kan føre til at prosjektet blir mislykket og oppdragsgiver kan nekte å godkjenne resultatet.

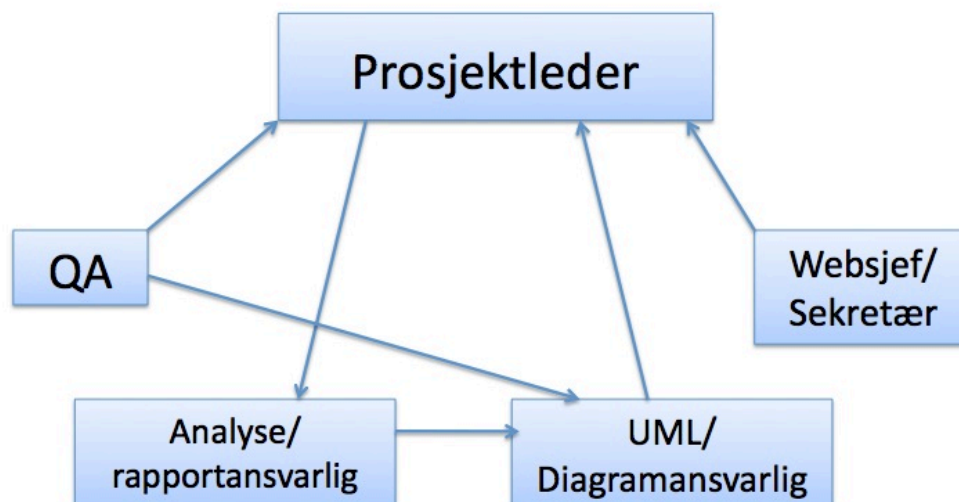
Det er naturlig at det dukker opp problemstillinger underveis og ting ingen tenkte på i starten, men da skal dette tas opp med oppdragsgiver (eller stud-assistent) før man i så fall endrer kurs.

Hvert prosjektmedlem fikk følgende roller ut i fra kompetanse:

Rolle	Navn	Kompetanse	Tidsperiode
Prosjektleder	Evy Litovchenco	LHS, livets harde skole. Alderspoeng	Hele prosjektet
Analyse, rapportansvarlig	Deniz Zeybek	Har vært rapportansvarlig før	Mot innleveringer
Websjef, sekretær	Tonje Henriksen	God på HTML, hatt ansvar	Kontinuerlig

		for websider før	
QA	Helge Nesje Kyrkjebø	Har erfaring med QA	Spesielt før hver innlevering
UML/ diagramansvarlig	Sigrid Endsjø	Flink med IT verktøy	Under hele prosjektet

Prosjektet ble organisert som følgende:



Alle har sine hovedområder, men hver enkelt skal også delta i idéprosessene og ha innblikk i hva de andre gjør. Dette for å unngå krise ved feks sykdom.

Her er oversikt over den enkeltes ansvarsområde:

- Prosjektleder vil delegere oppgaver til medlemmene, og foreslå relevant fagstoff underveis til hver enkelt som det forventes av alle leser, i tillegg til å arrangere møter. Ellers bistå med motivasjon, trøst og kompetanse.
- Analyseansvarlig har fokus på første del av konstruksjonsfasen, og sørge for at systemet blir oversiktlig. Han er ansvarlig for sekvensdiagrammer og visualisering i startfasen. I tillegg er han rapportansvarlig, og skal sørge for at både krav og frister for innlevering blir nådd.
- QA skal jevnlig sjekke kvaliteten på systemet og at man til enhver tid i prosjektet oppfyller selskapets krav. Use-cases skal gjennomgå kontinuerlig for å se om de holder mål.

- QA og analyseansvarlig skal formidle de forskjellige systemoversiktene til UML-ansvarlig for videre behandling.
- UML og diagramansvarlig har hovedfokus på å skissere ideene som gruppen kommer frem til ved hjelp av profesjonelt verktøy, som Rational Rose og Microsoft Project.
- Websjef og sekretær skal skrive møterapport etter hvert møte, og holde orden på gruppenettsiden så den alltid er a jour. Ellers bistå med analyse og arkitekturen.

Prosjektplanen ordnes på ved hjelp av Dropbox. Dette gir god mulighet for alle å oppdatere samtidig, og jobbe hjemmefra. (kapasiteten på skolen er overskredet og det er problematisk å få grupperom).

Vi delte opp prosjektet i 8 iterasjoner, med 4 innleveringer fordelt på annenhver iterasjon. Dette ble gjort fordi vi hadde ukentlige gruppemøter og fant det enklest å fordele prosjektet i biter som er lett å holde styr på. I tillegg fikk vi muligheten til å rette opp tilbakemeldinger fra hver innlevering tidlig i neste iterasjon (unntatt den siste da alt skal være klart).

På iterasjon 3, 5 og 7 justerte vi plan og rapport i henhold til tilbakemeldingene fra stud-ass, slik at prosjektet hele tiden oppfylte de satte kriterier.

3 ANALYSE

3.1 KRAVSPESIFIKASJON

Her er en oversikt over kravspesifikasjoner for Spotify Live.

Funksjonelle krav (lar seg implementere):

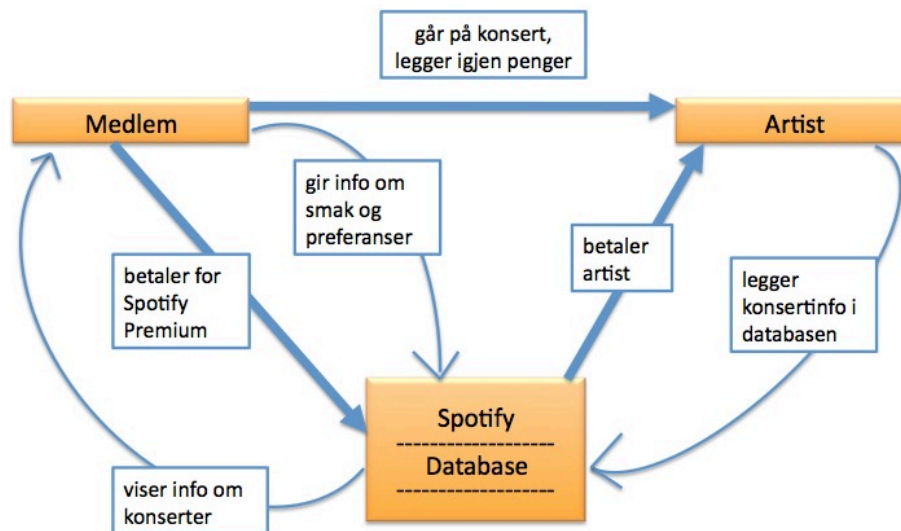
- Man er avhengig av å være innlogget på sin Spotify Premium konto for å få tilgang til sin egen Spotify Live.
- Antall band/artister som inkluderes i tjenesten skal vise de 25 mest spilte som default.
- Brukeren skal ha mulighet til å personliggjøre sin egen oppdaterings-liste, ved å slette eller legge til artister (favorisere).
- Det skal være mulig å øke antallet artister fra 25 til 50 og 100 mest spilte, for å øke antall konsertoppdateringer.

- Artister skal kunne legge inn konserter, lokale og tidspunkt i egen database. Spotify Live vil kun hente informasjon fra denne databasen, mens den oppdateres av artistene.
- Begge databasene (musikk og konsert) må snakke sammen på en logisk måte.
- Det skal være egen menyknapp inne i Spotify Premium, som ikke fører til et eksternt vindu men være en del av helheten.
- Det skal ikke være reklame i Spotify Live.
- Applikasjonen skal gi oversikt over fremtidige konserter i en kalenderform.
- Man skal kunne deaktivere oppdateringene ved en egen fane inne i konsertfunksjonen.

Ikke-funksjonelle krav (lite målbare forhold):

- Brukergrensesnittet skal ha en logisk plassering.
- Brukeren skal oppleve Spotify som en nær venn som kjenner dine preferanser og følger med så du ikke går glipp av noe.
- Skal ikke gi et masete inntrykk.
- Spotify Live skal ikke være tregere enn resten av programmet, og benytte alle de samme protokoller som allerede er i bruk.

Her har vi satt opp et flytdiagram for å skissere hoveddelene av systemet med de forskjellige aktørene:

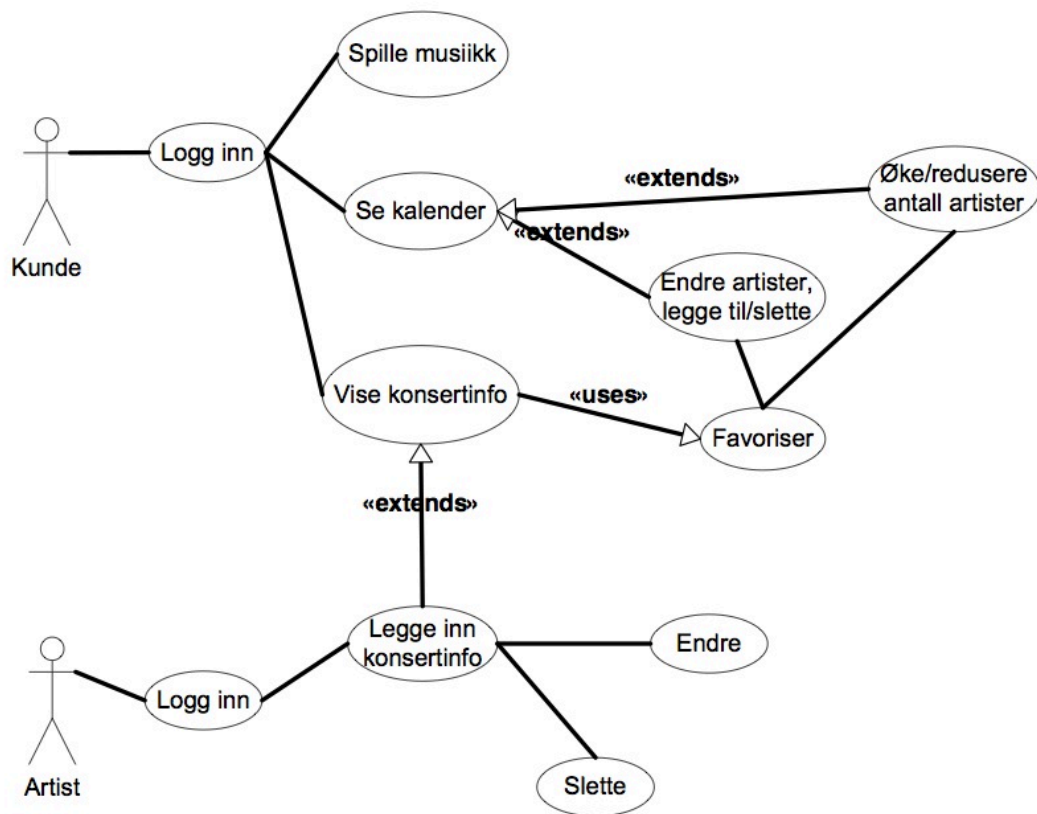


Vi har valgt å oppgi Spotify som et eksternt system selv om de i teorien *er* systemet. Dette har vi gjort fordi Spotify Live er avhengig av funksjonalitet som allerede eksisterer i en normal Spotify Premium-konto. Det som skjer i Spotify skjer stort sett automatisk, men fra vårt ståsted kan man si at Spotify opptrer som en admin, med muligheter til å lagre, endre og opprette ting i kulissene. Ettersom Spotify Live ikke rår over disse delene av systemet syntes vi det er hensiktsmessig å fremstille det på denne måten.

3.2 USE CASE MODELL

3.2.1 USE CASE DIAGRAM

Her er use case for Spotify Live. Fagstoff har vi hentet fra boken Systemutvikling³, UML 2 Toolkit⁴, Learning UML⁵ og Wikipedia⁶.



I dette use caset har vi beskrevet de forskjellige handlingene til de to eksterne aktørene: kunde (medlem) og artist. De nedenforstående use case beskrivelsene har derfor kun tatt utgangspunkt i kundens muligheter og viser at disse er avhengige av at artisten har gjort jobben sin.

3.2.2 DETALJERTE USE CASE BESKRIVELSER

Vi har valgt å beskrive to av use casene som vi syns spiller de viktigste rollene i systemet. Disse to omhandler use case for fremvisning av konsertinformasjon og favorisering av konsertinformliste.

Use Case: Fremvisning av konsertinformasjon

Aktører: Kunde, Spotify, Artist

Trigger: Kunden klikker på menyknappen for konsertinformasjon

Prebetingelser:

- A) Kunden er medlem av Spotify Premium, og logget inn på systemet.
- B) Kunden navigerer i databasen ved å bruke søkefunksjonen og spiller av musikk.
- C) Artisten har lagt ut informasjon om konserter i databasen.

Postbetingelse:

- A) Artistpreferanser er silt ut og linket med konsertinformasjon.
- B) Systemet formidler korrekt konsertinformasjon.

Hovedflyt:

1. Kunden logger inn på systemet.
2. Kunden søker etter musikk og spiller disse. Har man ikke spilt noen sanger enda vil man ikke ha noen oppføringer verken i Spotify eller i konsertinfo listen.
3. Databasen samler inn informasjon om kunden ut i fra handlinger/historikk, uavhengig av hvor lenge man har spilt en sang vil denne registreres som ett "spill".
4. For hver spilte sang setter systemet sammen en dynamisk konsertinfo liste basert på kundens mest spilte artister.
5. Kunden kan se denne listen ved å trykke på menyvalget "Konsertinfo".

Variasjoner/avvik:*A) Kunden velger å redigere konsertinfo liste*

1. Systemet oppfordres til å redigere informasjonen i konsertinfo listen.
2. Systemet endrer informasjonen og fremviser en oppdatert liste.

B) Eventuell feil i systemet ved valg av låt

1. Systemet oppfordres til å spille av en låt som kunden har valgt.
2. En feil i databasen fører til at låten ikke blir spilt og systemet viser en feilmelding til kunden.
3. Systemet lagrer feilmeldingen og sender denne til administratoren for feilsøking.

C) Artisten er død, evt spiller aldri konserter.

1. Systemet viser alle artister uavhengig av om de spiller konserter eller ikke.
2. Kunden må selv inn og redigere informasjon og fjerne de som ikke antas å være aktive. (dette kan leses i seksjonen *biografi* på hver artist, hvor man får informasjon om artisten er avdød eller fremdeles spiller.)

Use Case: Favorisering av konsertinformaliste

Aktør: Kunde, Spotify, Artist

Trigger: At kunden er inne på konsertinformalisten og velger endre eller slett

Prebetingelse:

- A) Kunden må være logget inn og ha spilt av musikk.
- B) Database av mest spilt musikk må være opprettet.
- C) Artisten må ha publisert konsertinfo til Spotify.

Postbetingelse:

- A) Endringer er skjedd uten feilmeldinger.
- B) Konsertinformalisten viser oppdatert informasjon.

Hovedflyt:

1. Kunden går inn på konsertinformalisten
2. Listen viser de 25 mest spilte artister.
3. Kunden plukker ut de artistene som kunden ikke vil ha og velger om hun vil slette eller endre. Kunden kan også velge å øke listen fra 25 til 50 eller 100 artister.
4. Systemet oppdaterer endringen, og fremviser en ny konsertinformaliste.
5. Ved spilling av musikk vil den oppdaterte listen være den aktuelle.

Variasjon/avvik:

A) Konsertinformalisten er ikke blitt oppdatert

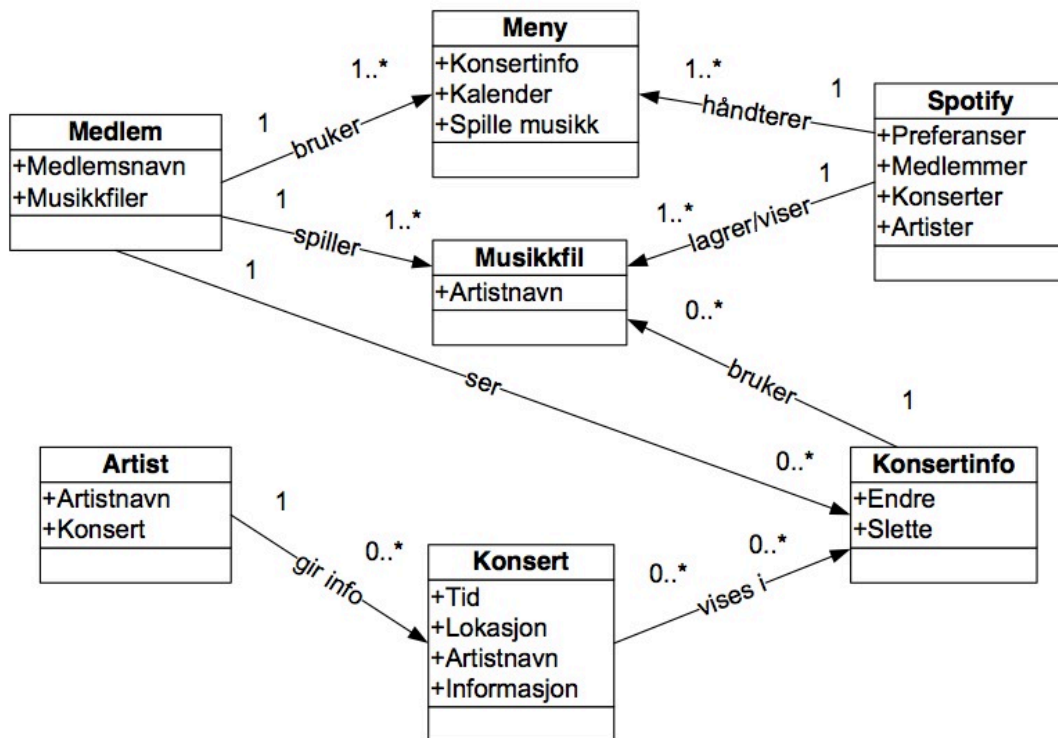
1. Kunde må gjenta steg 1-3 i hovedflyt.

B) Eventuell feil i systemet ved endring av spilleliste

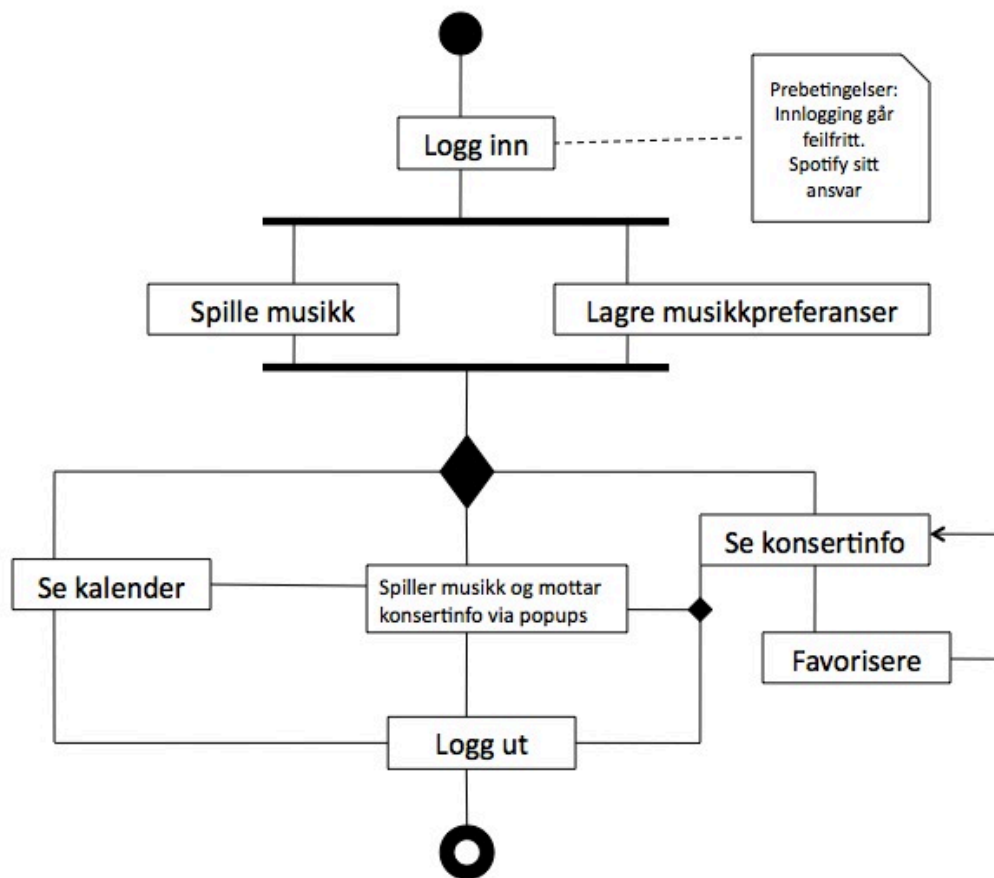
1. Systemet oppfordres til å endre en oppføring in konsertinfoen.
2. En feil i databasen fører til at endringen ikke blir lagret og systemet viser en feilmelding til kunden.
3. Systemet lagrer feilmeldingen og sender denne til administratoren for feilsøking.

3.4 DOMENEMODELL

Her presenterer vi en domenemodell av systemet som viser relasjonene mellom de konseptuelle klassene:



3.5 AKTIVITETSDIAGRAM



(Pop-ups i dette diagrammet henviser til at menyknappen lyser opp).

4 DESIGN

4.1 KLASSEDIAGRAM

Før vi tok tak i klassediagrammet opprettet vi noen CRC kort (Class Responsibility Collaborator) for å få oversikt over de forskjellige ansvarsområdene til hver klasse for at ikke en klasse skulle ha for mye å gjøre. Vi ville bruke dette for å se hvordan klassene burde være. (med Spotify-klassen menes det her Spotify Premium). Det var likevel litt vanskelig å bruke denne modellen helt etter punkt og prikke til klassediagrammet, for selv om enkelte klasser avhenger av hverandre, er det ikke nødvendigvis slik at de jobber sammen.

Medlem	
Ansvar	Samarbeider med klasse
Spiller sanger	Spotify, Gui
Legge inn favoritter i databasen	Spotify, Musikkfil

SpotifyLive Kontroller	
Ansvar	Samarbeider med klasse
Henter input fra Gui	Gui
Snakker med databasene	Spotify

Gui	
Ansvar	Samarbeider med klasse
Viser konsertinfo	SpotifyLive Kontroller, Medlem
Viser kalender	SpotifyLive Kontroller, Medlem
Viser spillelister	SpotifyLive Kontroller, Medlem

Spotify	
Ansvar	Samarbeider med klasse
Lagre sanger	Medlem
Spiller sanger	Medlem
Formidle konsertinfo	Artist, Kunde, Gui
Oppretter konsertinfo	Kunde, Konsertinfo

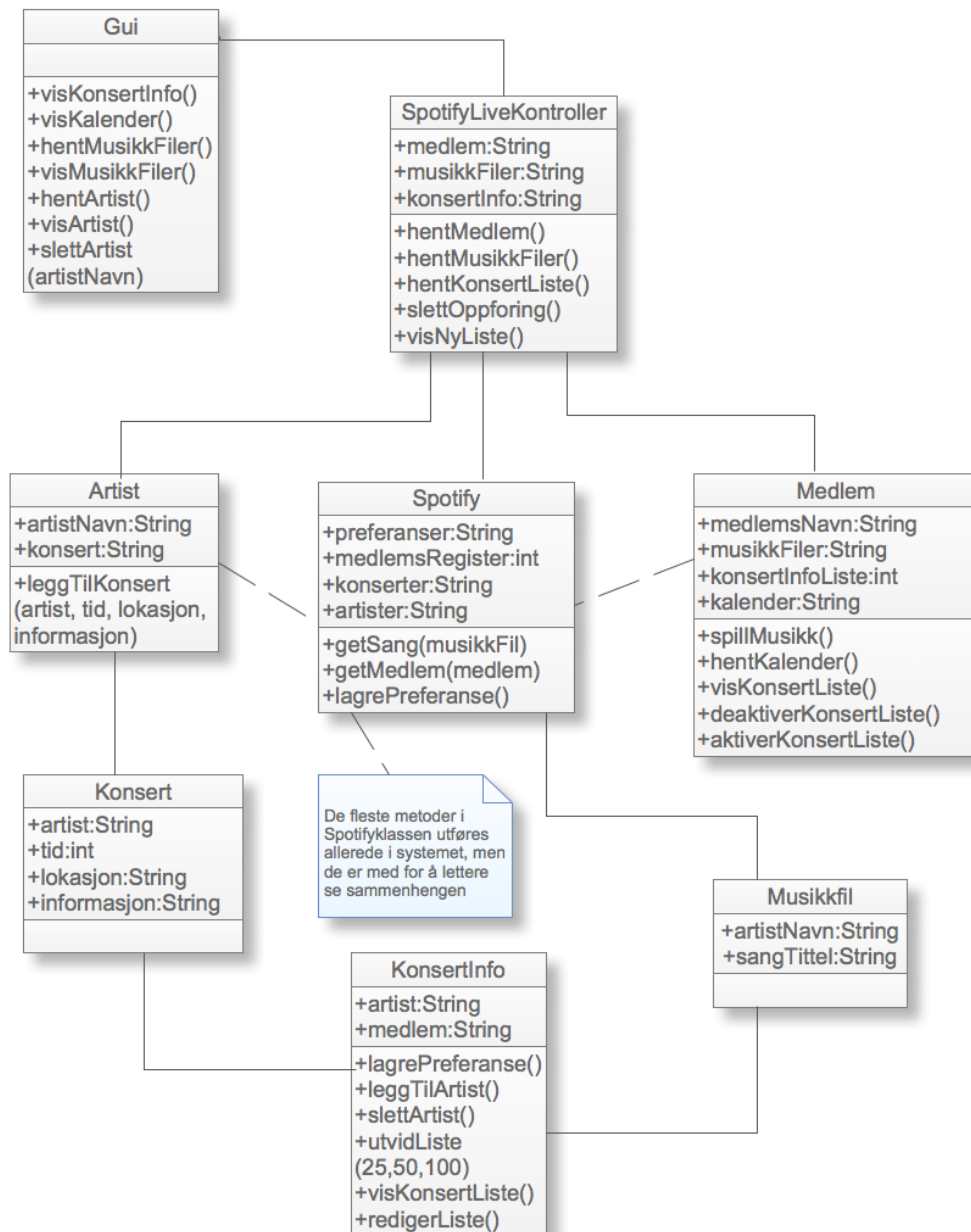
Musikkfil	
Ansvar	Samarbeider med klasse
Spilles	Spotify, Medlem
Viser artistnavn på hver låt	Spotify, Medlem

Konsert	
Ansvar	Samarbeider med klasse
Viser tid	Spotify, Artist
Viser sted	Spotify, Artist
Viser lokasjon	Spotify, Artist
Viser øvrig informasjon	Spotify, Artist

Konsertinfo	
Ansvar	Samarbeider med klasse
Kan endres	Spotify, Artist
Kan slettes	Spotify, Artist

Artist	
Ansvar	Samarbeider med klasse
Lagre konsert	Spotify, Konsert, Konsertinfo

Klassediagram:



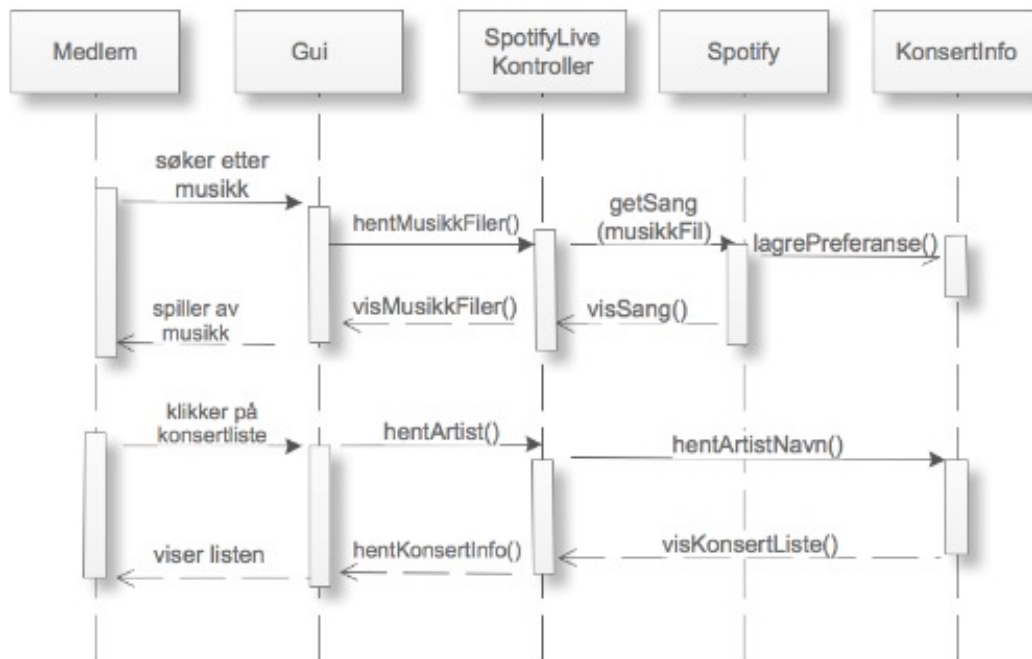
I klassediagrammet går vi ut i fra at registrer medlem og sanger og preferanser til hvert medlem allerede utføres av Spotify, så dette har vi ikke oppført. Med preferanser her menes kun de 25 mest spilte artister som lagres i konsertlisten. Kundene er avhengige av at artistene

legger ut informasjon om konsertene i databasen. Spotify Live er avhengig av at kundeklassen spiller musikk for å kunne opprette preferanseliste som danner grunnlaget for konsertinfo listen.

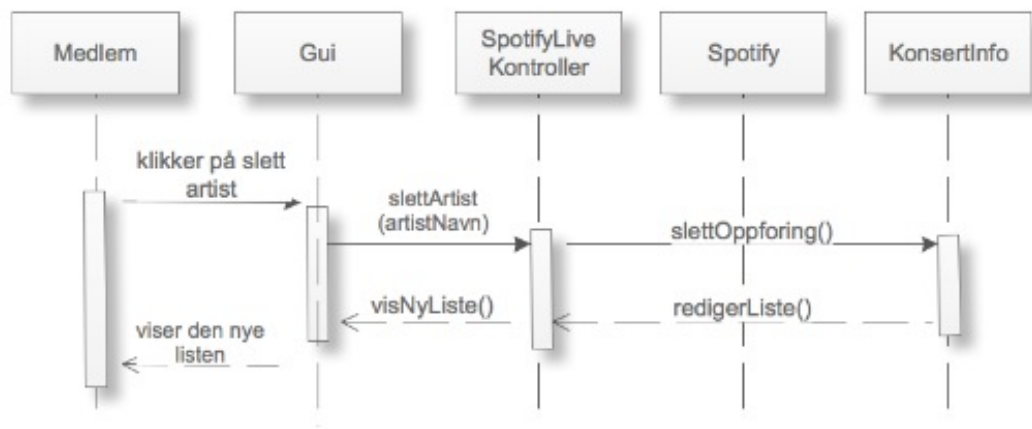
4.2 SEKVENSDIAGRAM

Her viser vi sekvensdiagram som modellerer de to viktigste scenarioene (en normalflyt og en variasjon) for den første av use casene beskrevet i 3.2.1.

Opprettelse av konsertinfo liste / Normalflyt:



Opprettelse av konsertinfo / Variasjon (ved redigering av listen):



4.3 LOGISK ARKITEKTUR


Spotify Live er bygget opp med en 3-lags arkitektur som inkluderer brukergrensesnittet (Gui), selve applikasjonen (SpotifyLiveKontroller) og de to databasene (Spotify og Konsertinfomodellen). Kontrolleren er avhengig av Spotify Premium sitt allerede eksisterende system for å plukke ut sanger og lagre disse. Den mottar input fra GUI og fungerer som bindeledd mellom GUI og Spotify, ettersom det ikke er så sikkert å la GUI snakke direkte med databasene. Den ene av databasene (musikkdatabasen) er også allerede en eksisterende database, og Spotify Premium benytter denne for å hente ut informasjon om musikkpreferanser. Denne informasjonen ligger til grunn for å kunne opprette konsertinfo. I tillegg har Spotify Premium allerede en medlemsdatabase, og musikkpreferansene vil alltid ligge lagret i denne. Det blir her nødvendig å utvide denne medlemsdatabasen med et felt for konsertliste. Fra denne går det en logisk kobling til riktig artist, og deres konsertoppføringer.

4.4 BRUKERGRENSESNITT

For å følge kriteriene til brukergrensesnittet har vi brukt samme farger og fonter som allerede eksisterer i Spotify Premium. Brukergrensesnittet skal være diskret og ikke masete. Det er forøvrig nødvendig å informere om den nye fanen. Dette gjøres ved at fanen er farget i samme tone som Spotify sin logo. Dette er det mulig å endre etter en viss tid når man ser at brukerne har begynt å benytte den. I tillegg vil det i startfasen dukke opp en liten live-logo.

Denne kan man klikke på og lese om det nye konseptet. Etter informasjonen er lest vil logoen forsvinne. Fanen vil lyse opp i grønt i startfasen, og ellers vil den kun lyse om det har skjedd en oppdatering, og vise feks Konsertinfo(2), i likhet med å få en epost.

Overview
Biography
Related artists
Artist radio
Konsertinfo









Jahn Teigen

Share

Jahn Teigen is a Norwegian singer and musician. Jahn was born on September 27, 1949, in the Norwegian city of Tønsberg. He represented Norway in the Eurovision Song Contest three times, in 1978, 1982 and 1983 respectively. His given name was Jan, he a...

Related artists









Top hits

123

Rank	Song	Buy	Duration	Progress	Collaborators
1	Det Vakreste Som Fins	Buy	5:10		
2	Optimist	Buy	2:54		
3	Friendly	Buy	3:32		Anita Skorgan, Jahn Teige
4	Jahn Teigen Grand Prix Medley	Buy	9:59		
5	Min Første Kjærlighet	Buy	3:21		

Albums



Buy album

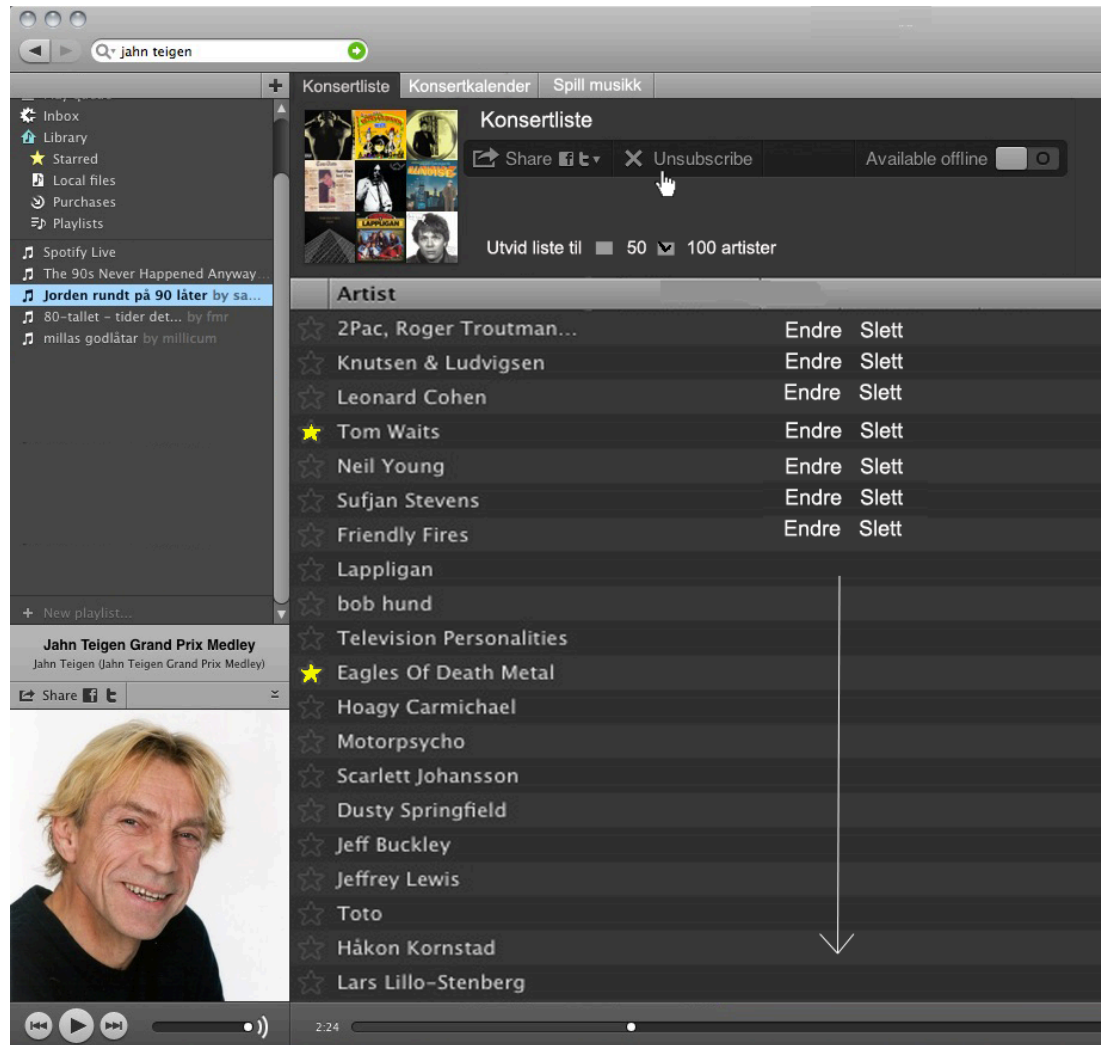
Teigen - 40 største hits (2009)

★ Star

1

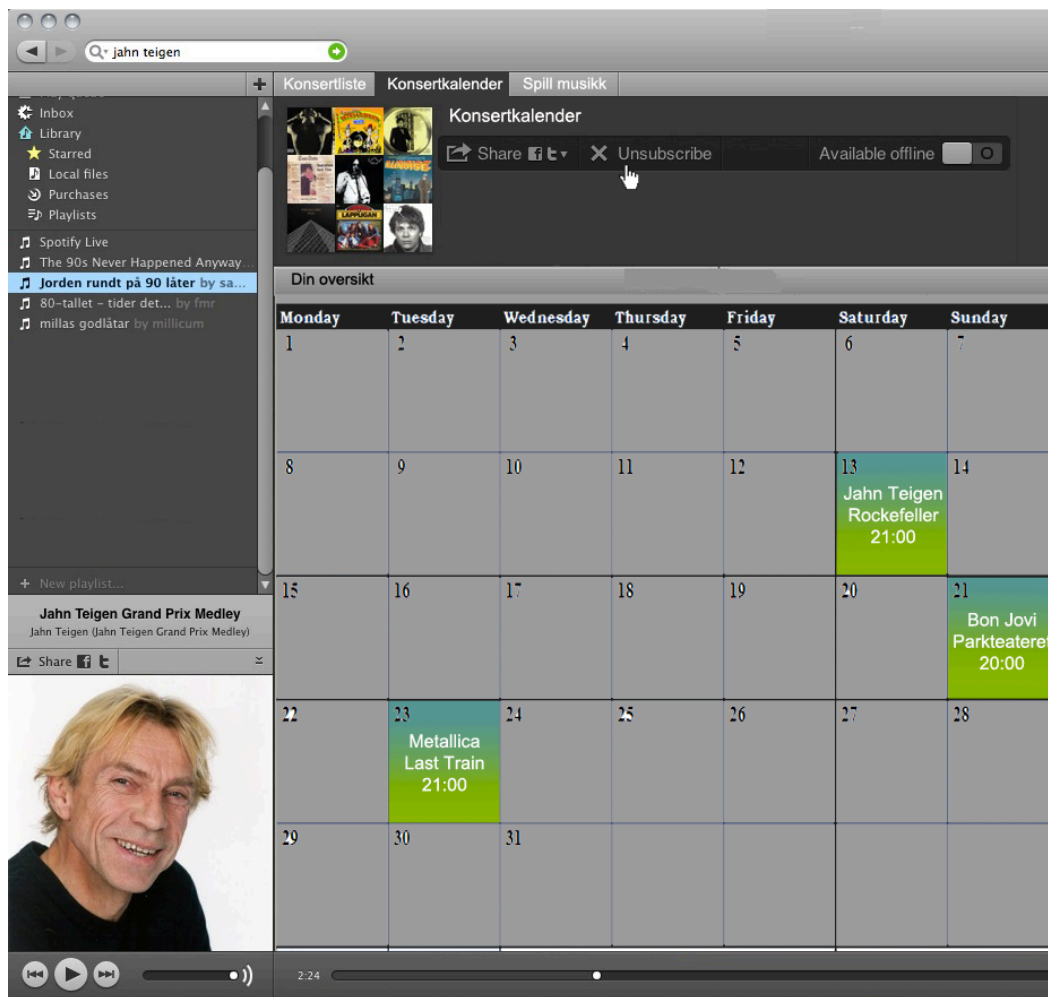
Rank	Song	Buy	Duration	Progress
1	Mil etter mil	Buy	3:06	
2	Claudius	Buy	2:55	
3	I natt er jeg din	Buy	5:24	
4	Jeg gi'kke opp	Buy	3:03	
5	They Don't Clap Losers	Buy	2:47	
6	En dags pause	Buy	2:14	
7	Min første kjærlighet	Buy	3:20	
8	Førstesidebilde	Buy	3:50	
9	I Hamburgs natt	Buy	3:13	
10	Risk	Buy	2:45	
11	Sala Palmer	Buy	2:44	
12	Ja	Buy	2:41	
13	Du skulle sagt ifra	Buy	4:01	
14	Bli bra igjen	Buy	3:21	
15	Adieu	Buy	2:22	

Etter klikk på fanen kommer man til informasjon om konserter. Har man lest de siste oppdateringer vil man gå rett inn på sin egen konsertliste. Denne vil da se slik ut:



I denne listen skal man ha mulighet til å endre og slette artister, og velge å utvide listen fra 25 til 50 eller 100. Man kan når som helst klikke på de aktuelle artistene (her illustrert med en stjerne) og komme tilbake til informasjonsvinduet om deres kommende konsert.

Fra denne modusen kan man også klikke på *kalender* som også er et valg i sidemenyen til enhver tid. Her får man oversikt over datoer det er konsert og man kan bla seg frem fra dagens dato.



Man kan alltid velge å deaktivere tjenesten ved å huke av for dette (unsubscribe i denne illustrasjonen). Kunden vil da ikke motta oppdateringer, men menyvalget vil fortsatt være en del av systemet.

5 VURDERING

5.1 VURDERING AV FORESLÅTT LØSNING

Løsningen vår avhenger i stor grad av at artistene blir grundig informert om fordelene ved å delta og oppdatere databasen med konserter. Når denne rutinen er innarbeidet vil systemet fungere optimalt ved å tilpasse informasjonsflyten til hver individuelle bruker. Brukerne kan strømlinjeforme sin egne oppdateringer, men løsningen kan senere videreutvikles til å gi brukeren mer innflytelse. Vi ser for oss at det kunne vært gunstig med en feedbackfunksjon

på konserter man har vært på, og samtidig sende informasjon til sine venner eller synke konsertkalender med andre, på samme måte som man i dag sender spillelister ved hjelp av en URL.

Det er i tillegg et problem med artister som ikke har mulighet for å holde konsert, og vi foreslår her å implementere en metode som fjerner de uaktuelle artistene fra Spotify Live. Men dette får nesten Spotify vurdere selv, ettersom vi ikke kjenner til avtalene mellom selskapet og de involverte og hvordan de helst ønsker å løse dette.

Designmessig gir systemet et troverdig inntrykk ettersom det er laget i tråd med resten av Spotify, og med knapper og valg som brukeren kjenner til. Reklameringen for Spotify Live er essensiell, men utenfor vårt område.

5.2 VURDERING AV VALG UTVIKLINGSMODELL

Vi har brukt Unified Process som utviklingsmodell, og tilpasset denne med passende iterasjoner basert på hvilken dato hver innlevering falt på. UP gir oss muligheten til å sakte gå tilbake og endre ting underveis, men at det er progresjon i riktig retning. Det er en litt overlappende utviklingsmodell som ikke er like streng som feks timeboxing hvor man blir gjør seg helt ferdig med hver iterasjon før man går over i neste. For oss var dette en fin måte å styre prosjektet på ettersom dette er et skoleprosjekt hvor vi skal lære ting underveis. Da gir det lite mening å ikke kunne gå tilbake og endre etter hvert som vi lærer, så denne modellen var meget effektiv.

5.3 VURDERING AV EGET PROSJEKTARBEID

For modellering brukte vi Omnigraffle, som var et veldig enkelt og hendig UMLprogram, men sluttet å fungere halvveis uti pga gratis-lisens. Vi måtte dermed fortsette med Visio og ConceptDraw som modelleringsverktøy og lære ting litt på nytt hele tiden, men det har selvsagt vært lærerikt å teste forskjellige programmer. Ellers har vi benyttet diverse UMLhåndbøker for hvordan å sette opp de forskjellige modellene.

Selve gruppearbeidet ble litt workshop-aktig. Det var nemlig vanskelig å booke grupperom og det ble derfor mye møter i kantinen hvor det er bråkete og møtene ble derfor preget av dette. Men de gangene vi fikk grupperom gikk arbeidet veldig bra og engasjementet steg hos alle gruppemedlemmene. Det at alle hadde egne roller førte til at man fikk eierskap over sin del av

prosjektet, og alle merket at deres roller var avgjørende i sluttproduktet. Vi hadde heller ingen sykdom eller frafall, så risikoplanen slapp vi å ta så mye stilling til.

6 KONKLUSJON

Vi har lært mye om UML i dette prosjektet og nødvendigheten av forskjellige visuelle fremstillinger av systemer. Ettersom software kan fremstilles på mange måter og at det er veldig abstrakte termer det går i, er det nok veldig behjelpelig for programmerere om man kan vise det i forskjellige modeller. Slik kan man få en bredere presentasjon og forhåpentligvis unngå forvirring. Forvirring kan uansett oppstå om modellene er veldig ukonsekvente, og vi har gjort så godt vi kan å unngå dette. Men det er ingen tvil om at det er et komplisert fag.

Vi har også fått erfare viktigheten av god struktur og planlegging for å gjennomføre prosjektet og ikke avvike for mye fra de kravene vi satte oss i starten. Det er veldig greit å timeplaner og roller og datoer å forholde seg til så ting ikke svever i løse luften, og denne prosjektmåten å jobbe på kan også brukes på flere områder i livet.

Vi føler vi har oppnådd de fleste av målene foruten dem som vil kunne måles etter at systemet er satt i drift (hvordan konseptet påvirker antall betalende brukere osv). Men løsningen danner hvertfall grunnlaget for videre utvikling og så er det opp til Spotify om det er lønnsomt å legge mer penger i denne funksjonen.

7 LITTERATURLISTE

1. <http://www.spotify.com>
2. <http://eu.techcrunch.com/2010/09/15/spotify-10-million/>
3. Thor E.Hasle, *Systemutvikling*, Cappelen Akademisk Forlag, 2008. ISBN 978-82-28605-7 s.94
4. Hans-Erik Eriksson, Magnus Penker, Brian Lyons, David Fado, *UML 2 Toolkit*, Wiley publishing Inc, 2004. ISBN 0-471-46361-2
5. Sinan Si Alhir, *Learning UML*, O'Reilly, ISBN 0-596-00344-7
6. http://en.wikipedia.org/wiki/Use_case

Til slutt vil vi si takk til stud-assistentene Eirik Risnes og Thomas B. Martinsen for veldig gjennomgående tilbakemelding på hver innlevering!