Anders Hesselbjerg - 21/09-1991
Daniel Causevic -
Lano Jalal -
Alexander Stohn -

Github link: https://github.com/Alex8943/Eksamen2020

21/12-2020

Eksamensprojekt - Projektkalkulationsværktøj

Indholdsfortegnelse

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Indledning (DC)

Vi vil ved hjælp af forskellige værktøjer og metoder gennemgå og analysere vores process ift. problemformuleringen. Læseren vil gradvist blive introduceret til vores metoder, hvor vi først vil beskrive, hvad vi har at gøre med, hvorefter vi vil beskrive, hvordan det bliver implementeret i vores projekt ved at give eksempler fra koden vha. screendumps.

Problemformulering

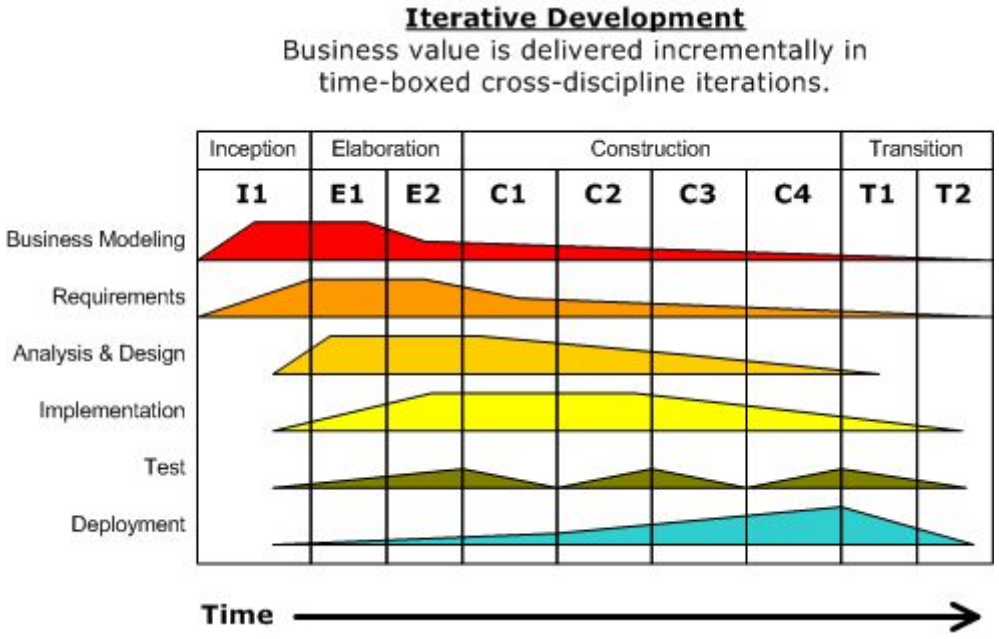
Vores kunde Alpha Solutions har bedt os lave projektkalkulationsværktøj, sagt med andre ord: Vi skal lave et produkt, der gør det overskueligt med alle de projekter Alpha Solutions har kørende.

- Produktet skal kunne give en opsummering over tidsforbruget af projekter, delprojekter og opgaver.
- Derudover skal den give tidsforbrug på arbejdsdage, så man ved, hvor mange timer der skal arbejdes hver dag.

Systemudvikling

Unified Process (AH og AS)

Unified process er en iterativ metode. Processen består af fire faser - inception phase, elaboration phase, Construction phase og Transition phase. Metoden fokuserer på at identificere risici og faldgruber, og den tvinger projektdeltagere til at overveje, om det overhovedet er fornuftigt at gennemføre projektet. Unified process er inkrementel og iterativ.[1]



Inception phase - 23/11-2020 - 25/11-2020:

Første iteration 23/11-2020 - 25/11-2020:

Efter vi fik udleveret kravene til projektet, prøvede vi at skabe et overblik over opgavens omfang. Vi diskuterede, om vi havde den samme opfattelse af kravene til projektet, og vi fastslog de grundlæggende visioner for systemet. Derudover lavede vi en risikoanalyse, som skulle sørge for, at vi kunne forudse forskellige faldgruber. Vi lavede en plan for at minimere risikoen for at disse problemer skulle opstå, og hvis problemer alligevel opstod, havde vi en plan b klar, så udviklingen af projektet ikke

[1] https://da.wikipedia.org/wiki/Unified_Process

gik i stå. Ofte beslutter man i denne fase, om projektet overhovedet skal gennemføres, men da det er et obligatorisk studieprojektet, var der ingen tvivl.

Vi designede en prototype af systemarkitekturen. Vi beskrev systemets nøglefunktioner som Use Cases.

Det indebar blandt andet de centrale Use Cases - log ind, create user, create project og create subproject. Derudover lavede vi supplementary specifications for at fastlægge ikke-funktionelle krav til systemet. I denne fase lavede vi også et skema, der viser, hvor meget vi hver især skulle arbejde for at kunne komme i mål med projektet. Der blev også lavet en plan for, hvor længe hver fase skulle vare.
Planen var at afsætte. 3 dage til Inception phase, 7 dage til Elaboration phase, 20 dage til Construction phase.

I første fase går man fra ideen til virkelighed af projektet. Denne fase svarer på 3 vigtige spørgsmål:

1) Hvem er brugerne af produktet? Hvad gør produktet godt for dem? besvarelse af dette spørgsmål: miniudgave af use case)
2) Hvordan kommer systemet til at se ud (besvarelse af dette spørgsmål: lave diagrammer)
3) Hvad er planen og hvor meget vil det koste? (det er gennem undersøgelse med klienten)

Elaboration phase - 26/11-2020 - 01/12-2020:

Første iteration 26/11-2020 - 01/12-2020:

I Elaboration phase validerede vi vores valgte system arkitektur. Vi lavede use case diagram, klassediagram og pakkediagram. I denne fase sad vi og lavede vores brainstorm til virkelighed, denne process er stadig før selve koden er gået i gang. Meningen med de diagrammer, er at have en ide til ens projekt, så man har noget at gå ud fra, når man sidder og koder. Ofte, altså man snakker 9/10 tilfælde, så vil ens diagrammer altid ændres undervejs, men meningen bag dem er heller ikke, at de er perfekte fra starten af - det gør bare processen nemmere i stedet, for så er man på bar bund med koden.

Som vi kan se i denne på billedet, så er business modelling høj i starten, men bliver lavere hen af vejen i denne elaboration periode.
Requirements og analysing & design er de 2 faser, der er mest fokus på i denne fase. Grunden til det, er at kravene til opgaven begynder at dukke frem i og med

man stille og roligt begynder at lave krav til rækkefølge i opgaven, hvordan hjemmesiden skal designes, og hvad programmet skal kunne. Hvem er brugerne af produktet og så videre.

I denne fase startede vi også med at programmere helt grundlæggende klasser i systemet. Klassediagram, Domænemodel og Use Cases gjorde det klart, at vi havde brug for 5 klasser i models. User, projects, subprojects, tasks og admin. Under repository havde vi brug for 'mapper' og og klassen Dbmanager(forbindelse til databasen). Det endelige klassediagram blev dog først lavet i Construction phase, fordi klasserne blev ændret senere i processen.

Construction phase 02/12-2020 - 21/12-2020

I Construction phase skal man sørge for at der er mange tests for at sikre sig at den kode der skrives virker. I denne fase er det helt afgørende, at al nødvendig kode færdiggøres.

I denne fase er der stadig mulighed for at man dropper projektet, og det kan der være flere grunde til. Det kan være følgende grunde til:

1) at økonomien ikke er stabil nok til at færdiggøre dette produkt. Det kan være at du som virksomhed, har bedt kunden om et for stort beløb, til at lave produktet så kunden ikke vælger dig som virksomhed alligevel

2) Kunden eller virksomheden har ikke tid nok til at færdiggøre produktet som kunden kræver.

3) At din use cases og de andre diagrammer bare ikke er blevet gode nok til at kunden er tilfreds med din ide til produktet.

Første iteration - 02/12-2020 - 06/12-2020

Vi ændrede use cases og UML-diagrammer, da de første udkast ikke var helt nøjagtige længere. I denne iteration var der stor fokus på at kode alt, som vi havde planlagt i Inception Phase og Elaboration Phase.

Anden iteration - 07/12-2020 - 14/12-2020

Jo længere construction fasen skred frem, jo mindre tid brugte vi på at lave diagrammer. I denne iteration gik vi mere i dybden med mindre funktionaliteter.

Tredje iteration - 15/12-2020 - 21/12-2020

Denne iteration blev også brugt til at programmere og fintune mange aspekter af programmet. Vi havde planlagt, at alle nøglefunktioner skulle virke - og det gjorde de stort set, da tredje iteration startede. Derfor brugte vi en del tid på at finjustere ‘view’ (css og html).

Transition phase:

I dette projekt er det ikke fornuftigt at bruge lang tid på overdragelses fasen. Årsagen er, at vi ikke skal aflevere projektet til Alpha Solutions. Vi når ikke at få feedback på projektet fra virksomheden, som vi udvikler systemet til, før den endelige version afleveres. Vi kommer heller ikke til at være en del af oplæringen af brugerne.

I virkeligheden skal vi have have feedback på produktet. Hvis vi var en rigtig virksomhed og vi handlede med en rigtig kunde, vi ville have lavet flere tests, altså hvor vi har samlet en rigtig gruppe af vores målgruppe. Men det kommer vi ikke til, i og med dette er et eksamensprojekt, så får vi ikke feedback fra Alpha Solutions.

Stakeholder analyse (AH)

Formålet med at lave en interessentanalyse er at finde ud af hvem der har interesse i projektet og hvordan disse stakeholders skal håndteres. Det fremgår ikke af opgavebeskrivelsen om interessentanalysen skal laves for selve projektet eller virksomheden Alpha Solutions, men vi har vurderet, at det giver mest værdi at lave analysen for selve projektet.

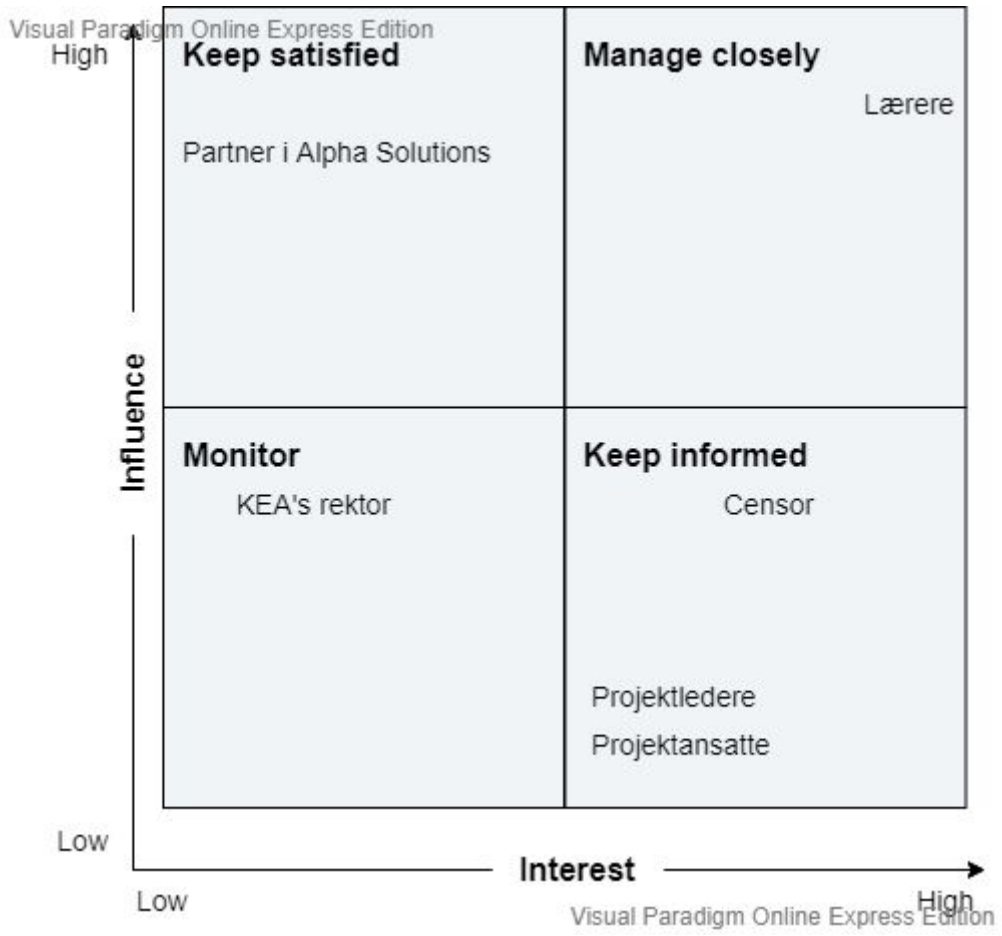
Vurderingen af hvilke interessenter der er relevante afhænger af hvilket produkt der udvikles og om vi i det hele taget antager, at vi er studerende eller f.eks. er udviklere i en softwarevirksomhed. Dette fremgår ikke af opgavebeskrivelsen, og vi har derfor valgt at antage, at systemet er et studieprojekt, og dermed er lærere vigtige interessenter. Derudover skal det besluttes om systemet kun skal bruges af Alpha Solutions, eller det skal være et produkt som skal sælges bredt til mange virksomheder. Det antages at produktet kun bliver solgt til Alpha Solutions.

Identifikation af stakeholdere:

- Lærere
- Censor
- Projektledere
- Projektansatte
- Partner i Alpha Solutions
- Rektor på KEA
- Bestyrelse på KEA

Vi lavede en power grid matrix, der viser hvor stor interesse interessenterne har i projektet og hvor stor indflydelse de har. Modellen viser, at lærere er de vigtigste interessenter i vores projekt. Vi skal være særligt opmærksomme på disse interessenter.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal



Lærere:

Lærerne har stor indflydelse på projektet og har stor interesse i projektet, fordi de har udleveret opgavebeskrivelsen. Der er klare krav til, hvilke værktøjer der skal benyttes til at udvikle systemet. Både i forhold til programmeringssprog og kodestruktur, men også i forhold til dokumentation, diagrammer, GUI, arbejdsproces (Unified Process), rapportskrivning, osv.

Vi har løbende kontakt med lærere i forbindelse med projekt-vejledninger, og vi kan få indikationer på om vores ideer til systemudviklingen opfylder kravene til projektet. Derudover er det i sidste ende lærerne, som skal bedømme om projektet opfylder kravene.

Censor:

Censor har en relativ stor interesse i vores projekt, men har ingen indflydelse på hvordan projektet gennemføres. Censor har dog stor indflydelse på vurderingen af slutproduktet, så man kan argumentere for at vedkommende er en relativ vigtig stakeholder.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Projektledere:

I matrixen ses det, at projektledere har lav indflydelse på projektet og middel interesse i projektet. Man kan argumentere for, at projektlederne har indflydelse på projektet i den forstand, at meningen er at lave et brugbart system, der kan skabe værdi for en virksomhed. Man kan dog ikke forvente, at et revolutionerende kalkulering værktøj kan udvikles af studerende på blot 3-4 uger. Så det er nok forholdsvis sandsynligt, at projektledere ikke kommer til at se systemet medmindre det er helt ekstraordinært.

Projektansatte:

I matrixen ses det at projektansatte har lav indflydelse på projektet og middel interesse i projektet. Man kan argumentere for at projektansatte har indflydelse på projektet i den forstand, at meningen er at lave et brugbart system, der kan skabe værdi for en virksomhed. Hvis vi var ansat som softwareudviklere hos en virksomheden, havde analysen som sagt set helt anderledes ud. Især projektledere havde været vigtige stakeholders, men også projektansatte havde haft stor interesse i projektet, fordi de ville være en del af målgruppen, som ville prøve at ramme.

Partner i Alpha Solutions:

Partneren i Alpha Solutions har stor indflydelse på projektet og middel interesse i projektet. Han har præsenteret projektemnet og dermed sat en ramme for hvilket produkt som skal leveres. Hans interesse i projektet er mere tvivlsom, da det ikke er sikkert, at han følger udviklingen af systemet og det færdige produkt.

KEA’s rektor:

Man kunne have undladt at tage rektor med i analysen, men vedkommende har muligvis indflydelse på udformningen af opgavebeskrivelsen i forhold til studieordninger. Derudover har vedkommende nok også nogen interesse i at studerende klarer sig godt og får gode karakterer, men det er tvivlsomt om interessen er stor for enkelte projekter.

Interessentanalyse for den enkelte interessent:

På næste side ses en analyse af interessenterne hver for sig. Den giver forslag til, hvordan interessenterne skal håndteres, hvilket er vigtigt både i forhold til at løse og forebygge problemer.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Interessent analyse af hver stakeholder

Interessent	**Mål**	**Reaktion i andre projekter**	**Forventet reaktion**	**Indflydelse** **1-5**	**Ideer til at forebygge og løse problemer med interessenter**
Lærere	At modtage et projekt, der opfylder alle krav og leveres til tiden.	Det er afgørende at kravene til projektet opfyldes. Det er ofte vigtigere at koden er af høj kvalitet i stedet for at have mange funktioner som er skrevet med ukonventionel kode.	Giver en dårlig karakter hvis projektet er dårligt	5	Det er vigtigt at inkludere lærerne i vores arbejdsproces og ideer løbende under hele projektet
Partner i Alpha Solutions	At se interessante løsninger af projektoplægget	Ingen tidligere erfaring	Forventer kun positiv feedback hvis projektet er ekstraordinært	3	Skrive mails til partneren eller en anden ansat hos Alpha Solutions for at få svar på problemstillinger
Censor	At modtage et projekt, der opfylder alle krav og leveres til tiden.	Forventer mange af de samme ting som lærerne	Giver en dårlig karakter hvis projektet er dårligt	3	Det er vigtigt at forberede sig godt til eksamen, så man kan præstere når projektet skal forsvares

Projektansatte	Forventer et system, der skaber værdi for virksomheden	Ingen tidligere erfaring	Ansatte vil muligvis klage over systemet, eller ønske ikke at bruge programmet.	1	Give ansatte mulighed for at beta teste systemet
Projektledere	Forventer et system, der skaber værdi for virksomheden	Ingen tidligere erfaring	Projektledere ville klage over programmet hvilket ville være skidt i forhold til evt. fremtidigt samarbejde.	1	Løbende kommunikere med projektledere for at forventningsafstem me
KEA's rektor	Forventer en opgave der opfylder studieordningen	Ingen tidligere erfaring	Forventer kun en reaktion hvis projektet er meget utilfredsstillende	1	Ikke nødvendigt

Konklusion

De vigtigste stakeholders er lærere, partneren i Alpha Solutions og censor. Lærerne har både mest indflydelse og har størst interesse i projektet. Dog kan man ikke have kontakt til censorer før eksamen, så det er begrænset hvad man kan gøre for at løse problemer med denne stakeholder. Vi kunne have undladt at have censor og lærere med i analysen, hvis vi havde antaget, at projektet ikke var et studieprojekt. Man kunne også havde medtaget konkurrenter eller andre ansatte i Alpha Solutions, men umiddelbart kommer vores løsning ikke til at påvirke andre end de stakeholders, som allerede er blevet nævnt. Hvis systemet ville blive dyrt og revolutionerende for virksomheden Alpha Solutions, kunne man også have taget CEO'en og CTO'en med i analysen. Man kan sagtens forestille sig, at enten en CTO eller CEO har taget beslutningen om at købe systemet, og de ville i så fald være meget vigtige stakeholders.

Udvidet risikoanalyse (AH)

En risikoanalyse bruges til at vurdere, hvilke risici der er for at et projekt ikke lykkes eller ikke lever op til kravene. Først undersøges alle potentielle risici og derefter vurderes det hvor sandsynligt det er, at disse kan have en negativ betydning for udviklingen af projektet. En risikoanalyse kan bruges på mange forskellige måder. Mange virksomheder laver også risikoanalyser på et mere overordnet plan for at finde ud hvad der kan true virksomheden. Det kan f.eks. være analyser som indeholder konkurrenceforhold, produktionsforhold, indtjenings- og kapitalforhold og samfundsøkonomiske forhold. Analysen i denne rapport fokuserer på at analysere risici i forhold til at gennemføre projektet og at resultatet bliver tilfredsstillende.

Identifikation af potentielle risici:

Projekt:

- Sygdom.
- Hvis nogen stopper med at studere, kan det forsinke visse dele af projektet.
- Hvis vores gruppe eller Alpha Solutions ikke definerer klare krav til systemet kan det resultere i misforståelse af hvordan det endelige produkt skal fungere.
- Bruge for lang tid på nøgleopgaver.
- Unøjagtige tidsestimeringer.
- Begrænset erfaring med at bruge Spring og SQL.

Produkt:

- Forkert tidsestimering.
- Hvis vores gruppe eller Alpha Solutions ikke definere klare til systemet kan det resultere i misforståelse af hvordan det endelige produkt skal fungere.
- Sygdom.
- Hvis nogen stopper med at studere, kan det forsinke visse dele af projektet.
- Begrænset erfaring med at bruge Spring og SQL.

Man kunne også have valgt at undersøge hvilken risiko hele virksomheden Alpha Solutions løber ved at betale for projektet, men da Alpha Solutions ikke betaler for produktet, lader det til, at risikoen er så lille, at det ikke er relevant at tage med. Alpha Solutions skal dog bruge ressourcer på at lave opgavebeskrivelsen og give løn til de ansatte, der bruger tid på dette.

Risikoanalyse tabel:

Følgende risikoanalyse tabel viser de mest relevante risikomomenter. Tabellen viser sandsynligheden for, at disse problemer bliver til virkelighed og hvor stor konsekvensen er for projektudviklingen. Sandsynlighed og konsekvens ganges for at give et billede af, hvor vigtig det er at håndtere risikomomentet.

Risikoanalyse tabel

Risici	Sandsynlighed	Konsekvens	Produkt
Unøjagtige tidsestimeringer	6	6	36
Misforståelser af krav til projektet	2	8	16
Sygdom eller nogen forlader gruppen	1	5	5
Manglende erfaring med at benytte Spring Framework og SQL	2	9	18

Udvidet risikoanalyse tabel

Følgende risikoanalyse tabel viser sandsynlighed, præventive tiltag og løsninger. Især præventive tiltag og løsninger er vigtige når der laves en risikoanalyse. Det nytter ikke noget at opstille sandsynlighed for at forskellige problemer opstår, hvis analysen ikke indeholder løsningsforslag og præventive tiltag.

Udvidet risikoanalyse tabel

Risici	Sandsynlighed	Konsekvens	Produkt	Præventive tiltag	Løsninger

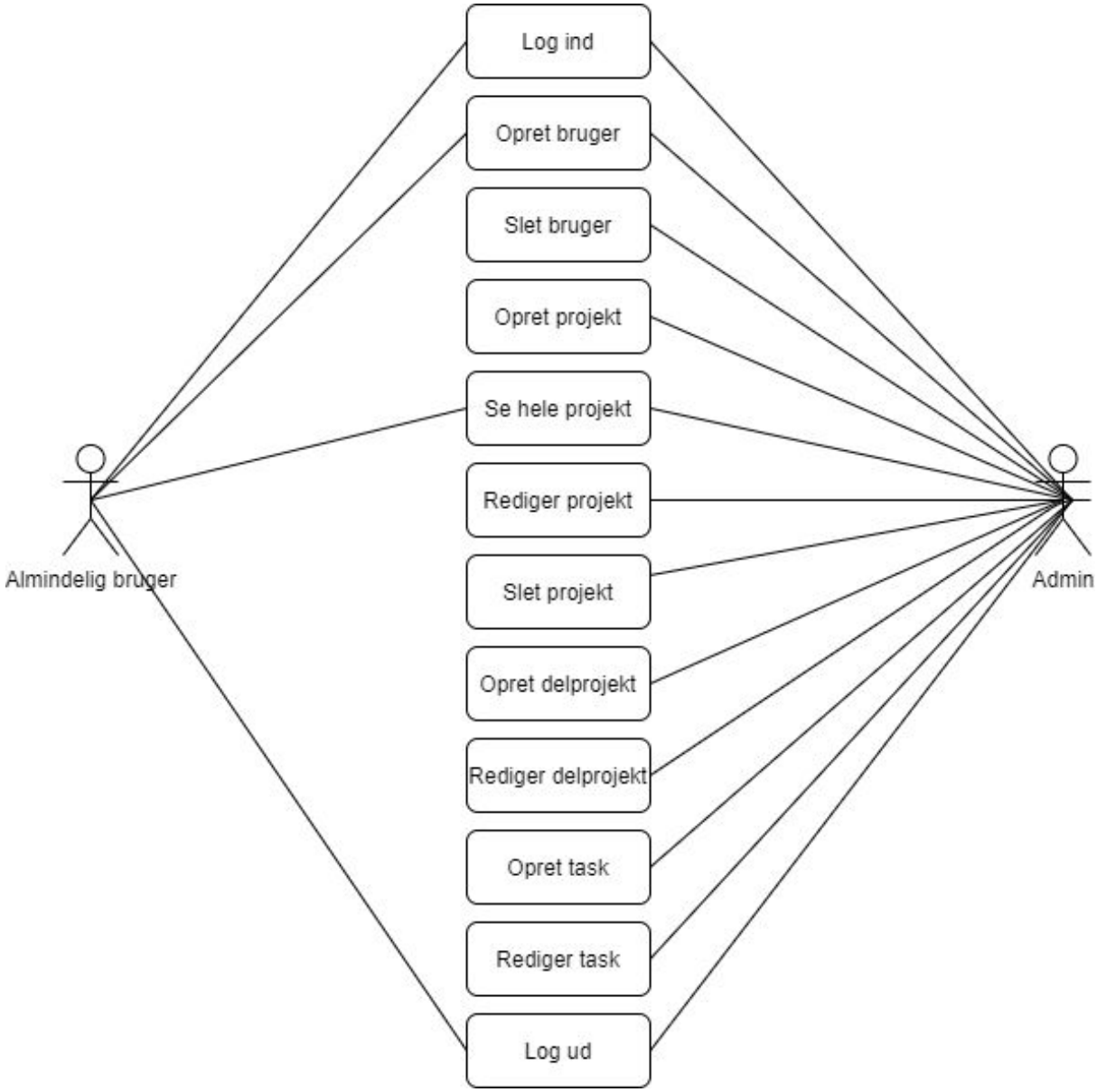
Unøjagtige tidsestimeringer	6	6	36	Gennemgå tid estimeringer flere gange og evt. få en ekstern person til at gennemgå planen.	Hvis nogle af opgaverne ikke bliver løst til tiden, skal man være forberedt på at arbejde mere end planlagt.
Misforståelser af krav til projektet	2	8	16	Brug lang tid på at diskutere hvilke krav, der skal opfyldes. Gennemgå kravspecifikationer med lærere og evt. Alpha Solutions.	Hvis det sent i processen opdages, at kravene til projektet er blevet misforstået, kan det være nødvendigt, at fokusere på de mest vitale features af programmet.
Sygdom eller nogen forlader gruppen	1	5	5	Godt arbejdsmiljø og sørg for at mindst to personer (helst alle) forstår alle dele af projektet. Sørg for at overholde retningslinjer for forebyggelse mod COVID-19	Hvis nogen bliver syg eller forlader gruppen, er det planlagt hvem der skal overtage vedkommendes arbejdsopgaver.
Begrænset erfaring med at benytte Spring Framework, SQL og Thymeleaf.	2	9	18	Afsætte tid til at blive bedre til at bruge Spring Framework, SQL og Thymeleaf	I værste fald må visse features udelades, så alle andre krav bliver opfyldt. Det er en svær beslutning at træffe, men det kan være en stor fejl at sidde fast i små opgaver for længe.

Konklusion

Det kan konkluderes, at der er størst risiko for at der laves forkerte vurderinger af, hvor længe nøgleopgaver tager at løse. Dette er en fare, som vi skal være meget opmærksomme på at kunne håndtere. Derudover er begrænset erfaring med at benytte Spring og SQL også vigtig, fordi konsekvensen er meget høj.

Use cases (AH og AS)

Use case diagram:



Use case diagrammet viser alle de måder en bruger kan interagere med systemet. Formålet med diagrammet er at give et overblik over hvordan de forskellige 'actors' kan bruge systemet. I dette tilfælde vises det, at en almindelig bruger (projektansat) kun har muligheden for at logge ind, oprette bruger, se hele projektet og logge ud. Admin kan benytte alle systemets funktioner.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Brief Use Case: (AS)

Use Case id: Opret bruger

Goal Level: Oprette en bruger, så du kan logge ind.

1. Kunden åbner forsiden.
2. Trykker på opret bruger.
3. Skriver information til opret bruger.
4. Føres tilbage til forsiden.
5. Logger ind.

Use Case id: Log ind

Goal Level: At logge ind.

6. Kunden åbner forsiden.
7. Logger ind.
8. Føres videre til profil.

Use Case id: log ud

Goal Level: At kunne logge ud.

9. Kunden logger ind.
10. Inde på hvilken som helst side.
11. trykker på log ud.

Use Case id: Opret Projekt

Goal Level: At kunne oprette et projekt

12. Kunden åbner forsiden.
13. Trykker på opret projekt.

14. Skriver informationer for at lave et projekt.
15. Trykker på opret projekt
16. Føres derefter videre til profil forside.

Use Case id: Slet projekt

Goal Level: At kunne slette et projekt.

17. Kunden åbner forsiden.
18. Trykker på opret bruger.
19. Skriver information til opret projekt.
20. Føres tilbage til profil forside.
21. Trykker på slet projekt.
22. Her vælges hvilket projekt du vil slette.
23. Trykker på slet projekt.

Use Case id: Rediger projekt

Goal Level: At kunne redigere projekt.

24. Brugeren åbner forsiden.
25. Trykker på opret bruger.
26. Skriver information til opret bruger.
27. Føres tilbage til forsiden.
28. Logger ind.
29. Trykker på rediger projekt.
30. Derinde ændres projektoplysninger.
31. Derefter føres du til profil forside.

Use Case id: Se projekt

Goal Level: At kunne se alle ens projekter.

32. Brugeren åbner forsiden.
33. Trykker på opret bruger.
34. Skriver information til opret bruger.
35. Føres tilbage til forsiden.
36. Logger ind.
37. Trykker på Se projekt inde på userProfile siden.

Use Case id: Opret delprojekt
Actor: Admin

En admin ønsker at oprette et delprojekt. Vedkommende logger ind ved at indtaste e-mail og adgangskode. Admin trykker opret projekt og indtaster information om projektet i de tomme felter. Efterfølgende indtaster admin informationer om et delprojekt og trykker opret delprojekt. Til sidst indtaster admin informationer om en task og trykker opret task.

Use Case id: Rediger delprojekt
Actor: Admin

En admin ønsker at redigere et delprojekt. Vedkommende logger først ind ved at indtaste e-mail og adgangskode. Hvis admin ikke har oprettet et projekt, skal dette gøres som i ovenstående use case: opret delprojekt. Efterfølgende skal admin trykke se projekter og trykke rediger delprojekt.

Use Case id: Opret task
Actor: Admin

En admin ønsker at oprette en task. Vedkommende logger først ind ved at indtaste e-mail og adganskode. Admin trykker opret projekt og indtaster information om projektet i de tomme felter. Efterfølgende indtaster admin informationer om et delprojekt og trykker opret delprojekt. Til sidst indtaster admin informationer om en task og trykker opret task.

Use Case id: Rediger task
Actor: Admin

En admin ønsker at redigere en task. Vedkommende logger først ind ved at indtaste e-mail og adgangskode. Hvis admin ikke har oprettet et projekt, skal dette gøres som i ovenstående use case: opret delprojekt. Efterfølgende skal admin trykke se projekter og trykke rediger task.

Casual Use Case(AS)

Use Case id: Opret bruger

Goal Level: Oprette en bruger, så alle funktionaliteter kan benyttes.

38. Kunden åbner forsiden.
39. Trykker på opret bruger.
40. Skriver information til opret bruger.
41. Føres tilbage til forsiden.
42. Logger ind.

Extensions (alternativer)

1) At brugeren ikke har mulighed for at tilgå vores hjemmeside (pga internetforbindelse eller lign.)

Use Case id: Log ind
Goal Level: At kunne logge ind

43. Brugeren åbner forsiden.
44. Trykker på opret bruger.
45. Skriver information til opret bruger.
46. Føres tilbage til forsiden.
47. Logger ind.

Extensions (alternativer)
Failer:

1) Brugeren ikke kan logge ind, grundet manglende internet.
2) Brugeren har ikke lavet en bruger på vores hjemmeside.
3) At Brugeren ikke har en eksisterende mail, til at logge ind med.

Use Case id: Log ud

Goal Level: At kunne logge ud.

48. Kunden logger ind.
49. Inde på hvilken som helst side.
50. Trykker på log ind.
51. Herefter kan man kan trykke log ud.

Extension (Alternativer)
Failer
1) Knappen ikke fører til log ind side.
2) Man er stadig logget ind selvom man har forsøgt at logge ud.

Use Case id: Opret projekt

Goal Level: At kunne oprette et projekt

52. Brugeren åbner forsiden.
53. Trykker på opret projekt.
54. Skriver informationer for at lave et projekt.
55. Trykker på opret projekt
56. Føres derefter videre til profil forside.

Extension (Alternativer)
Failer
1) At oprette et projekt ikke er muligt, pga teknisk fejl
2) dato ikke er muligt at oprette

alternativ til
2: at skrive dato i beskrivelsen

Use Case id: Slet projekt

Goal Level: At kunne slette et projekt.

57. Brugeren åbner forsiden.
58. Trykker på opret bruger.
59. Skriver information til opret bruger.
60. Føres tilbage til profil forside.
61. Trykker på slet projekt.
62. Her vælges hvilket projekt du vil slette.
63. Trykker på slet projekt.

Extension (Alternativer)

Failer
3) At slette projekt ikke er muligt, pga teknisk fejl.
4) Internetforbindelse ikke gør det muligt.
5) Fejl i databasen, der gør at projekt ikke kan slettes.

alternativ til
5: at skrive i beskrivelsen, at projektet er slettet.

Use Case id: Rediger projekt

Goal Level: At kunne slette bruger.

64. Brugeren åbner forsiden.
65. Trykker på opret bruger.
66. Skriver information til opret bruger.
67. Føres tilbage til forsiden.
68. Logger ind.
69. Trykker på rediger projekt på profil index.
70. Derinde kan du ændre dine projektoplysninger.
71. Derefter føres du til profil forside.

Extension (Alternativer)
Failer
6) At det ikke er muligt at redigere projekt, pga teknisk fejl

alternativ til
2: at skrive sine opdateringer i beskrivelsen, så man kan se sine seneste ændringer.

Use Case id: Se projekt

Goal Level: At kunne se alle ens projekter.

72. Brugeren åbner forsiden.
73. Trykker på opret bruger.
74. Skriver information til opret bruger.
75. Føres tilbage til forsiden.
76. Logger ind
77. Trykker på se projekt inde på account index.

Extension (Alternativer)
Failer

7) At redigere projekt ikke er muligt, pga teknisk fejl

alternativ til
2: at skrive sine opdateringer i beskrivelsen, så man kan se sine seneste ændringer.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Fully dressed use cases:(AS)

https://www.youtube.com/watch?v=zid-MVo7M-E

Use case ID:	**1**
Use Case Name:	**Opret bruger**
Created By:	**Alexander**
Create Date:	**23/11-2020**
Description:	**Brugeren går ind på vores hjemmeside, på forsiden ved toppen af hjørnet er det muligt at trykke på opret bruger.** **her føres brugeren til en ny side hvor du meget simpelt skal skrive din mail, og en adgangskode, som gemmes i databasen, når du så trykker på opret bruger, føres man tilbage til forsiden, så du kan logge ind med de informationer du lige har skrevet i opret bruger.**
Primary Actor:	**User og admin**

Husk

- **user goal**
- **alternativer**

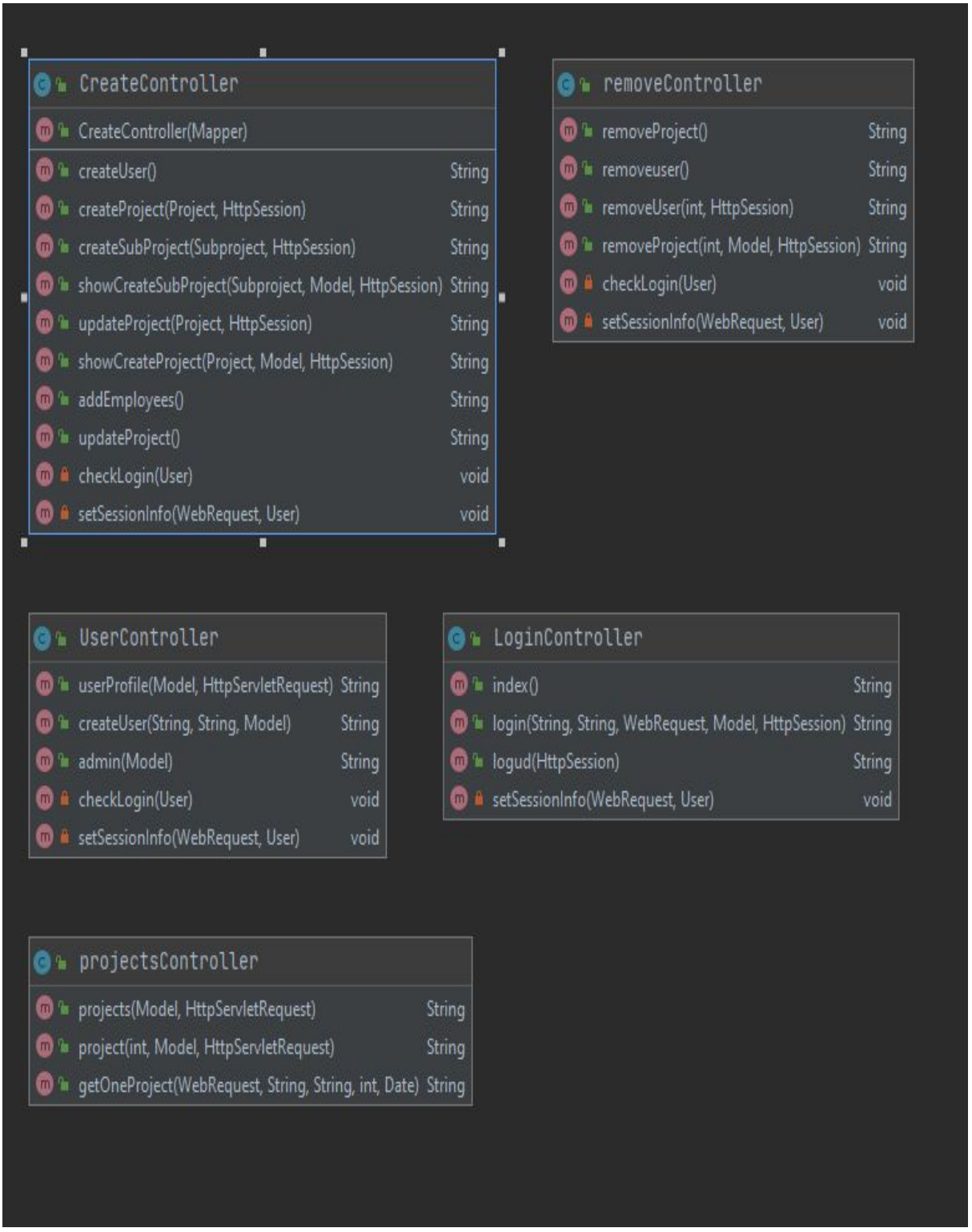
Use case ID:	**2**
Use Case Name:	**Log ind**
Created By:	**Alexander**
Create Date:	**23/11-2020**
Description:	**Kunden går ind på vores hjemmeside, hvis kunden har lavet en bruger (ved at trykke på opret bruger) på vores hjemmeside, så kan man logge ind simpelt på forsiden. Når man er logget ind, føres man til sin profil.**
Primary Actor:	**Bruger og admin**

Use case ID:	**3**
Use Case Name:	**Opret projekt**
Created By:	**Alexander**
Create Date:	**26/11-2020**
Description:	**Kunden går ind på vores hjemmeside så kan man logge ind simpelt på forsiden. Når man er logget ind, føres man til sin profil. Herfra kan man i hjørnet til højre oprette et projekt ved at skrive:** **- navn på projektet** **- deadline dato** **- beskrivelse af projektet** **- antal medarbejder** **- dagsdato** **Admin har kun adgang til at oprette projekter, så admin af projektet vil kunne se alt information af projektet efter (han/hun) har lavet projektet på sin profil.**
Primary Actor:	**Admin**

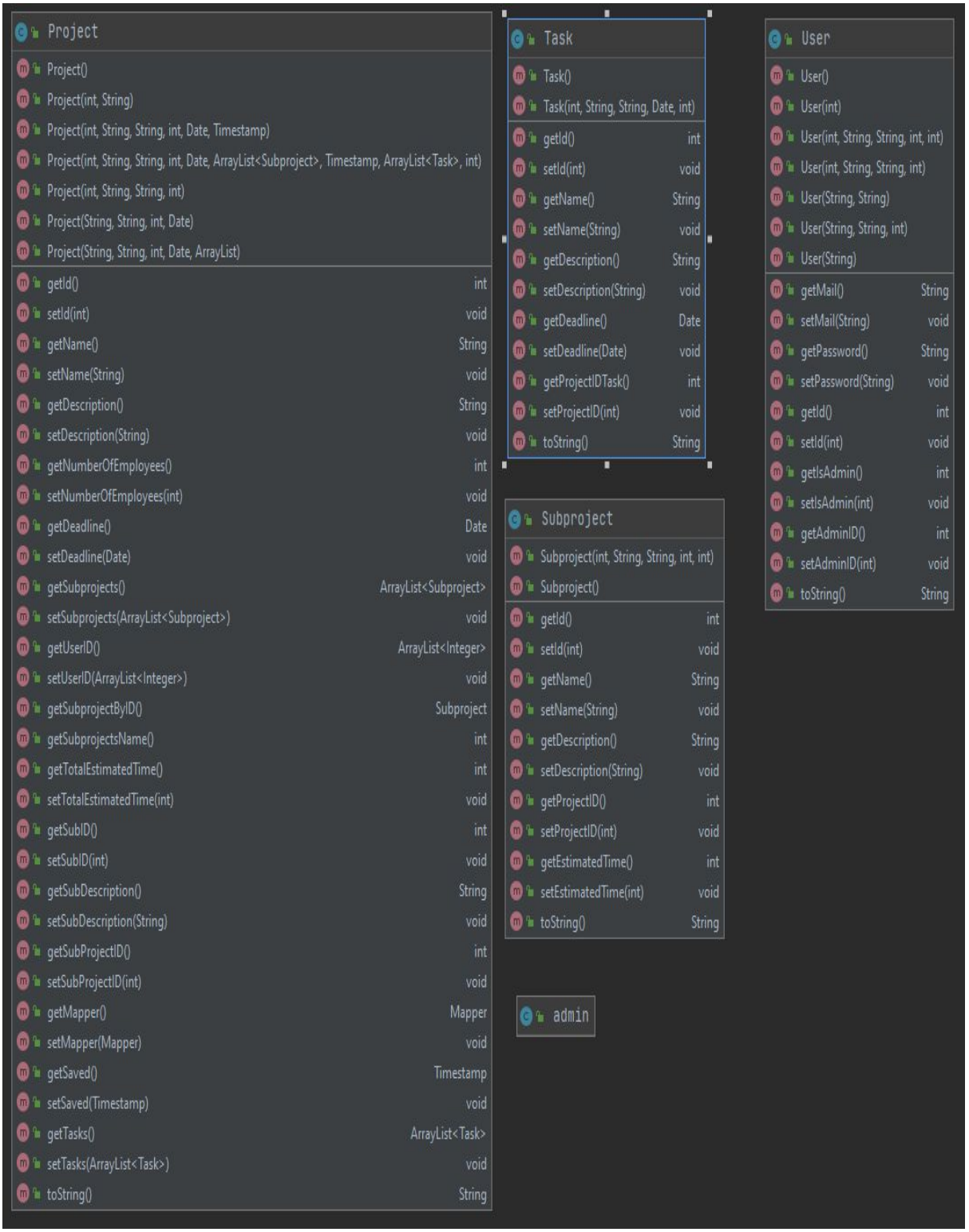
Use case ID:	**4**
Use Case Name:	**Slet projekt**
Created By:	**Alexander**
Create Date:	**26/11-2020**
Description:	**Admin har adgang i sin profil side, til at slette de projekter han har lyst til. Det gør han ved at trykke på slet projekt knappen, hvorefter projektet slettes.**
Primary Actor:	**Admin**

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

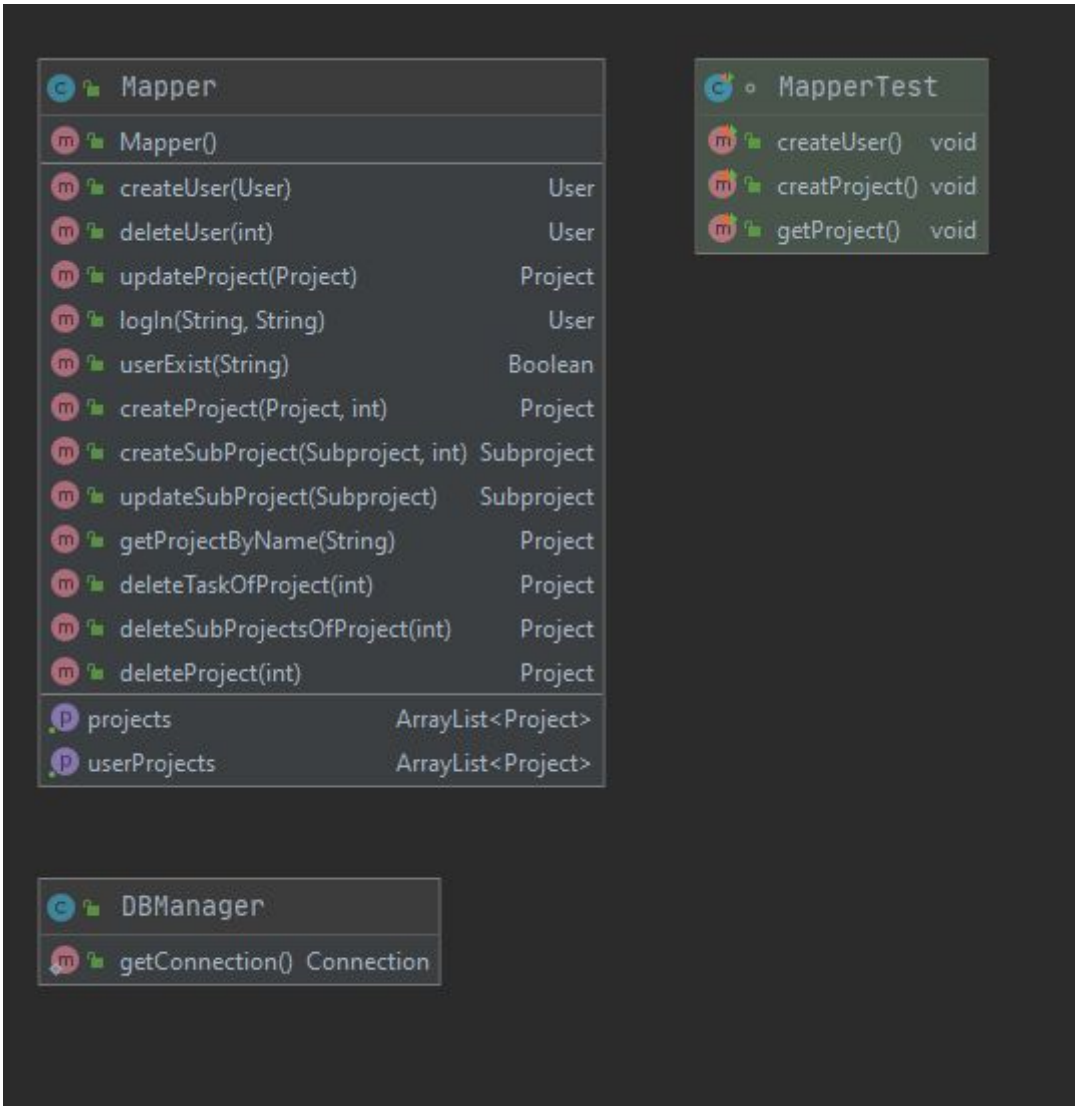
Klasse diagram: (LJ)



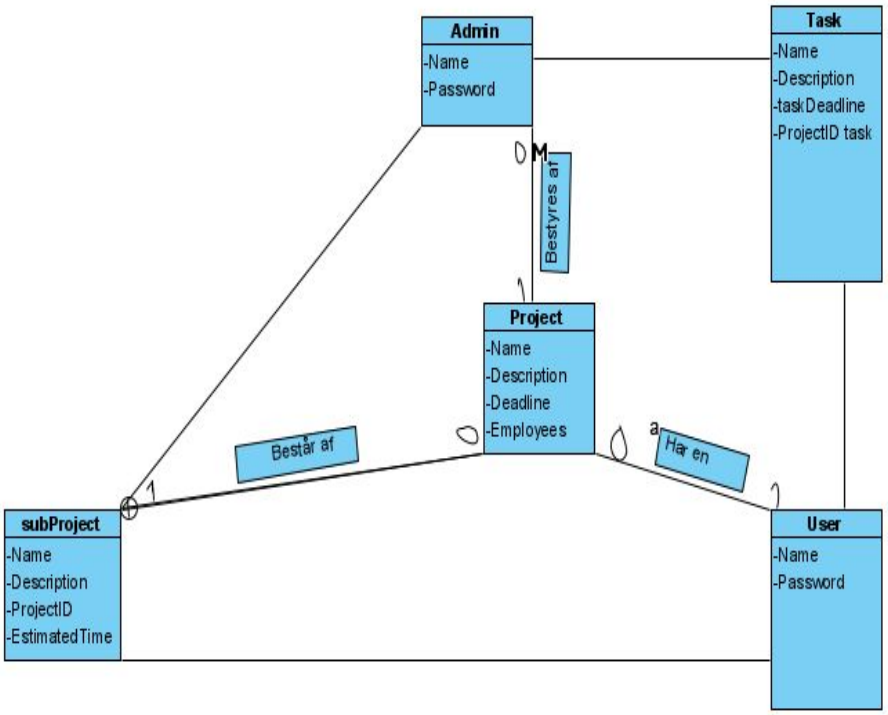
AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal



AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal



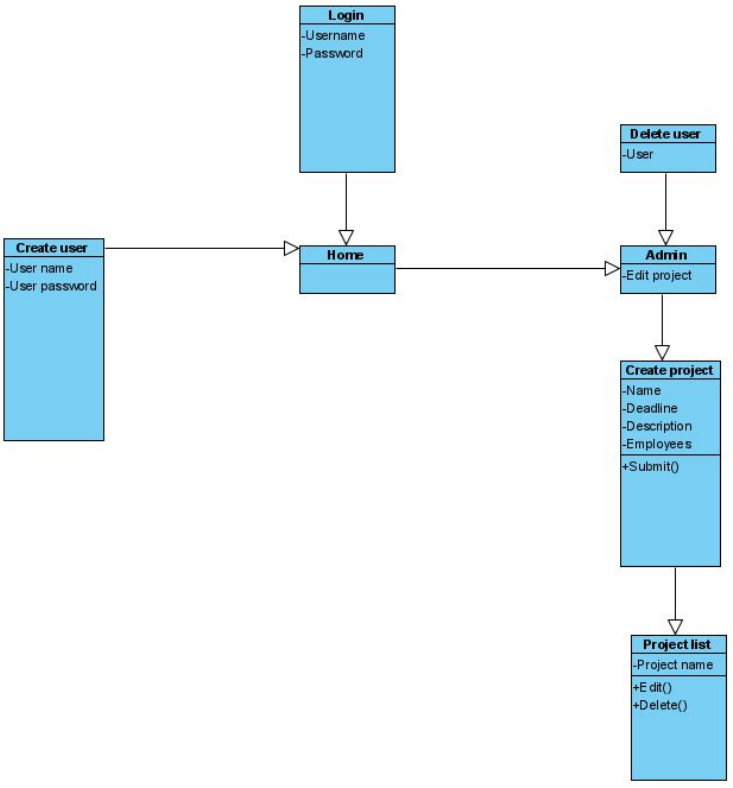
Domain model (LJ)



Ud fra dette diagram kan man se at et projekt er tilknyttet en Admin og User som brugere af projektet. Admin har adgang til at oprette “project” hvor der vælges navn, beskrivelse, deadline for projektet og antal medarbejdere.

Derudover kan admin oprette “subproject”, hvor der så vælges navn, beskrivelse, projekt id og det estimerede tidsforbrug for projektet. Der kan tilføjes tasks til et projekt. User har kun mulighed se informationer om projektet.

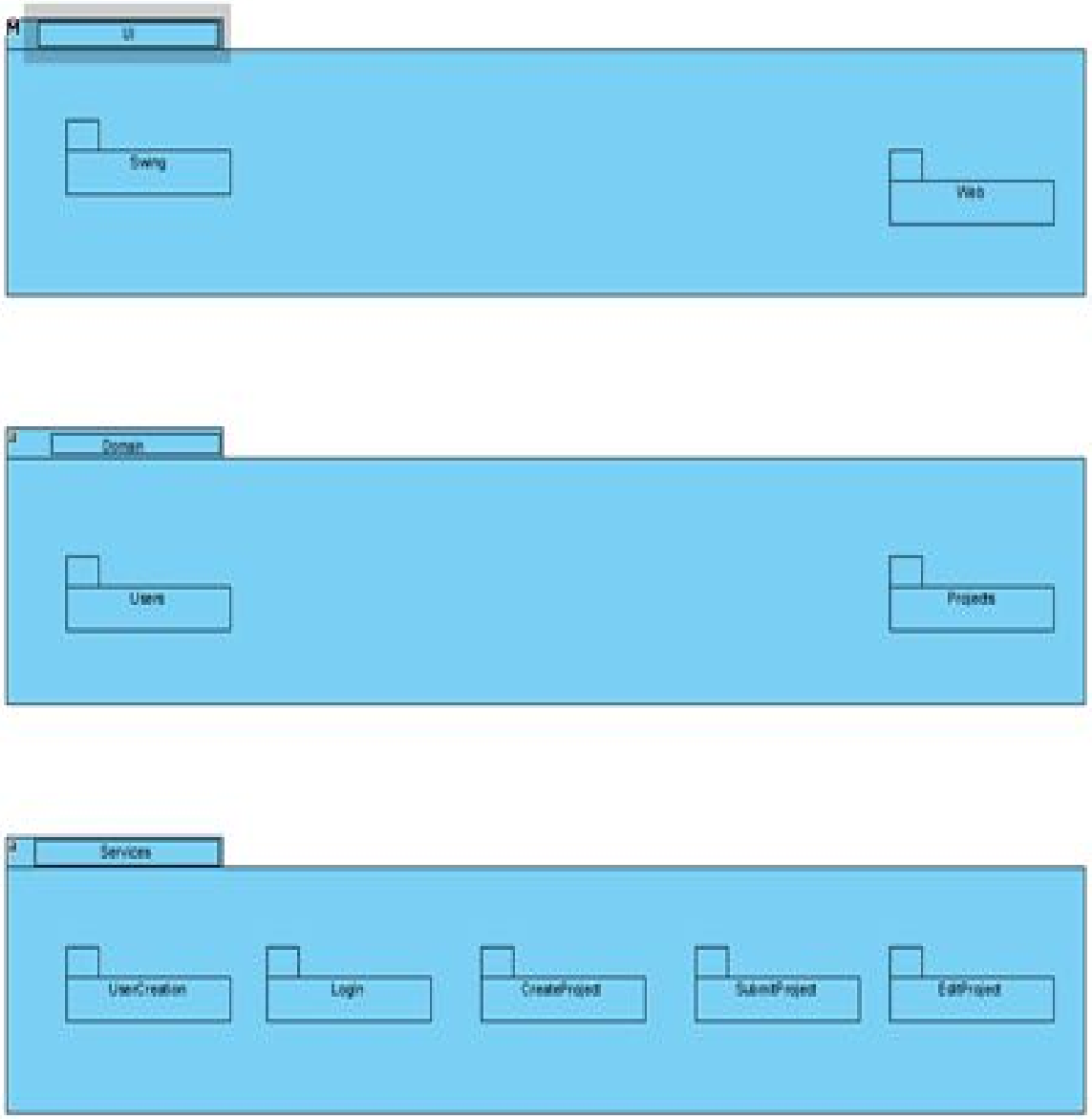
Design class diagram: (LJ)



Dette diagram viser de forskellige attributter og metoder som klasserne indeholder.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Pakkediagram: (logisk arkitektur) (LJ)



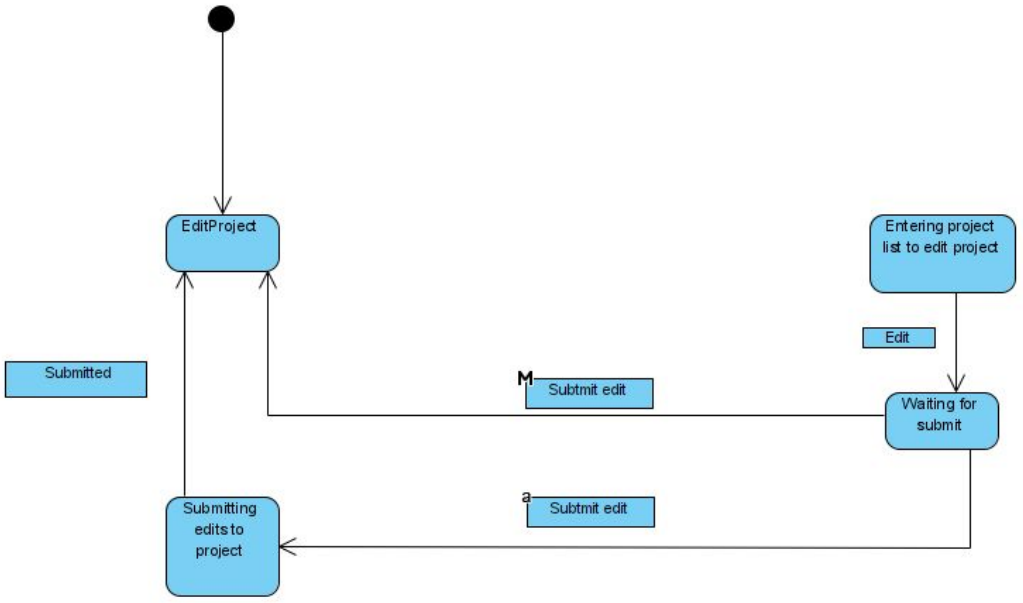
Dette pakkediagram viser strukturen af systemet.

I UI er der gjort brug af Swing der gør det muligt at systemet bliver fremvist på internettet (web)
I Domain er der gjort brug af MySQL der gør det muligt at gemme data fra Users og projects.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

I services er der UserCreation hvor brugeren har mulighed for at oprette sig som bruger, der er Login der gør at brugeren kan logge ind på systemet, bruger kan så være admin eller medarbejdere. Der er CreateProject der gør det muligt at oprette project. SubmitProject der gemmer projekter og EditProject der gør det muligt at redigere i projektet.

State machine diagram: (navigationsflow) // done (LJ)



Dette diagram viser begivenhederne når man som bruger trykker på EditProject, kommer man ind på projektet hvor man kan redigere projektet. Når brugeren er færdig med at redigere, kan brugeren trykke submit og ændringerne bliver så gemt.

Supplementary Specification (AH)

Supplementary Specification bruges til at give et overblik over hvilke funktionelle- og ikke funktionelle krav systemet skal opfylde. I dette tilfælde fokuseres der kun på ikke-funktionelle krav. Det var oplagt at bruge dele af FURPS+ model, hvor de mest relevante punkter blev udvalgt.

Usability:

Programmet skal være let at bruge for folk uden it-erfaring, og UI'et skal være brugervenligt. Systemet skal kunne bruges i forskellige browsere - Firefox, Google Chrome og Microsoft Edge. Det forventes, at mindst 90 % af alle mellem 18-60 har kompetencerne til at bruge programmet - dvs. oprette brugere, logge ind, oprette projekter, delprojekter og tasks.

Reliability:

Systemet forventes at være tilgængeligt 24 timer i døgnet. Det forventes, at systemet kører minimum 95 % af tiden.

Performance:

Programmet køres på en lokal database, så det er ikke afhængigt af eksterne datacentre. Der er få ting, der ville kunne få systemet til at crashe, fordi programmet er forholdsvis simpelt. Hvis MySQL Workbench er nede, kan programmet ikke køre. Antallet af brugere kræver også en vis hukommelse og processorkraft, hvilket kan give problemer, hvis trafikken er for høj. Det forventes ikke at blive et problem for systemet, fordi det kun er forholdsvis få ansatte i Alpha Solutions, der skal benytte programmet.

Supportability:

Først og fremmest er der en klar kørselsvejledning til hvad der kræves for at bruge programmet. Databasen oprettes lokalt, så det forventes, at Alpha Solutions selv skal kan vedligeholde den. Der skal være klar dokumentation for metoder og løsninger i programmet, hvilket kan findes i denne rapport. Dog er både det meste dokumentation samt UI skrevet på dansk, så det er nødvendigt, at brugerne af systemet forstår sproget.

Legal:

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

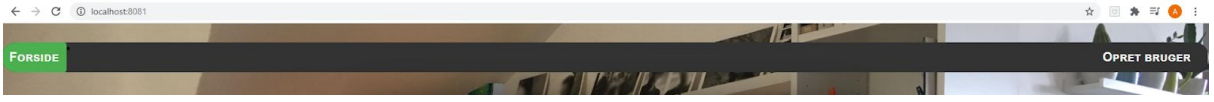
Systemet skal efterleve de strenge GDPR-regler. Desværre blev det senere tydeligt, at projektet nok ikke ville leve op til alle krav som fx. databeskyttelse, fordi der ikke var tid til at lægge stor fokus på sikkerhed, så det er en del af de ønskede funktionaliteter, som vi blev nødt til at undlade.

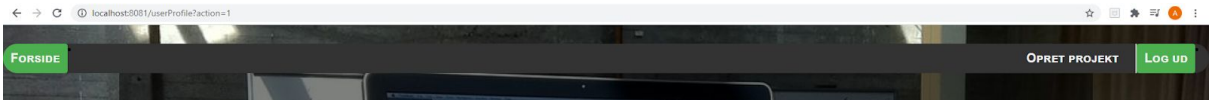
GUI - Golden rules og Gestaltlovene (AH)

Golden rules indeholder mange forslag og ideer til hvad en god brugergrænseflade er. En brugervenlig brugergrænseflade øger chance for, at brugeren kan løse dens opgaver og mål. I dette projektet har vi så vidt muligt prøvet at følge disse retningslinjer. Gestaltlovene beskriver, hvordan vi sanser og giver bud på hvordan hjernen opfatter sammenhænge i omverdenen. Disse love kan altså benyttes til at skabe mere et brugervenligt webdesign.

Konsekvent :

Vi har forsøgt at bruge nogenlunde ens design patterns på alle html-sider. Følgende billeder hvordan der bruges samme struktur og farver i forhold til headers.





Genveje:

Genveje er ofte en del af et brugervenligt GUI. Som det kan ses på ovenstående billede indeholder GUI'en en Forside-knap, så man altid kan komme tilbage til forsiden og dermed ikke behøver at ændre teksten i URL'en.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Fejlhåndtering:

Ideelt ville det være bedst, hvis der ikke opstår fejl eller opstår få fejl, men dette er ikke altid muligt. I tilfælde af fejl er det vigtigt, at brugeren bliver orienteret om, hvorfor fejlen opstår og hvordan problemet kan løses. I vores system har vi lavet en error html side, hvorpå man kan trykke på følgende links - Opret projekt, User Profile, log ud og Gå tilbage til forsiden.

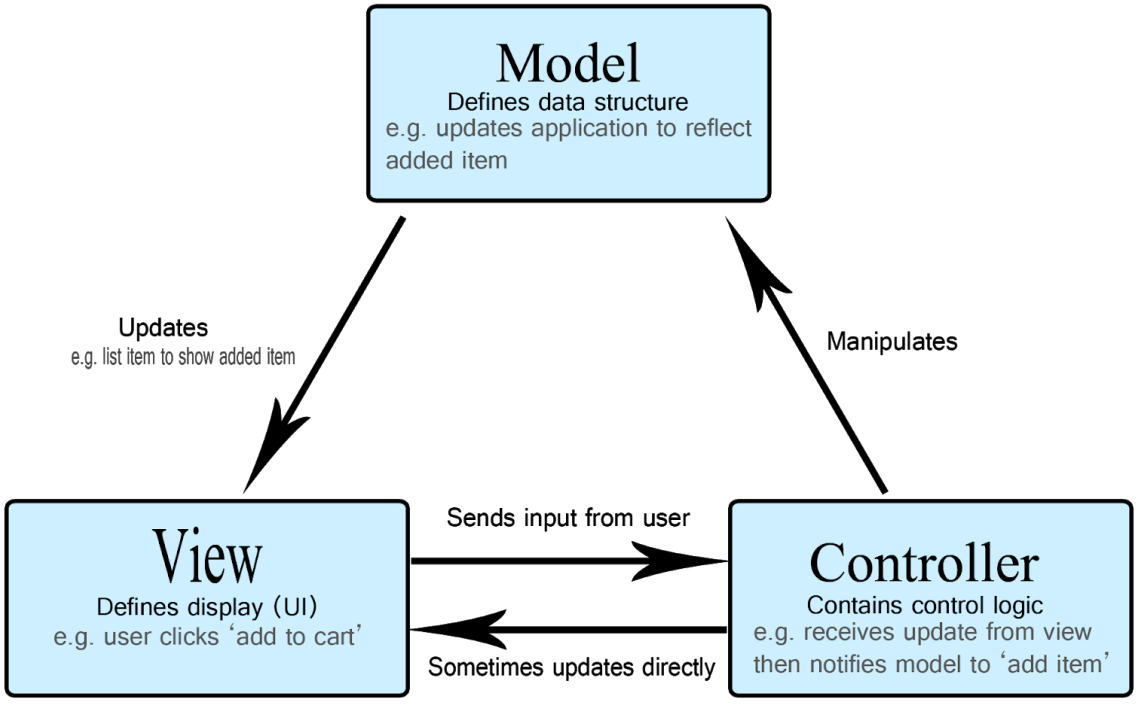
Programdokumentation

Design Patterns

Spring Model-View-Controller (AH)

Spring MVC er et design pattern, som bruges til at bygge webapplikationer. Program-strukturen deles op i models, views og controllers. I vores program ligger alle vores models admin, Project, Subproject, tasks og User i models-pakken. Vores view ligger i resources hvor alle html/ccs filer ligger. Controllers ligger i controllers-pakken.

Nedenstående model giver et overskueligt billede af, hvilken struktur dette design pattern har.



Her fremgår det, at **model** svarer på al data-relateret logik som brugeren arbejder med. **View** bliver brugt til alt logikken i forhold til brugergrænsefladen af applikationen. **Controller** er et slags grænseflade mellem **Model** og **View**, som behandler alt forretningslogik og alle forespørgsler.

GRASP (DC)

Grasp er en forkortelse for **General Responsibility Assignment Software Patterns**. Disse principper indeholder retningslinjer for, hvordan der skal uddelegeres ansvar til klasser og objekter i objekt-orienteret software design. Overordnet set er der 9 principper i GRASP, men for at rapporten ikke skulle blive for omfattende, har vi valgt at holde os til de mest relevante

Controller:

Controllernes opgaver er at håndtere forespørgsler, der bliver sendt af brugerne. Controlleren

Ideelt set skal koden bestå af flere controllere, der ikke har for meget ansvar eller for mange opgaver, hvor den hovedsageligt skal hente kode, såsom metoder, udefra. I princippet kunne vi have lavet én controller klasse, men netop dette skal helst undgås. I vores system er der en CreateController, LoginController, ProjectController, removeController og UserController. Herunder er eksempler på hvordan to @Getmapping-controllere er simple og kun har få opgaver.

```
@GetMapping("/createUser")
public String createUser(){

   return "createUser";
}

@GetMapping("/removeProject")
public String removeProject(){
   return ("removeProject");

}
```

Overordnet set gør vi brug af to forskellige notationer i vores controllere; Getmapping og Postmapping. Den første henter data ned fra dens angivne sti. Eksempelvis vil @GetMapping(“/removeProject”) returnere html-siden “removeProject.html”, når man indtaster /removeProject i URL’en.

High Cohesion:
High Cohesion betyder, at man giver de enkelte klasser så stort et formål og ansvar som muligt, så de skaber så meget “forståelse” som muligt. Det er dét, high cohesion betyder. Du vil som programmør være interesseret i at skabe så høj cohesion som muligt, da dine klasser vil være nemmere at vedligeholde. Ud over dette vil det være nemmere for dig at benytte klasserne sammen, da koden lettere vil blive genbrugt.

```
@PostMapping("/createSubProject")
public String createSubProject(Subproject subproject, HttpSession session){
    User user = (User) session.getAttribute( s: "login");
    int userid = user.getId();
    checkLogin(user);
    mapper.createSubProject(subproject, userid);
    int checkIfAdmin = user.getIsAdmin();
    if(checkIfAdmin == 1){
        return "redirect:/projects";
    } else{
        return "redirect:/userProfile";
    }
}
```

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Ovenfor ses et udklip af en controller, som står for at oprette delprojekter. Har har vi sørget for at gøre koden så forståelig som muligt ved at kalde variablerne navne, som direkte understreger, hvad det er der sker i koden. F.eks. “checkLogin”-metoden tjekker, om brugeren er logget ind. Koden er så at sige “idiotsikret” forståelsesmæssigt, da vi heller ikke har kommentarer med, da man direkte kan forstå, hvad der foregår. Dette vil vi mene er et eksempel på high cohesion.

Low Coupling:
Low coupling betyder derimod, at klasser skal afhænge så lidt af hinanden som muligt, men det vil også skal high cohesion, da man stadig vil tilegne klasserne så stort et formål og ansvar som muligt.
Hvis klasserne er for afhængige af hinanden, kan det skabe problemer, når det bliver ændret noget i én af dem. Der kan så at sige ske en kædereaktion, hvor hvis noget bliver ændret i den ene klasse, kan der opstå problemer i en anden klasse. Men på den anden side kan der specielt ved større projekter opstå problemer med for lav coupling, hvor klasserne og koderne generelt er så uafhængige af hinanden, at de vil blive for svære at administrere.

```
@GetMapping("/projects")
public String projects(Model model, HttpServletRequest servletRequest) {
    ArrayList<Project> projectList = mapper.getUserProjects();
    model.addAttribute( s: "project", projectList);
    HttpSession session = servletRequest.getSession();
    session.setAttribute( s: "projectList",projectList);
    return "projects";
}

@GetMapping("/project/{id}")
public String project(@PathVariable("id") int id, Model model, HttpServletRequest servletRequest){
    HttpSession httpSession = servletRequest.getSession();

    ArrayList<Project> projectList = (ArrayList<Project>) httpSession.getAttribute( s: "projectList");
    Project oneProject = null;
    for(Project project:projectList){
        if(project.getId()==id){
            oneProject = project;
        }
    }
    model.addAttribute( s: "project", oneProject);
    return "project";
}
```

I ovenstående udklip er et eksempel på high coupling, da controllerne er dybt afhængige af hinanden, da øverste controller tilføjer et projekt til sessionen, mens den nederste får ét fra sessionen og et id fra URL’en. Ergo hvis den ene øverste ikke fungerer som den skal, virker den nederste heller ikke.

Diverse benyttede værktøjer (AH)

Github:

Github er et versions kontrol system. Programmet gør det muligt at gemme ændringer i koden i et centralt repository. Dette gør samarbejde om udviklingen af software meget lettere, fordi man hurtigt kan hente en ny version af softwaren, lave ændringer og uploade en ny version. Softwareudviklere kan dermed få et godt overblik over alle ændringer, som er blevet foretaget. Hvis nogle ændringer skaber problemer i koden, kan tidligere versioner altid hentes, så det kan være lettere at se hvor fejl opstår.

Vi startede med at oprette vores program på Github og delte det til alle projektdeltagere. Vi aftale, at alle commits skulle indeholde klare beskrivelser, således at alle ændringer fremgik tydeligt. Generelt oplevede vi, at det var hurtigst at foretage pull, commit og push via Intellij, så da projektet skred frem, stoppede vi med at benytte Git Bash. Vi overvejede at oprette flere branches, men konkluderede, at det ikke var nødvendigt for et projekt i denne størrelsesorden. Vi oplevede enkelte merge-konflikter, men løste dem hurtigt.

Google Drive:

Google Drive er et Cloud-baseret program, der gør det muligt at gemme filer online og kunne tilgå dem overalt. En af de store fordele ved dette produkt er muligheden for at kunne redigere i et dokument samtidig. Alle projektmedlemmer har erfaring med at benytte Google Drive. Især Google Docs hjalp med at give gruppen et overblik over hvor langt gruppemedlemmer var med deres tildelte arbejdsopgaver. Dermed kunne det let ses om deadlines beskrevet i faseplanen i Unified Process blev overholdt.

Microsoft Teams:

Kommunikation mellem gruppemedlemmerne foregik hovedsageligt via Teams og Messenger. Vi havde ellers aftalt at mødes i alle hverdage, men corona-epidemien satte en stopper for dette, så vi var nødsaget til at snakke sammen virtuelt. Det er ærgerligt, fordi visse dele af projektet var lettere at lave, når vi var samlet. Dog var der også fordele ved at arbejde sammen via Teams. Skærmdeling er en funktion, som var meget brugbar og medførte, at Pair Programming fungerede næsten lige så godt via Teams som i virkeligheden.

Frameworks og templates

Spring:

Spring framework er en Java-platform, der giver hjælp til infrastrukturen i processen med at udvikle Java-applikationer.

Fordel

De fleste programmører der arbejder med udvikling af websites, ville mene at Spring er et utrolig godt værktøj, til at bygge hjemmesider med, uden brug af en server. Men det er de færreste der rent faktisk ved hvad der foregår bag skærmen bag skærmen. Her vil vi forklare fordelen ved at bruge spring og senere hen forklarer fordelen ved det.

Spring, har de fordele at den når vi kører vores kode bruger vi det der hedder en lokal server, hvilket gør det nemt og hurtigt at få en pæn hjemmeside i gang, sådan at man skal simpelt bare trykke på compile knappen i ens kode, for at få localhost til at køre på internettet.

En anden fordel, med Spring er at ingen andre har adgang til det du sidder og laver da det er serveren på ens egen computer som kan vise ens kode på localhost.

tredje fordel er også at Spring er meget nemt at arbejde med, eksempelvis hvis du har noget andet kørende i baggrunden som også kører på samme port som dit java program så fortæller java dig det. Så kan man ændre ens port så den i stedet for at kører det på localhost:8080 så kører den på en.

grunden

Grunden til vi har brugt er at, det er nemt at arbejde med. Ser vi eksempelvis på, den funktion med @Controller som vi bruger til at forklarer koden at det er her, vi gerne vil have vores get og post mappers skal ligge, det er en utrolig smart metode til at opdele vores kode. Det er ikke noget, vi kan med et standard set-up i et java program.

Hele flowet med at vi kan arbejde med koden, og rent faktisk se resultatet på en lokal server er utrolig lækker, og det er vores egen lille grund til vi synes Spring er utrolig godt værktøj at arbejde med.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Thymeleaf:

Thymeleaf er hjertet af forbindelsen mellem det data du har i databasen, og det data du gerne vil have ud på din hjemmeside. Dens formel er at lave den bro der skal til at få alt det information du vil have ud på din applikation. Hvilket er utrolig smart, hvis du godt vil sparer tid, og ikke lave din egne metoder til det, så skal du altså bruge det her thymeleaf til det.

fordel

Fordel med Thymeleaf er at, vi her kan overføre data ved hjælp af det vi kalder for model som blandt andet indeholder i Springframework biblioteket.
Ved hjælp af thymeleaf bruger vi nu, model til at indeholde alt dataen fra thymeleaf. Det data som Thymeleaf indeholder, er alt det information vi har fra databasen (mySQl), derfor er det utrolig smart at bruge thymeleaf til at hive data fra databasen og bruge det i ens kode, så man kan se de informationer man skal bruge på ens hjemmeside.

Så man kan helt bestemt sige, at thymeleaf har den fordel at vi sparer en masse tid, ved at bruge thymeleaf til at fange det data der ligger i databasen.

hvorfor har vi brugt det

Ser vi på userProfile.html i koden, kan vi se at thymeleaf meget nemt kan bruges til at hente data ned, og få det ud på hjemmesiden, ved at loope igennem indtil der ikke er flere rækker tilbage (linje 24 - 29). Det er et meget smart eksempel på hvordan thymeleaf kan blive brugt utrolig effektivt på bare 5 linjer kode. ‘

Et andet eksempel på thymeleaf er nemt, er når vi referere til thymeleaf. Det kan ses i starten af næsten alle html-filer i projektet. Vi skriver ‘th’ som et objekt til det her thymeleaf link, som er nemt og hurtigt og det er grunden til vi bruger thymeleaf - og det gør det endnu nemmere og vi sparer en masse tid, ved brug af Model objektet i forbindelse med thymeleaf fordi så kan man bruge det data vi har i thymeleaf ud i vores java-klasser.

eksempler.

Vores første eksempel er:

Her looper vi igennem vores objekter via thymeleaf.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

På linje 16, som er markeret, er et eksempel på hvordan vi kombinere 'th' med noget java kode.

beskrive hvad det betyder de eksempler

- Eksempel 1:
  Eksemplet er taget fra userProfile.html klassen.
  Hvis vi ser på første eksempel, kan vi at vi looper igennem vores objekter fra vores database. Loopet begynder, fra th:each på linje 24 og slutter lidt længere nede hvor denne: '</…' står på en bestemt linje. Grundet at loopet fylder en del, har vi valgt ikke at tage den med på billedet men i stedet valgt at fokusere på loopet.

- Eksempel 2:
  Dette eksempel er taget ud fra userProfile.html klassen.
  Ser vi på nummer 2 billede, er det koden til knappen "opret nyt projekt".

Det, de to knapper har tilfælles er at de begge indeholder thymeleaf, og måden det virker på i det her tilfælde er:

- Login metoden i loginControlleren har bla. model. Model (lige nu) er tom, idet login metoden returnerer userProfile, så alt det data som vi har i thymeleaf, det bliver ført videre til den th:action metode, så at vi kan bruge dataen til hvad vi har lyst til, uden at skulle bruge ekstra meget tid på at bygge vores egen metode, så gør thymeleaf det for os.

Test case (AS)

Når vi arbejder med test cases, fokusere vi på 2 ting.

1) Hvad vi gerne vil teste.
2) hvordan vi gerne vil teste.

Nogle andre ting vi gerne vil fokusere på når vi tester er:

3) Hvad vi forventer af resultat
4) Hvad resultat er i virkeligheden.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Testcase ID:	Test Case Description	Dependencies	TC ready for revior	Forventning af resultat:	Faktiske resultat:
TC 1: Creat User	index → opret bruger → udfyld oplysninger	x	Test lavet og succesfuldt	Created en bruger i databasen	bruger blev created
TC 2: Creat Project	login as admin → creat project→ udfyld oplysninger → gemmes i database	x	Tes lavet og succesfuld	Create projekt i database og på hjemmeside n	Create projekt i database og på hjemmeside n
TC 3: Get connection	Rammer index → tjekker forbindelse med database	x	test lavet og succesfuldt	Get connection til database	Connection var succesfuld

Database (AH)

Vores database indeholder fire tabeller – user, subprojects, projects og tasks. User behandler alt i forhold til brugeren – oprettelse af brugere og sletning af brugere og dermed muligheden for at kunne logge ind og bruge programmet. Projects og subprojects indeholder data om alle projekter, som er nødvendigt for at kunne oprette og slette projekter i systemet. Vi har brugt følgende statements: **Select, From, Where, Delete, Update og Insert Into**.

Select bruges til at hente data fra en tabel, som returnerer dataen som et result-set.

From bruges fx. til at vælge data eller slette data fra den “rigtige” tabel.

Where bruges til at angive en betingelse, når der hentes data fra en tabel. Når en angiven betingelse opfyldes, returneres en specifik værdi fra en tabel.

Delete bruges til at slette data fra en tabel.

Update bruges til at redigere data i en tabel.

Insert Into bruges til at indsætte ny information i en tabel i databasen.

Normalisering (AH)

Normalisering af database sikrer, at ændringer i databasen kan laves med mindst muligt indflydelse på det oprindelige system.

En af de store fordele ved normalisering er muligheden for at minimere redundant data, hvilket betyder at de samme informationer ikke skal ligge flere steder.

Vores database er normaliseret til 3. normalform. Der findes ikke felter uden for primærnøglen som er indbyrdes afhængige. Der er en primær nøgle i hver tabel, der entydigt bestemmer indholdet.

Database Integritet (AH)

Entity integrity:

Entity integrity omhandler primærnøgler, dataredundans. Ingen felter må være null. Årsagen til dette er at primary keys formål er at identificere unikke 'records'. Hvis to records i samme kolonne har en NULL værdi, bliver kolonne værdierne ikke betragtet som lige - to NULL værdier kan altså ikke være lige. Primary keys kan altså ikke have NULL værdier, fordi de ikke bliver sammenlignet med andre værdier.

Vores fire tabeller indeholder hver især en primærnøgle (primary keys) som er unik og not NULL – henholdsvis id, id, subId og taskID.

Referential integrity:

Referential integrity omhandler regler, som sikrer at data bliver gemt og brugt korrekt. Dette punkt er konceptet om fremmednøgler(foreign keys)**.** Foreign keys bruges til at sammenkæde tabeller. Det er meningen, at en foreign key i en tabel skal referere til en primary key i en anden tabel.

Vores projekt indeholder følgende foreign keys: projectID i tabellen subprojects, ProjectIDTask i tabellen tasks og userID i tabellen projects. Vi brugte fx foreign keys til at sørge for, at projekter kunne tilhøre en bruger eller tasks kunne tilhøre et projekt.

Domain integrity:

Data Definition Language og Data Manipulation Language

Data Definition Language (DDL): (AH)

I løbet af projektet har vi gjort brug af Data Definition Language og Data Manipulation Language. Data Definition Language består af SQL kommandoer som kan blive brugt til at definere database-skemaer. Det håndterer beskrivelser af

database-skemaer og bliver brugt til at oprette og modificere strukturen af database-objekter.

I forbindelse med oprettelsen af vores database har vi brugt følgende kommandoer: ALTER, CREATE, DROP og RENAME. Vi brugte ofte DROP når vi foretog ændringer i databasen, fordi SQL-scriptet skulle ændres, så alle projektmedlemmer kunne arbejde med databaser, der var ens. CREATE brugte vi i starten til at oprette nye tabeller i skemaet, men brugte den ikke så ofte senere, fordi vi hurtigt var klar over, at vi kun skulle bruge tabellerne projects, subprojects, tasks og user. ALTER benyttede vi løbende gennem hele projektet, fordi var vi lidt tvivl om foreign keys og hvilke attributter de forskellige klasser skulle have.

Data Manipulation Language (DML): (AH)

SQL kommandoerne fra DML manipulerer data i databasen. SELECT, INSERT, UPDATE, DELETE, MERGE, CALL, EXPLAIN PLAN og LOCK TABLE er statements fra DML.

SQL kommandoerne i DML har afgørende betydning for mange af de vigtigste funktioner i vores system. SELECT, UPDATE, DELETE og INSERT sørger for, at vores CRUD funktioner kan fungere.

```
public User createUser(User u) {
    Connection connection = DBManager.getConnection();
    String sqlstr = "INSERT INTO user (mail, password ) VALUES (?, ?);";
    PreparedStatement preparedStatement;
    User user = null;
    try {
```

Fx. I createUser oprettes en bruger og der indsættes værdier for mail og password i rows i databasen. VALUES betyder værdier, vi bruger spørgsmålstegne til, at lade os ufylde de værdier vi har i vores database.

```
public User deleteUser(int id) {
    Connection connection = DBManager.getConnection();
    String sqlRemove = "DELETE FROM user WHERE id = '?' ";
    PreparedStatement preparedStatement;
```

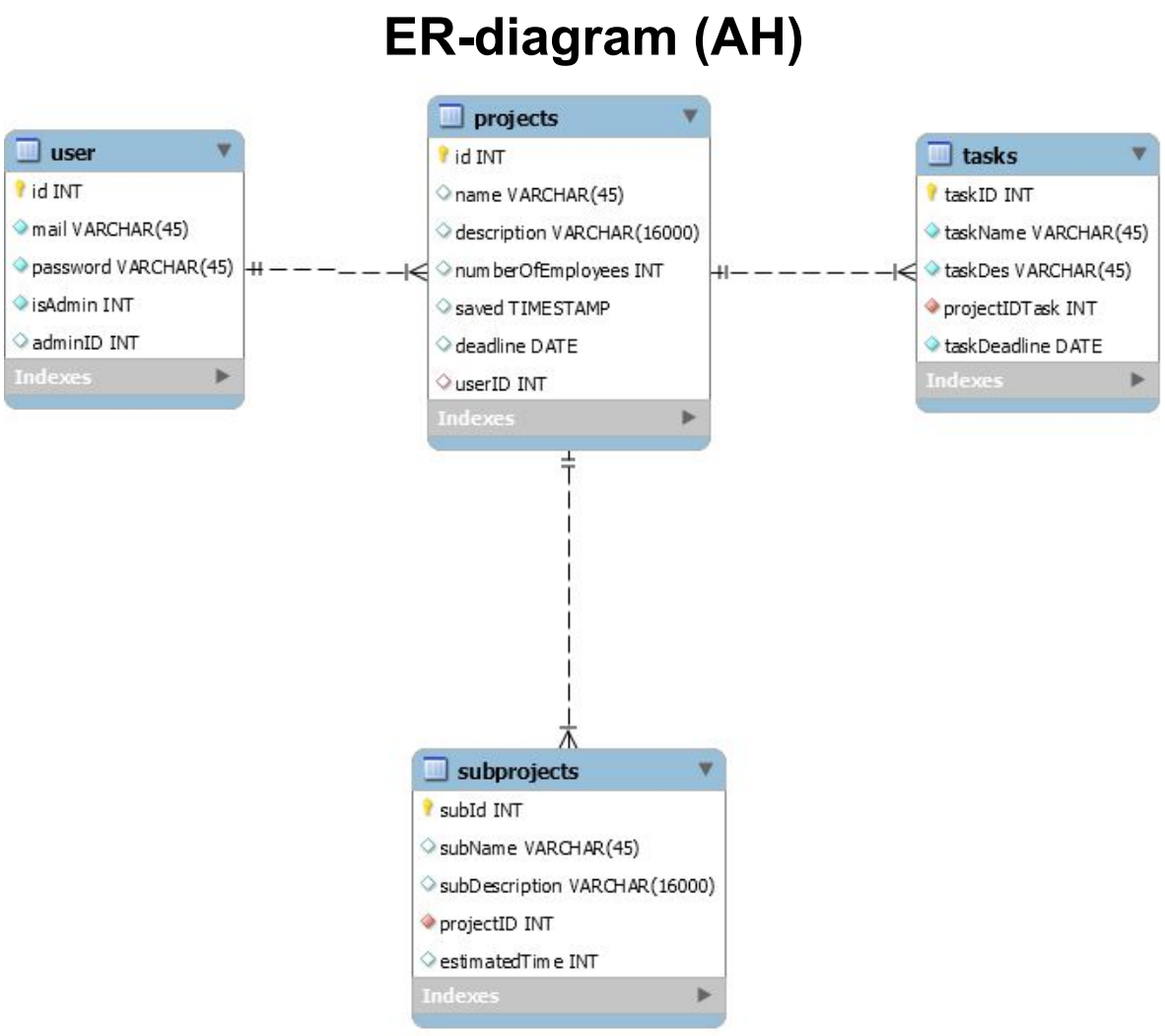
I deleteUser slettes en bruger fra user tabellen i databasen hvis id matcher.

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

```
public Project createProject(Project project, int userID) {
    Connection connection = DBManager.getConnection();
    String sqlstr = "INSERT INTO projects(name, description, numberOfEmployees, deadline, userID) VALUES(?, ?, ?, ?, ?)";
    PreparedStatement preparedStatement;

    try {
        preparedStatement = connection.prepareStatement(sqlstr);
```

I createProject indsættes description, numberOfEmployees, deadline og userID i tabellen projects.



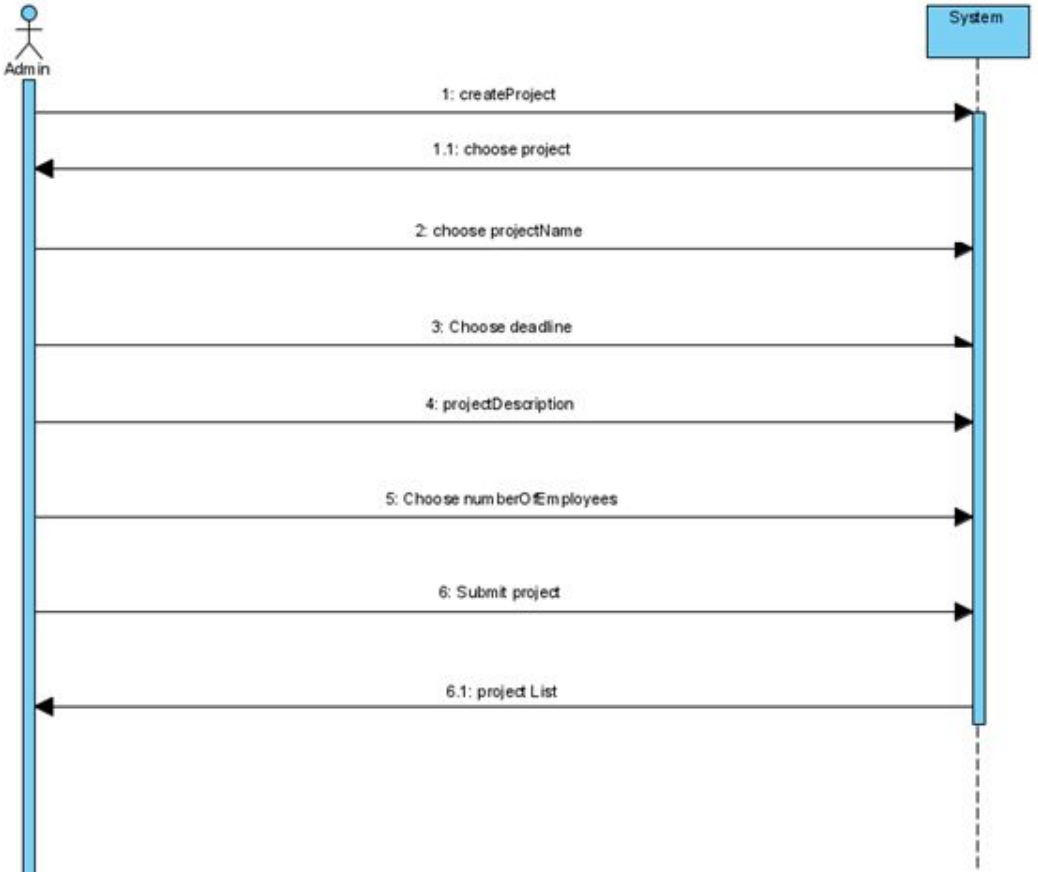
Ovenstående model er et Entity-Relationship diagram (ER diagram). Formålet med at lave et ER-diagram er at give et overblik over relationerne mellem enheder/objekter, der er gemt i databasen. Ved at vise enheder, attributter og relationerne illustrerer ER diagrammet den logiske struktur af en database. Det er en model af databasedesignet.

Diagrammet viser de 4 tabeller, der findes i databasen. Det viser også alle tabellernes kolonner og hvilken datatype værdierne i kolonnerne har. Fx. bruges

datatypen INT til isAdmin, fordi denne kolonne kun skal indeholde talværdier. Description har datatypen VARCHAR, fordi beskrivelsen kan være en blanding af tal og bogstaver. Værdierne i deadlines skal bruge datatypen DATE, fordi denne datatype kan vise datoer.

De gule nøgler viser hvilke kolonner, der er primary keys. De røde firkanter viser hvilke kolonner, der er foreign keys. Diagrammet viser også relationerne mellem tabellerne. Pilene viser, at der kan være flere tasks og subprojects til et project, men der kan ikke være flere projects til et subproject eller en task. Det kan også ses, at der kun kan være en bruger til flere projekter, hvilket ikke var efter hensigten. Man burde kunne tilføje flere medarbejdere til et projekt, men må implementeres senere.

System sequence diagram (LJ)



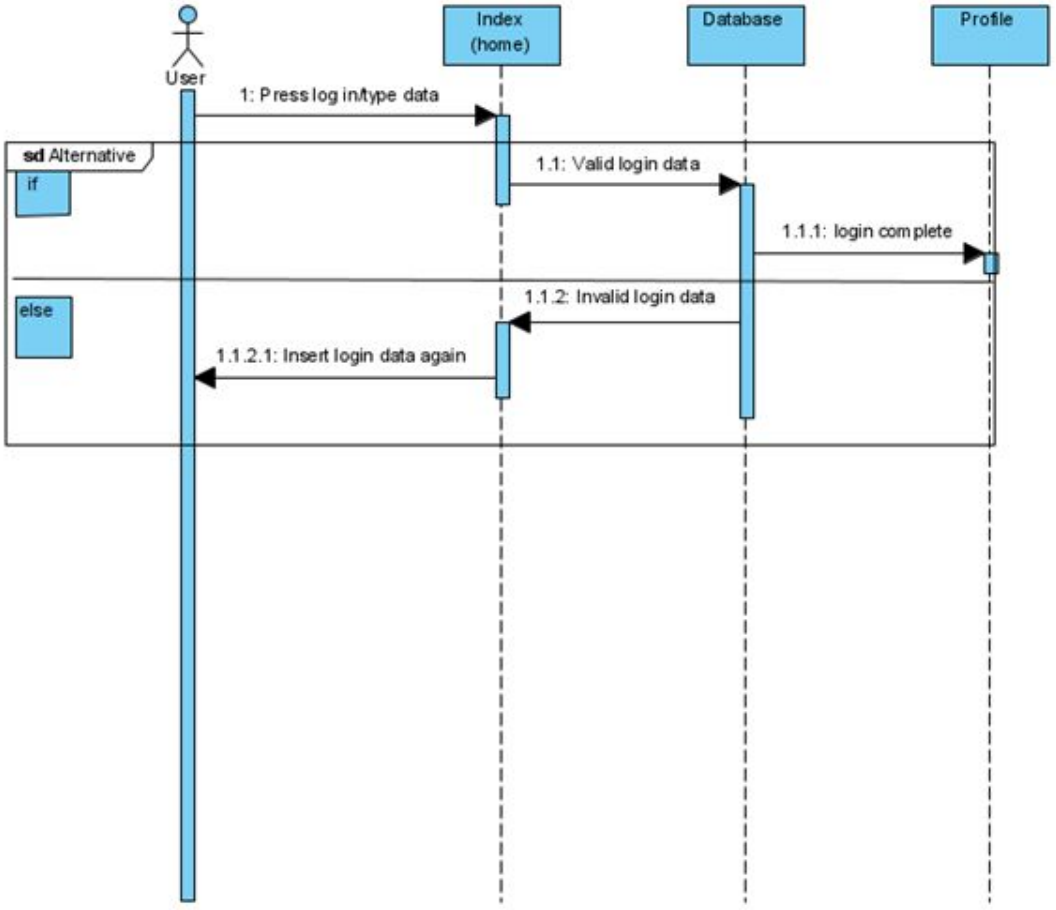
Dette diagram viser hvordan administrator laver et nyt projekt.

Diagrammet starter med at brugeren (administrator) opretter et nyt projekt. Systemet viser første trin af fire hvor brugeren skal indtaste et navn for det nye projekt. Nu skal

brugeren indtaste en dato som deadline for projektet, efterfulgt af en beskrivelse af selve projektet.

Til sidst skal brugeren indtaste det antal medarbejder, som skal være med i projektet. Når alle informationer om projektet er indtastet, trykker brugeren på opret projekt. Data sendes til databasen, og systemet puttet projektet i en projektliste og viser listen for brugeren.

Sequence diagram (LJ)



Dette diagram viser hvordan man som bruger logger ind på siden.

Brugeren starter på forsiden og logger ind på egen bruger. Data bliver sendt til databasen hvis data er korrekte bliver brugeren sendt videre til egen profil. Er data ikke korrekte bliver brugeren sendt tilbage til forsiden, og skal indtaste data igen

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

Dette diagram viser hvordan brugeren redigere i et projekt.

Avanceret kodebeskrivelse

Vi vil tage udgangspunkt i vores mest avanceret metode create project metode, metoden har tre steps: 1) creater et projekt. 2) creater subprojekt. 3) creater task.

Så nu, prøver vi at breake det down i små bider og går i dybden med hvad der egentlig sker. Vi har vores getMapping metode som i dette tilfælde tjekker vores user stadig er logget ind ved hjælp af session. Model.addAtrribute gør at, de værdier vi ser på linje 56 kan blive brugt i html. til sidst retunere vi egentlig bare en html-side og intet andet (som også kan ses på billedet)

```
51
52      @GetMapping("/createNewProject")
53  @   public String showCreateProject(Project project, Model model, HttpSession session) {// Model model fletter data, og tager dem fr
54          User theuser = (User) session.getAttribute( s: "login");
55          checkLogin(theuser);
56          model.addAttribute( s: "project", project);
57          return "createProject";
58      }
59
```

Vores getMapping metode, fører os videre til vores html side, hvor den læser linje for linje hvad der sker i koden. Det er her Thymeleaf blandt andet er interessant og vi vil derfor vælge at gå lidt i dybden og fortælle om hvad det gør her.

Meget simpelt sagt, bruges Thymeleaf til at gøre objekter fra domain klasser pæne på din webapplikation. Det vil sige, at man IKKE henter database informationer ved hjælp af thymeleaf men at man derimod bruger thymeleaf til at hente objekter fra din domain over på din hjemmeside.

```
ıoveController.java × UserController.java × CreateController.java × User.ja
1   <!DOCTYPE html>
2   <html lang="en" xmlns:th="http://www.thymeleaf.org">
3   <head>
4       <link rel="stylesheet" href="myCSS.css">
5       <link rel="stylesheet" href="profile.css">
6       <meta charset="UTF-8">
7       <title>Opret nyt projekt</title>
8   </head>
9
```

Billedet ovenover viser hvordan et thymeleaf objekt ser ud, og hvordan linket ser ud i thymeleaf billedet er taget fra create Projekt klassen,

I realiteten er det ikke nødvendigt at have thymeleaf for at hente de oplysninger om bruger og så videre vi skal bruge. Men det gør det alligevel en del nemmere at have med at gøre,

AS = Alexander Stohn. AH = Anders Hesselbjerg. DC = Daniel Causevic. LJ = Lano Jajal

når man har thymeleaf til rådighed, da det indeholder mange metoder, som er en kæmpe fordel som programmør at kunne udnytte. Måden, vi så henter det her data vi nu har liggende i thymeleaf, er ved hjælp af Model som er en del af det her java bibliotek vi har til rådighed.

Model bruges i vores Postmapping “createUser”, den indeholder så det her data vi har fra thymeleaf som vi så videre bringer til vores post mapping. Måden at den ved den at det er en postMapping den skal finde er ved at kigge på action og hvilken metode:

```
12    <div class="slider" />
13    <th:block th:include="/includes/header :: header"></th:block>
14    <div class="container">
15        <h3 id="header3">Opret nyt projekt</h3>
16        <!--- Her er et eksempel på en action knaps retur værdi -->
17        <form th:action="@{/createProject}" th:object="${project}" method="post">
18
```

Hvilket vi også kan se ovenpå.

Kørselsvejledning (AH)

Følgende guide forklarer hvordan programmet kan køres med Intellij som integreret udviklingsmiljø (IDE) og en database oprettet i MySQL Workbench.

Vi har bevidst valgt at oprette databasen uden en Cloud-løsning, fordi det ikke var et krav til projektet. Set i bakspejlet kunne det have været smart at bruge en Cloud-løsning fra AWS eller Oracle, da vi alle kunne have brugt den samme database. I stedet for blev løsningen at lægge SQL-scriptet ind i en fil i for sig selv. Hver gang der laves ændringer i databasen bruges forward engineering til at lave et nyt script. Scriptet kan dermed sendes sammen med andre ændringer i koden via Github. Dette var en lidt besværlig løsning, da der i løbet i projektet blev lavet mange ændringer i databasen.

Guide til MySQL Workbench og Intellij:

1. Installer Intellij
2. Installer MySQL Workbench
3. Åbn Intellij.
4. Klon koden fra https://github.com/Alex8943/Eksamen2020.git.
5. Åbn application.properties og sørg for, at db.user og db.password indeholder brugernavn og password, der matcher ens egen forbindelse.
6. Tryk Database -> new -> Data Source ->MySQL -> Indtast User og Password -> kopier url **jdbc:mysql://localhost:3306/projektoplysninger?serverTimezone=UTC->**

7. Tryk Apply.
8. Åbn MySQL Workbench.
9. Kopier SQL-scriptet fra Intellij eller Github og tryk execute. Dette sørger for at oprette tabeller, der er identiske med vores.
10. Opret brugere, projekter og delprojekter i databasen eller i browseren.
11. Kør programmet i Intellij
12. Åben browser og skriv http://localhost:8081/ i URL'en.

Hvis vejledningen følges, er det muligt at benytte systemet i browseren.

Guide til at bruge programmet:

1. Opret en bruger eller brug mail og password som allerede er indtastet i databasen.
2. Log ind som enten admin eller almindelig bruger.
3. Hvis man er logget ind som admin, har man muligheden for at oprette og slette projekter. Hvis man er logget ind som almindelig bruger, har man kun muligheden for at se projekterne.
4. Tryk opret projekt efter alle felter er blevet udfyldt.
5. Tryk opret delprojekt efter alle felter er blevet udfyldt.
6. Tryk opret task efter alle felter er blevet udfyldt.
7. Ønsker man at logge af, skal man trykke på Log ud.
8. Ønsker man at komme tilbage til forsiden, skal man trykke på Forside.

Konklusion (AH)

Opgaven fra Alpha Solutions og KEA handlede om at lave et kalkulationsværktøj, der kunne give et overblik over tidsforbruget for projekter.

Krav til virksomheden i form af Stakeholder analyse og Udvidet Risikoanalyse er opfyldt. Systemudviklingen er baseret på Unified Process, og vi lavede en fase plan som vi prøvede at følge.

Generelle krav til software konstruktionen er blevet opfyldt. Applikationen bruger Java klasser, Spring Controllere, HTML, CSS og Thymeleaf. Programmet er forbundet til en MySQL database og kører på en Tomcat server uden problemmer.

Rapporten indeholder programdokumentation af koden i form beskrivelser af særlige komplekse program-dele. Derudover indeholder den alle obligatoriske diagrammer, der giver overblik over systemet. Rapporten indeholder også dokumentation af test, beskrivelse af databasen og GUI'en er designet med hensyntagen til Gestaltlovene og The Golden Rules.

De vigtigste nøglefunktioner i systemet virker. Det er muligt at oprette projekter, delprojekter og tasks. Det er også muligt at redigere og slette disse. Det estimerede tidsforbrug for det enkelte projekt kan beregnes, hvis der indtastes informationer om de delprojekter, der tilhører hovedprojektet.

Der var desværre flere features, som vi ikke nåede at implementere. I risikoanalysen stod det klart, at de største risici ved projektgennemførelsen var forkert tidsestimering og begrænset erfaring med at bruge Spring Framework, SQL og Thymeleaf. Disse faktorer blev også udslagsgivende for, hvorfor vi blev tidspressede. Derudover blev kommunikationen mellem gruppemedlemmer ramt af, at det ikke var muligt at mødes på studiet pga. COVID-19.

I en fremtidig version skal systemet kunne tilføje flere brugere til et projekt. En 'almindelig bruger' skal have mulighed for at indtaste hvor mange timer vedkommende har brugt på tasks og delprojekter.

Vi kan konkludere, at vi grundet manglende kommunikation i gruppen ikke kunne nå det vi satte os for. Vi fik alligevel opfyldt kravene til opgaven. De opgaver vi ikke fik løst har vi alle overvejet løsninger til.

Bilag

Litteraturliste

https://da.wikipedia.org/wiki/Unified_Process

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

https://www.3schools.com/

https://www.techopedia.com/

APPLYING UML and PATTERNS - Craig Larman

Building Java Programs - Stuart Reges, Marty Stepp

https://www.geeksforgeeks.org/

https://sagaratechnology.medium.com/