
Softwareteknologi 2. semester Gruppe 05

Udvikling af cyber-physical softwaresystemer

Forfattere

Aleksander Soender 168256750	alsoe19@student.sdu.dk
Anders Wylardt Nielsen 167007542	anni119@student.sdu.dk
Christopher Kas 426413	chkas15@student.sdu.dk
Jakup Malik Davut Karagoz 481562	jakar19@student.sdu.dk
Mikail Cengiz 166839407	micen19@student.sdu.dk
Simon Jessen 454932	sijes16@student.sdu.dk

Vejleder

Jeppe Schmidt

*Teknisk fakultet
Syddansk Universitet
Mærsk Mc-Kinney Møller Instituttet
Odense*



FORÅR 2020

3. Februar - 29. Maj 2020

Abstract

The research question which is attempted solved in this report is:

How can a creditsmanagementsystem help TV2 with the job of removing end-credits and thereby freeing up 30 seconds of air time, by the end of a broadcast?

In order to resolve the problem at hand, most thoroughly, a solution was thought of, through the case problematics, by picking the most desired parts.

The proposed solution is a two part program, containing both a public and a more private part. The public part is an incomplete mockup emulating a TV overlay, that anyone of TV2s viewers will be able to access, via an integrated online database. The second part consists of a login page, and direct access to the database. The program supports user entries, so producers and administrators can add, edit, delete broadcasts and credits to the database.

Furthermore this paper includes the implementation process. SCRUM and a modified *Unified Process* has been a cornerstone for the project.

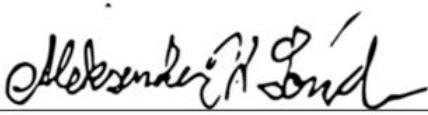
Finally, the conclusion of the work is that the developed prototype is acceptable. The system implements various features, desired in the finished version and do so in a simple way. The system can not be portrayed as a finished solution, however, it does provide a possible solution to the problem at hand.

Titelblad



Christopher Kas
Underskrift

chkas15@student.sdu.dk
Christopher Kas



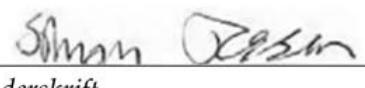
Aleksander Soender
Underskrift

alsoe19@student.sdu.dk
Aleksander Soender



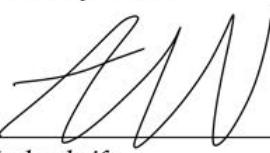
Jakup Malik Davut Karagoz
Underskrift

jakar19@student.sdu.dk
Jakup Malik Davut Karagoz



Simon Jessen
Underskrift

sijes16@student.sdu.dk
Simon Jessen



Anders Wylardt Nielsen
Underskrift

anni119@student.sdu.dk
Anders Wylardt Nielsen



Mikail Cengiz
Underskrift

micen19@student.sdu.dk
Mikail Cengiz

Udvikling af cyber-physical softwaresystemer 2. Semesterprojekt, Gruppe 05

Softwareteknologi

Vejleder: Jeppe Schmidt

Teknisk fakultet

Syddansk Universitet Odense

Mærsk Mc-Kinney Møller Instituttet

Foråret 2020

3. Februar - 29. Maj

Forord

Denne rapport er udført i forbindelse med et 10 ECTS-point projekt, ved Mærsk Mc-Kinney Møller Instituttet, Syddansk Universitet, som opfølgning på det udviklingsarbejde projektgruppen ST05 foretog i foråret 2020.

Udviklingsarbejdet tog udgangspunkt i Credits Management, en case fra TV2 Danmark A/S. Gruppen havde til opgave at udvikle et bud på et krediteringssystem, der skulle hjælpe TV2 til at kunne fjerne rulleteksterne efter endt program på sendefloden. Denne rapport beskriver de konkrete problemstillinger og opgaver gruppen har valgt at arbejde med, samt hvad der er blevet udført under projektarbejdet.

Rapporten bygger på et fagligt vidensgrundlag og er skrevet således personer med kendskab til grundlæggende software engineering, databaseopbygning, samt objektorienteret programmering kan forstå koncepterne rapporten igennem. Målgruppen er hovedsageligt rettet mod gruppens vejleder, samt censor. Vi takker Jeppe Schmidt for god vejledning, råd og opbakning.

Læsevejledning

I rapporten bliver der henvist til diverse figurer og bilag; i disse tilfælde kan det være relevant at kigge på figurerne, eller gå til bilaget, efter teksten er læst. Henvisninger er som udgangspunkt beskrevet i teksten, men det kan det være relevant at have figurene ved siden af, mens der læses. Hvis rapporten læses digitalt bør den være zoomet ind til 100% for at diagrammerne er tydelige.

Det er som udgangspunkt muligt at læse afsnittene uafhængigt af hinanden, så længe læser er opmærksom på titlen af kapitlet. Søger læser information om f.eks. database design kan læser sagtens gå direkte til dette afsnit. Forståelsen for beslutninger og hvorfor visse elementer er implementeret som de er, kan gå tabt ved ikke at læse hele rapporten, men for at forstå selve koden er det muligt at læse enkelte dele for sig. Rapporten bør læses af de som har interesse i udarbejdelsen, samt slutresultatet af systemet.

Redaktionelt

Dette afsnit beskriver, hvem der har haft ansvaret for de forskellige afsnit i rapporten, samt hvem der har bidraget til og kontrolleret hvert afsnit. Dette kan ses i tabel [0.1].

Afsnit	Ansvarlig	Bidrag fra	Kontrolleret af
<i>Abstract</i>	Mikail	Anders	Christopher
<i>Forord og Læsevejledning</i>	Anders	-	Simon
<i>Indledning</i>	Alle	-	Alle
<i>Metode og Planlægning</i>	Aleksander, Simon	Mikail	Christopher
<i>Krav</i>	Aleksander	-	Anders
<i>Analyse</i>	Anders, Christopher	Aleksander	Mikail, Jakup
<i>Design</i>	Mikail, Jakup	Christopher	Anders
<i>Implementation</i>	Christopher	-	Anders
<i>Implementation: Database</i>	Aleksander	Mikail	Jakup
<i>Test</i>	Anders	-	Mikail
<i>Diskussion</i>	Anders, Christopher	Mikail, Jakup	Jakup
<i>Konklusion</i>	Alle	-	Aleksander
<i>Procesevaluering</i>	Christopher, Anders	Simon	Mikail

Tabel 0.1: Tabel over arbejdsfordeling i rapporten

Indholdsfortegnelse

1 Indledning	8
1.1 Introduktion	8
1.2 Problemanalyse	8
1.3 Undersørgsmål	10
1.4 Afgrænsning	10
1.5 Kritiske Risici	11
2 Fagligt Vidensgrundlag	12
2.0.1 Begrebsdefinitioner	12
2.0.2 Teori	12
3 Metode og planlægning	14
3.0.1 Unified Process (Lightweight)	14
3.0.2 Scrum rammeværktøj	16
3.1 Udviklingsprocessen	18
3.2 Øvrige værktøjer	20
4 Krav	21
4.1 Scope	21
4.2 Modellering af krav	22
4.2.1 Brugsmønstre	23

5 Analyse	27
5.1 Krediteringer	27
5.1.1 Historisk kontekst	27
5.1.2 Visning af kreditering	27
5.1.3 Forskel i udsendelser	28
5.1.4 Notifikationer	28
5.1.5 Grafik	28
5.1.6 Database	30
5.2 Analyseklassediagram	30
5.2.1 Systemsekvensdiagrammer	31
6 Design	34
6.1 Softwarearkitektur	34
6.1.1 Trelagsdeling	34
6.2 Pakkediagram	37
7 Implementation	38
7.1 Fladfil System	38
7.2 Brugere	39
7.3 Programmer	40
7.4 Kreditering	41
7.5 Notifikationer	41
7.6 Visning på TV	42
7.7 Database	43
7.7.1 Kortlægning af entiteter og relationer (EER)	43
7.7.2 Generering af tabeller (UML)	45
7.7.3 SQL - konstruktion af script	46
8 Test	48
8.0.1 1. Iteration	48
8.0.2 2. Iteration	48
9 Diskussion	49
9.1 Perspektivering	50
9.1.1 Brugbarhed	50
9.1.2 Næste skridt	50
10 Konklusion	52
11 Procesevaluering	53
11.1 Vejlederaftale	53
11.2 Samarbejdsaftale	53
11.3 COVID-19	54

11.4 Teamroller	54
11.5 Inceptionsfasen	55
11.6 Elaborationsfasen	56
11.6.1 1. Iteration	56
11.6.2 2. Iteration	56
11.7 Brugen af metoder	56
11.7.1 Kanban	56
11.7.2 Github	57
11.7.3 Belbin	57
11.7.4 Pairprogramming	57
Litteratur	58
12 Bilag	59
12.1 Scrum	59
12.2 Samarbejds og vejlederaftale	59
12.3 Logbog	59
12.3.1 Øvrige værktøjer	59
12.4 Design klassediagrammer	60
12.4.1 Datalag	60
12.4.2 Domænelag	61
12.4.3 Præsentationslag	62
12.5 SQL script	63
12.6 Cyber-physical systemer	65
12.7 Inceptionsdokument	71
12.8 TV2 Casebeskrivelse	88

Indledning

1

1.1 Introduktion

Ved afslutningen af film og TV-udsendelser er der tradition for at have krediteringer af skuespillere og produktion, som ofte vises med tekst i varierende størrelser centreret på sort baggrund. TV2 ønsker forslag til et nyt krediteringssystem, der ultimativt kan betyde en besparelse/fortjeneste på op til 60 millioner DKK ved at frigøre op mod 30 sekunder per program, der i dag anvendes til krediteringer. Med ønsket om forslaget til en løsning er der opstillet en række tekniske krav, som opgaveløser frit kan vælge i mellem.

Udledt af casebeskrivelsen lyder gruppens problemstilling således:

Hvordan kan et krediteringssystem hjælpe TV2 med at fjerne slutkrediteringer og derved frigøre 30 sekunder fra sendefladen, efter endt program?

1.2 Problemanalyse

Følgende områder af tekniske krav til krediteringssystemet fra TV2 er valgt på baggrund af interesse:

1. Registration of Credits (registreringer af krediteringer)
2. Notifications (notifikationer)
3. Access Control (specifikke brugertyper)
4. Making Persons Unique (unikke personer)

Der skal udvikles en grafisk brugerflade, der har til formål at simulere en TV-skærm, hvor krediteringer for det enkelte program kan vises henover programfladen. Dette giver TV2 mulighed for at fjerne slutkrediteringer fra udsendelser, og derved spare op til 30 sekunder efter hvert enkelt afsluttet program. Herved opnås TV2s ønske om at få frigjort tiden i sendefladen og hente en fortjeneste på op til 60 mio. kr.

Som en yderligere besparelse tilføjes mulighed for, at producenter selv kan indtaste krediteringer og oprette nye udsendelser/programmer. Det frigør mandetimer for både TV2 og producent, der i dag har ressourcekrævende mailkorrespondancer om udførelsen af krediteringslisten. Ved at give producenter mulighed for at skrive krediteringer direkte ind i krediteringssystemet gøres processen altså mere strømlinet.

Når producenter tilføjer krediteringsberettigede personer for første gang i databasen vil de oprettes. Herefter, hvis en producent bruger samme person, kan de blot tilføje den nye kreditering til denne. Derved kan det ses at en fiktiv skuespiller *Hans Hansen* er medvirkende i alle afsnit af *Badehotellet* og der ikke bliver oprettet en ny *Hans Hansen* for hvert afsnit af tv-serien. Skulle det ske at der var navnesammenfald for en lydmand, kan de adskilles, ved at se andre krediteringer under den unikke person.

Specifikke brugertyper oprettes til krediteringssystemet, således TV-seeren kun har adgang til at se oprettede krediteringer og oprettede programmer. Her er login ikke nødvendigt, ej heller muligt. Brugeren kendes ved et unikt login, hvorved brugeren - en producer - kan se dennes programmer og deres krediteringer, samt tilføje, redigere og slette programmer og krediteringer. TV2 kan sidde med en administrator og holde styr på krediteringer og eventuelle fejl heri. Administrator brugertypen vil have adgang til hele databasen og kunne tilføje, redigere og slette i samtlige programmer og krediteringer. Dertil oprette, redigere og slette brugere.

Brugertyperne er lavet ud fra hierarkisk pyramide struktur, således at administratoren har adgang til alle funktioner, brugeren har adgang til sine egne ting og seeren har adgang til at se krediteringer og intet andet. Det er bygget op således, for bedst muligt at kunne vedligeholde og fremtidssikre systemet.

Opgaven, at indskrive krediteringer er givet til de enkelte producere, i stedet for TV2 selv. Derfor er et notifikations system nødvendigt, således at administrator fra TV2 kan holdes opdateret på nye programmer, krediteringer og aktører. Der kan køres kvalitetskontrol og TV2 har mulighed for at rette eventuelle fejl, hvis der skulle være et behov. Alle ændringer der laves i programmet lander herved i notifikationssystemet og skal godkendes af administrator, før det bliver endeligt tilføjet i systemet. Dette sikrer at fejlinformation ikke kommer ud til seeren eller videre til f.eks. Registrering Danmark eller evt. brancheorganisationer.

1.3 Underspørgsmål

For at bryde problemformuleringen op har gruppen formuleret en række underspørgsmål, der anvendes til at besvare hovedspørgsmålet.

- Hvad er et krediteringssystem?
- Hvilke regler og konventioner for kreditering skal overholdes?
- Hvilken forskel skal der være på de forskellige brugere?
- Hvordan kan en database benyttes til denne udarbejdelse?
- Hvordan tilføjes og ændres kreditering til krediteringssystemet?
- Hvordan rettes fejl eller snyd i systemet?
- Hvordan kan dette hjælpe TV2 med at fjerne kreditering i slutningen af et program?

1.4 Afgrensnings

TV2s krediteringsformalia følges ikke, da der fokuseres på nye krediteringer og ikke ældre krediteringer, der skal tilføjes til systemet. Det vil ikke være muligt at importere eller eksportere data til databasen igennem f.eks .xml filer. Der vil ikke være kompatibilitet med EPG systemer og systemet vil ikke indeholde en API, som andre kan tilkoble sig. Andre organisationer vil ikke kunne modtage notifikationer omkring nye krediteringer til udsendelser, da det ikke bliver integreret i andre systemer.

Organisationer som f.eks. Center for ophavsret og Producentforeningen vil ikke kunne modtage notifikationer, da de kun distribueres til interne administratorer. Ønsket om at systemet skal være offentligt tilgængeligt bliver kun delvist mødt, idet systemet kun kan tilgås af det offentlige via TV og ikke kommer på internettet, igennem f.eks. en hjemmeside.

1.5 Kritiske Risici

Som en del af inceptionsdokumentet blev der udarbejdet en liste over kritiske risici. De blev diskuteret og i nogen grad inddelt. Listen er vist i tabel 1.1

Risiko	Risiko vurdering	Indvirkning	Håndtering	Kritisk?
Belbin	Middel	Middel	Gruppen er gennem projektforløbet fokuseret på de svagheder der er, i forhold til gruppens belbinroller, da dette kan medføre til et mindre godt projekt.	Ja
Tidsplan	Middel	Høj	Gruppen overholder tidsplanen nøje for ikke at overskride afleveringsfrister, dermed får gruppen også brugt tiden ligeligt på de forskellige opgaver i projektet.	Ja
COVID-19	Høj	Høj	Projektet bliver udarbejdet hjemme, hvor gruppen håndtere den indbyrdes kommunikation over programmet <i>Discord</i> . Hertil bliver vejledermøder også håndteret på <i>Microsoft Teams</i> .	Ja
Sygt gruppe-medlem	Lav	Middel	Et sygt gruppemedlem meddeler først og fremmest til gruppen at personen er syg. Hvis personen gentagende gange i streg melder sig syg bliver dette drøftet i gruppen.	Nej
Gruppemedlem melder sig ud	Lav	Middel	Tidsplanen og forventningen til projektet bliver revurderet for at omfordele arbejdsbyrden på passende vis.	Nej

Tabel 1.1: Tabel over risikovurderingen af forskellige scenarier

Fagligt Vidensgrundlag

2

2.0.1 Begrebsdefinitioner

Der gøres brug af en række begreber. De væsentligste gennemgås her:

Ord	Betydning
<i>creDB</i>	Det udviklede krediteringssystem
<i>UI</i>	User Interface (brugergrænseflade)
<i>Query</i>	Forespørgsel til en database
<i>EPG</i>	Kort for <i>Electronic Programme Guide</i> , er betegnelse for programoversigten over udsendelser på TVet og deres tilhørende data.
<i>API</i>	Kort for <i>Application Programming Interface</i> er en dokumenteret facade i et program som andre programmer kan kommunikere med.
<i>Overlay</i>	Her refereres til et lag der ligger ovenpå et andet med en anden funktionalitet

2.0.2 Teori

Vidensgrundlaget for udarbejdelsen af prototypen creDB og tilhørende projektrapport er baseret på og tager udgangspunkt i en række teorier og tilgange til udvikling af software. De enkelte tilgange beskrives kortfattet her, da de er berørt andetsteds i rapporten i mere dybdegående form. Der er angivet centrale kilder, som interesserede kan dykke ned i. Se evt. kapitel 3.

General Software Engineering

Unified Process har været en gennemgående teoretisk tilgang til udviklingen af produktet. Der er arbejdet indgående med artefakter som generering af modeller og brugerfladekoncepter samt aktiviteter som konstruktion af domænemodel og implementering af konkrete

klasser. [4]

Processværktøjer

Scrum rammeværktøj til projektstyring har været anvendt til udarbejdelsen af projekt og rapport. Der er taget udgangspunkt i den officielle tilgang, men i en modificeret udgave. Der er opereret med artefakter som Backlogs, Roller, Sprints og Daily Scrum Meetings. Se evt. kapitel 3. [5]

Objektorienteret Programmering

Der er anvendt basalt objektorienteret programmering til udvikling af systemet. Dette indebærer kendskab om læsning og skrivning til filer, tråde, opbygning af softwarearkitektur såsom facade- og singleton-klasser, og slutteligt viden om forbindelse og samspil med en database. [2]

Data Management

Der er anvendt teori indenfor databasemodellering såsom verb/noun analyse af entiteter og deres relationer. Derudover modellering af EER diagrammer og tabelgenerering. [3]

Metode og planlægning

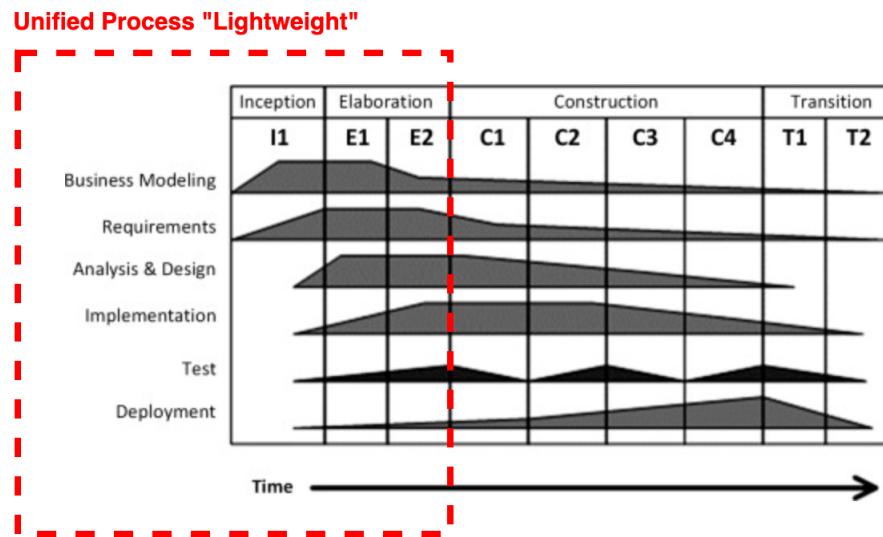
3

Under udarbejdelsen af projektet er en række redskaber blevet benyttet for at strukturere udviklingsarbejdet. Dette spænder over værktøjer fra afdækning af personlige kompetencer vha. Belbin til Scrums rammeværktøj for projektstyring, samt en lang række underliggende værktøjer til f.eks. visualiseringer vha. diagrammer. De væsentligste værktøjer præsenteres i dette kapitel, og det gennemgås i hvilken form de er anvendt i produktudviklingen af færdig prototype, og i tilblivelsen af den tilhørende rapport. Ydermere gennemgås planlægningen af ressourceforbrug i projektets forskellige faser.

To overordnede metoder har været styrende i udviklingsprocessen af løsningsforslaget til projektet. Det er en modificeret udgave af *Unified Process* kaldet *Unified Process Lightweight* og rammeværktøjet *Scrum* til projektudarbejdelsen. Begge metoder har haft til formål at sikre højkvalitets udvikling.

3.0.1 Unified Process (Lightweight)

Unified Process er en metode til softwareudvikling baseret på løbende udvidelse og forfining af et system. Systemet udvikles trinvist over tid i forskellige iterationer. UP indeholder overordnet fire faser, se figur 3.1, med en række underliggende discipliner. Faserne er fortløbende og hver fase repræsenterer en generation af produktet i tilblivelsen.



Figur 3.1: Illustration af faser og discipliner i *Unified Process*.

Et af de formelle krav til projektarbejdet er at gruppen skal udvikle en prototype, der løser problemstillingen beskrevet i casebeskrivelsen. *Lightweight* begrebet er defineret ved kun at indeholde de to første faser i processen: Inception og Elaboration, dvs. at Construction og Transition er udeladt i processen. Faserne er indrammet med rød stiblet markering i figur 3.1.

Inceptionsfase

Inceptionsfasen er størrelsesmæssigt den mindste af faserne. Her præciseres scope for projektet og kravene eliciteres. Alle eksterne entiteter som systemet skal interagere med skal præciseres. Alle brugsmønstre skal identificeres, og enkelte bør beskrives.

Følgende punkter opsummerer resultatet af inceptionsfasen [1]:

- Generel vision for projektets krav, kernefunktioner og afgrænsninger.
- Domænemodel.
- Indledende brugsmønstre (10-20% færdige).
- Indledende risikovurdering.
- Projektplan med faser og iterationer.
- En el. flere prototyper.

Elaborationsfase

Fasen omtales som den vigtigste af de fire faser i UP. Projektgruppen skal have en klar forståelse af kravene, og en klar forståelse af problemet som projektet skal løse. Der udvikles en projektplan med en klar systemarkitektur. Arkitekturen skal afspejle forståelsen af hele systemet: scope, kernefunktioner og ikke-funktionelle krav.

Følgende punkter opsummerer resultatet af elaborationsfasen [1]:

- Brugsmønstre (80% færdige).
- Supplerende krav afdækker funktionelle/ikke funktionelle krav.
- Softwarearkitektur beskrivelse.
- Kørende prototype.
- Revideret risici.
- Udviklingsplan for det *endelige* projekt.
- Anvendt proces til udvikling.
- Eventuelt indledende brugermanual.

Anvendte diagrammer og modelleringsværktøjer

Der er anvedt en lang række visualiseringskoncepter i form af diagrammer i udviklingsfasen. I tabel 3.1 er faserne angivet, hvor arbejdet med den enkelte model er påbegyndt, men typisk har arbejdet med modeller gået på tværs af de to faser. F.eks. er brugsmønstre påbegyndt i inceptionsfasen og raffineret i elaborationsfasen.

3.0.2 Scrum rammeværktøj

Scrum[5] er et rammeværktøj til projektstyring af bl.a softwareudvikling. Det gør brug af artefakterne *Product Backlog* og *Sprint Backlog*. Arbejdet udføres i såkaldte *Sprints*, der er en fast defineret opgaveliste, der udføres indenfor en given tidsramme, der typisk varer 2-4 uger. Et sprints omfang planlægges i det såkaldte *Sprint Planning* som omfangsmæssigt afhænger af længden på et sprint. Der afvikles daglige møder *Daily Scrums* typisk af 15 minutters varighed, hvor alle gruppemedlemmer rapporterer om hvad der er opnået, udfordringer og fremtidige opgaver. Såkaldte scrum-butts er afvigelser fra rammeværktøjet for en given projektgruppe.

Scrum koordineres i en relativt flad struktur med tre typer af rollefordelinger: *Scrum Master*, *Product Owner* og *Development Team*. *Product Owner* godkender tilføjelser til backlogs,

Modeller	Faser
Brugsmønstre 1.0	Inceptionsfase
Domænediagram	Inceptionsfase
Design mockups	Inceptionsfase
EER diagram 1.0	Inceptionsfase
Brugsmønstre 2.0	Elaborationsfase
Analyseklassediagram	Elaborationsfase
Designklassediagram	Elaborationsfase
Systemsekvensdiagram	Elaborationsfase
EAST-ADL diagram	Elaborationsfase
EER diagram 2.0	Elaborationsfase

Tabel 3.1: Tabel over modeller, og den fase de er påbegyndt i.

bestemmer om et sprint skal forlænges eller stoppes før tid, og sørger for at gruppen prioriterer de vigtigste opgaver.

Scrum Master styrer det daglige møde, af en typisk varighed på 15 minutter sammen med resten af holdet, løser de problemstillinger opsat af *Development Teamet* og hjælper *Product Owner* med at udarbejde backlog elementer.

Development Team er ansvarlig for udførelsen af et sprint, og bestemmer hvornår et sprint er klar til at blive vist til *Product Owner*.

Product Backlog er en sorteret liste over alle de krav, der umiddelbart er kendte ift. det endelige produkt. Listen er karakteriseret ved at være dynamisk og aldrig færdiggjort.

Sprint Backlog er en liste over opgaver, som indgår i et sprint. Opgaverne er punkter fra Product Backloggen. Opgaverne bevæger sig fra To Do, Doing og til Done. Progressionen er visuelt synlig for alle gruppemedlemmer. Et eksempel på en sprint backlog ses i figur 3.2.

Scrum sprint backlog: creDB			
User story (forecasts)	# Tasks	Status	Ansvarlig
• Som administrator kan jeg oprette nye brugere • Som administrator kan jeg slette og redigere brugere • Notifikationer holder administrator opdateret om ændringer foretaget af brugere • Administratorer har samme anvendelsesmuligheder som brugere i programmet	1 Lav Account klasse + tilhørende subklasser 2 Lav Datalag som gemmer Users 3 Lav knytning mellem præsentations og domænelag 4 Opret grafik til Users og Admins 5 Funktionalitet i FXMLController for Admin 6 Lav Notifications klassen og knyt den til Users	Done Done Done Done Done Done	▼ Mikail og Simon ▼ Christopher og Simon ▼ Simon og Christopher ▼ Yakup og Aleksander ▼ Yakup og Aleksander ▼ Simon og Christopher
• Brugere kan logge ind • Brugere kan oprette programmer • Til programmer kan der tilføjes krediteringer • Brugerne kan få vist en liste over alle krediteringer til et program	1 Lav login funktionalitet, der kan anvendes på Account 2 Lav Broadcast klasse 3 Knyt Broadcast klasse til datalaget 4 Knyt Broadcasts til bestemte Users i datalaget 5 Funktionalitet i FXMLController for User 6	Done Done Done Done Done Done	▼ Christopher og Anders ▼ Mikail og Simon ▼ Mikail og Simon ▼ Christopher ▼ Christopher & Anders ▼
• Seer af TV programmer kan få vist krediteringer ved at åbne overlay på TV	1 Opret grafik for Viewer i præsentationslaget 2 Lav Interface skiftende programmer 3 Tilføj mulighed for at lave uddrag fra datalaget 4 Tilføj krediteringer tekst på viewer præsentationslaget	Done Done Done Done	▼ Mikail og Anders og Christopher ▼ Mikail og Anders ▼ Christopher og Simon ▼ Mikail og Anders

Figur 3.2: Sprint Backlog fra 1. iteration

Scrum implementering

Gruppen implementerede Scrum ved påbegyndelse af programmering og anvendte det til projektets ende. Scrumrollerne, ceremonier og scrum-buts er beskrevet i specifikke dokumenter som kan ses i bilag 12.1. Ceremonien består af et dagligt opstartsmøde på omkring 30 minutter, og et afsluttende møde på omkring 15 minutter, hvor hvert enkelt medlem har ordet skiftevis. På hvert møde blev der diskuteret:

- Hvad det enkelte gruppemedlem havde lavet, udfordringer, eventuelle nye muligheder, resultater og planer for fremtidige opgaver.
- Hvad gruppen bør fokusere på.
- Status på *Product Backlog*.
- Nedskrive spørgsmål til vejledermøder.

Disse Scrum-buts blev udarbejdet for at tilpasse Scrum til gruppens behov:

- Scrum møder bliver ikke afholdt dagligt, men i stedet i starten og slutningen af arbejdssdage.
- En uge var ikke tilstrækkeligt til at implementere al funktionalitet, derfor var sprintvarighed sat til samme varighed som hver iteration.
- For at facilitere læring er rolleopdelingen løs for at alle kan lære om Scrumelementer som f.eks. backlogs.

I forbindelse med introduktionen af Scrum, var sprint backlogs blevet oprettet til at beskrive de overordnede arbejdsopgaver. På disse backlogs oprettes alle de generelle opgaver, som skulle være implementeret i det givne sprint. Opgaverne oprettes ud fra user-stories, som forklarer brugen af systemet. Medlemmer blev tildelt specifikke opgaver, og hver opgave havde tre stadier (*ikke påbegyndt, påbegyndt, og færdigt*).

3.1 Udviklingsprocessen

Inception

I starten af denne fase gennemgik gruppen kravene knyttet til TV2s case og udvalgte en håndfuld af dem, som dannede scope for projektet. Gruppen besluttede hvilke krav det endelige produkt skulle opfylde og modsat hvilke krav gruppen ikke ville fokusere på. Kravene blev præciseret ved opfølgende møde med en repræsentant fra TV2, hvor gruppen kunne stille spørgsmål og drøfte ideer. Gruppen havde altså en generel vision

for projektet og der var bred konsensus omkring overordnede krav, kernefunktioner samt afgrænsinger. Kravet omkring cyber-physical tilgang til problemstillingen blev også drøftet i inceptionsfasen, hvor simple design illustrationer af kobling mellem knap på fjernbetjening og visning af krediteringer som metadata til specifikt program blev udarbejdet, som set på figur 3.3.



Figur 3.3: Illustration af første prototype for simulering af tryk på fjernbetjening for at aktivere krediteringsvisningen.

Dette fungerede principielt som en prototype af programmet, da det skabte enighed om retningen for projektet, i gruppen. Der blev i inceptionsfasen lavet brugsmønster- og aktørlistenter med brugsmønsterdiagrammer over de udvalgte krav. Dette havde til formål at tydeliggøre hvilke funktionaliteter programmet skulle have.

Udstyret med brugsmønsterdiagrammet påbegyndte gruppen derefter udarbejdelse af CRC kort, UML og EER diagram. Inceptionsfasen har normalvis også et stærkt fokus på selve forretningsmodellen i produktet, men denne vinkel er udeladt i dette projekt.

Elaboration - 1. Iteration

Arbejdet havde fokus på udarbejdelse af CRC kort og klassediagram over domænemodellen for at kunne påbegynde programmering. I løbet af de første par uger i fasen tilføjes et analyseklassediagram, som skabte et bedre overblik over de udfordringer, der skulle forceres.

Samtidigt integreredes Scrum i den eksisterende arbejdsstruktur, som beskrevet i 3.0.2.

Der blev oprettet en Scrum backlog for 1. iteration. I forbindelse med udarbejdelse af cyber-physical elementer blev der udarbejdet funktionelle og ikke funktionelle krav samt en EAST-ADL model, som kan ses i bilag [12.6](#). Fokus i midten til afslutningen af 1. iteration var at få færdiggjort prototypen samt at lægge fundamentet i koden til implementering af databasen. Her blev processer som brugsmønsterrealisering, kontrakter og delvise sekvensdiagrammer udarbejdet.

For at sikre lav kobling, en god softwarearkitektur og høj samhørighed blev systemets udformning ofte diskuteret.

Elaboration - 2. Iteration

En vigtig del af denne fase for gruppen var en revurdering af brugsmønstrene og en gennemgang af hvorvidt de var blevet implementeret. Dette blev gjort med mange af de eksisterende modeller.

Inden udviklingen af databasen startede blev EER diagrammet revideret og der blev lavet en grundig plan for ændringerne i det eksisterende system.

Par Programmering

Gruppen har gjort brug af par programmering, projektet igennem, til kode opgaver. Par programmering udføres ved at der er 2 personer på én opgave, med hver sin rolle. Der er *Driveren*, hvis opgave er at skrive og opsætte koden og *Observeren*, hvis opgave det er at kigge efter for fejl og hjælpe med idéer. Ved 2 programmører til én opgave opnås fordele til mere gennemtænkte løsninger. Gruppens medlemmer har alle forskellige erfaringer fra tidligere og tænker forskelligt. Dette fører til færre fejl og overordnet set bedre kode. Samtidig kan gruppens medlemmer få styrket eventuelle svagheder, ved først at kigge koden igennem som *Observer*, for derefter at give sig i kast med, selv at tage *Driver*-rollen.

3.2 Øvrige værktøjer

Liste af software anvendt til at producere hhv. systemet; creDB og tilhørende rapport ses i bilag [12.3.1](#).

Krav

4

4.1 Scope

Afsnittets formål er at levere en deltaljeret beskrivelse af processen omkring udformning af kravene til systemet creDB. Relevante interesser til produktet afdækkes vha. brugsmønstre. De funktionelle og ikke-funktionelle krav beskrives. Afsnittet er anvendeligt som en forventningsafstemning mellem opgavestiller og opgaveløser.

Visionen med produktet er et krediteringssystem til TV2. En seer af TV2s kanaler skal kunne tilgå krediteringsoplysninger for et specifikt program ved at klikke på en allokeret knap på fjernbetjeningen. På denne måde frigøres op til 30 sekunder efter hver udsendelse. Lovkrav, kutyme og traditioner forbundet med kreditering af værker i film og TV industrien udfordres på denne måde ikke, da krediteringerne stadig figurerer *på* værket. Systemet skal gøre det muligt for eksempelvis et produktionsselskab at oprette udsendelser med tilhørende krediteringer. Målet er at lave en smart og hurtig løsning, der er let at navigere rundt i for ulige brugertyper.

Kravteknik

Kravteknik er en bred term for de aktiviteter, der i udviklingsprocessen bygger bro mellem krav på den ene side og konstruktion og design på den anden. I projektet har tilgangen været at følge de traditionelt 6 aktiviteter, som forklares i tabel 4.1.

Aktiviteter	Beskrivelse
Inception	Grundlæggende forståelse af TV2s problem vedr. krediteringer. Mere generel forståelse af TV2 som virksomhed og deres forretningsområder. Overordnede krav til løsning og kommunikation etableres på tværs mellem projektgruppe, vejleder og repræsentant fra TV2.
Elicitering	Indsamling af krav fra bl.a casebeskrivelse og samtale. Forståelse af formålet med projektet. Dialog omkring kravene og en raffinering af eksisterende ideer til funktionaliteter baseret på opfølgende samtaler/foredrag.
Elaboration	Brugsmønstrene konkretiseres og raffineres. På baggrund af disse analyseres det hvordan, slutbrugerne af creDB skal interagere med klasser og attributter i programmet. Desuden er relationer imellem klasser identificeret.
Forhandling	Normalvis i en køb/salg proces vil der være en forhandling om hvorvidt produktet opfylder kravene. Dette har ikke eksplisit været en del af processen her. En demosession for øvrige projektgrupper midtvejs i forløbet fungerede principielt som noget tilsvarende, da gruppen fik feedback på kørende program
Specificering	Kravene er typisk dokumenteret i en artefakt kaldet krav-specificering som indeholder både tekst og modeller. Bliver normalvis først gyldigt efter valideringen.
Validering	En formel validering tjekker om krav og modeller stemmer overens med det interessenten rent faktisk efterspørger som løsning. Efter valideringen er specificeringen officiel.

Tabel 4.1: Aktiviteter i kravudvikling, med beskrivelser.

De sidste aktiviteter i den traditionelle fremgangsmåde er der afveget en smule fra i udarbejdelsen af dette projekt. Det kan dog siges, at der har været en form for specificering og validering løbende, hvor projektgruppen kontinuerligt har holdt produktet op imod TV2s problemstilling, for at forsikre at løsningen stadig løser problemstillingen.

4.2 Modellering af krav

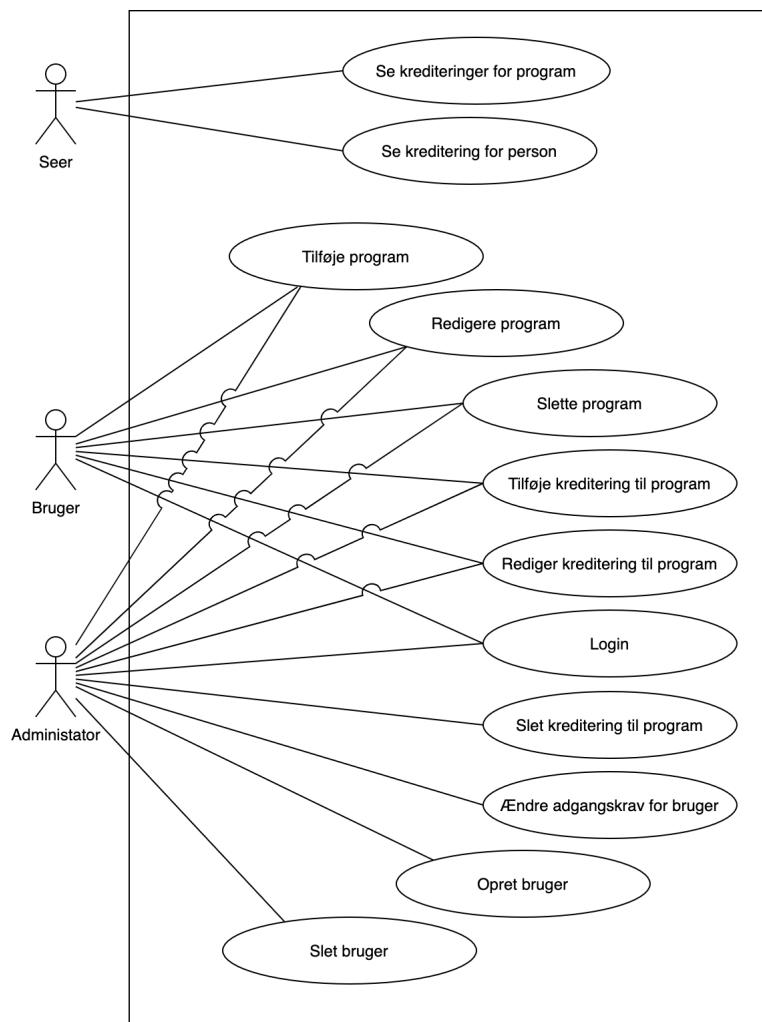
I inceptionfasen udarbejdede gruppen et projektdokument (inceptionsdokument), der indrammer visionen for projektet og planen for udførslen. Da processen omkring modellering af krav vha. f.eks. UML klassediagrammer har kørt sideløbende med en introduktion til teorien har visse områder fået for meget opmærksomhed i f.eks. inceptionfasen. Her kan brugsmønstre nævnes som et godt eksempel. Teorien beskriver en 10-20% færdiggørelse af brugsmønstre i inceptionfasen og en raffinering til op mod 100% færdiggørelse i elaborationsfasen. Realiteten blev at brugsmønstrene blev omrent 95% færdige i inceptionfasen, da de simpelthen fik for meget opmærksomhed pga. manglende

erfaring/færdigheder indenfor praktisk anvendelse af UP.

4.2.1 Brugsmønstre

Brugsmønstre er sekvenser af aktiviteter udført af en eller flere aktører, som enten kan være personer eller organisationer/systemer, der producerer et eller flere resultater til en eller flere af aktørene. Brugsmønstre er super anvendelige indenfor afdækning af krav og til at generere analyse, design og implementering.

Projektgruppen har aktivt anvendt brugsmønstre gennem hele udviklingsfasen. I figur 4.1 ses udførligt brugsmønster fra inceptionsfasen.



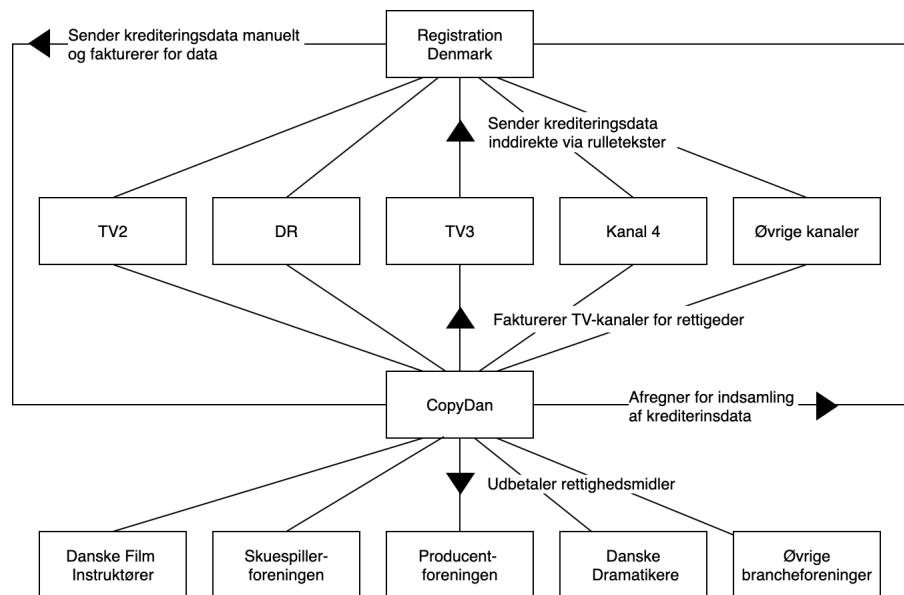
Figur 4.1: Brugsmønstre for de tre primære aktører: seer, bruger og administrator.

Identificering af aktører

Ud fra brugsmønstre er tre forskellige potentielle brugertyper af selve programmet identificeret, der kan karakteriseres som primære aktører:

1. Administrator
2. Bruger
3. Seer

I *Notifications* under krav i casebeskrivelsen 12.8 er det nævnt at øvrige aktører som *Samrådet for Ophavsret* og *Producentforeningen* kunne være eksempler på interesserter, der kunne være interesseret i at modtage notifikationer på ændringer foretaget i databasen. Disse aktører kan karakteriseres som sekundære. For at forstå interesserne i kreditering på tværs af markedet udarbejdede projektgruppen et diagram over samtlige interesserter.



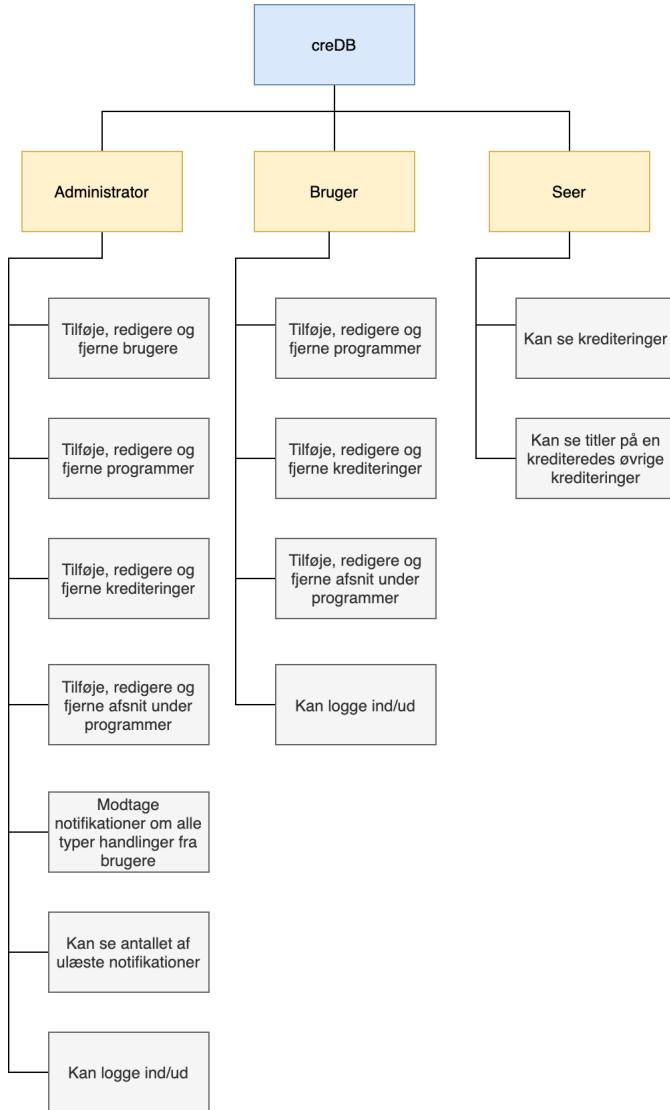
Figur 4.2: Diagram over sekundære aktører.

Research på strukturene omkring krediteringer og aktører bidrog til en bedre overordnet forståelse af problemstillingen omkring sekundære aktører, som ikke fremgår helt tydeligt af casebeskrivelsen. Diagrammet (forståelsen) bidrager endvidere til at generere idéer omkring nye løsninger til funktionalitet. Diagrammet åbner også op for at kigge på hvilke fordyrende led i ophavsrettighedsstrukturene der kunne arbejdes på at fjerne i en konstruktions fase.

Funktionelle krav

Ud fra casebeskrivelsen blev de primære brugertyper afdækket. Administrator og bruger kan tilgå systemet med ID og password. Brugere kan oprette, ændre og fjerne programmer

og krediteringer. Administrator har adgang til samme funktioner som bruger men kan derudover tilføje, fjerne og redigere brugere. Ydermere kan administratorer overvåge tilføjelser, ændringer og fjernede programmer og krediteringer foretaget af brugere via et notifikationssystem. Seeren kan tilgå krediteringer via konkret TV-udsendelse på sit private fjernsyn. Informationerne aktiveres via fjernbetjeningen og seeren kan i et lag (overlay) se f.eks. medvirkende skuespillere i den pågældende TV-udsendelse. Trædiagrammet i figur 4.3 visualiserer kernefunktionerne i programmet.



Figur 4.3: Trædiagram med de forskellige aktører, og de funktioner, som er tilgængelige for dem.

Ikke-funktionelle krav

I forbindelse med arbejdet omkring udvikling af cyber-physical er der udarbejdet en særskilt rapport der udelukkende fokuserer på funktionelle og ikke-funktionelle krav.

Se bilag [12.6](#)

Supplerende krav (FURPS)

De supplerende krav er blevet opstillet ved hjælp af FURPS model. Denne model er et hjælpeværktøj, til at sikre at et område af kravspecifikationen ikke bliver overset, og har en hierarkisk opbygning.

FURPS	Krav
Functionality	Skal kunne indeholde kreditering for alle de roller som er krediteringsberettigede.
Usability	Grænsefladen skal have høj kontrast mellem baggrund og tekst, så det er let læseligt. Systemet skal være intuitivt at benytte både for brugere og seere i systemet.
Reliability	Systemet må forventes at være oppe og tilgængeligt 24 timer i døgnet
Performance	Opslag i systemet skal returnere et korrekt resultat på under 5 sekunder.
Supportability	Krediteringen skal opbygges således at det kan indsættes i en relationel database.

Tabel 4.2: Aktiviteter i kravudvikling med beskrivelser

Krav om cyber-physical vinkel på projektet

Der lå et overordnet krav om at projektet skal have en vinkel på kobling mellem løsningsforslag og cyber physical produkt. Allerede i inceptionsfasen blev en model for hvordan denne integrering i løsningen kunne konstrueres. Ved at anvende fjernbetjeningen til at aktivere krediteringerne er der etableret en kobling. Se arbejdet med denne kobling i bilag [12.6](#).

Krav om anvendelse af fladfilssystem og relationel database

Der lå et overordnet krav om at løsningsforslaget skal være forbundet til hhv. fladfilssystem og database i hver af de to iterationer i elaborationsfasen.

Analyse

5

Analysen bygger på problemformuleringen, teorien og de valgte metoder, som sammenlagt danner et godt grundlag. Det vil altså være forklarende for de empiriske beslutninger, overvejelser, og resultater, som er beskrivende for programmet og projektets udformning.

5.1 Krediteringer

Krediteringer er typisk en liste over navne på bidragsydere, der har skabt pågældende film, serie eller udsendelse. Der er kutyme for krediteringer ved både begyndelsen og afslutningen, hvor såkaldte slutkrediteringer er mere omfattende. Projektet har alene fokus på krediteringer ved afslutningen af film eller udsendelser. Listen over medvirkende fremvises typisk i slutningen eller tæt på slutningen af film eller TV program. Listen kan uddover ovennævnte indeholde sponsorer, distributionsselskaber, musikere, forfattere, rettighedshavere mm. I denne, ofte omfattende, liste af bidragsydere eksisterer der en form for hierarki, der er vigtig at afdække for at kunne skabe en alternativ digital løsning til fremvisning af krediteringer.

5.1.1 Historisk kontekst

Krediteringer i spillefilm blev introduceret i 1970'erne. Før dette tidspunkt indeholdte produktioner kun krediteringer ved begyndelsen af værket. Op gennem historien findes der eksempler på film og TV-serier uden krediteringer ved begyndelsen og helt op til 37 minutters slut-krediteringer. Der er i dag konsensus om at både film og TV-serier indeholder omfattende lister over bidragsydere.

5.1.2 Visning af kreditering

Rulleteksterne i dag kan ses som resterne af fortidens TV. Der forsvandt megen information fra selve TV-apparatet, da tekst-TV forsvandt. Al tekst-TV lignende information er i dag, i stedet at finde online. I et forsøg på at lade TV'et beholde noget funktionalitet,

er der til creDB blevet udviklet et seer-overlay, således at den enkelte seer selv kan tilgå krediteringer for et program, og have så længe til at studere dette, som ønsket. Der er godt belæg for, at dette kunne være lagt online på en hjemmeside, men da der blot er tale om krediteringer for dansk-producerede programmer, blev beslutningen om overlay taget. Dette synes at passe bedre til målgruppen for systemet. Der kan via overlayet ses en række af de seneste, samt nuværende og kommende programmer, dertil disses krediteringer. I den udviklede version er der ikke forbundet til TV2s TV-signal. Der er derfor lagt stilbilleder af TV2-programmer ind, hvor den næste kreditering, samt titel og udgivelsesår bliver hentet fra databasen og vist via overlayet. Dette er altså blot en demo for at vise hvordan kreditering kan hentes fra databasen og blive vist ovenpå et program.

5.1.3 Forskel i udsendelser

Der er blevet skabt 3 udsendelsestyper i creDB, som brugere kan vælge fra, når der oprettes nye programmer. De er blevet inddelt i, Film, Serie og Liveshow. De har alle en titel, beskrivelse og udgivelsesår. Serie har derudover også sæson nummer, samt episode nummer og episode titel. Liveshow har som noget unikt en lokation, hvor dette foregår. Det blev valgt således, da det ikke gav mening at en film havde en sæson, eller at der kunne opstå tomme værdier i databasen, da en serie ikke har en lokation. Det er både mere brugervenligt, at disse er inddelt sådan, men også lettere for administratorer at kende forskel på de enkelte ting i databasen. F.eks. kunne der komme en dokumentarfilm om X-faktor, der i forvejen er et liveshow. Skulle der ændres i én af tingene er det lettere at kende forskel på den respektive film og liveshow.

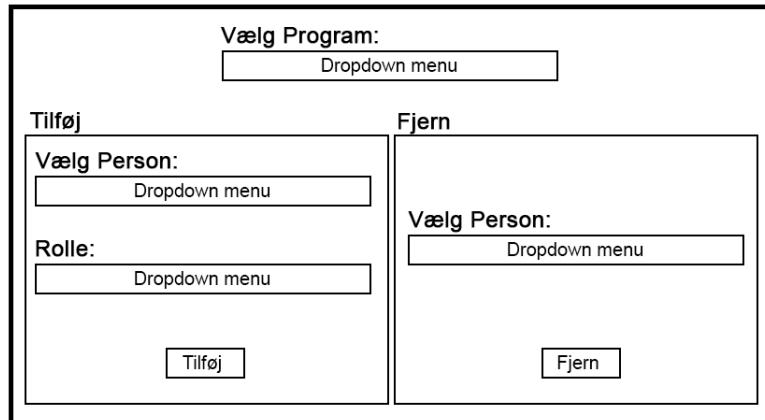
5.1.4 Notifikationer

Et af kravene omhandler et notifikationssystem, som skal notere hver gang en ændring sker. Den kan underrette systemadministratorer, ejerne af den ændrede udsendelse, eller måske endda eksterne partnere såsom *Samrådet for Ophavsret* eller *Producentforeningen*. I creDB er et notifikationssystem blevet implementeret med fokus på at underrette systemadministratorerne. Dette er fordi de er vurderet til at være de mest essentielle modtagere af notifikationer, da de har til formål at opretholde pålideligheden af indholdet i databasen. Hver gang en bruger af systemet opretter, sletter eller ændrer en udsendelse eller kreditering, vil en notifikation som indeholder data om hvornår, hvem og hvad der er ændret blive tilføjet til administrators notifikationsliste.

5.1.5 Grafik

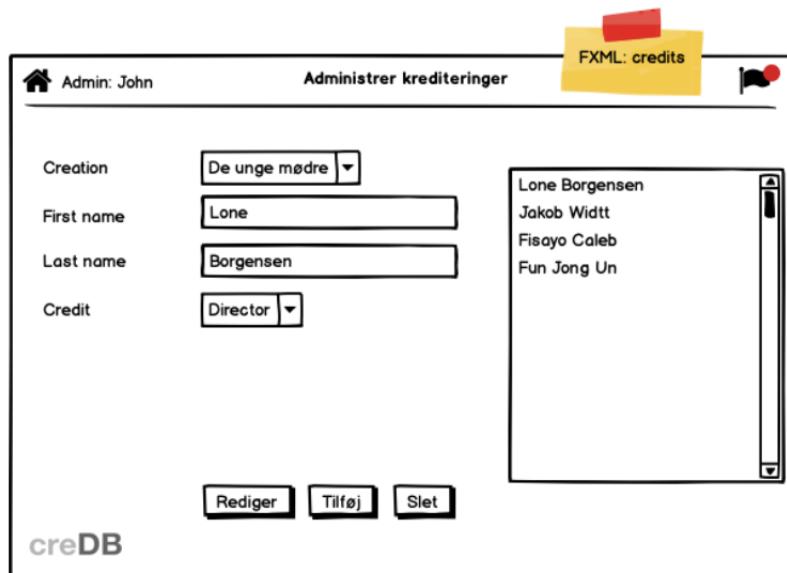
I starten af projektet var der en del diskussion omkring hvilke af kravene, der var mest interesse for, og hvilke der derved skulle fokuseres på. Under denne proces som skulle formulere indholdet af programmet, blev der udviklet en række skitser for at

illustrere gruppens vision. I dette afsnit vil der ses på udviklingen fra start til slut i den del af brugergrænsefladen, som omhandler tildeling af kreditering til personer. På figur 5.1 ses hvordan den tidligste skitsering af dette var udtænkt. Det er lavet med en hær af dropdown menu'er, og det stod hurtigt klart at der var brug for en bedre måde at visualisere alle de krediterede personer.



Figur 5.1: Første skitse over kreditering af personer.

På figur 5.2 ses hvordan dette problem er løst ved at implementere en altid synlig liste til visning af de krediterede personer. Ligeledes er oprettelse af personer rykket ind i denne visning. Tidligere var det tænkt at have en separat side til at oprette personer, som var gyldige valg til kreditering. Grunden til dette er, at skabe mindre forvirring over at skulle igennem mange forskellige visninger for at oprette en udsendelse, og endeligt færdiggøre den ved at tilføje kreditering.



Figur 5.2: Endelige skitse over kreditering af personer.

På figur 5.3 ses et billede af hvordan visningen for kreditering af personer endeligt så ud, og at den nærmest er identisk med den udviklede skitse.

Fornavn	Efternavn	Rolle
Lars	Kaalund	Instruktør
Jannik	Johansen	Instruktør
Mille	Dinesen	Hovedrolle
Nikolaj	Groth	Birolle
Lisa	Bastrup	Birolle
Carsten	Bjørnlund	Birolle
Sara Hjort	Ditlevsen	Birolle
Morten Vang	Simonsen	Birolle

 The footer of the application says 'creDB'."/>

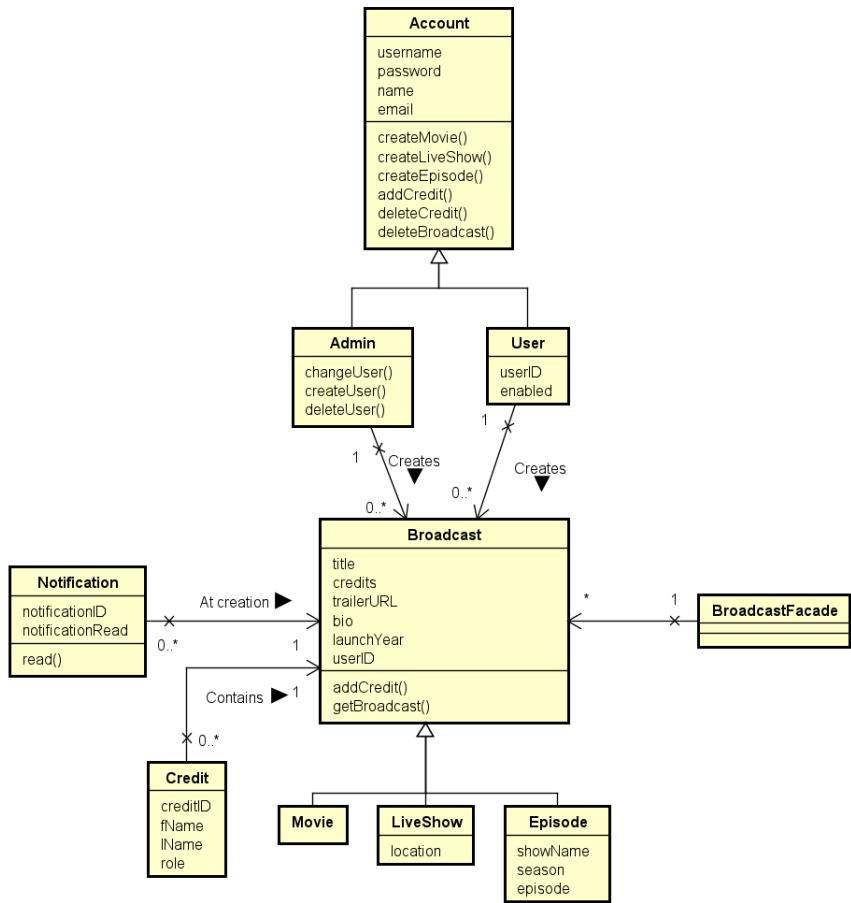
Figur 5.3: Billede af visningen for kreditering af personer i programmet creDB.

5.1.6 Database

I den anden iteration af projektet blev filhåndteringssystemet rykket over på en relationel database. Her var det muligt at lave en lokal database, eller have en database i skyen, som alle kan forbinde til. Der blev besluttet at investere tid og energi i at oprette en database i skyen. Det er gjort med *Amazon Web Services* igennem *Github Student Developer Pack*. Dette er på baggrund af fokus i læring og viden om hvordan en sådan database skal sættes op, og hvordan det skal håndteres når flere personer arbejder med den samme database. Det var desuden fordelagtigt at ændringer i databasens struktur var tilgængelige for alle således at der ikke blev arbejdet på outdatede versioner lokalt. Dette ville også gøre det nemmere for folk udenfor projektgruppen at afprøve programmet. Dette har desværre også nogle ulemper. Al data ligger hos en tredjepart, og det kan forårsage tab af data. Desuden vil det også betyde at en internetforbindelse er påkrævet for at køre programmet.

5.2 Analyseklassediagram

Til projektets 1. iteration blev et analyseklassediagram udarbejdet, således gruppen kunne nå til enighed om den generelle opbygning og udformning af det endelige program. Da diagrammet blev lavet meget tidligt i projektet, var der ikke blevet gået i dybden med, hvordan de forskellige ting skulle opbygges, uover i domænelaget. Derfor ses kun domæneklasser og deres relationer. Diagrammet afbilleder kun selve krediteringssystemet og ikke seerdelen. Seerdelen var tænkt på og der var således enighed om tilblivelsen senere hen, men ikke nok til at det kunne tilføjes til analyseklassediagrammet.



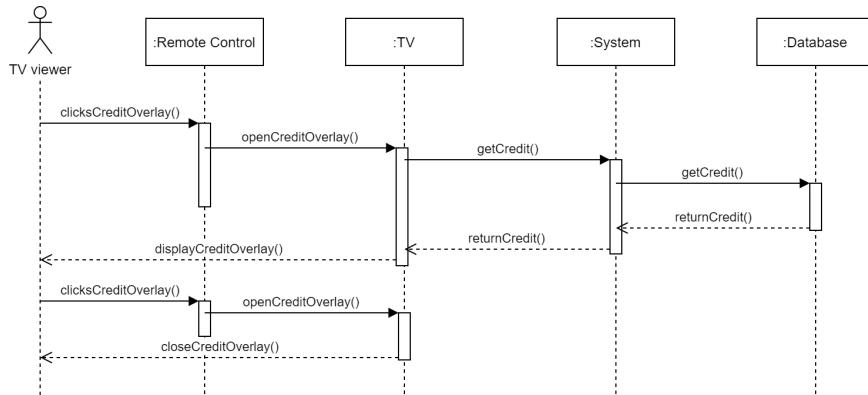
Figur 5.4: Analyseklassediagram lavet i starten af første iteration

5.2.1 Systemsekvensdiagrammer

Der er blevet udvalgt vigtige brugsmønstre, der skulle laves systemsekvensdiagrammer på. Disse er valgt på baggrund af MoSCoW og FUPRS+ analyser.

Se kreditering for program, B01

Det gennemgående problem for TV2 er, at de ikke kan vise alle krediteringer, grundet begrænsningen på 30 sekunder. Derfor er dette også grundlag for at udarbejde et krediteringssystem, således at alle bidragsydere kan blive krediteret for deres respektive medvirken. I creDB er det derfor muligt at skrive alle aktørroller ind, uanset om der modtages betaling fra CopyDan eller der er tale om frivillige aktører. Altså er det muligt at se den fulde krediteringsliste for et givet program i creDB. Det er gjort muligt for séeren, selv at kunne få krediteringerne frem, på et givent tidspunkt vha. fjernbetjeningen og derved fremkalde informationerne fra systemet, der henter dem fra databasen, som set på figur 5.5.



Figur 5.5: Systemsekvensdiagram for brugsmønster: Se kreditering for program, B01

Tilføj program, B03

Brugere og administratorer kan tilføje nye programmer via en menu, med både en indbygget dropdown menu, samt tekstfelter til diverse info. Dette synes vigtigt at give brugere mulighed for selv at kunne dette, da TV2 har udtrykt, at nuværende løsning, hvor der sendes krediteringslister for programmer via mail, er langt fra optimal. Selve oprettelsen lagres direkte fra programmet til databasen, og så snart et program er oprettet, kan der tilføjes krediteringer hertil.

Login, B09

Der er valgt login system vha. brugernavn og adgangskode for både brugere og administrator. Et af kravene fra TV2 var som bekendt, at en del af programmet skulle være offentligt tilgængeligt, derfor mener gruppen at login for seere ikke er nødvendigt. Login skærmen for brugere og administrator er ens, da der er oprettet rettigheder baseret på fornævnte brugernavn.

Administrere ændringer, B10

For at sikre sig korrekte oplysninger uden slåfejl eller andre små fejl, er det gjort muligt for administratorer at administrere ændringer. Dette er gjort muligt ved brugen af notifikationer, så administratorer bliver gjort bekendt med nye oprettelser og ændringer i krediteringer. Herfra var det oprindelige ønske, at administratorer skulle kunne godkende eller afvise ændringer, før disse blev lagt i databasen. I det udviklede program er meget af forarbejdet for dette lagt, men selve funktionaliteten nåede ikke at blive implementeret.

Opret bruger, B11

Det er muligt for administratorer at oprette brugere. Ideen om, at produktionsselskaber selv kunne få en administratorlignende bruger til deres producent individer var oppe at

vende, men denne blev lagt på hylden, da det ville blive rigtig meget arbejde, uden nogen særlig funktionalitet. Administratorer kan åbne en menu og herved indtaste oplysninger på den nye bruger. Disse lagres i databasen og den nye bruger kan straks logge sig ind.

Design

6

For at overdrage gruppens ideer og overvejelser til implementationsfasen, gjorde gruppen brug af forskellige modeller og metoder, til at vise hvordan programmets struktur skulle designes. Dette blev udarbejdet i designfasen. Her blev der gjort overvejelser, taget beslutninger og udarbejdet resultater, med hensyn til softwarearkitektur og trelagsdeling. I denne fase tog gruppen udgangspunkt i materiale, som blev udarbejdet i analysefasen.

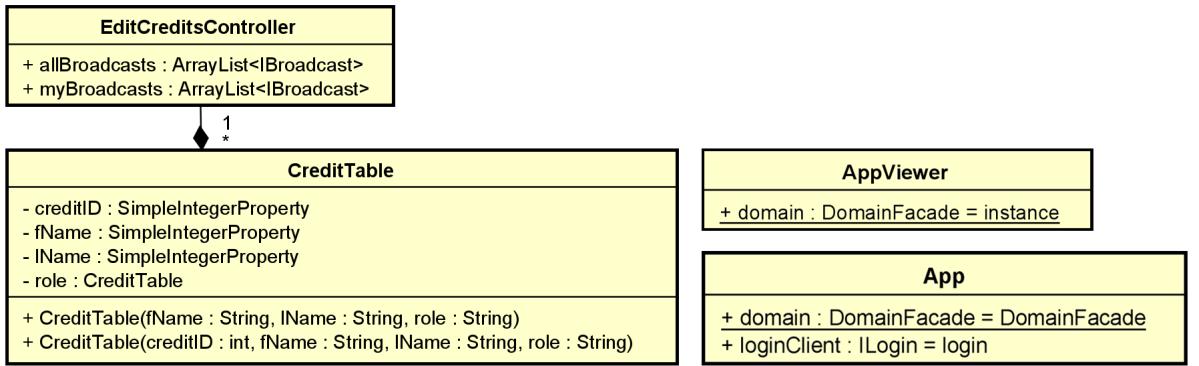
6.1 Softwarearkitektur

6.1.1 Trelagsdeling

CreDB er udviklet ud fra trelagsdeling i softwarearkitekturen. De tre delinger består af et data-, domæne- og præsentationslag. Ved at opdele systemets software vil de forskellige dele af systemet være isoleret. Det vil sige at de tre delinger har hver deres formål. Datalaget har til formål at kommunikere med databasen og derved håndtere lagring af data. Domænelaget indeholder kernelogikken for selve programmet, og bliver brugt som mellemmand til kommunikation mellem data og præsentationslagene. Præsentationslaget er brugergrænsefladen, og har derfor til formål at gøre programmet let brugbart for brugerne.

Præsentationslag

Præsentationslaget indeholder hovedsageligt Java controllerklasser. CreDB består af to visuelle programmer, *App* og *AppViewer*. I figur 6.1 ses et uddrag af klassediagrammet for præsentationslaget. Her ses klasserne for de visuelle programmer, og at de begge indeholder en attribut *domain* af typen *DomainFacade*. Dette gør det muligt for klasserne at benytte elementer fra domænelaget, og derved også datalaget. Ligeledes ses der en attribut *loginClient*, som kun er tilstede i *App*. Dette er fordi, *App* er det program som kræver login, imens *AppViewer*, er emuleringen af TVet som ikke kræver login. Resten af designklassediagrammet for præsentationslaget kan ses i bilag 12.4.3.

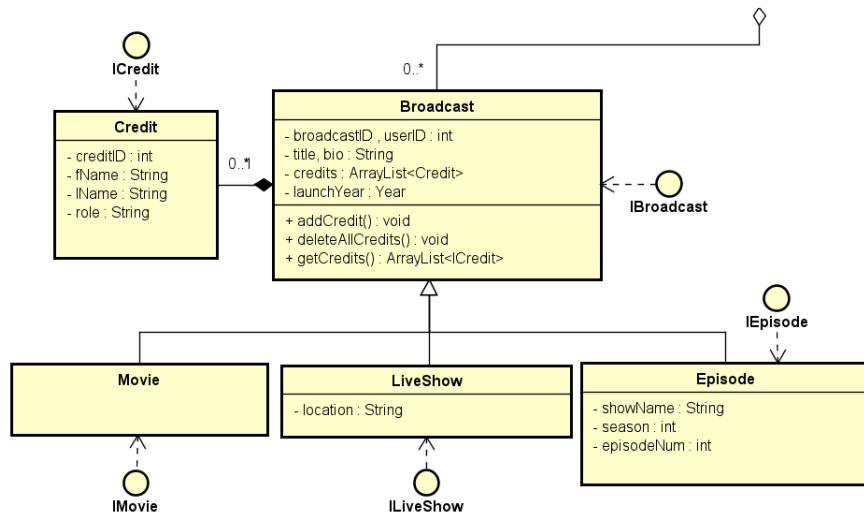


Figur 6.1: Uddrag af designklassediagrammet for præsentationslaget.

Domænelag

Der blev endnu engang diskuteret indholdet af de nødvendige klasser såvel som sammenhængen imellem dem. I denne forbindelse blev de specificeret ved tilføjelse af attributter og metoder. De fleste klasser i domænelaget har også implementeret et unikt interface, for at hjælpe med abstraktionen op til præsentationslaget. Et eksempel på dette ses i figur 6.2. Her ses klasserne omkring udsendelser og kreditering, og hvordan de alle har deres unikke interface. Den eneste klasse i domænelaget som ikke implementerer et interface er *DomainFacade*-klassen. Dette er fordi, som det lyder af navnet, den er en facade-klasse, som skal være med til at drage funktionaliteten op til det næste lag.

På figur 6.2 ses som sagt klasserne for udsendelserne og krediteringen dertil. Det er altså resultatet af overvejelsene gruppen har gjort sig, omkring opbygningen af forskellige typer af udsendelser. Det er derfor relevant, og sigende for hvordan udvidelse af flere typer udsendelser kan tilføjes til systemet. I denne version er der implementeret 3 typer: *Movie*, *LiveShow* og *Episode*. Der findes ikke en separat klasse til at håndtere serie eller sæsoner, da disse dækkes af attributter på episoden. Der ses også at klassen *Broadcast* indeholder en liste af objekter fra klassen *Credit* som indeholder kreditering. Resten af designklassediagrammet for domænemodellen kan ses i bilag 12.4.2.

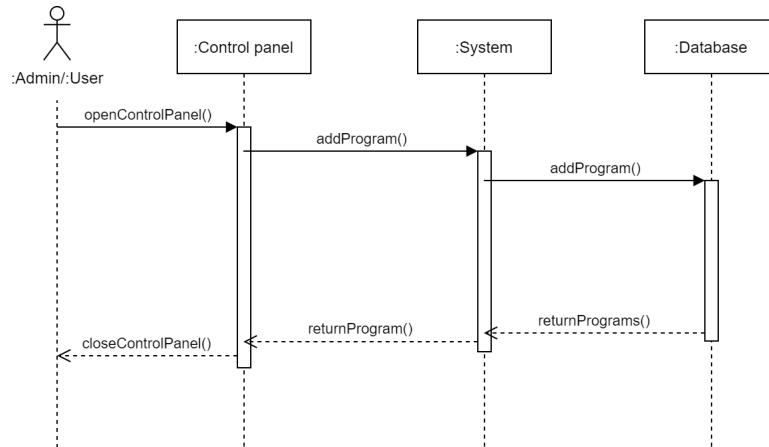


Figur 6.2: Uddrag af designklassediagrammet for domænelaget.

Datalag

Formålet med datalaget er at skabe persistens i programmet, på denne måde har programmet mulighed for permanent at gemme data. I første Iteration bestod denne data blot af tekstfiler. Datalaget har to klasser, en til at skrive til data, *Writer*, og en til at læse fra data, *Reader*. Al kommunikation med dataen foregår altså igennem disse klasser. Et designklassediagram for datalaget kan ses i bilag 12.4.1.

Da anden iteration begyndte kom designet af datalaget endnu engang på tale. Der skulle nu implementeres en relationel database som erstatning af tekstfilerne, hvilket krævede store ændringer af netop dette lag. Der blev oprettet en ny klasse *ConnectionDB*, som skulle sørge for at forbinde til databasen inden der bliver forsøgt at skrive, eller læse fra den for at undgå problemer. *Reader* og *Writer*-klasserne blev begge omskrevet til at implementere brugen af *queries* til databasen. Desuden betød dette at mange metoder nu blot behøvede et id som argument, da databasen automatisk kan oprette unikke id'er for data. Sluttligt blev *DataFacade*-klassen opdateret så resten af programmet var samhørigt med den nye database.

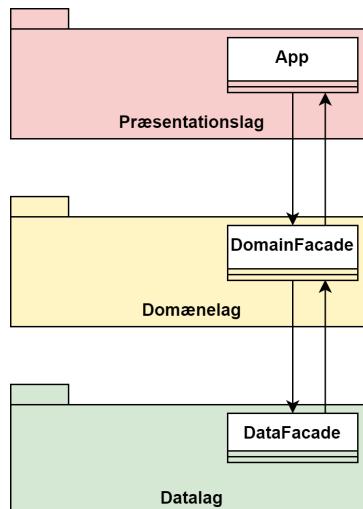


Figur 6.3: Systemsekvensdiagram for tilføjelse af programmer.

Et eksempel på sammenkoblingen af lagene ses i figur 6.3, som er et eksempel på tilføjelse af udsendelse til databasen. Der ses at kontrolpanelet/brugergrænsefladen for tilføjelse åbnes, og administratorer såvel som brugere kan her, tilføje programmer. Når en tilføjelse sker, ses det hvordan det starter en bølge ned igennem lagene, og der endelig returnes data om det tilføjede program for visning i brugergrænsefladen.

6.2 Pakkediagram

På figur 6.4 ses et pakkediagram, som beskriver snitfladen mellem de forskellige lag. Der ses hvordan der kun kommunikeres imellem lag, som er direkte naboer, og hvilke klasser (*App*, *DomainFacade*, *DataFacade*) som benyttes til kommunikationen.



Figur 6.4: Et pakkediagram, som viser sammenhængen imellem de forskellige pakker i programmet.

Implementation

7

Implementationen blev udført i to faser. I den første fase blev der fokuseret på at bygge skelettet for programmet. Der blev opsat en trelags-arkitektur, og oprettet alle de nødvendige elementer. Brugerfladen blev konstrueret, og så vidt muligt blev logikken nødvendig for loginsystemet, såvel som oprettelse af notifikationer, programmer, kreditering og visning af disse lavet. Der var dog i denne fase brug af tekstdokumenter til lagring af informationen. I den anden fase var det største fokus at implementere en relationel database til at erstatte tekstfilerne. Det er også her den endelige kobling imellem lagene blev færdiggjort, så al funktionalitet var på plads.

7.1 Fladfil System

Som tidligere nævnt blev der i den 1. iteration oprettet et fladfil system til at håndtere det data som skulle lagres. Dette system skulle indeholde 4 forskellige slags data: udsendelser, krediteringer, kontoe og notifikationer. Notifikationer og kontooplysninger ligger i hver deres mappe, med én *.txt* fil, som indeholder al data om dem, hver især. Udsendelser og krediteringer er lagret så hver udsendelse har sin egen *.txt* fil, og denne fil indeholder både information om udsendelsen og krediteringen for den. En tekstfil læses linje for linje i programmet, og på hver linje separeres information med et kolon (:).

Titel : Beskrivelse : Lanceringsår : Kontonummer : Movie
--

Tabel 7.1: Strukturen af information om en film i fladfil systemet.

Tekstfilerne for udsendelser er navngivet efter titlen på udsendelsen, og den første linje i dokumentet er altid dedikeret til information om udsendelsen. Hvis det er en film vil denne linje tage form efter strukturen i tabel 7.1, eller hvis det er en episode, tabel 7.2. Et liveshow vil se ud ligesom film, men med en tilføjet lokation. Den sidst værdi i linjen vil altid være en af (*Movie, Liveshow, Episode*), og er et hjælpemiddel til at differentiere dem i programmet.

Titel	: Beskrivelse	: Lanceringsår	: Serienavn	: Sæson	: Episodenummer	: Kontonummer	: Episode
-------	---------------	----------------	-------------	---------	-----------------	---------------	-----------

Tabel 7.2: Strukturen af information om en episode i fladfil systemet.

Efter den første linje kommer al krediteringen, som er bundet til udsendelsen. Hver kreditering har sin egen linje, og tager formen som er set i tabel 7.3.

Fornavn	:	Efternavn	:	Rolle
---------	---	-----------	---	-------

Tabel 7.3: Strukturen af kreditering i fladfil systemet.

Som tidligere nævnt bliver al brugerdata lagret i én fil. I denne fil er hver linje repræsenterende for én bruger, strukturen for denne ses i tabel 7.4. Der er på dette tidspunkt ikke tænkt på at gøre forskel på administrator og almene brugere, uover at reservere ID = 0. Værdien *aktivert* skulle være sigende for om brugeren har rettigheder til at ændre i systemet.

ID	:	E-mail	:	Kodeord	:	Fornavn	:	Efternavn	:	Aktiveret
----	---	--------	---	---------	---	---------	---	-----------	---	-----------

Tabel 7.4: Strukturen af brugere i fladfil systemet.

Notifikationer er ligeledes lagret i én fil, og følger samme regler som brugerne. Strukturen for disse kan ses i tabel 7.5. Værdien *set* er en boolsk værdi som beskriver om notifikationen allerede er set af administratoren. *Ændring* er en tekstbeskrivelse af hvad og hvordan noget er blevet ændret.

Set	:	Dato	:	Kontonummer	:	Ændring
-----	---	------	---	-------------	---	---------

Tabel 7.5: Strukturen af notifikationer i fladfil systemet.

7.2 Brugere

Som udgangspunkt er der i programmet kun én administrator konto. Der kan ikke oprettes flere igennem programmet, dette skal gøres direkte på databasen. Administratører af systemet kan dog oprette almene brugere. Disse almene brugere har adgang til at oprette og se deres egne programmer, såvel som at tilføje og fjerne kreditering fra dem. I listing 7.1 ses hvordan og hvilken information, der hentes fra databasen når der skal tjekkes om en bruger eksisterer. Hvis denne query returnerer en værdi findes brugeren,

og der opnås adgang til dennes *account_id* som benyttes i mange sammenhænge for at vide hvem der har gjort hvad.

```
1 SELECT account_id, email, password, first_name, last_name FROM accounts
2 WHERE email = ?
3 AND password = ?;
```

Listing 7.1: SQL Query som tjekker om der findes en bruger med de informationer, der forsøges at logge ind med.

7.3 Programmer

Det er muligt for administratorer såvel som almene brugere at oprette udsendelser i systemet. Afhængigt af hvilken type udsendelse, som ønskes oprettet, vil forskellig information være påkrævet, som set på figur 7.1. Der er på denne side også en tabel til at vise brugerens oprettede udsendelser, mens administratorer kan se alle udsendelser.

The figure displays three separate data entry forms, each representing a different type of broadcast:

- Film:** Fields include Titel (Title), Type (Type) set to FILM, Beskrivelse (Description), and Lanceringsår (Release Year).
- Liveshow:** Fields include Titel (Title), Type (Type) set to LIVE, Beskrivelse (Description), Lanceringsår (Release Year), and Lokation (Location).
- Serie:** Fields include Titel (Title), Type (Type) set to SERIE, Beskrivelse (Description), Lanceringsår (Release Year), Serienavn (Series Name), Episode, and Sæson (Season).

Figur 7.1: Illustrerer de datafelte, der kræves udfyldt afhængigt af hvilken type udsendelse der er valgt.

I listing 7.2, ses hvilke SQL-statements der bliver brugt til at opdatere et liveshow. For film og liveshows er dette relativt simpelt, mens for episoder er der en del ting, som skal tages højde for. Der skal tjekkes om den eventuelt nye sæson eller serie findes på forhånd, hvis de ikke gør skal de oprettes osv. Derfor er der sat begrænsninger på at typen af udsendelser og serienavnet ikke kan ændres med rediger knappen. Hvis disse værdier ønskes ændret bliver brugeren bedt om at slette udsendelsen og oprette den på ny, med de rigtige værdier.

```
1 UPDATE broadcasts SET title = ?, bio = ?, launchyear = ?
2 WHERE broadcast_id = ?;
3
4 UPDATE liveshow SET location = ?
```

```
5 WHERE broadcast_id = ?;
```

Listing 7.2: SQL Query som opdaterer informationen for et liveshow.

7.4 Kreditering

I sektion 7.1 blev der nævnt at hver udsendelse har krediteringer knyttet til sig. Dette blev senere droppet til fordel for at knytte krediteringen til en hel serie, således at brugeren ikke skal indtaste alle de samme personer for hver episode i en lang serie.

I listing 7.3 ses det SQL-statement som bruges til at slette krediteringer. Der ses at der udelukkende slettes fra tabellen *credits* selvom der også er en anden tabel som binder krediteringer med udsendelser. Dette er pga. opbygningen i tabellerne, som automatisk vil slette alle linjer, som er afhængige af det givne *credit_id*. Denne måde at slette globalt formindsker muligheden for menneskelig fejl, og er derfor forsøgt implementeret hele vejen igennem programmet.

```
1 DELETE FROM credits WHERE credit_id = ?;
```

Listing 7.3: SQL Query som sletter en kreditering

7.5 Notifikationer

Når en administrator er logget ind vil der i toppen af programmet altid være en knap som leder til notifikationssiden. På figur 7.2 ses i toppen en knap med et flag som leder til siden. Hvis der er usete notifikationer vil en rød cirkel med antallet af notifikationer vises. Inde på notifikationssiden kan administratoren se hvilke notifikationer der er usete da de er markeret med fed skrift. Når en notifikation er trykket på, vil den være set, og skriften vil ændre sig.

Date	User	Change
21/05/2020	John Doe	Tilføjet kreditering for Birthe Lyngsæ til Badehotellet som Scenografi
19/05/2020	Jane Doe	Tilføjet kreditering for Morten Vang Simonsen til Rita som Birolle
19/05/2020	Jane Doe	Tilføjet kreditering for Sara Hjort Ditlevsen til Rita som Birolle
19/05/2020	Jane Doe	Tilføjet kreditering for Carsten Bjørnlund til Rita som Birolle
19/05/2020	Jane Doe	Tilføjet kreditering for Lisa Bastrup til Rita som Birolle
19/05/2020	Jane Doe	Tilføjet kreditering for Nikolaj Groth til Rita som Birolle

Figur 7.2: Udklip fra programmet, som viser et eksempel på en række notifikationer såvel som påmindelsen om at der er nye usete notifikationer.

7.6 Visning på TV

Visningen af kreditering er udtaenkt at vise kreditering for den nuværende udsendelse, der bliver sendt i TV'et. Af åbenlyse begrænsninger er et TV blevet emuleret med 5 billeder, som skifter hvert 5 sekund for at *simulere* en TV-kanal med skiftende programmer. Programmet henter så alle de udsendelser der ligger i databasen, og hver gang billedet skifter, vil krediteringen for en ny udsendelse vises. Dette er et forsøg på at vise hvordan krediteringen kan illustreres på udsendelser. Et eksempel på dette ses på figur 7.3.



Figur 7.3: Billede af hvordan krediteringen bliver vist over et stillbillede, som skal forestille et kørende program.

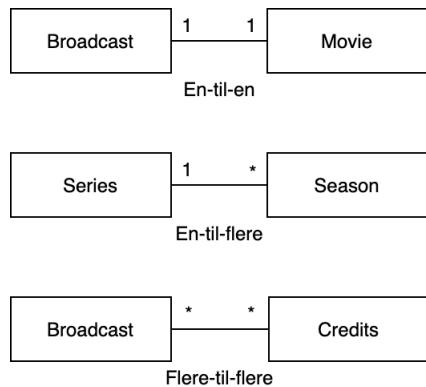
7.7 Database

Fremgangsmåden for at analysere, designe og implementere databasen har været at følge fremgangsmåden i Unified Process:

1. Analyse: ER, EER, UML (kortlægning af entiteter m.m.)
2. Design: UML (generering af tabeller)
3. Implementering: SQL (skabelse af script)

7.7.1 Kortlægning af entiteter og relationer (EER)

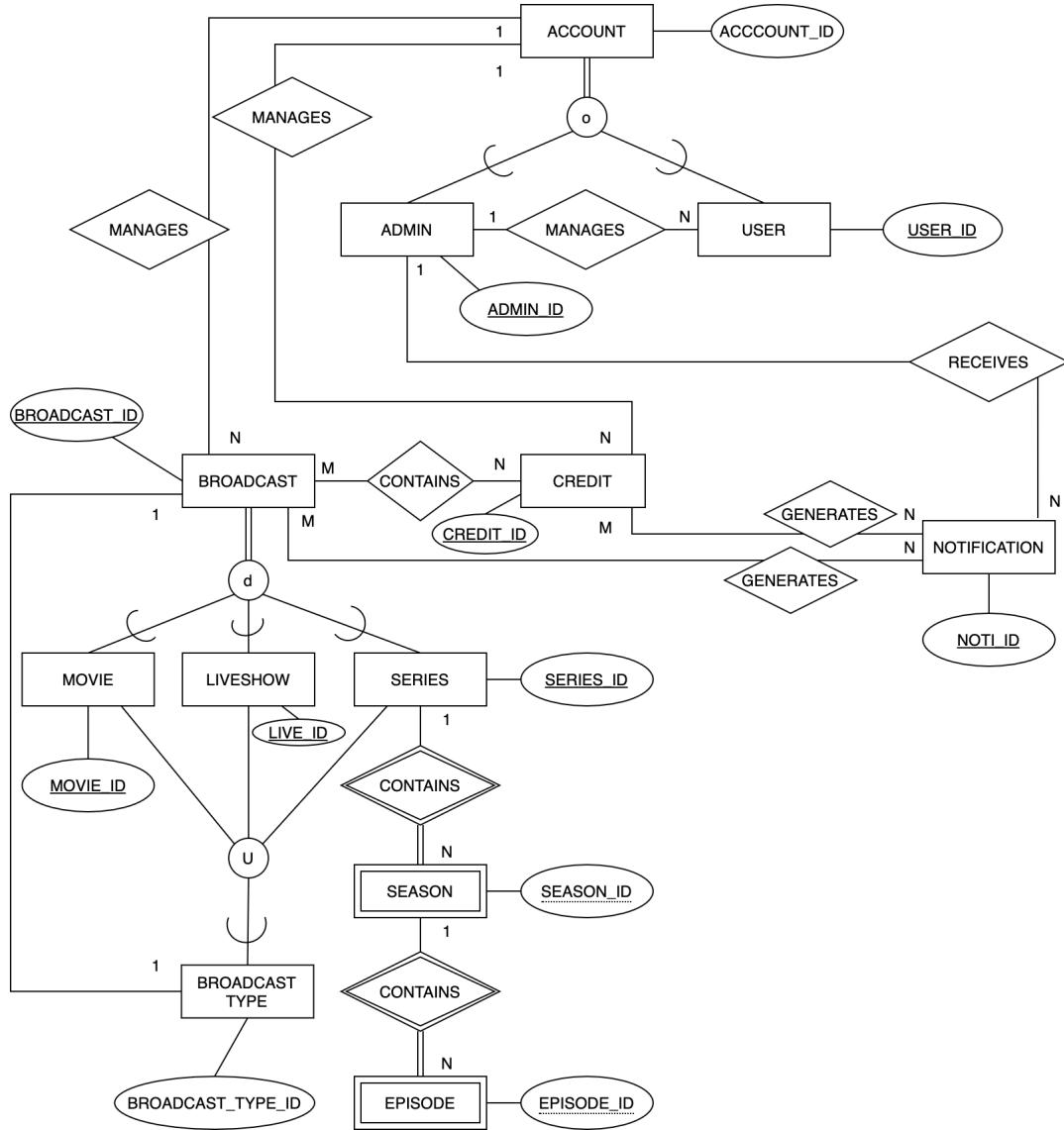
Første skridt i at få databasen normaliseret er at få afdækket entiteter i det kommende program. Fremgangsmåden har været at anvende verb/noun metoden på casebeskrivelsen fra TV2 for at få lokaliseret entiteter (noun) og deres indbyrdes relationer (verb) med eventuelle hierakier. Databasen til programmet omfatter de tre hovedtyper af relationer i en relationel database illustreret på figur 7.4. Relationerne varierer betragteligt i udformning, hvorfor de alle er medtaget i illustrationen her. I første scenarie, Broadcast-Movie, er det en en-til-en relation, hvor de to entiteter deler primary key, da de ikke individuelt kan eksistere. I Series-Season en-til-flere relation, men kun Series har en primary key, da Season er en svag entitet med en partial key. Broadcast-Credits relationen er en mange-til-mange relation, hvorfor det er nødvendigt med en såkaldt junction table imellem dem, hvor deres respektive primary key tilsammen udgør identifikationsnøglen. Alle kardinalitetsforhold er synlige på figur 7.5



Figur 7.4: Eksempler på kardinalitetsforhold mellem entiteter i programmet creDB.

EER indeholder alle modelleringskoncepter fra ER diagrammet. Kun EER er medtaget i rapporten her, da der opereres med arve-, kategoriserings- og aggregeringsmodelleringskoncepter. I programmet er disse koncepter relevante, fordi der gøres brug af generalisering/specialisering imellem entiteterne Account og Admin/User. Der er anvendt union

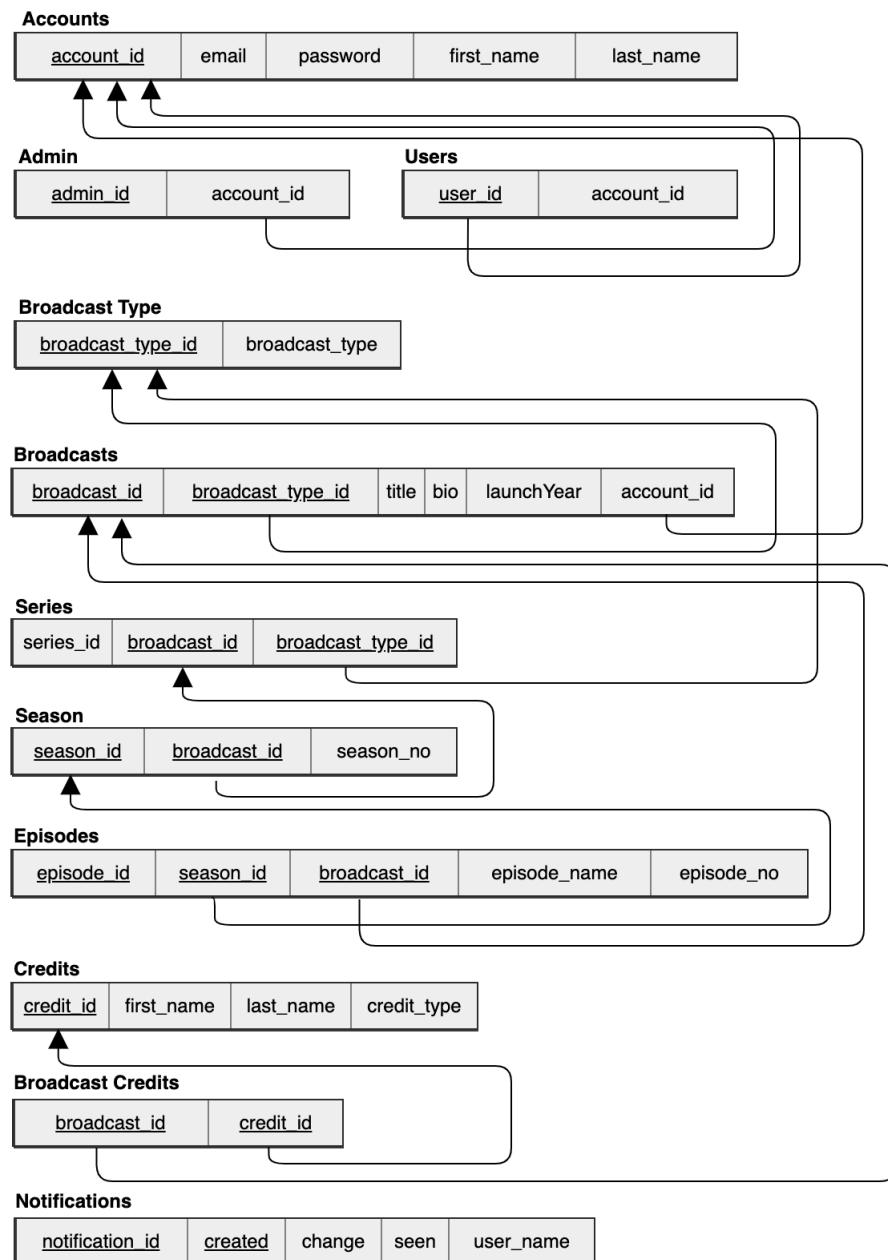
types (kategorisering) imellem Broadcast og hhv. Movie, Series og Liveshow. Regulære og svage entiteter kortlagt og hvorvidt relationerne er svage og total/partial. I det komplette EER diagram for creDB er kun væsentlige attributter medtaget.



Figur 7.5: EER diagram for creDB.

7.7.2 Generering af tabeller (UML)

Fra det komplette EER diagram kan tabeller genereres. Hver enkelt værdi i databasen skal som hovedregel kun lagres et sted for at undgå redundans. En normaliseret database hjælper med sikring af at operationer kører fejlfrit. Med EER diagrammet kan tabellerne konkretiseres og deres keys kan implementeres, så de passer til kardinaliteter og relationsstyrke. I kortlægningsdiagrammet herunder er gengivelser (Movies og Liveshow) ikke medtaget, da deres struktur er identisk med series.



Figur 7.6: Kortlægning af tabeller og deres relationer.

7.7.3 SQL - konstruktion af script

Udfra de genererede tabeller kan SQL koden konstrueres, så den kan håndtere de krævede operationer i programmet. der er kun medtaget de SQL-konstruktioner som volde største kvaler, at få implementeret. Hele scriptet kan findes i bilag 12.5.

Union types er en nævneværdig konstruktion, der sikrer at alle oprettede objekter af underklasser til broadcast (abstrakt) har en pre-defineret type. Konstruktionen sikrer at der ikke oprettes duplikater, da select statements skal have en værdi.

```
1 CREATE TABLE broadcast_types (
2     broadcast_type_id SERIAL PRIMARY KEY,
3     broadcast_type VARCHAR (10)
4 );
5
6 INSERT INTO broadcast_types
7 SELECT 1, 'Series' UNION ALL
8 SELECT 2, 'Movie' UNION ALL
9 SELECT 3, 'Liveshow'
```

Listing 7.4: Eksempel med Union type.

En speciel konstruktion, der blev investeret mange ressourcer i at få til at fungere korrekt er relationerne imellem broadcast (abstrakt objekt) og serier samt de afhængige og underliggende sæsoner og episoder. Det særlige her er at der eksisterer arv imellem broadcast og series, hvor broadcast skal være en type (i dette tilfælde series). Series har en svag entitet, sæsoner, knyttet til sig som igen har en svag entitet, episoder, tilknyttet. Altså kan objekter oprettes i alle lag. Slettes sæsoner, slettes alle underliggende episoder osv.

```
1 CREATE TABLE series (
2     series_id SERIAL UNIQUE NOT NULL,
3     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
4         CASCADE,
5     broadcast_type_id INT NOT NULL DEFAULT 1 CHECK (broadcast_type_id =
6         1), -- series
7     PRIMARY KEY (broadcast_id, broadcast_type_id)
8 );
9
10 CREATE TABLE seasons (
11     season_id SERIAL UNIQUE NOT NULL,
12     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
13         CASCADE,
14     season_no INT NOT NULL,
```

```
12     PRIMARY KEY (season_id, broadcast_id)
13 );
14
15 CREATE TABLE episodes (
16     episode_id SERIAL NOT NULL,
17     season_id INT REFERENCES seasons (season_id),
18     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
19         CASCADE,
20     episode_name VARCHAR (50) NOT NULL,
21     episode_no INT NOT NULL,
22     PRIMARY KEY (episode_id, season_id, broadcast_id)
23 );
```

Listing 7.5: Dobbelt koblet svag entitet.

Test

8

Programmet er løbende blevet testet på forskellige måder; der er udført unit-tests af enkelte klasser, så det sikres de virker individuelt. Derudover er der udført test for programmet som helhed i begge iterationer, for at finde både bugs og håndteringsproblemer.

8.0.1 1. Iteration

I 1. Iteration lå meget af fokusen på at få GUI op at køre, både login- og viewerdelen. Der blev lagt megen arbejde i at sikre koden var trelags delt fra start, og som følge blev en del integrationstest foretaget løbende. Derudover blev funktionaliteten i brugerdelen testet, således at der kunne lagres data i flad-filssystemet. Dette blev fremvist via Microsoft Teams konferencen - Demosession, hvor input i form af ros og ris blevet taget med videre til udviklingen af 2. Iteration.

8.0.2 2. Iteration

I 2. Iteration skulle flad-filssystemet skiftes ud med en relationel database. Der blev kørt mange unit-tests i databasen, i takt med at den blev oprettet. Dertil blev der kørt system tests, da koden og databasen blev samlet. Systemet er blevet testet igennem og rettet til, således det ikke kan crashe, hvis der skulle ske noget uventet eller forkert indtastning.

Systemet er kun testet igennem af gruppen internt, da det ikke virkede nødvendigt at bruge emperisk test til et system som det udviklede. Der er en forventning om, at skulle der videreudvikles på dette, eller skulle det tages i brug som det er, så ville der blive udbudt oplæring uanset, og derfor er intuitions-faktoren ikke en kæmpe faktor for et godt system.

Diskussion

9

I dette afsnit vil de forskellige beslutninger og valg fra projektet blive diskuteret. Der vil ligeledes blive diskuteret, hvis der er ting, der ikke er gået som planlagt. Overordnet set er creDB et system, der delvist løser TV2s problem. Der er mulighed for at producere kan skrive alle ønskede medvirkende fra en produktion ind i en krediteringsliste, og det er muligt for TV2 at fjerne rulleteksterne fra sendefloden.

Det udviklede system er lavet ud fra de valgte krav, stillet af TV2, uden at der har været fuld fokus på at få færdigimplementeret disse. Projektet fungerer i sin helhed som *proof of concept* for, at der faktisk kan udvikles et krediteringssystem som det ønskede.

/En alternativ løsning til kontooprettelse kunne være at man som bruger selv oprettede en konto, hvorefter man kunne opnå administratorgodkendelse og tildeles de korrekte rettigheder. Dette kunne dog også resultere i problemer med falske og uønskede konti. Det kan diskuteres hvorvidt det er optimalt med den nuværende løsning, hvor en administrator bestemmer en brugers password ved oprettelse, men det virkede for gruppen som den mest åbentlyse løsning for at undgå de nævnte problemer. Da dette er en demo version for at vise *proof of concept*, blev brugerens sikkerhed ikke vurderet over en visuel løsning. En anden ting som kunne have været løst anderledes, er kreditering specifikt af episoder. I den nuværende version af programmet hører al kreditering til serien og ikke de enkelte episoder. Alternativet til den nuværende løsning er at have dem til at tilhøre unikt til de enkelte episoder. Den bedste løsning findes nok et sted imellem de to. Kreditering af hoved og biroller, samt aktøre bag kameraet, som er ansvarlige for hele serien kan være krediteret på serien. De folk som er gæsteoptrædener, eller kun er hjælpende i få eller måske endda enkelte episoder kan så være krediterede på episoder. I nogle tilfælde kunne det måske også give mening at have kreditering knyttet på sæsoner.

I databasens tabeller indeholder *notifications*-tabellen en værdi, som er en sammenkobling af brugernes fornavn og efternavn; denne kobling er lavet i Java-programmet. Koblingen imellem tabellerne kunne være tydeligere beskrevet, hvis der i stedet havde været *account_id* på denne værdis plads. Når navnet så ville hentes kunne man query databasen for dette ID.

På nuværende tidspunkt vises der maksimalt de 20 nyeste notifikationer. Dette er grundet at formateringen, som skal vise om en notifikation er læst eller ej, er langsom til at opdatere, når der rulles i siden. Det er uvist præcist hvad problemet er, men det antages at være en kombination af at tabellen kun indhenter data om de linjer, som er på skærmen, og at databasen ligger i Virginia, USA hvilket gør forbindelsen langsom.

9.1 Perspektivering

9.1.1 Brugbarhed

CreDB er en løsning til TV2's case som bl.a indeholder krediteringer. TV2 har mulighed for at sælge det til andre TV stationer såsom DR. Systemet indeholder de essentielle elementer som er påkraævet, for at en producent kan tilføje krediteringer, samt metoder til at hente den data til visning. De grafiske elementer kan tilpasses til de fleste TV programmer og på den baggrund har TV stationer også mulighed for benytte den del af creDB systemet.

9.1.2 Næste skridt

Fremtiden for systemet kunne inkludere de tidligere overvejelser i processen som blev nedprioriteret af hensyn til at aflevere prototypen til tiden. Disse inkluderer, men er ikke begrænset til at:

Tilpasse brugergrænseflade designet til at passe med TV2's tema. Den nuværende version af creDB's brugergrænseflade, består af grålige og sorte farver. Her kan TV2's typiske design implementeres, som typisk bærerpræg af rød og hvid.

Gøre personer unikke, på database niveau, så det er muligt at se hvilke udsendelser, der er knyttet til en bestemt person. Dette vil reducere mængden af data, der skal indtastes for personer som er oprettet i systemet samt gøre det nemmere, at tilføje yderligere funktionalitet til systemet. I sådan et system vil det give mening at have en separat side, hvor personer oprettes i systemet før de kan krediteres.

Dele indhold fra databasen til andre systemer, således brancheorganisationer der udbetaler royalties til medvirkende kan hente oplysninger direkte fra creDB. Disse vil have information omkring hvornår og hvor ofte en udsendelse er blevet vist. Det vil derved blive muligt at springe *Registrering Danmark* over i rækken af kommunikation. Dette vil spare ressourcer og potentielt penge på sigt.

Have notifikationer med funktionalitet. Dette kunne være praktisk til at bekræfte eller afvise ændringer i systemet. Dette kunne eksempelvis være en bruger som ønsker at slette en udsendelse, men det er ikke tilladt fra TV2s side. En løsning på dette kunne bestå af endnu en kolonne i *notifications*-tabellen i databasen, som skal indeholde det *broadcast_id* som ønskes slettet.

Implementere de resterende krav fra TV2. For at give et endeligt bud på en løsning vil det være nødvendigt at revurdere de resterende krav fra den givne case. Dette ville potentielt også omfatte alle ovenstående punkter.

Konklusion

10

TV2 kan ved at implementere en løsning med udgangspunkt i prototypen creDB, muligvis frigøre op til 30 sekunder efter hver udsendelse, der er allokeret til sluttrediteringer og givetvis hente den fortjeneste på op til 60 mio. kr. der er helt central i problemstillingen.

Projektforslaget er designet i to dele. I den ene del kan en TV2-seer tilgå krediteringer via det aktuelle program, der vises som et overlay på TV-skærmen, som det bl.a kendes fra tekst-TV. I den anden del kan en producer relateret til TV2, redigere krediteringsoplysninger på en given udsendelse via et særskilt program.

Begge dele af forslaget korresponderer med samme cloudbaserede database hvilket gør løsningen smidig ift. tilføjelse af ny funktionalitet og evt. tilkobling af andre tjenester. Visionen for løsningen har været ganske klar fra starten og det har resulteret i en robust løsning med en solid arkitektur, der i høj grad opfylder de overordnede og valgte underliggende krav.

Det er uklart om konceptet bag løsningen creDB lever op til lovkravene på området for kreditering, da der ikke er lignende eksempler tilgængelige på markedet. Der er tradition for at kreditering figurerer på et værk indenfor film og TV-industrien, og det er i hvert fald opfyldt i løsningen.

Løsningen baserer sig på, at der kan allokeres en universel knap til krediteringsvisning på fjernbetjeningen. TV2s repræsentant kunne ikke svare entydigt på om det var muligt at anvende en knap til dette på møderne, men tilkendegav at det givetvis var muligt. Denne del af problemstillingen er ikke undersøgt til bunds, hvorfor det ikke er sikkert at fjernbetjeningsaktiviteten af krediteringer rent faktisk er teknisk mulig.

Får man afklaret spørgsmålene omkring teknik og jura ved hhv. fjernbetjening og lovkrav vil man fint kunne skabe en ny generation af systemet i en næste fase.

Procesevaluering

11

I dette afsnit reflekteres der over de forskellige faser i projektet, og hvordan gruppens arbejdsproces har haft indflydelse på dem. Derudover inddragelse af gruppens evne til samarbejde, og i hvilken grad det har påvirket processen.

11.1 Vejlederaftale

Et af de første punkter for gruppen var at udarbejde en vejlederaftale. Den etablerede fundamentet for den aftale, der blev indgået mellem gruppen og vejleder, Jeppe Schmidt. Vejlederaftalen kan ses i bilag [12.2](#)

Aftalen beskrev det ansvar, vejlederen forpligtede sig til i forbindelse med projektet, men specificerede samtidig de forventninger som vejleder havde til gruppen. Disse bestod bl.a. af at der skulle sendes en indkaldelse til møde, senest fredagen inden, hvori en dagsorden til mødet skulle vedhæftes.

Aftalen beskriver også hvornår det er forventet at vejlederen kommer til møderne, samt hvilke konsekvenser der vil være, hvis vejleder ikke møder op, og hvor lang tid han har til at svare på emails.

Vejlederaftalen har fungeret, som en forsikring for de studerende, i forhold til, hvad der kunne forventes i samarbejdet med vejleder. Denne er blevet overholdt af begge parter, hvilket har resulteret i et meget behageligt mødemiljø. Vejleder har været en solid sparringspartner igennem processen, så gruppemedlemmerne er blevet klogere på nogle af de svære opgaver, der indgår i projektarbejde.

11.2 Samarbejdsaftale

Samarbejdsaftalen fastlægger de generelle retningslinjer og rammer, som gruppens samarbejde skal fungere indenfor. Aftalen har de mest nødvendige værktøjer inkorporeret, samt en generel beskrivelse af forventningerne til projektet og hinanden. Samarbejdsaftalen

kan ses i bilag [12.2](#)

I samarbejdsaftalen er procedurer for mødepligt og lovligt/ikke lovligt fravær specifiseret. Derudover er også gruppens arbejdsstruktur under disse møder fastlagt i aftalen. Der kunne med fordel have været en opdatering af samarbejdsaftalen, da Scrum-rammeværktøjet blev implementeret da mødestrukturen ændrede sig.

Aftalen har været et godt grundlag for samarbejdet, og hjulpet med at skabe struktur, selvom det ikke har været nødvendigt at håndhæve samarbejdsaftalen pga. dårlig adfærd eller lignende. Det vurderes at aftalen har været et godt redskab i en ny gruppe for at forventningsafstemme, og danne grundlag for samarbejdet.

11.3 COVID-19

Der blev vendt op og ned på gruppens måde at arbejde på, midt i semestret, da SDU blev lukket ned, grundet SARS-CoV-2. Gruppen fik dog hurtigt styr på en måde at mødes på, da alle medlemmer allerede havde erfaring med online møder. Der blev oprettet en Discord server, hvor der kunne holdes møder, samt mindre rum, så gruppen kunne deles op i 2 eller 3 mands hold for at opnå mere effektivt arbejde. Efter nedlukningen har gruppen fortsat holdt arbejdssdag om tirsdagen, og ligeledes holdt vejledermøde. Sidst nævnte blev grundet SDU's interne politik rykket til Microsoft Teams, hvilket også fungerede rigtig fint. Der har været små problemer i form af fjernkontakten, men med de faste møde dage og skærmdeling er vi alle enige om, at coronakrisen ikke har påvirket arbejdet i de store træk.

Man kan måske næsten sige at COVID-19 situationen har styrket evnerne iblandt gruppe-medlemmerne til at løse komplekse udviklingsopgaver i fælleskab, hvilket har været en positiv erfaring.

11.4 Teamroller

Alle medlemmer i gruppen har taget en Belbintest. De enkelte resultater af testen er blevet sat ind i en Belbin gruppeprofil, som kan ses tabel [11.1](#).

Rolle	Aleksander	Anders	Christopher	Jakup	Mikail	Simon
 Kontaktskaber						
 Formidler						
 Koordinator						
 Idemand						
 Analysator						
 Specialist						
 Opstarter						
 Organisator						
 Afslutter						

Tabel 11.1: Alle medlemmer har to farvede celler, hvoraf de mørke er medlemmets primære rolle og de lysere er den sekundære.

Gruppens styrker:

Der er en god repræsentation i det sociale område og kan forventes at der er særlig fokus på at få samarbejdet til at virke, samt skabe en god stemning. Der er en stor samling af specialister så der bliver adgang til god faglig kompetence. Der er to opstartere som kan være med til at sørge for at projektet bliver drevet fremad. De to kontaktskabere vil kunne sørge for, der bliver kontaktet andre grupper, vejledere eller andre muligheder.

Svagheder:

Action er dårlig repræsenteret idet der ikke haves nogen organisator eller afslutter. Det betyder at gruppen kan risikere ikke at holde ordentligt styr på detaljerne, og går glip af formelle krav. Ligeledes haves en analysator ikke, hvilket sammenlagt med en overflod af specialister kan betyde at der kunne komme til at blive gået for meget i dybden på specifikke områder, og dermed glemme det store billede.

11.5 Inceptionsfasen

Vores inceptionsfase afviger lidt fra den originale, der handler mest om krav eliciting og udarbejdelse af brugsmønstre. Vi havde lidt for meget fokus på programmeringen fra starten og udkom meget hurtigt med CRC-kort og UML. Vi burde have haft bedre fokus på domænemodeller og udarbejdet brugsmønster detaljering og opsætning af krav.

Udarbejdelsen af inceptionsfasen var meget lærerig, ikke mindst som erfaring til fremtidige projekter, hvor alle i gruppen er bedre klar over, hvor fokus skal ligge, uanset personlig interesse.

11.6 Elaborationsfasen

I elaborationsfasen skulle vi udvide brugsmønstre, lave en række prototyper og beskrive softwarearkitekturen. Vi manglede noget skarphed i brugen af brugsmønstre og var nok for hurtige i planlægningen, i det der senere hen opstod småproblemer, der kunne have været undgået ved bedre planlægning. Brugsmønster prioriteringen vi havde udformet, burde have været fulgt mere nøje, så vi havde implementeret de vigtigste, først.

11.6.1 1. Iteration

Der blev udarbejdet et noget generisk UML diagram, der var halvt analyseklassediagram og halvt designdiagram. Til fremtidige projekter har vi nu lært, hvor stor en forskel det ville gøre, hvis der fra start blev udarbejdet bedre versioner af de respektive diagrammer. Da vi havde særligt fokus på kobling og samhørighed i vores trelags arkitektur, fik vi en rigtig god software struktur.

11.6.2 2. Iteration

Den generelle eksekvering af 2. Iteration gik meget bedre end første. Vi lavede tidligt EER-diagram, tabeller og diskuterede brugsmønstre og systemsekvensdiagrammer en sidste gang. Dette gav os et godt overblik over, både hvad der skulle laves og hvordan det ville ende med at se ud. Vi lavede rigtig mange tests, løbende, for hele tiden at sikre os at programmet kunne køre. Det var vi enige om var en god beslutning, eftersom det resulterede i positive oplevelser, da koden næsten altid virkede som forventet.

11.7 Brugen af metoder

De brugte metoder har overordnet set hjulpet gruppen i udarbejdelsen af processen, derudover er der blevet lært en masse om bl.a. Scrum, hvilket har været positivt. Dog hersker der lidt tvivl om, hvorvidt alle metoderne er blevet brugt som de oprindeligt var udviklet til.

11.7.1 Kanban

Brugen af Kanban via *Glo Boards* har været meget nyttig, når der skulle vælges nye opgaver, samt udpensles en forklaring af disse. Der blev oprettet 2 boards i starten af

projektet, ét til code tasks og ét til generelle project tasks. Der er løbende blevet tilføjet nye ting til de respektive boards, i takt med at vi i gruppen har fået nye idéer og er kommet længere i processen. Der blev diskuteret om, hvorvidt det havde været en fordel at lave alle arbejdsopgaver, inden der blev taget hul på dem, men da vores projekt har udviklet sig og taget en drejning op til flere gange, er vi alle tilfredse med måden det er blevet håndteret på. Mange gange er arbejdsopgaverne først blevet tilføjet til boardet når vi sad fast, og det kunne nok have forbedret processen, hvis vi havde været bedre til at definere dem på forhånd. Dette kan være grundet manglen af belbin rollen organisator, som kan være med til at holde styr på det organisatoriske.

11.7.2 Github

Github har været gruppens fælles mappe under hele projektet. Det har været en klar fordel at gøre brug af Github, da flere af gruppens medlemmer har kunne skrive i kildekoden på samme tid. Derved har vi kunne uddele flere opgaver på samme tid, og dele os op i mindre grupper. Dog er der tendens til misvisning i commits, da der har været arbejdet i grupper og ikke individuelt, som commits ellers viser. Dette er forsøgt løst ved at skrive de medlemmer der har arbejdet på de pågældende opgaver ind i både Kanban boardet og Scrum-backlogs. Det er altså her det tydeligst ses, hvem der har arbejdet på hvilke opgaver.

11.7.3 Belbin

For at overkomme gruppens svagheder, baseret på belbin rolle kompositionen er en række redskaber og metoder taget i brug. Der kan i gruppen være en tendens til ikke at holde styr på detaljer, og det organisatoriske. For at modarbejde dette er der sat fokus på brugen af kanban boardet, og scrum-backlog, i et forsøg på at klargøre ting som ellers ville blive overset. For at modarbejde specialisterne i gruppens tendens til at gå for meget i dybden med specifikke områder er Scrum møderne et fantastisk redskab. Det tvinger alle medlemmer til at opsummere hvad der bliver arbejdet på, og vi kunne i fællesskab holde hinanden i ørene og få folk på det rette spor.

11.7.4 Pairprogramming

Pairprogramming har været en rigtig stor hjælp til udviklingen af koden til projektet. Da der er forskel på programmerings niveau blandt medlemmerne i gruppen har det været raret at kunne sparre imens kodningen er foregået. Specielt da vi ikke har kunnet sidde sammen fysisk i løbet af det meste af kodeprocessen. Discord rummene har været brugt, så vi kunne dele os op i par-opkald, men også hurtigt har kunne komme i kontakt med de andre medlemmer, hvis vi stødte ind i problemer, eller havde spørgsmål til en del af en opgave.

Litteratur

- [1] IMB. Rational unified process. https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf, 2003. Accessed: 26-05-2020.
- [2] Y. Daniel Liang. *Introduction to Java Programming and Data Structures*. 11th edition, 2017.
- [3] Ramez Elmasri Shamkant B. Navathe. *Fundamentals of Database Systems*. 7th edition, 2017.
- [4] Roger Pressmann. *Software Engineering: A Practitioner's Approach*. 9th edition, 2020.
- [5] Scrum.org. The scrum guide™. <https://www.scrumguides.org/scrum-guide.html>, 2017. Accessed: 26-05-2020.

Bilag

12

12.1 Scrum

Link: <https://github.com/Kaszz/semesterprojekt2/tree/master/Files/Scrum>

12.2 Samarbejds og vejlederaftale

Link: <https://github.com/Kaszz/semesterprojekt2/tree/master/Files/Aftaler>

12.3 Logbog

Link: <https://github.com/Kaszz/semesterprojekt2>

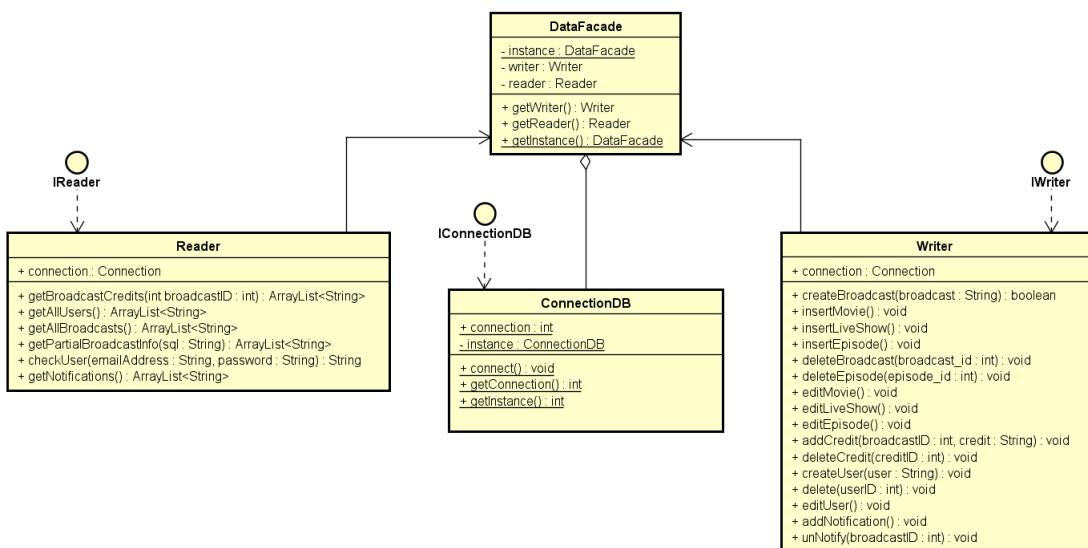
12.3.1 Øvrige værktøjer

Navn	Beskrivelse
LaTex	Tekst redigeringsystem anvendt til rapport
Overleaf	GUI klient til LaTex
Glo Kanban	Traditionelt kanban board til allokering af opgaver
Github	Software til versioneringsstyring
IntelliJ	IDE til bl.a Java
GitKraken	GUI klient til Github
Discord	Videotelefoni-, chat- og fildelingssoftware
Diagrams.net	Diagram redigeringsystem
Astah	Diagram redigeringsystem
Google Drive	Filhåndtering, tekst redigering og tabel redigering
PGAdmin4	GUI klient til PostgreSQL script redigering
Sublime Text	SQL redigering
AWS RDS	Online hosting af relationel database

Tabel 12.1: Tabel over de benyttede værktøjer, og en beskrivelse af disse.

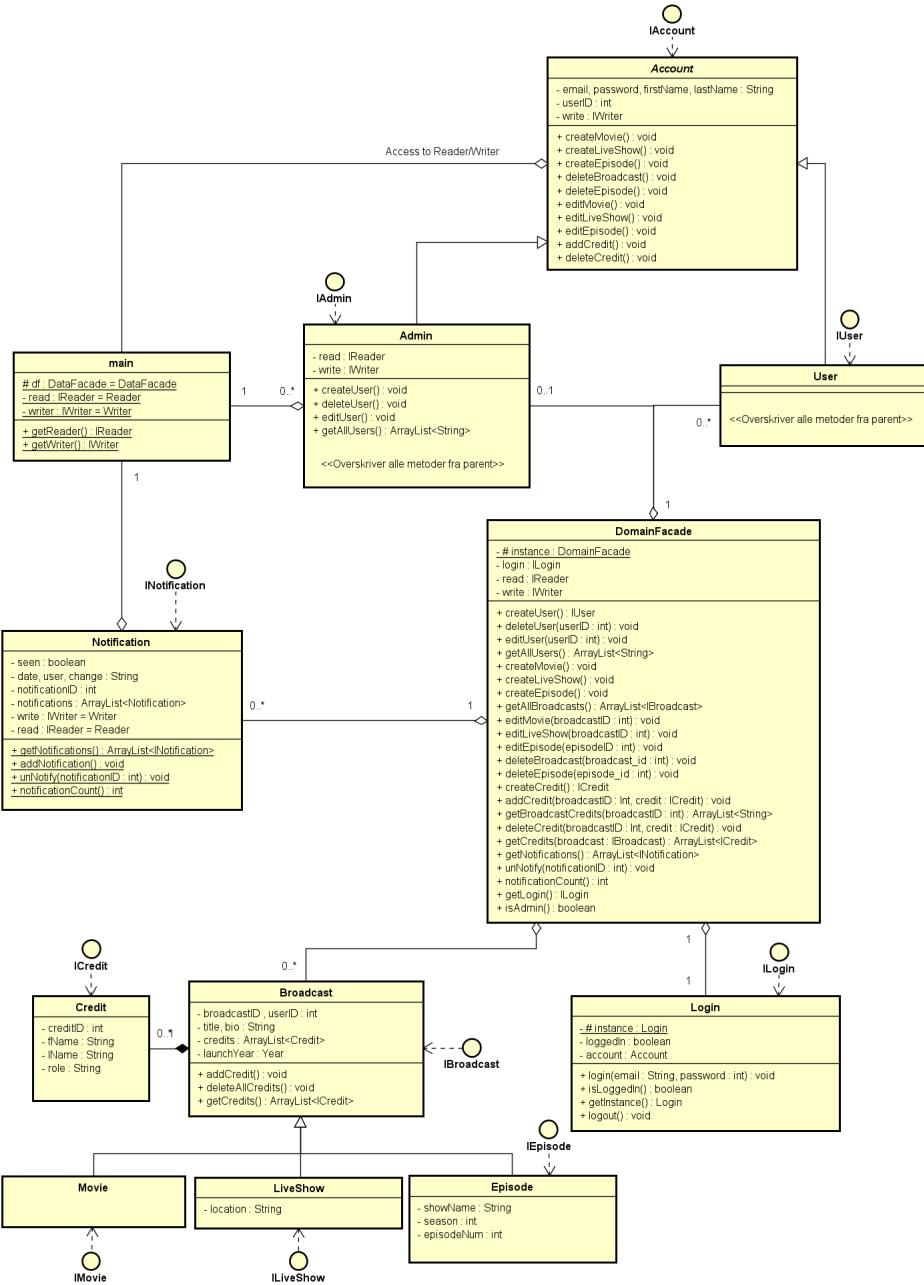
12.4 Design klassediagrammer

12.4.1 Datalag



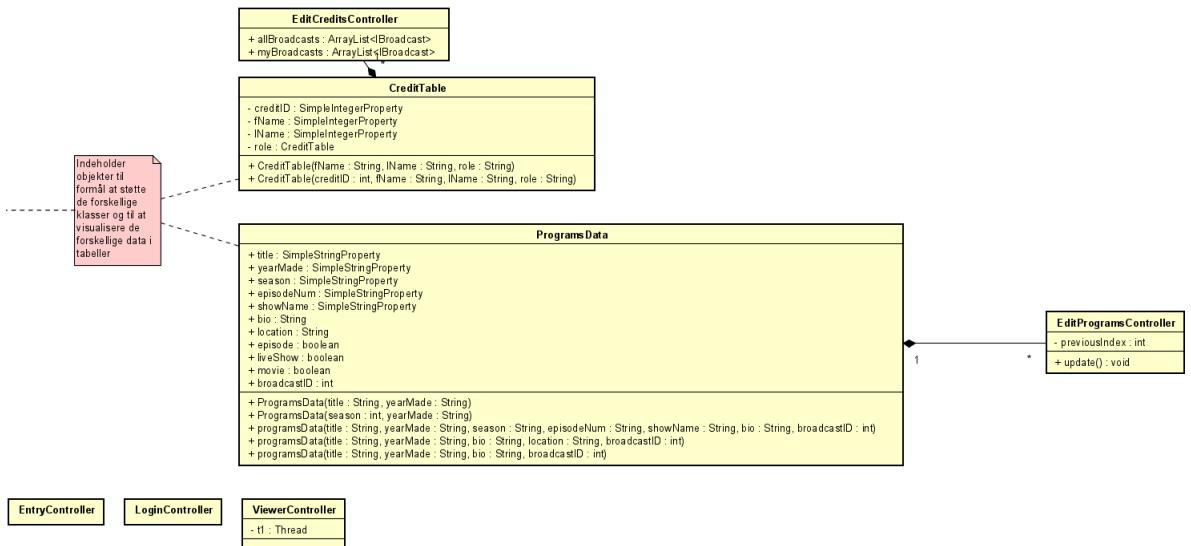
Figur 12.1: Design klassediagram over datalaget.

12.4.2 Domænelag

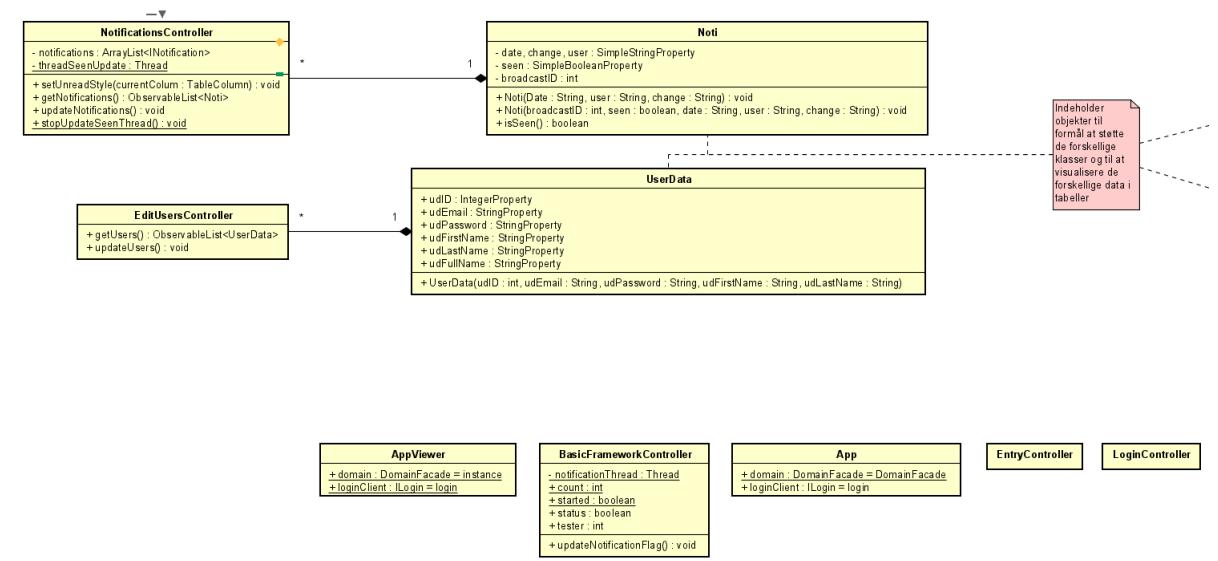


Figur 12.2: Design klassediagram over domænelaget.

12.4.3 Præsentationslag



Figur 12.3: Et uddrag af præsentationslaget i designfasen, resten af diagrammet ses i figur 12.4.



Figur 12.4: Et andet uddrag af præsentationslaget i designfasen, resten af diagrammet ses i figure 12.3.

12.5 SQL script

```
1 CREATE TABLE accounts (
2     account_id SERIAL PRIMARY KEY,
3     email VARCHAR (100) UNIQUE NOT NULL,
4     password VARCHAR (50) NOT NULL,
5     first_name VARCHAR(50) NOT NULL,
6     last_name VARCHAR(50) NOT NULL
7 );
8
9 CREATE TABLE admin (
10    admin_id SERIAL PRIMARY KEY,
11    account_id INT NOT NULL REFERENCES accounts (account_id)
12 );
13
14 CREATE TABLE users (
15    user_id SERIAL PRIMARY KEY,
16    account_id INT NOT NULL REFERENCES accounts (account_id)
17 );
18
19 CREATE TABLE broadcast_types (
20     broadcast_type_id SERIAL PRIMARY KEY,
21     broadcast_type VARCHAR (10)
22 );
23
24 INSERT INTO broadcast_types
25 SELECT 1, 'Series' UNION ALL
26 SELECT 2, 'Movie' UNION ALL
27 SELECT 3, 'Liveshow'
28
29 CREATE TABLE broadcasts (
30     broadcast_id SERIAL PRIMARY KEY,
31     broadcast_type_id INT REFERENCES broadcast_types (broadcast_type_id),
32     title VARCHAR (200) UNIQUE NOT NULL,
33     bio TEXT NOT NULL,
34     launchYear INT NOT NULL,
35     account_id int NOT NULL REFERENCES accounts,
36     CONSTRAINT broadcasts_AltPK UNIQUE (broadcast_id, broadcast_type_id)
37 );
38
```

```

39 CREATE TABLE series (
40     series_id SERIAL UNIQUE NOT NULL,
41     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
42         CASCADE,
43     broadcast_type_id INT NOT NULL DEFAULT 1 CHECK (broadcast_type_id =
44         1), -- series
45     PRIMARY KEY (broadcast_id, broadcast_type_id)
46 );
47
48 CREATE TABLE seasons (
49     season_id SERIAL UNIQUE NOT NULL,
50     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
51         CASCADE,
52     season_no INT NOT NULL,
53     PRIMARY KEY (season_id, broadcast_id)
54 );
55
56 CREATE TABLE episodes (
57     episode_id SERIAL NOT NULL,
58     season_id INT REFERENCES seasons (season_id),
59     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
60         CASCADE,
61     episode_name VARCHAR (50) NOT NULL,
62     episode_no INT NOT NULL,
63     PRIMARY KEY (episode_id, season_id, broadcast_id)
64 );
65
66 CREATE TABLE movies (
67     movie_id SERIAL UNIQUE NOT NULL,
68     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
69         CASCADE,
70     broadcast_type_id INT NOT NULL DEFAULT 2 CHECK (broadcast_type_id =
71         2), -- movies
72     PRIMARY KEY (broadcast_id, broadcast_type_id)
73 );
74
75 CREATE TABLE liveshow (
76     liveshow_id SERIAL UNIQUE NOT NULL,
77     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
78         CASCADE,

```

```

72     broadcast_type_id INT NOT NULL DEFAULT 3 CHECK (broadcast_type_id =
73         3), -- liveshow
74     location VARCHAR(100),
75     PRIMARY KEY (broadcast_id, broadcast_type_id)
76 );
77
78 CREATE TABLE credits (
79     credit_id SERIAL PRIMARY KEY,
80     first_name VARCHAR (50) NOT NULL,
81     last_name VARCHAR (50) NOT NULL,
82     credit_type VARCHAR(50) NOT NULL
83 );
84
85 CREATE TABLE broadcasts_credits (
86     broadcast_id INT REFERENCES broadcasts (broadcast_id) ON DELETE
87         CASCADE,
88     credit_id INT REFERENCES credits (credit_id) ON DELETE CASCADE,
89     PRIMARY KEY (broadcast_id, credit_id)
90 );
91
92 CREATE TABLE notifications (
93     notification_id SERIAL PRIMARY KEY,
94     created DATE NOT NULL DEFAULT CURRENT_TIMESTAMP,
95     change VARCHAR (500) NOT NULL,
96     seen boolean NOT NULL,
97     user_name VARCHAR (100) NOT NULL
98 );

```

12.6 Cyber-physical systemer

List of requirements draft (April 14th)

Introduction

Identification of *functional requirements* [FR] and *non-functional requirements* [NFR] starts with the requirements elicitation. The elicitation is defined by the practise of collecting the requirements of a system from a customer (TV2) stakeholders (users) and other stakeholders (eg. Registration Denmark).

Requirement elicitation methods typically contain: interviews, questionnaires, user observation, workshops, brainstorming, use cases and prototyping.

According to Kevin Adams¹ the FR is defined by the following essential characteristics:

1. Define *what* the system should do.
2. Be action oriented.
3. Describe tasks or activities.
4. Are associated with the transformation of inputs to outputs.

While a NFR is defined by the following essential characteristics:

1. Define a property or quality that the system should have.
2. Can be subjective, relative, and interacting.
3. Describe how well the systems must operate.
4. Are associated with the entire system.

ReQtest.com² lists some examples of categories related to FR and NFR. These categories are useful to analyse the case description from TV2. While this is not a complete list but a custom compressed one, it is representative of the major FR and NFR associated with the TV2 credit case.

Some typical functional requirement are:	Some typical non-functional requirements are:
Transaction corrections, adjustments and cancellations Administrative functions Authentication Authorization levels External Interfaces	Performance Scalability Availability Reliability Maintainability Security

¹ Non-functional Requirements in Systems Analysis and Design by Kevin Adams.

² <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

Legal or Regulatory Requirements	Regulatory Data Integrity Usability Optimization
----------------------------------	---

The requirements are derived from the SI2 Project Case provided by TV2³. On the left side all FRs are collected and NFRs on the right side. The title categories for each cluster of features are taken from the table above.

Functional Requirements	Non-functional Requirements
Administrative functions System Administrators should be able to maintain, ie. create, read, update, and delete persons and credits, as well as users in the system.	Availability Digitalization of credits. All credits should be visible for the viewers
Users need capability to add programs and associated credits.	Optimization Reduction or elimination of credits screen time during broadcasts.
Users should only be able to edit programs they own.	Usability Needs a graphical interface. It is very important that the system makes it easy for TV users to see the credits for the program they have just seen.
A notification system that will alert administrators each time something happens.	There should also be a publicly available part of the system, where it is possible to see the credits for a production without logging in. The system should be in danish.
External Interfaces TV2 should be able to export a specific amount of data in different various formats.	Security There should be some kind of access control for data such as edit, delete, add.
There should be an API that can be used to connect with internal as well as external publishers such as YouSee and Stofa.	

³ <https://docs.google.com/document/d/1p6lQjWV76TX67uTLst2OAdmV07XfMRn5fObelzBpd2EI/edit#>

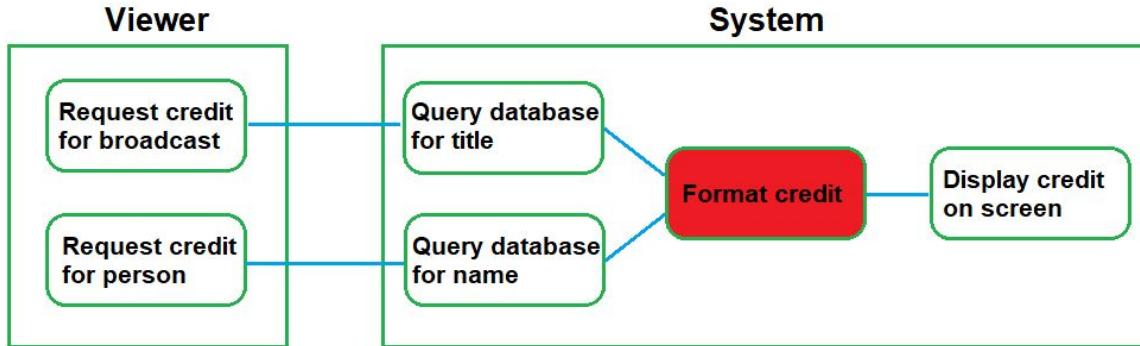
Functional Requirements	Non-functional Requirements
<p>It should be possible to import publications from the EPG system</p> <p>Transaction corrections, adjustments and cancellations</p> <p>It should be possible to link/merge persons who refer to the same person.</p> <p>Find a way to distinguish between people with the same name, so that the correct credits can be attributed to them.</p> <p>It should be possible to search the data directly so that it's possible to find persons, programs, series and credits.</p>	<p>Scalability</p> <p>Notifications should be sent to external partners such as "Samrådet for Ophavsret" and "Producentforeningen".</p>

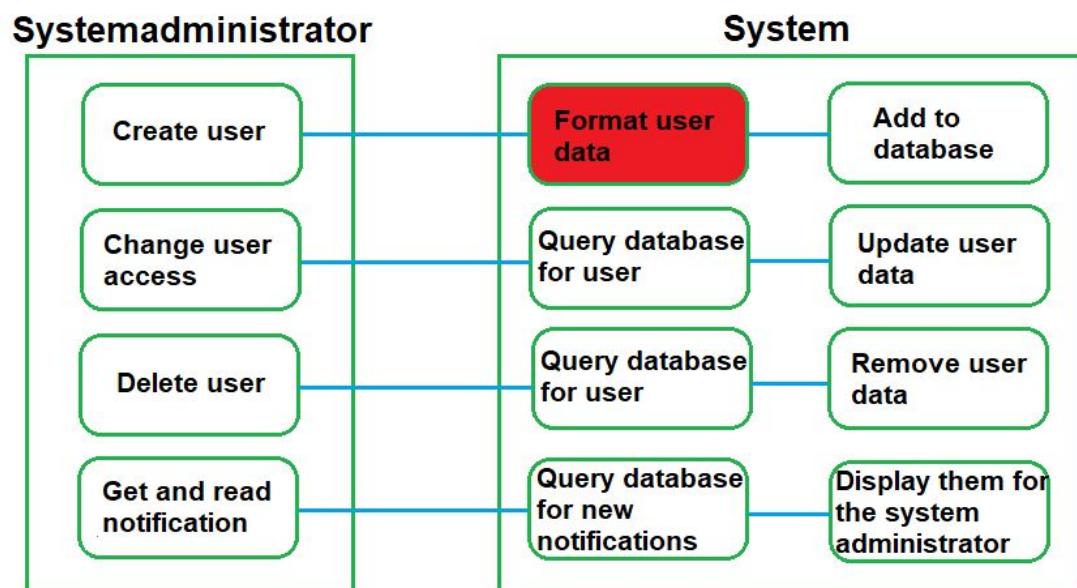
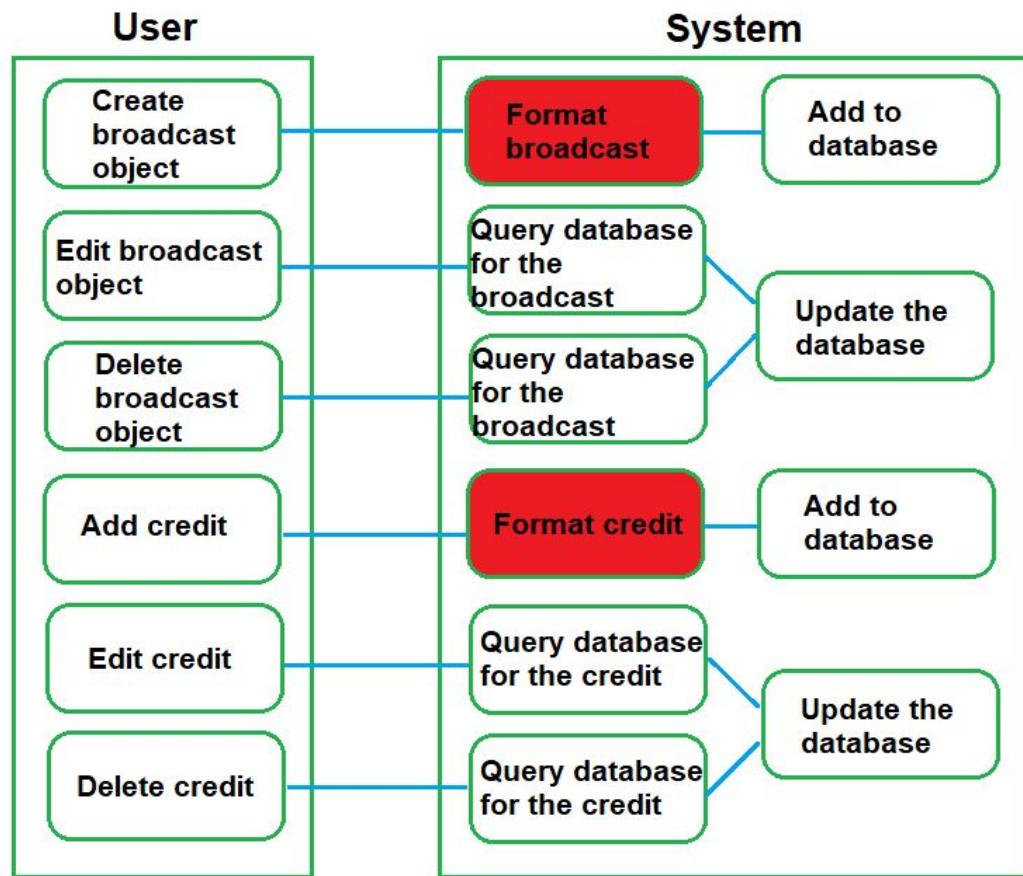
EAST-ADL

Since there is no actual physical component in the project description, the EAST-ADL models are very minimalistic. A model for both types of physical aspects - a viewer & a user - has been developed. Viewer and user are graphical user interfaces that can interact with the system.

After the meeting with TV2 we didn't feel the need to change our functional and non-functional requirements.

The models are in ranked order. Made so, administrators also have the access/capabilities of the users, and users of viewers.





12.7 Inceptionsdokument

Indledning

Ved afslutningen af film og TV-programmer er der tradition for at have krediteringer til skabere af værket f.eks. skuespillere og produktion. De vises ofte med tekst i varierende størrelser centreret på sort baggrund.

TV2 ønsker tekniske forslag til et nyt krediteringssystem [KS], der ultimativt kan betyde en besparelse/fortjeneste på 60 millioner DKK for medievirksomheden ved at frigøre op til 30 sek. pr. film eller udsendelse, der i dag anvendes til krediteringer. Med ønsket om forslag til løsning er der opstillet en række tekniske krav, som opgaveløser frit kan vælge i mellem.

Vi har valgt at navngive løsningen/produktet “creDB”, som er et portmanteau af *credits database*. Efterfølgende vil creDB derfor henvise til den tekniske løsning, der udarbejdes af gruppe 05.

Formål med inceptionsfasen

Formålet med inceptionsfasen er at danne et fast projektgrundlag for gruppen, og at få fastsat projektets rammer. Herunder bliver projektet afgrænset, samt udarbejdet en overordnet kravspecifikation (requirements outlined) - kravene hertil bliver prioriteret. De opgaver, der løses i inceptionsfasen, er en hjælp til at danne overblik over, hvad der skal arbejdes ud fra i elaborationsfasen.

Målene for inceptionsfasen opstillet i punktform:

- Få fastsat projektets rammer
- Udarbejde en overordnet kravspecifikation
- Danne projektets metoder i elebarationsfasen

Problemanalyse

Følgende områder af tekniske krav til KS fra TV2 er valgt på baggrund af interesse:

1. Registration of Credits (registreringer af krediteringer)
2. Notifications (notifikationer)
3. Access Control (specifikke brugertyper)
4. Using Data (søgefunktion)

KS skal indeholde mulighed for at tilføje, fjerne og redigere i bidragsydere til film og TV-programmer. For TV2 betyder denne funktionalitet at opgaven med indtastning kan frigøres fra selskabet og pålægges eksempelvis produktionsselskabet. Svagheden ved et KS, der ikke er synkroniseret med betalingssystem er at incitamentet for eksempelvis produktionsselskab til at indtaste krediteringsoplysninger er mangelfulde. KS vil derudover indeholde specifikke brugertyper (administrator, producer og bruger) med adgang til forskellig funktionalitet.

Der ligger en ressourcemæssig besparelse ved at f.eks. en producent kan tilføje og redigere krediteringer. En ulempe ved løsningen er at KS ikke konstrueres, så der på en smidig måde kan tilføjes yderligere brugertyper i fremtiden. Brugertyperne er ikke synkroniseret med TV2s øvrige IT-systemer, hvilket en bruger givetvis ville forvente i et endeligt produkt. Offentlig adgang til databasen via f.eks. producer-brugertypen øger sandsynligheden for utilsigtet adgang. Der implementeres en søgefunktion for at give en bruger intuitiv adgang til indhold i databasen og styrke brugervenligheden.

Søgefunktioner er ressourcetunge og kan betyde restriktioner i design af databasen. KS vil indeholde notifikationer, der gør opmærksom på tilføjelser og ændringer realtime til administrator. Det skaber overblik og er dermed også ressourcebesparende samt reducerer fejl og misbrug. Det er ydermere forventeligt at et system har denne funktionalitet. Der kan eksistere en forventning om cross-platform funktionalitet, så notifikationer kan modtages på flere tjenester og enheder f.eks. e-mail og smartphone.

Anvendelse af database betyder at flere systemer på sigt kan tilkobles og data fylder mindre samt minimerer redundans. Derudover reduceres fejl ved opdateringer og styrker konsistens på tværs af systemet. Ønsker TV2 at udvide med f.eks. mobil app kan samme database anvendes. U

Impen ved databaser er at de generelt er komplekse og kan være ressourcekrævende at designe. KS vil indeholde et grafisk brugergrænseflade [UI], der letter forståelsen af systemet for en ikke teknisk bruger.

Faglige Vidensgrundlag

Begrebsdefinitioner

Begreb	Definition
creDB	Er navnet på det krediteringssystem, som udvikles i dette projektforløb.
MoSCoW	MoSCoW er en prioriteringsmetode hvor man kan prioritere emner i 4 bestående ord: Must (have) Should (have) Could (have) Want (have)
SQL	SQL er programmeringssproget til relationelle databaser.
PostgreSQL	PostgreSQL er en database server som bruges til at holde databaser liggende.
Systemadministrator	Systemadministrator er rollen der har ansvar for TV2's system.
Bruger (producent)	Bruger er producenten der er ansvarshavende for de enkelte TV-programmer.
Gæst (TV seer)	Dette er en almen konsument af et TV-programmer som også kan tilgå informationer om krediteringer via systemet.
Java	Java er et programmeringssprog.
Gitkraken	Gitkraken er et Git GUI klient
Program	Her refereres til en TV-udsendelse (produktion)

Teori og Fagligt Vidensgrundlag

TV2, har gjort opmærksom på at de skal følge det samme regelsæt for kreditering som DR¹. Det udgår herfra at ophavsretsloven siger at ophavsmanden har krav på at blive krediteret, samt de personer, som har ophavsretligt, overenskomstmæssigt eller kontraktligt krav på det. For at kvalificere som ophavsmand, skal man have

¹ https://www.dr.dk/NR/rdonlyres/00221a7b/dpikscstjptklixndnigywgeuakhwpog/DR_kreditmanual_050810.pdf

bidraget med en selvstændig og kreativ indsats, som har haft afgørende betydning på programmet indhold og eller form.

Denne kreditering skal så vidt muligt implementeres i programmet med relevant billede og lyd. Det er dog ikke muligt at få alt kreditering med således, og derfor gælder disse regler om maksimal gyldig varighed for kreditering efter et program.

- 20 sekunder for programmer med en varighed under 60 min.
- 30 sekunder for programmer med en varighed over 60 min.
- 45 sekunder for større danske dramatik.

Det faglige vidensgrundlag for gennemførsel af projektet inkluderer derfor en generel viden omkring krediteringsregler. Det er ligeledes essentielt at have teknisk viden omkring implementering af et sådan system. Dette kunne inkludere kendskab til at programmeringssprog såsom Java, til udarbejdelse af selve applikationen. Kyndighed med databaser, og i den forbindelse et SQL såsom PostgreSQL. Desuden må forventes at disse vidensgrundlag skal kunne spille sammen for at danne en fuldendt løsning.

For at hjælpe denne process antages der nødvendig viden omkring brug og forståelse af UML til at modellere både systemets adfærd og struktur.

Relevante Eksisterende Løsninger

IMDb (Internet Movie Database) har en database med information omkring film, serier, streaming og indeholder ligeledes information om de medvirkende. Denne database er offentlig tilgængelig via internettet, hvor det er muligt at søge og se relationer mellem udsendelser og alle de medvirkende.

De har den største samling af data på dette område i verden.

Overordnet kravspecifikation

Kravspecifikation

creDB er et system, der holder styr på krediteringer til alskens TV-programmer for eksempelvis medievirksomheden TV2 samt deres interesser. I krediteringssystemet er det tanken at flere brugertyper har forskellige muligheder for at redigere i indholdet af krediteringer. Der vil i creDB være en overordnet *systemadministrator*, der kan tilføje *brugere* (producenter), der er ansvarlige for et eller flere TV-programmer. En *bruger* kan tilføje TV-programmer og redigere i krediteringslisten forbundet med det specifikke TV-program. *Systemadministratoren* har tilsvarende redigeringsmuligheder som en *bruger* i creDB. En krediteret bidragsyder kan ikke redigere i indholdet men kan eventuelt anmode producenten

(og systemadministratoren) om at ændre i indholdet, hvis det ønskes. Den sidste brugertype er i systemet *gæst* som alene kan tilgå krediteringerne, men har ikke mulighed for at ændre i indholdet. Overordnet opererer løsningen creDB altså med tre brugertyper i systemet:

Systemadministrator	dette er f.eks. ansvarshavende for systemet hos TV2. Rollen kan tilføje, ændre og fjerne brugere samt ændre i alle TV-programmer, der er oprettet af brugere i systemet.
Bruger (producent)	dette er f.eks. ansvarshavende for det enkelte TV-program. Rollen kan tilføje, ændre og fjerne bidragsydere (krediteringer) på et givent TV-program. Brugertypen kan alene redigere i TV-programmer som er knyttet til egen konto.
Gæst (TV seer)	dette er en almen konsument af et TV-program som også kan tilgå informationer om krediteringer via systemet. Denne brugertype har ikke nogle muligheder for at redigere i indholdet.

Aktørliste

ID	Aktør	Beskrivelse
A01	Systemadministrator	<p>Som systemadministrator skal man have følgende muligheder:</p> <ul style="list-style-type: none"> - Logge ind til systemet. - Adgang til systemets logside. - Håndtere brugere og deres adgangsrettigheder. - Håndtere og opdatere krediteringer af medvirkende personer af en specifik titel. - Administrere og overse ændringer i systemet.
A02	Bruger	<p>Som bruger skal man have følgende muligheder:</p> <ul style="list-style-type: none"> - Logge ind til systemet.

		<ul style="list-style-type: none"> - Opdatere og tilføje sine krediteringer samt tilføje medvirkende personer.
A03	Gæst	<p>Som gæst skal man have følgende muligheder:</p> <ul style="list-style-type: none"> - Se kreditering for et program. - Se alt kreditering en person er knyttet til.

Brugsmønsterliste

ID	Brugsmønster	Beskrivelse
B01	Se kreditering for program	Det skal være muligt at se krediteringen, der er knyttet til et program
B02	Se kreditering for person	Det skal være muligt at se krediteringen, der er knyttet til en person.
B03	Tilføj program	Systemadministratoren og brugere skal kunne tilføje nye programmer.
B04	Rediger program	Systemadministratoren og brugere skal kunne redigere programmer.
B05	Slet program	Systemadministratoren og brugere skal kunne slette programmer.
B06	Tilføj kreditering til program	Systemadministrator og brugere skal kunne tilføje kreditering til eksisterende programmer.
B07	Rediger kreditering til program	Systemadministrator og brugere skal kunne redigere i kreditering knyttet til et eksisterende program.
B08	Slet kreditering til program	Systemadministrator og brugere skal kunne slette kreditering knyttet til et eksisterende program.
B09	Login	Systemadministratoren og brugere skal

		kunne logge ind i systemet.
B10	Administrere ændringer	Systemadministratoren skal kunne godkende eller afvise ændringer foretaget af brugere.
B11	Opret bruger	Systemadministratoren skal kunne oprette nye brugere.
B12	Slet bruger	Systemadministratoren skal kunne slette brugere.
B13	Ændrer adgangskrav for bruger	Systemadministratoren skal kunne ændre i adgangskravene for de forskellige brugere.

Prioritering af brugsmønstre

Brugsmønstre	Navn	Nytte for bruger og virksomhed	Indflydelse på arkitektur	Risiko	Sum
B01	Se kreditering for program	5	5	5	15
B02	Se kreditering for person	5	4	3	12
B03	Tilføj program	5	5	5	15
B04	Rediger program	5	3	2	10
B05	Slet program	5	2	1	8
B06	Tilføj kreditering til program	5	3	1	9
B07	Rediger kreditering til program	5	3	1	9
B08	Slet kreditering til program	4	2	1	6
B09	Login	5	5	5	15
B10	Administrere ændringer	5	5	5	15
B11	Opret bruger	5	5	5	15

B12	Slet bruger	5	3	1	9
B13	Ændrer adgangskrav for bruger	3	4	1	8

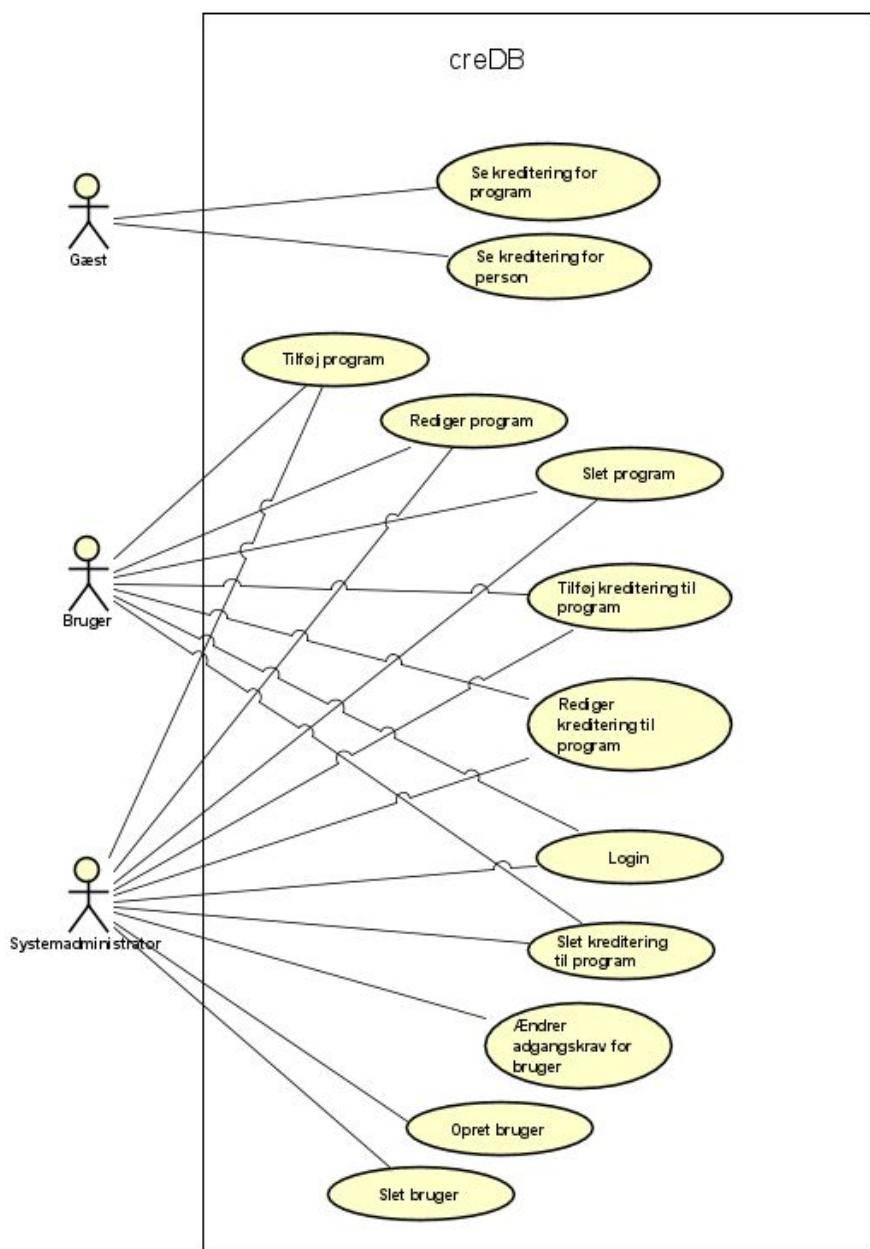
Ud fra det udarbejdede prioriteringsskema af brugsmønstre, kan man se de vigtigste brugsmønstre i projektet.

Herunder er de højest prioriterede brugsmønstre opstillet:

1. Se kreditering for program
2. Tilføj program
3. Login
4. Administrere ændringer
5. Opret bruger

Brugsmønsterdiagram

De tre brugertyper i systemet er repræsenteret på venstre side og de forskellige handlinger i gule ovale indramninger. Her illustreres det f.eks. at systemadministrator har samme redigeringsmuligheder som bruger.



Detaljeret brugsmønsterbeskrivelse

Brugsmønster: Tilføj program
ID: B03
Primære aktører: Bruger
Kort beskrivelse: Brugere af systemet kan oprette et program med de dertilhørende krediteringer. Programmet er knyttet specifikt til brugeren, der har oprettet programmet, så andre brugere ikke har mulighed for at redigere programmet.
Formål: Tilføje nye programmer er at give mulighed for at indsætte kreditering til programmer, således at krediteringer kan fjernes fra sendefladen.

Brugsmønster: Login
ID: B09
Primære aktører: Systemadministrator, Bruger
Kort beskrivelse: Alle brugertyper kan logge ind i systemet med email og password. Typen af bruger bestemmer adgangen til funktioner i systemet og hvilke programmer brugeren har adgang til at redigere indholdet i.
Formål: Systemet kræver login da funktionerne varierer ift. brugertype samt adgangen til redigeringsmuligheder på programmer.

Brugsmønster: Opret bruger
ID: B11
Primære aktører: Systemadministrator
Kort beskrivelse: Systemadministratoren af creDB har muligheden for at oprette en eller flere brugere i systemet. Brugeren oprettes med email og password. Brugeren kan eksempelvis være ansvarlig for et produktionsselskab, der håndterer krediteringer til et givent TV-programmer.
Formål: Formålet med at oprette en bruger vil være at mindske krediteringsarbejdet for creDB administrator, da produktionsselskaberne selv kan indtaste deres kreditering.

Tekniske krav (Fra TV2)

- Registration of Credits (registreringer af krediteringer)
- Notifications (notifikationer)
- Access Control (specifikke brugertyper)

Funktionelle krav

- Systemet skal sende en notifikation til administrator når brugere laver tilføjelser eller ændringer i systemet.
- Der skal være et login system for administratorer såvel som brugere.
- Brugere skal kunne oprette deres egne programmer, og knytte personer til dem.
- Administratorer skal have adgang til alt funktionalitet.
- Før en brugers ændring er endelig i systemet, skal den godkendes af en administrator.

Ikke funktionelle krav

- Alle krediteringsberettigede skal krediteres.
- Gæster i systemet skal have mulighed for at se krediteringen for alle programmer i systemet.
- Opslag i systemet skal altid returnere et korrekt resultat hurtigt og konsekvent.

Supplerende krav

De supplerende krav er blevet opstillet ved hjælp af FURPS model. Denne model er et hjælpeværktøj, til at sikre at et område af kravspecifikationen ikke bliver overset, og har en hierarkisk opbygning.

Krav kan altså tilhøre en af de fem forskellige niveauer vist i tabellen:

FURPS	Krav
Functionality	<ul style="list-style-type: none">- Skal kunne indeholde kreditering for alle de roller som er krediteringsberettigede.
Usability	<ul style="list-style-type: none">- Grænsefladen skal have høj kontrast mellem baggrund og tekst, så det er nemt læseligt.- Systemet skal være intuitivt at benytte både for brugere og gæster i systemet.
Reliability	<ul style="list-style-type: none">- Systemet må forventes at være oppe og tilgængeligt 24 timer i

	døgnet
Performance	<ul style="list-style-type: none"> - Opslag i systemet skal returnere et korrekt resultat på under 5 sekunder.
Supportability	<ul style="list-style-type: none"> - Krediteringen skal opbygges således at det kan indsættes i en relationel database.

Prioritering af krav

Der er gjort brug af MoSCoW metoden, til at prioritere gruppens opstillede krav. Prioriteringen kan ses på figur 2.



Figur 2

Kritiske risici

Risiko	Risikovurdering	Indvirkning	Håndtering	Kritisk?
Belbin	Middel	Middel	Gruppen er gennem projektforløbet fokuseret på de svagheder der er, i forhold til gruppens belbinroller, da dette kan medføre til et mindre godt projekt.	Ja
Tidsplan	Middel	Høj	Gruppen overholder tidsplanen nøje for ikke at overskride afleveringsfrister, dermed får gruppen også brugt tiden ligeligt på de forskellige opgaver i projektet.	Ja
COVID-19	Høj	Høj	Projektet bliver udarbejdet hjemme, hvor gruppen håndterer den indbyrdes kommunikation over programmet “Discord”. Hertil bliver vejledermøder også håndteret på Discord.	Ja
Sygt gruppemedlem (evt. COVID-19)	Lav	Middel	Et sygt gruppemedlem meddeler først og fremmest til gruppen at personen er syg. Hvis personen gentagende gange i træk melder sig syg bliver dette drøftet i gruppen.	Nej

Gruppemedlem melder sig ud	Lav	Middel	Tidsplanen og forventningen til projektet bliver revurderet for at omfordèle arbejdsbyrden på passende vis.	Nej
----------------------------	-----	--------	---	-----

Afgrænsning

Krediteringssystemet kan udvikles i mange retninger, og det er derfor nødvendigt at afgrænse omfanget, således at brugbarhed er opnåelig.

Projektet er begrænset til udelukkende at arbejde med krediteringssystemet i den forstand, at der ikke bliver taget højde for hvordan TV2 ønsker at erstatte kreditering på skærmen. Der bliver blot set på at udvikle et brugbart system, altså ikke deres business case.

Ligeledes bliver der afgrænset i forhold til de ønskede systemspecifikationer, som er fremstillet af TV2. Der bliver ikke arbejdet med områderne, som handler om at brugere i systemet skal være unikke og integration med andre systemer.

Inden for punkterne omkring brug af data vil det udelukkende være en søgefunktion, og for notifikationer ses bort fra muligheden for 3. parts samarbejde med "Samrådet for Ophavsret" og "Producentforeningen". Dette betyder altså at det ikke er et færdigt produkt, ud fra de krav TV2 har stillet.

Under udviklingen af den grafiske brugerflade vil der ikke blive taget højde for TV2's designmanual, da det færdige projekt blot vil være en prototype af det endelige system.

Metoder i elaborationsfasen

Unified Process [UP] er en procesmodel. Processen hænger sammen med Unified Modelling Language (UML), der er et universelt visuelt modelleringsprog. UP består af 4 faser: Inception, Elaboration, Construction og Transition.

I projektet anvendes UP, der strukturerer projektet så alle faser dækkes. Hermed kan man udføre projektet med en god afslutning, men da det er en prototype, som der skal laves vil projektet kun dække ind over inception og elaboration. Under inceptions fase indgår analyse og en smule design. Elaboration indgår analyse, design, implementation og test.

Scrum er en agil softwareudviklingsmetode som består af en række "sprints". Sprints er kortsigtet aktiviteter, der typisk varer to til fire uger og som ender ud i en inkremental prototype. Hver sprint har en backlog hvilket er en prioriteret liste af

artefakter som mangler implementering. På samme måde er der en product backlog hvilket er en prioriteret liste over krav og funktionerne som kunden ønsker implementeret.

Et Scrum hold består af tre til seks mennesker hvor en er “product owner”, en “Scrum master” og resten er en del af et “development team”.

Product owner godkender tilføjelser til backlogs, bestemmer om en sprint skal forlænges eller stoppes før tid og sørger for at holdet udfører kundens vigtigste backlog først.

Scrum master holder daglige møder, med en typisk varighed på 15 min, sammen med resten af holdet, løser problemstillinger opsat af development team og hjælper product owner med at udarbejde backlog elementer.

Development team er ansvarlige for udførelsen af en sprint og bestemmer hvornår en sprint er klar til at blive vist til product owner.

I vores projekt gennemgår vi to sprints: Første og anden iteration.

I første iteration laves kode rundt om databasen implementationen hvor principippet i en database vises gennem et fladfil system.

I anden iteration implementeres en database og den omkringliggende kode tilpasses.

Ressourcer

Til at danne overblik over hele projektets proces bruger vi et Gantt diagram. Den beskriver de faser vi kommer til at gennemgå.

[Gantt diagram²](#)

Tjeneste	Forklaring	Link
Gitkraken Glo	Administrering af ugentlige opgaver gøres gennem et Kanban board hvor vi bruger Gitkrakens glo. Her kan de enkelte medlemmer oprette et kort under enten “To do”, “In progress” eller “Done”. Alle kort har en	https://www.gitkraken.com/glo

² <https://drive.google.com/file/d/1N93m0NXMliWDESxPgIblCH9FTpqle3k8/view?usp=sharing>

	beskrivende titel, en eller flere ansvarlige for opgaven samt en deadline. Glo board opdateres løbende så alle kan se hvem der arbejder med hvad.	
Discord	Video konference værktøj med tekstchat og fildelingsfunktioner	https://discordapp.com/
Google Drive + underliggende tjenester	Online tekst redigeringsværktøj	https://www.google.com/drive/
Visual Paradigm Online	Anvendt til at lave en MoSCoW krav prioritering.	https://online.visual-paradigm.com/

Konklusion

Gennem projekt casen fik gruppen set på det pågældende problem, samt udarbejdet de rammer der skulle arbejdes indenfor. Kravene stillet fra TV2 og hvad de indebar, samt mulige risici blev identificeret. Der blev afgjort, hvilke aktører der havde hvilke funktioner, samt hvor relevante de var, de pågældende steder og hvilke brugsmønstre de fulgte, respektivt. Heraf kunne der opstilles et brugsmønster diagram, der visuelt viser systemet. Ved hjælp af MoSCoW blev det klart, hvilke brugsmønstre der var mest essentielle og kunne gå dybere til værk med disse.

Procesdokumenter

Vejlederaftale og samarbejdsaftale kan findes på gruppens git eller via dette link:

<https://github.com/Kaszz/semesterprojekt2/tree/master/Aftaler>

Belbin gruppeprofilen er at finde i samarbejdsaftalen.

12.8 TV2 Casebeskrivelse

Credits Management

A case by TV 2 Denmark A / S¹

Introduction

TV 2 is a Danish TV station that was established in 1988. TV 2's business is centered around making quality content for the Danish population. TV 2 generates its revenue by making its channels available in TV packages (through distributors like YouSee, Waoo, Stofa etc.), as well as showing commercials between programs.

Market

There's many ways to measure a TV-broadcaster's position in the market. One way is to look at how many minutes a typical dane watches in a given week. [Figure 1](#) shows the different broadcasters' "viewer minutes" (ie. how many minutes have the Danes on an average spend watching channels from the given broadcaster) in week 3, 2020. The first number from left is the average number of minutes viewed on a daily basis, whereas the next number is a total for the entire week. The last number, to the right in the diagram, is the market share based on the viewer minutes expressed in percentages. Looking at TV 2, we can see that the company has a market share in week 3 (2020) of 51,5%. This includes TV being watched on any channel in the TV 2 family. Compared to the other broadcasters, this is quite a lot and is partly due to some new agreements with distributors like Yousee, where the TV 2 channels get a better placement in their TV packages.

Broadcaster	Antal min. dagligt	Antal min. ugentligt	Seerandel %
TV 2	74	519	51,5
DR	45	314	31,1
NENT	12	83	8,2
Discovery	6	43	4,2
Fox	3	19	1,9
Viacom	1	9	0,9
Disney	1	6	0,6
Turner	0	1	0,1
Andre kanaler	3	18	1,8

Figure 1 Viewer minutes²

¹ The case is made in collaboration between TV 2 and SDU

² http://tvm.tns-gallup.dk/tvm/pm/2020/pm2003_Consolidated.htm

Technology

Technologies at TV 2 can be split into two main categories. First we have all the technology and equipment involved in the production of programmes. This includes (but is not limited to) cameras, microphones, teleprompters, robots, drones, etc.

The other part is technology supporting the rest of TV 2's ecosystem. This includes digital services like our subscription based video-on-demand streaming service TV 2 Play, but also all of our free services, like tv2.dk, all of our apps (News, Weather, TVID, etc.), tvid.dk and more. Behind these services is a lot of internal services that delivers data to other systems, that can then be presented to the end user.

For hosting, TV 2 uses a combination of on-prem and cloud solutions³. This gives us the best of both worlds, and enables us to choose whether a service should be hosted in the cloud or on-prem on an individual basis.

The main development language for the backend development at TV 2 is Java, but a lot of other programming languages (and frameworks) are used in the house. To name a few examples: C#, Python, HTML/CSS/JS, PHP, Swift and Kotlin. Regarding persistence there's multiple technologies being used as well, including Postgresql/SQL, no-SQL (like MongoDB and DynamoDB) and GraphDB.

Digital Transformation

Like most other companies, TV 2 is undergoing a digital transformation. Because of that, we are actively looking for ways in which we can optimize our workflows as well as exploring the potential possibilities we have for creating more value, both for the organisation as a whole and to the end customer.

TV 2 has also digitized other services such as our video streaming platform, TV 2 PLAY. We also operate a website with news, sport, weather forecasts, a TV guide and more that can be found at www.tv2.dk.

The vast majority of our digital solutions are developed and maintained in-house, and right now we have about 150 employees working with software on a daily basis.

³ https://en.wikipedia.org/wiki/On-premises_software

Problem

When a program has been broadcasted on a TV station, its credits must be shown. Today, this is done by showing the credits on screen at the end of each program, limited to a maximum duration of 30 seconds. In practical terms, this means that there is not always time to show all the credits, and therefore it is imperative that the credits are prioritized before they are shown on TV. This prioritization is done solely because there is not enough time to show all the credits on screen for each program.

Freeing up these 30 seconds at the end of every program, would open up opportunities for the Danish TV stations to use this time to show something else (for example an additional commercial). If TV 2 gets the opportunity to digitize the credits and thus spend these 30 seconds to show advertisements instead, TV 2 will be able to increase it's yearly revenue by up to 60 million Danish kroner. At the same time, all credits will become visible as they no longer need to be prioritized for a duration of 30 seconds.

To meet these ends TV 2 needs a crediting system, a credits management system, that can handle credits for TV content produced in Denmark. This includes the possibility to enter new credits into the system when a new production has been made, as well as being able to search for a given production (programme/series) and get a list of credits tied to that production. It should be possible to see which role a given person has had on a production, as one person can have different roles on different productions.

System Context

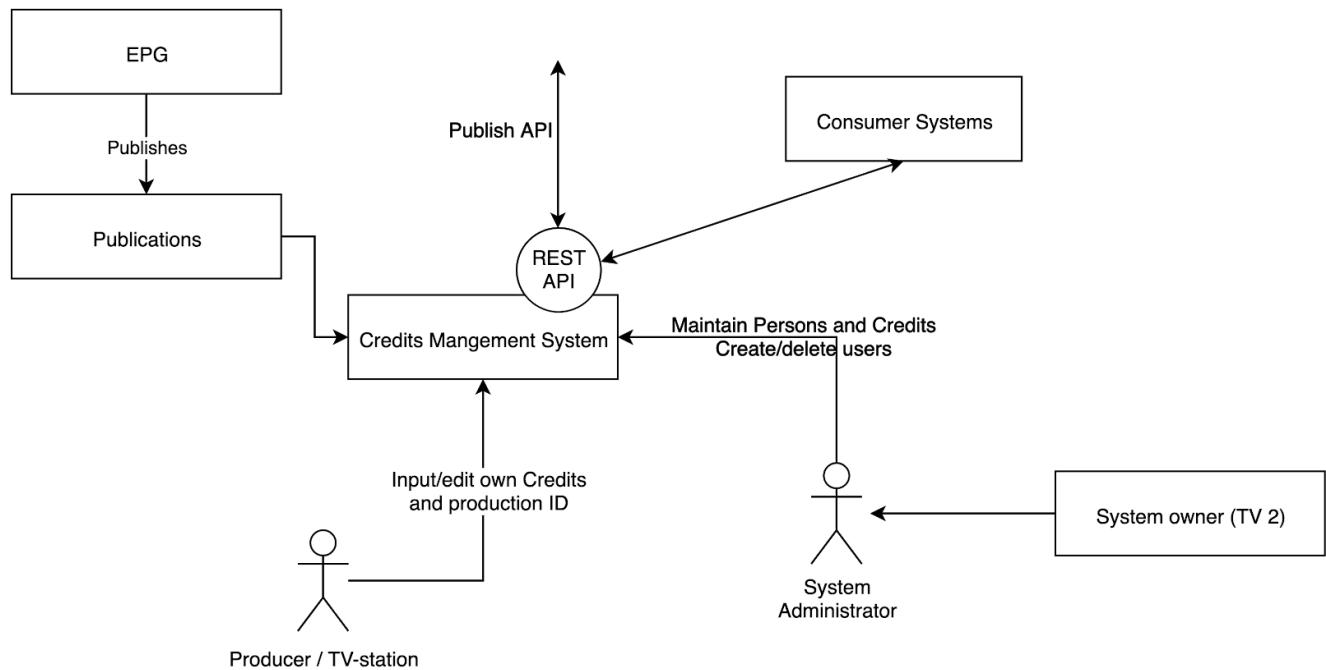


Figure 2 The context of the Credits Management System⁴

The Credits Management System's context can be seen in [Figure 2](#) above. The Credits Management System should receive Publications from the Electronic Program Guide system (EPG).⁵ These Publications contains data about the programs broadcast by all TV stations in Denmark.

Producers/TV-stations should be able to input and edit credits for the programs/productions that they own. They should also be able to edit the production IDs for these program/productions. System Administrators should be able to maintain, ie. create, read, update, and delete persons and credits, as well as users in the system.

Finally the Credits Management System should publish a service for other systems to consume. The consumer systems can for example be a web portal or an app. But other systems should also be able to consume the API, so the data can be used in other existing systems, such as TVTID.dk (TV 2's TV-Guide).

⁴ Integration with other systems could be by exposing data via a REST API as shown in figure 2, but a REST API is outside the scope of the project

⁵ Electronic Program Guide (EPG) is a system that contains data about the programs broadcast by all TV stations in Denmark. This data includes broadcast times and the channels where the programs are broadcast. The EPG system sends this data to TV providers such as Yousee.

Crediting rules

Detailed information about crediting rules may be obtained from TV 2 on the scheduled meetings.⁶

The System's needs

In connection with the realization of this system, TV 2 would like to have shed some light on how to design a comprehensive solution for digitizing the credits. This should be done by developing proposals for solutions to one or more of the system's needs, which must form the entirety of the system's full functionality. The proposal should include requirements, analysis, design, implementation and test.

Registration of Credit

It should be possible for producers to use a graphical interface in which they can add their programs and the associated credits. It is important that producers can only edit the programs they own themselves.

Making Persons Unique

It should be possible to link/merge persons that refer to the same person in the real world. When two different producers want to create a credit for a program, that credit must be associated with a person and its role.

It can be difficult to tell if a person is the same or two different persons when entering a name like *Hans Pedersen*. Therefore, it should be possible for Producers to create Persons that can be merged into the same Person.

TV 2 is not allowed to store sensitive data about persons in this system (like a Personal Security Number (CPR) or Phone Number). Therefore it is necessary to find another way to distinguish persons in the system and make sure the credits are correctly linked to the right person.

⁶ For inspiration see: BBS Credits and branding:
<https://www.bbc.co.uk/commissioning/tv/production/articles/credits-branding-trademarks> or DR:

DR's krediteringsregler for TV:
https://www.dr.dk/NR/rdonlyres/00221a7b/dpikscstjptklixxdnjgywgeuakhwpog/DR_kreditmanual_050810.pdf

Using Data

It should be possible for TV 2 to export a specific amount of data in various formats, such as XML and CSV, so that TV 2 can easily use the data for other purposes. At the same time, it should also be possible to search the data directly so that it's possible to find persons, programs, series and credits in a simple and meaningful way when using the graphical interface.

It is very important that the system makes it easy for TV users to see the credits for the program they have just seen. There should be a clear (easy to use) reference from the TV program to the place in the Credits Management System where users can see the associated credits for the corresponding program.

Integration with other systems

It could be interesting for TV 2 to explore the possibilities of integrating the Credits Management System with other systems, both TV2's own systems (internal and external) as well as third party systems (like Yousee Play, Boxer Play, Stofa, etc). It is therefore important that the crediting information in the Credit Management System is compatible with the crediting information in other systems.⁷

Notifications

It could also be interesting for TV 2 to investigate the possibility of making notifications, each time something new happens in the system. This could bring value to both the System Maintainers (so they know when there's new data they have to look through) and TV Stations/Producers (for example to notify them when a Program, Series or Season they own have been changed - a Credit may have been removed, updated etc. by a System Maintainer). This is just examples, there's a lot of possible uses for a Notification system in this context.

“Samrådet for Ophavsret” and “Producentforeningen” (and other parties involved in paying royalties to the people involved in the production of a program), could also be interested in getting a form of notification/message each time something new has entered the system, where they have to confirm the credit lists and pay out royalties based on these.

⁷ Integration with other systems could be by exposing data via a REST API as shown in figure 2, but a REST API is outside the scope of the project

Access Control

Some kind of access control should be implemented for the protected parts of the system (entering/editing/deleting data etc.)

There should also be a publicly available part of the system, where it is possible to see the credits for a production without logging in.