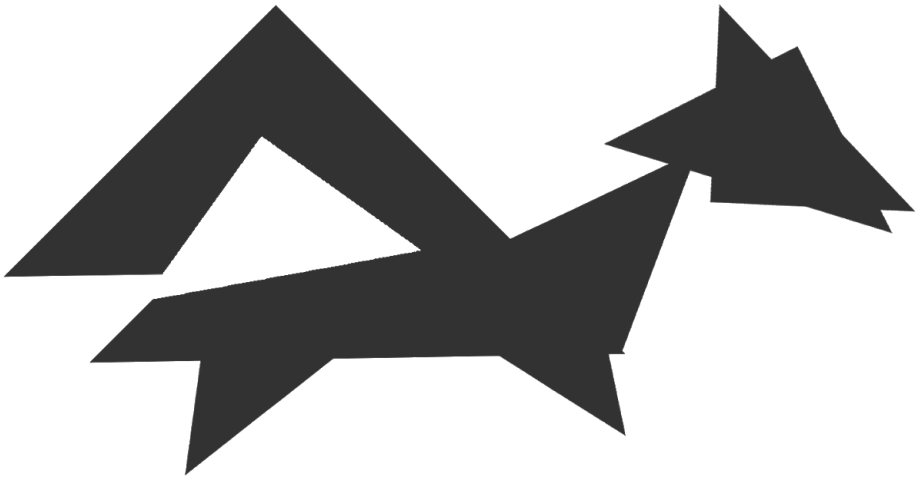
Eksamensprojekt

Game Master's Tools



Zealand Erhvervsakademi - Forår 2019

Rasmus Drotved | rasm781y@edu.easj.dk | GitMaster, developer
Laura Vilen | laur165n@edu.easj.dk | ScrumMaster, sekretær, developer
Kristian Hornbøll | kris195y@edu.easj.dk | Formidlingsansvarlig, developer

Indholdsfortegnelse

Problemformulering

Hvordan kan agil systemudvikling (Scrum) og iterativt arbejde, C#-programmering og relationelle databaser bruges til udvikling og implementering af et IT-system, der kan hjælpe Game Masters med forberedelse, styring og administration af D&D spil? For mere information og forklaring af D&D se afsnit: Projektetablering - Projektbeskrivelse.

Hvordan kan vi lave en database der kan indeholde alle de nødvendige data, som en bruger skal bruge til at styre et D&D spil?

Hvordan kan man lave en god og overskuelig brugerflade i en UWP app, ved brug af MVVM og xaml, samt få data fra databasen repræsenteret i vores UI?

Indledning

I denne rapport vil beskrive vores arbejde med programmet Game Masters Tools, samt besvare ovenstående problemformulering. Vi vil beskrive ideen til og formålet med vores program, samt give læseren den nødvendige baggrundsviden til at forstå konceptet.
Rapporten indeholder desuden en forretningsanalyse af det marked programmet kunne blive solgt på.

Vi vil forklare arbejdsmetoden Scrum, og hvordan vi har benyttet denne agile process, samt beskrive de kodeværktøjer og teknologier vi har benyttet i udviklingen.
Derudover vil vi forklare hvordan vi har oprettet en database, hvordan vi tilgår denne via en webservice, samt hvordan denne er forbundet til vores solution og brugerflade.

Link til vores Scrum-board
Vores database og web service i Azure
Vores projekt på GitHub.com - Vores repository er privat, så skal man have adgang bedes man sende mail til laur165n@edu.easj.dk. Vi har dog vedlagt projektet som zip hentet direkte fra GitHub.

Vil man teste programmet med en bruger der har noget data tilknyttet kan man logge ind med:

Brugernavn	Adgangskode
Bobby	123456
Hans	12345678

Projektetablering

Projektbeskrivelse

Af Laura Vilen

I dette projekt arbejder vi med at udvikle et program til styring af Tabletop RPG'er, mere specifikt Dungeons & Dragons.
For at forstå programmets formål er det nødvendigt at have en basal forståelse af, hvad Dungeons & Dragons er, hvordan det foregår, og hvem der er involveret i aktiviteten.

Hvad er Dungeons & Dragons?

Dungeons & Dragons - blot forkortet D&D - er et tabletop RPG. RPG står for roleplaying game og er en type spil eller computerspil, hvor man spiller en karakter der er vidt forskellig fra en selv og lever sig ind i rollen som denne karakter. Tabletop refererer til den måde aktiviteten foregår, at man sidder en gruppe rundt om et bord og spiller, modsat Live Rollespil hvor man ofte klæder sig ud, har rekvisitter og går rundt under aktiviteten.
Dungeons & Dragons er ét ud af mange forskellige tabletop RPG'er, men nok det mest kendte og populære. Typisk er scenariet sat i en eventyr- og middelalderverden, og spillet har et fast regelsæt, mekanikker og elementer, som deltagerne benytter.

Hvordan foregår Dungeons & Dragons?

Spillet foregår som nævnt som en siddende aktivitet for de deltagende. Man medbringer typisk ikke elektroniske genstande som computer, telefon eller tablet, medmindre disse fungerer som noteværktøjer. Man vil typisk blot benytte sig af papir og blyant, så der ikke forekommer unødige distraktioner fra spillet.
Et vigtigt udtryk i Dungeons & Dragons er udtrykket: "Kooperativt Narrativ". Dette beskriver rigtig godt aktiviteten, da de deltagende ved bordet samarbejder om at fortælle og udvikle en historie ved hjælp af spilmekanikker og rollespil.

Hvem deltager i aktiviteten?

I Dungeons & Dragons vil man typisk være 4-6 deltagere, selv om det er muligt at være færre eller flere. Disse deltagere vil have en af to roller: Spiller eller Game Master. En spiller medbringer én karakter - også kaldet en Player Character eller blot PC - som de styrer og

rollespiller. Al interaktion og aktivitet i spillet foregår med og gennem denne ene karakter, og man forventes kun at handle på viden som denne karakter har, også selv om spilleren selv har større viden om eller kendskab til dele af spillet. De fleste deltagere rundt om bordet vil være spillere.

Modsat er der oftest kun én Game Master i et spil - selv om om det er muligt at have flere især ved store grupper af spillere - og Game Masterens rolle er meget forskellig fra spillerens. Game Masteren fungerer som regeldommer og organisator. De formidler reglerne, beskriver omgivelserne i spillet og styrer alle fjender og NPC'er.
Det er altså Game Masterens opgave at sætte scenen for eventyret samt give spillerne ting at reagere på og interagere med i spillet.

Denne rollefordeling betyder oftest, at Game Masteren skal lægge meget mere arbejde i spillet end resten af gruppen: De skal sørge for at planlægge beskrivelser, kort over verdenen, sammenstød med fjender, interessante NPC'er, holde styr på og læse op på reglerne og holde styr på spillere og taletid.

Dette projekt er derfor dedikeret til at gøre planlægning og styring af D&D nemmere for Game Masters. Programmet skal gøre det simpelt at planlægge og skrive indhold til sine eventyr som forberedelse til spilgange samt gøre det nemt og overskueligt at finde den relevante information frem under spilgangen.

Ønsker læseren mere information om hvad D&D er og hvordan det spilles, henvises der til materiale fra Wizards of the Coast, som har udviklet spillet.
Regler og information kan findes i deres bøger eller på deres hjemmeside: DnDBeyond.com, hvor de bl.a. publicerer afsnit fra bøgerne, eventyr og regler online.

Introduktion til D&D: https://www.dndbeyond.com/sources/basic-rules/introduction
Oversigt over basisregler og mekanikker: https://www.dndbeyond.com/sources/basic-rules

Nogle gode termer at kunne:

- Game Master - den der styrer spillet og fortællingen.
- PC - står for "player character" og er en karakter i spillet styret af en spiller.
- NPC - står for "non-player character" og er en karakter i spillet der ikke styres af en spiller. Ofte har de en bestemt og afgrænset rolle i historien som kroejeren eller vagtsoldaten.
- Eventyr - kan også kaldes en "Quest" og er et afsnit af historien, som har et bestemt tema, mål eller en afgrænset handling.
- Kampagne - en sammenhængende historie opbygget af flere eventyr, som ofte har et overordnet tema. En kampagne kan vare alt fra uger til år at afvikle.

Værktøjer

Software (Visual Studio 2017, Trello.com, Google Drive, Google Docs, Lucidchart.com)
Hardware (Computer, kamera)

Studierum på skolen med whiteboard og skærm til fælles arbejde.

Gruppekontrakt

Fremmøde

- Er du her ikke, har teamet lov til at tage beslutninger uden dig
- Er du forsinket eller forhindret i at møde op, skal gruppen have besked

Faste mødetider på skolen

- Tirsdag - fredag
- 9.10 - 13.45
- Alt udover aftales efter behov

Hjemmearbejde

- Arbejdes der på projektet hjemmefra, skal det aftales på forhånd
- Der arbejdes ikke hjemmefra på noget, som gruppen ikke er enige om

GitHub

- Kun én fra gruppen har lov til at merge pull-requests
- Der pushes IKKE i master branch, medmindre alle har gennemset ændringen

Kode

- Vi skriver i C#, xaml, sql
- Vi bruger Visual Studio og Github
- Der skrives kommentarer til koden
- Vi aftaler fælles navngivning for klasser mv.
- Xaml-elementer skal navngives for overskuelighed

Inception

Inception deck[1]

Hvorfor er vi her - fra idé til virkelighed

Formålet er at lette arbejdsbyrden og mindske adgangsbarrierer for nye såvel som allerede etablerede Game Masters. Dette vil vi gøre ved at samle informationer i en app, som kan hjælpe med at gøre det hurtigt og nemt at slå noget op, følge et plot og samtidig formidle en historie til sine medspillere.
Der er mange hjælpemidler til rådighed på nettet og i bøger, men vi vil gerne prøve at få meget af det samlet i én pakke.

Mød teamet

Rasmus, Laura, Kristian

Gruppen har valgt dette projekt, da vi alle har større eller mindre kendskab til Dungeons & Dragons. To af gruppens medlemmer er selv Game Masters, og den sidste er spiller fra tid til anden. Dermed har vi alle kendskab til hvordan spillet foregår samt motivation for at udvikle en løsning på den problematik, vi selv sidder med til daglig. Derudover har vi andre venner og bekendte, som vores program ville kunne gavne.
For at sikre at vi som team udvikler det bedst mulige produkt, har vi derfor fået en god ven af gruppen - Kasper Gregersen, som har stor erfaring med D&D, tabletop RPG og rollespil og er ansat i rollespilsbutikken Hack&Slash i Holbæk - til at agere som product owner.

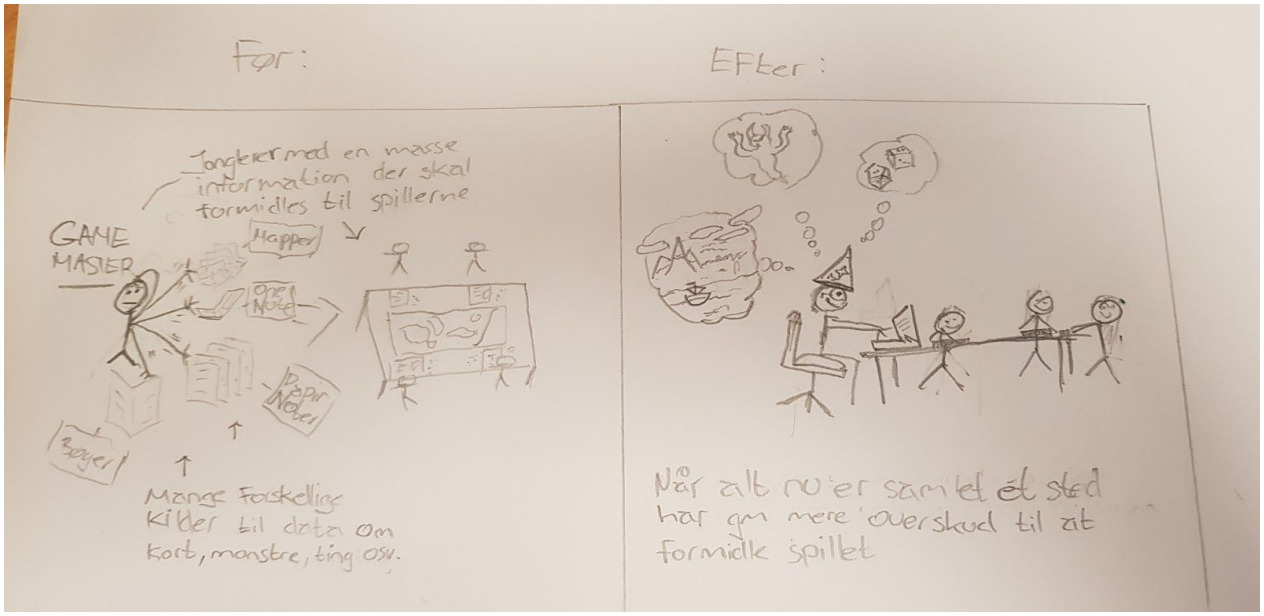
Vi har i teamet forskellige styrker og svagheder, men som som samlet gruppe mener vi at have gode kompetencer og kundskaber til at kunne fremstille et grundlæggende og brugbart produkt. Vi har alle været involverede i både analyse, design, implementing og test.

Problemet & løsningen

Under projektetableringen har vi udarbejdet nedenstående rich picture, som skal illustrere hvordan situationen er nu for en Game Master, og hvordan det kunne blive ved brug af vores program.

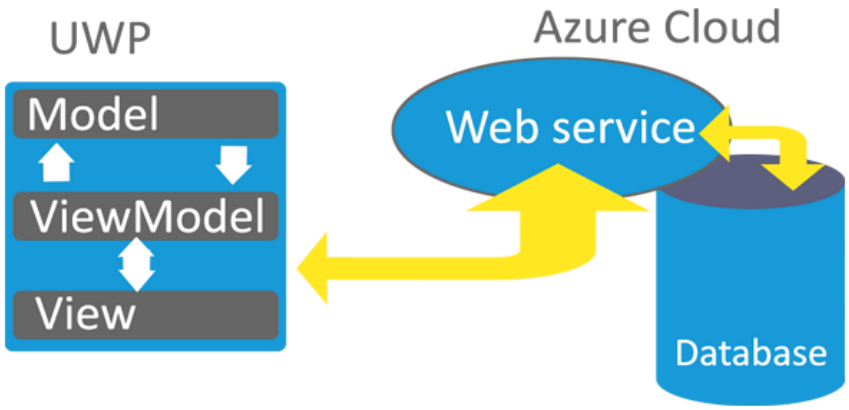
[1] Rasmusson, Jonathan: The Agile Samurai: How Agile Masters Deliver Great Software, 2010

“Før”-billedet viser den problematik der kan opstå, når Game Masteren skal bruge tid på at lede efter oplysninger i forskellige opslagsværker. “Efter”-billedet viser en Game Master, der i stedet kan koncentrere sig om spillet og spillerne, når alle oplysninger er lige ved hånden.



Vi har forestillet os en god gammeldags Game Master, der sidder foran sin gruppe med noter, diverse mapper og bøger samt opslagsværk omkring sig - billedet sætter hurtigt gang i tanker om stress. Det er så her vores produkt kommer ind i form af en app, som eventuelt kunne køres på en tablet. Et par swipes senere, og historien fortsætter ufortrødent.

Arkitekturen

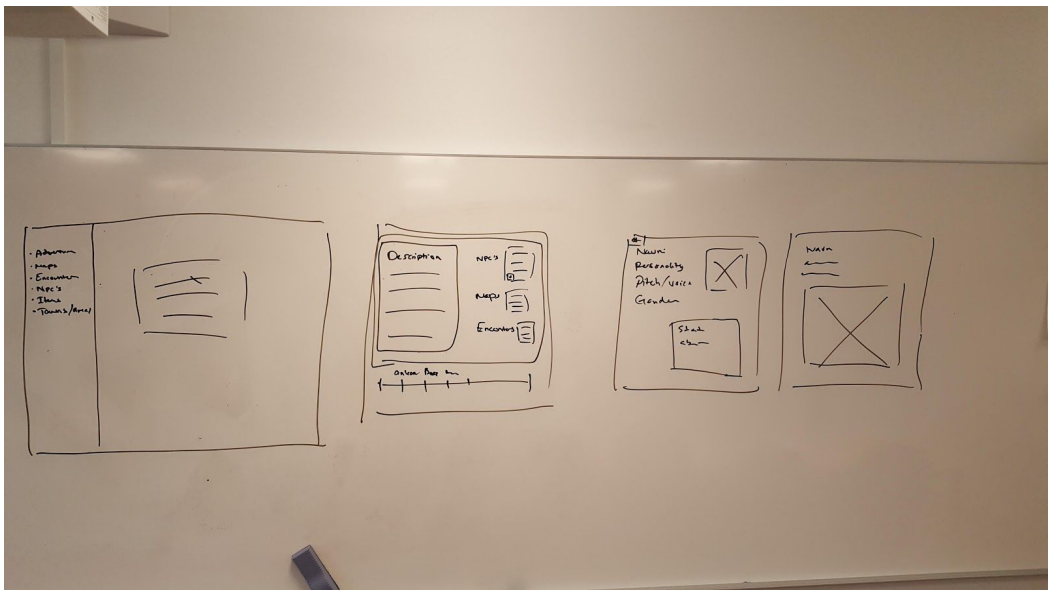


Projektet er lavet som et MVVM-projekt, dvs. Model-View-ViewModel. Det grundlæggende indenfor MVVM er at holde domænet med modelklasser og adfærd adskilt fra den grafiske brugergrænseflade, så det at teste produktet vil være enklere. Derudover vil man kunne genbruge den bagvedliggende logik i andre former for platforme - altså er der uafhængighed mellem lagene. Ud over de tre lag kommer også et fjerde lag i form af et persistens-lag, der gør at man kan gemme og hente sine data i en database.

Ud fra vores rich picture kan der udledes nogle objekter fx i form af Campaign, Item, PC(player character), NPC(non player character), Maps, Locations samt måske Chapter, så campaigns kan opdeles yderligere. Derudover vil vi gerne lave et login system, så en brugerklasse skal vi også have (User). Ud fra de oplysninger får vi oprettet en model der viser klasserne i vores domæne, og hvordan deres forhold til hinanden er.

User Interface

I vores opstartsfase overvejede vi forskellige designs. Her er et par mockups fra begyndelsen.



Stakeholders

Udviklingsteam
Product Owner
Vejledere (lærere)

Risks

Hvad kan forhindre os i at nå målet? Uventede ting som fx sygdom kunne være en stor risiko, da vi kun er tre i vores gruppe. Ud over det ferie (påske) midt i projekt perioden.
Tab af data, forventninger til det endelige projekt - måske har vi haft forskellige idéer om det færdige produkt. Vi kan dog sige at vi er mere eller mindre på samme side efter at have diskuteret i vores opstartsfase - why are we here.

How to deliver

Vi arbejder i Scrum, som er en agile arbejdsmåde. Vi inddeler vores arbejdstid i mindre dele kaldet sprints og arbejder så hver især på vores del, sørger hele tiden for at være på samme side og kommunikere, og når et sprint er gået, sammenfattes hvad der er nået og hvad der mangler, og så starter vi forfra med et nyt sprint. I hvert sprint laves analyse, design, implementering og test, så vi efter hver gang kan fremvise noget, som rent faktisk kan kaldes færdigt.

What to give

Vi skal have sat en realistisk tidsramme samt et scope for projektet, der kan nås.
Kvaliteten er vigtig for os, men inden for tid og ramme vi vil gerne levere et produkt, som kan vises frem.

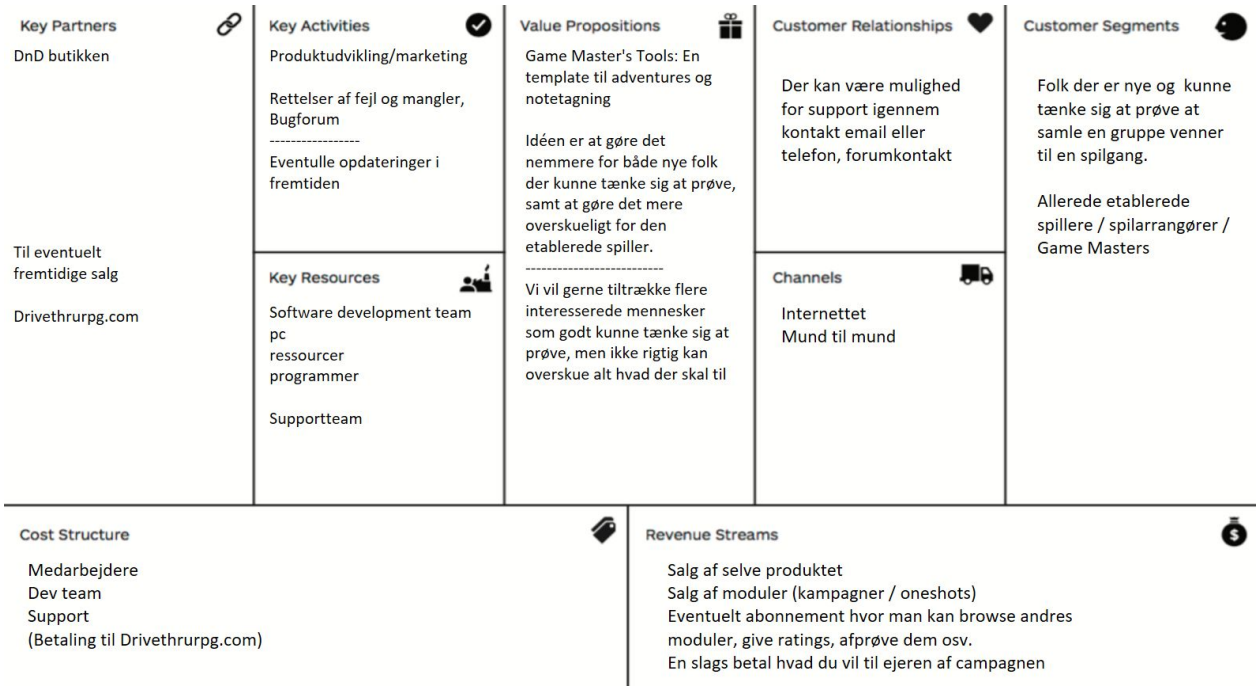
Virksomhedsanalyse

Af Kristian, Laura og Rasmus

Da vi arbejder med vores egen idé som et software development team/virksomhed og ikke for at lave et produkt til en anden virksomhed, vil vores forretningsanalyse være baseret på det.
Vi vil bruge SWOT analyse til at undersøge teamets styrker og svagheder, Porter's Five Forces til at analysere konkurrencen på markedet og Business Canvas til at undersøge forretningsstrukturen for den type virksomhed, der ville udvikle et program som vores.

Vi opstiller et business canvas for at udvikle en forretningsmodel til vores projekt. Den viser nogle nøglepunkter i form af værdi over for kunder og partnere samt hvilke aktiviteter der skal udføres og hvad en indtjeningsmetode kan være.

Business Canvas



Problemet er reelt. Vi kender det selv og har hørt det tit i miljøet omkring D&D, så kundegrundlaget burde være der. Derudover vil vi også gerne tiltrække nye ved at gøre det enklere at starte. Forretningsmodellen ville være enten en lille valgfri brugerbetaling ved benyttelse af appen og/eller ved at tage en lille bid ved deling/salg af kampagner brugere imellem.

SWOT

Med swot analyserer vi vores egne egenskaber som team/virksomhed. Med swot kommer vi hele vejen rundt, styrker og svagheder kommer frem i lyset og kan derfor nemmere håndteres indbyrdes.

Strengths

- Godt team
- Vi er ikke bundet af konventioner
- Vi er selvkørende
- Kender problemstillingen og forstår produktets formål

Weaknesses

- Vi er nye og har ikke et omdømme
- Mangel på ansatte
- Vi kender ikke så meget til markedet

Opportunities

- Der findes ikke noget andet program/web page, der samler brugerskabte adventures og modules
- De fleste hjemmesider tilbyder dele af det program vi vil lave eller enkeltstående templates, ikke det hele samlet

Threats

- De etablerede virksomheder, som har et stærkt brand
- Den enorme mængde af lignende produkter, der er tilgængelig via nettet

Vi har et rigtig godt fundament i forhold til produktet, da vi kender selve spillet med dets fordele og ulemper og let kan sætte os ind i eventuelle problemer. Vi er dog nye i udviklings-gamet og har ingen erfaring.

Porter's 5 forces

Med Porters forsøger vi at danne os et overblik over situationen rent konkurrencemæssigt blandt konkurrerende produkter. Hvilke ydre kræfter er på spil, hvordan kommer vi ind i miljøet med vores produkt, er området mættet, eller er der plads til flere produkter?

Threat of New Entrants

- High: Det er netbaseret og billigt at fremstille

Bargaining Power of Buyers

- Medium: Vi laver det pay-what-you-want, men bør stadig lytte til folks ønsker. Brugervenlighed er essentielt.

Bargaining Power of Suppliers

- Low: Der er mange andre hjemmesider, hvor man kan sælge produktet. Det er også en mulighed at oprette egen hjemmeside til salget.

Threat of Substitute Products

- Medium: Der findes en del lignende produkter, men ingen der er helt det samme. Der er også folk der klarer sig med papir-noter og OneNote.

Competition: Medium

Vi har vurderet markedskonkurrencen til at være medium, da det er nemt for køber at finde alternative produkter. Til gengæld tilbyder vi et unikt produkt, som samler mange af de templates/programmer, der findes spredt på nettet, i ét produkt.

Med vores kendskab og de voksende krav til og muligheder for et produkt som vores mener vi at vi er ret godt stillet trods konkurrencen.

Vision Statement

Af Kristian, Laura og Rasmus

Med et vision statement ender vi med en liste over needs og features, som vores produkt meget gerne skulle indeholde. Dem kommer vi frem til ved systematisk at analysere problemet: hvem har problemet, hvem vil bruge produktet og hvordan, hvad der allerede findes på markedet og hvad vi kan gøre bedre.

Problem statement

Problemet	At have for mange steder at skulle kigge efter information
Har indflydelse på	Game Masters evne til at køre spillet effektivt
Effekt af problemet	Uorganiseret information og mindsket overblik
Mulig løsning	At samle så meget information som muligt ét sted og gøre den overskuelig

Product position statement

Til	Game Masters
Som	Let mister overblik, har for meget at holde styr på eller ønsker mere overskud
Produktet	Er et Information Management System
Der	Samler al information ét sted og gør den mere overskuelig
Modsat	Bøger, papir og DndBeyond
Vores produkt	Øger produktiviteten for Game Masters

Stakeholder summary

Kristian, Laura og Rasmus	Os (udviklingsteam og delvis produktejer)	Udvikler og designer systemet
Kasper	Product owner	Kommer med feedback til systemet (Hjælp med salg/marketing)

User environment

Antal personer involveret i opgaven:
Der er som regel én person, der er Game Master og planlægger spilgang og eventyr.
Deres opgave er delt op i forberedelse og afholdelse af spil.

Opgavecyklus:
Planlægning af eventyr kan tage alt fra et par dage til et par måneder.
En spilgang afvikles typisk på 2-10 timer.
Tiden brugt på planlægning vil som udgangspunkt ikke ændre sig, men under afholdelse af spilgang vil Game Master have mere tid til at fokusere på selve spillet end på værktøjer og opslag.

Begrænsninger:
Umiddelbart ingen, men spillet foregår som regel indendørs. Adgang til internet er anbefalet.

Platforme der bruges i dag:
Bøger, papir, DndBeyond website, online/elektroniske note-værktøjer.

Fremtidige platforme:
Computer, tablet mv. (platforme med internetadgang).
I fremtiden vil det måske blive muligt at få adgang til DndBeyonds database gennem vores applikation.

Needs and features

Behov	**Prioritet**	**Features**	**Beskrivelse**
Timeline	Medium	Organiserings-værktøj	En tidslinje over eventyr/hændelser, som skal gøre det nemmere at holde styr på disse for brugeren
Dice roller	Lav	Ekstra feature	En virtuel terninge-kaster, hvis brugeren ikke ønsker at rulle manuelt
Tilføjelse og sletning af informationer	Meget høj	Informations-værktøj	Det er vigtigt at brugeren kan indtaste egne informationer i programmet
Gemme i database (azure)	Meget høj	Filhåndtering	Brugeren skal kunne gemme sine data i en database og dermed kunne tilgå informationerne overalt
Gemme i fil (lokalt)	Høj	Filhåndtering	At kunne gemme lokalt hvis der ikke er noget internet tilgængeligt
Initiative tracker	Lav	Ekstra feature	Et redskab til at holde styr på rækkefølgen af spillernes ture under spillet.
Dele kampagner med andre brugere	Lav	Dele-værktøj	At kunne dele de kampagner eller oneshots man har lavet med andre brugere af systemet
Informationsorganisering	Høj	Organiserings-værktøj	Brugerens data skal være organiseret i forskellige kategorier, så det er nemmere at overskue og håndtere

Combat tracker	Medium	Organiserings-værktøj	Brugeren skal kunne indtaste og behandle data for ingame combat-visning
Login system	Meget høj	Organisering	For at den enkelte bruger af systemet kan gemme sine egne data

Andre nødvendigheder

Requirement	**Priority**	**Planned Release**
Brugervenlighed	Meget høj	Det skal være nemt og intuitivt at bruge programmet
Organisering	Meget høj	Brugerens data skal være organiseret på en måde, der gør dem nemme at overskue
Ydeevne	Medium	Det bør være nemt og hurtigt for brugeren at navigere rundt i systemet
Genbrugelighed	Meget høj	Brugeren skal kunne genbruge information fra andre eventyr og kampagner
Pålidelighed	Meget høj	Risikoen for tab af data skal være minimal
Enkelthed/overskuelighed	Meget høj	Programmet skal være overskueligt og enkelt at bruge.

Size It Up

Af Laura Vilen

Arbejdet med projektet er startet allerede i uge 14, hvor vi har arbejdet med ideudvikling og gruppedannelse samt projektetablering.
Herefter ligger påskeferie i uge 16, og det officielle arbejde kan derfor starte i uge 17.
Med aflevering den 28 maj giver dette os en fast tidsramme på 5 uger til udviklingen af programmet og rapportskrivning og dokumentering.

Vores program kræver dog et langt større team eller længere tid til udviklingen for at få alle features implementeret.
Derfor har vi besluttet at afgrænse projektet til kun at omhandle udviklingen af en del af programmet, således at vores produkt kan agere som et eksempel på, hvordan et program til styring af D&D kan se ud. Dermed har vi defineret et fleksibelt scope.

Derfor vil vores fokuspunkter i projektet være:

- Oversigt over kampagner med mulighed for oprettelse, redigering og slet
- Oversigt over eventyr/kapitler med mulighed for oprettelse, redigering og slet
- Notefelter, som giver bruger mulighed for at føje deltaljer til ovenstående elementer
- Overskueligt design
- Intuitiv navigation
- Eksempelsider

Vi vil arbejde med features og udvælge user stories fra backloggen at arbejde med ved at tage udgangspunkt i følgende:

- De minimumskrav vi har defineret for at have et basalt og præsenterbart produkt
- Den prioritering vores product owner har foretaget af backloggen
- De kodemæssige krav der er stillet til opgaven

Tidsestimering

I projektetableringsfasen har vi fået udarbejdet en liste af user stories. For at kunne estimere hvilke og hvor mange user stories vi vil kunne nå at implementere i løbet af projektet, har vi forsøgt os med at lave et system til tidsestimering baseret på story points.
Vi har besluttet at 5 story points svarer til en dags arbejde for ét teammedlem.
Dette giver os 60 story points om ugen.
Vi vil således vurdere hver user story med et antal story points og med udgangspunkt i dette estimat udvælge et antal user stories til sprintloggen i hvert sprint.

Konklusion

Af Kristian, Laura og Rasmus

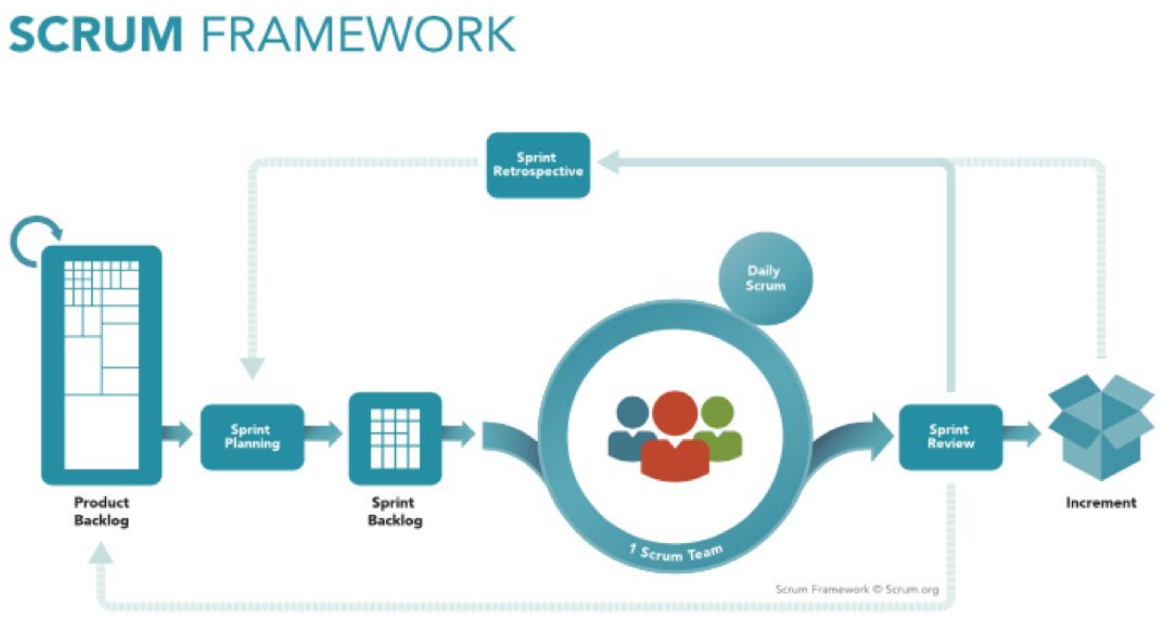
Vi har i denne fase af projektet arbejdet med at definere vores problemstilling, og hvordan vi ønsker at løse problemet.
Til dette har vi, i samarbejde med vores product owner, udarbejdet en liste af features, som vi ønsker i programmet, samt vurderet vigtigheden af de forskellige elementer. Disse elementer omskrives til user stories, der vurderes med et tidsestimat og prioritering.
Herefter har vi prøvet at begrænse vores scope, så vi kun arbejder med udviklingen af den vigtigste del af programmet.

SCRUM

Af Kristian Hornbøll

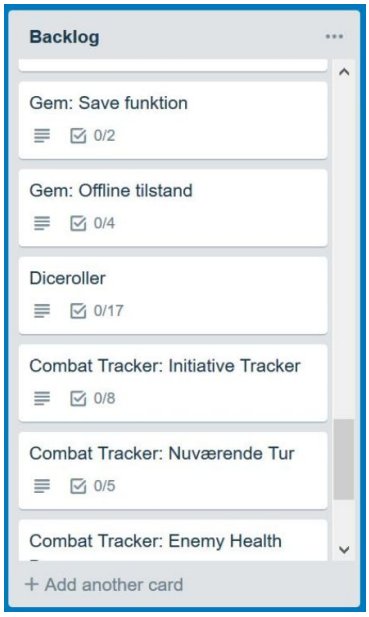
Introduktion

Scrum metodologien anvendes i dag af mange udviklingsvirksomheder pga. dens fleksibilitet både i arbejdsmåden og kommunikationen i teamet. Man arbejder i iterationer, hvor design, implementering og test foregår hele tiden. Et af hovedpunkterne er at kunne levere noget af værdi efter hver iteration. På den måde sikres kvalitet ved gennemgående inspektion.



Fra Scrum.org - flowchart af scrum metoden.

Product backlog



Vores produkt backlog er små opgaver, som vi har defineret ud fra needs and features-listen. Inden for agile development og i vores tilfælde scrum bruger man disse såkaldte user stories til at kunne arbejde enkeltvis og samtidig levere værdi efter hver iteration (sprint).

Invest[2] [3]

Ved udviklingen af vores user stories har vi så vidt muligt fulgt INVEST konceptet inden for agile udvikling, dvs. at de skal overholde nogle retningslinjer.

En user story skal være:

Independent: Den skal være uafhængig af andre, dvs. uanset hvilken prioritet og rækkefølge en backlog er sat i, skal den kunne færdiggøres ud fra de kriterier den har nu. Vi har været ekstra opmærksom på dette, eftersom det var skyld i megen spildt tid i et tidligere projekt.

Negotiable: Den skal være åben for diskussion. En god user story er derfor ikke låst fast i en bestemt retning. Et vigtigt koncept i agile udvikling, da der tit dukker en ekstra Task op, når implementeringen går i gang. Det oplevede vi flere gange.

---

[2] https://www.agilealliance.org/glossary/invest 15/5-2019

[3] https://agileforall.com/new-to-agile-invest-in-good-user-stories/ 15/5-2019

Valuable: En user story skal give noget til helheden, noget af værdi som er værd at forfølge.

Estimable: Alle user stories skal kunne give en idé om, hvad de vil koste at færdiggøre. En backlog prioritet er ofte bygget op om forholdet mellem kompleksitet og værdi, så hvis det ikke er muligt at fastsætte kompleksitet eller tidsestimering, så kan den heller ikke prioriteres ordentligt. I sådan et tilfælde vil man typisk gå ind og se, om det ikke er muligt at dele den yderligere op.

Small: Man vil typisk opbygge dem forskelligt afhængig af hvordan man arbejder i sit team, så de kan nås inden for en iteration. I forlængelse af det forrige krav vil man gerne have, at de er små nok til at der kan fastsættes en tidsramme på dem.

Testable: For at en user story kan opnå status some ”Done” skal den kunne testes. En god idé er at sætte kriterier på allerede fra starten. Hvad skal den kunne? Hvad skal der ske, hvis...? osv. På den måde er der lagt op til at kunne begynde at teste allerede inden implementeringen går i gang, såkaldt test driven development.

Typisk udarbejdes user stories af en product owner evt. i samarbejde med en fra teamet. Vi har siddet sammen i gruppen og oprettet dem selv, og derefter har vores product owner hjulpet med at prioritere dem i forhold til værdi for produktet og brugeren.

Sprint planning

Under sprint planning planlægges det næste sprints user stories: hvad der skal med i sprintets sprintlog, og hvem der har ansvaret for hvilken user story.

Vi har i vores user stories tilføjet business value samt complexity: hvad deres værdi for brugeren er, og hvor lang tid de ca. vil tage at designe, implementere og teste. Ud fra de to værdier afgøres det, hvordan de sorteres i backloggen.

Vi har afsat 60 point til hvert sprint, hver af en uges varighed.
User stories udvælges herefter fra backloggen og overføres til sprintloggen for det kommende sprint, således at antallet af story points for vores user stories stemmer overens med estimatet.
Ved et sprints begyndelse flyttes kort på trello-boardet fra en “To do”-liste - Sprintbackloggen - til en “Doing”-liste, således at hele teamet har overblik over, hvilke user stories der arbejdes med.
De enkelte user stories flyttes til slut over i en “done”-kategori, når underopgaver og kriterier er lavet og opfyldt.

Sprint review & sprint retrospective

Efter hvert sprint holdes der et review møde, hvor man typisk inviterer product owner samt eventuelle eksperter, som ikke er at finde i teamet, til at gennemgå kode og/eller design.

Til dette møde kan product owner sige god for en user story, og den kan erklæres “done-done”.
Derudover holdes der også et retrospective møde, hvor teamet internt vurderer arbejdsgangen i dette sprint: hvad har virket, og hvad har ikke været så godt.

Vi har dokumenteret vores møder i hvert sprint, som kan ses under de enkelte sprints.

Daily scrum

Under arbejdet i et sprint samles teamet en gang om dagen for at holde et kort møde. Dette finder typisk sted om morgenen og har til formål at give hvert medlem af teamet et overblik over, hvor langt gruppen er nået, hvad der arbejdes på, og hvilke udfordringer der er i teamet.
Problemer løses ikke på mødet, men nævnes i stedet, således at det er muligt for folk med de rette kompetencer at hjælpe den/dem der har problemerne.

Sprint 0

Af Laura Vilen

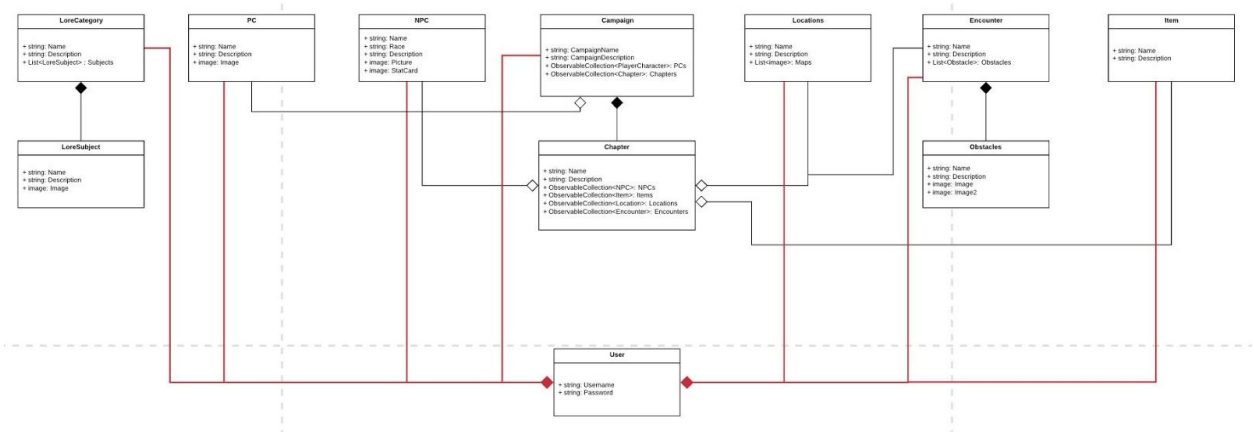
Den først iteration af vores projekt strakte sig over en periode på 2-3 uger.
Perioden blev brugt på at få etableret projektet ved at udforme et inception deck, udføre forretningsanalyse, formulere et vision statement samt udarbejde en arbejdskontrakt, som alle medlemmer kunne være tilfredse med.
Herefter havde vi brug for at få formuleret user stories for vores program.
Det er vigtigt at pointere, at vi er klar over, at det normalt er product owners opgave at formulere og prioritere en liste af user stories, som skal forme backloggen. Men i dette tilfælde har vores product owner ikke nogen tidligere erfaring med konceptet, og undervisning i proceduren ville koste for meget tid i forhold til udbytte. Derfor har vi selv udarbejdet user stories.
Vores product owner har dog haft indflydelse på prioriteringen af user stories i backloggen.
For fuld Backlog se *Bilag #.*

Herefter havde vi brug for at klargøre og opsætte forskellige værktøjer, der skulle bruges til udvikling af projektet, som GitHub og Azure.

Derudover ønskede vi at nå frem til en fælles forståelse af det program, der skulle udvikles.
Dette gjorde at vi udarbejdede følgende:

1. Udarbejde mock-ups af programmets brugerflade. Dette for at vi kunne få en idé om layout, funktionalitet og brugervenlighed, som er en af de vigtigste kvaliteter for vores program.
2. Lave første udkast til domænemodellen. Dette så vi bedre kunne danne os et overblik over programmets indre struktur. I flere tilfælde blev der rettet og tilføjet elementer ved at jævnføre med vores GUI mockups.
3. Lave første udkast til Entity Relations-diagrammet, så vi bedre kunne danne os et overblik over vores database og hvilke tabeller og attributter, vi regnede med at få brug for.

Domæne model



Domænemodellen bruger vi til at skabe et overblik over de klasser, vi gerne vil have med.
Når vi så begynder at tænke på at gemme vores objekter i en database, kan vi tage vores klasser og flytte dem næsten en til en over i et entity-relations-diagram. Dette viser relationer mellem klasserne og er beskrevet i detaljer senere.

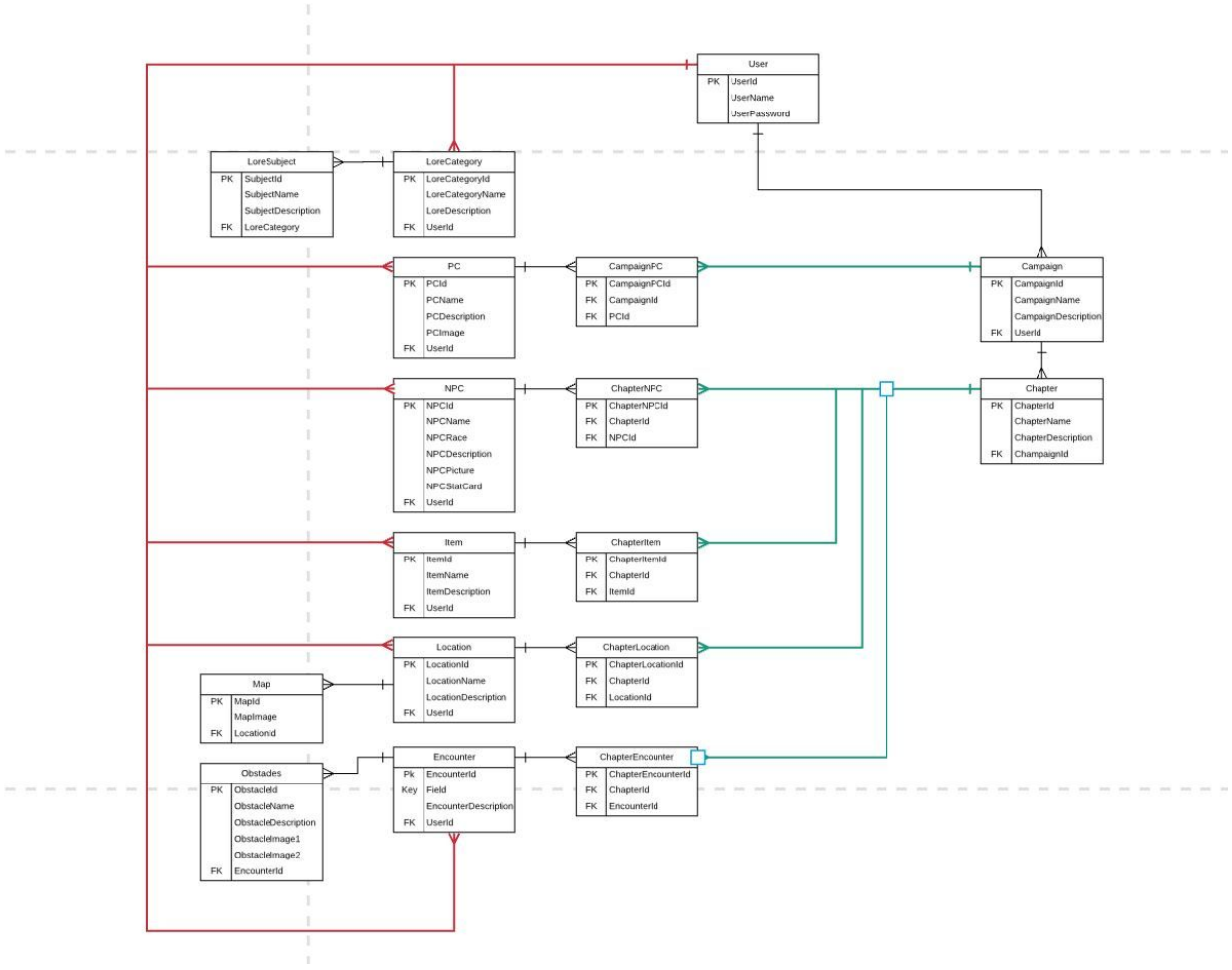
Alle vores modeller og diagrammer i dette projekt er oprettet i UML (Unified Modelling Language), som er et “sprog” der bruges til at visualisere et softwaredesign eller en funktion i systemet. Dette gøres på en detaljeret tidslinje hvor det beskrives, hvornår objekter oprettes i systemet samt hvad en metode gør, hvornår den gør det og de beskeder den får retur.
UML er en standard inden for objektorienterings-modellering og diagrammer. Man kan fx beskrive forhold mellem klasser i et klassediagram, og hvilke designmønstre der er brugt. I vores domænemodel er der helt almindelig associering samt komposition og aggregation. Ved hjælp af disse kan man se, hvordan de skal implementeres som kode. Med komposition menes at en klasse ejes af en anden. Man siger ofte at den er “part of” den anden; den kan altså ikke eksistere alene. Aggregation er når en klasse bruger en anden, en “uses”-relation.

Entity Relations Diagram

Af Rasmus Drotved

Ud fra vores domænemodel kan vi nu så småt begynde at tænke over, hvad der skal være i vores database, som består af diverse tabeller, mere om det i sprint 2.
Ved at se på vores domænemodel beslutter vi, hvilke objekter der skal opbevares i databasen, og begynder at tegne et nyt diagram: et Entity Relations diagram.
Dette diagram giver et overblik over, hvilke tabeller og dermed hvilke typer Entities vi har i databasen samt hvilke Relationer tabellerne har til hinanden.

Vores Entity Relations diagram ses herunder:



Mere om databasen

Af Rasmus Drotved

Tabeller

En tabel (aka. relation) er en samling af data, der har noget til fælles og opdeles i kolonner (aka. attributes) og rækker (aka. records), hvor kolonnerne er ting som fx Name og Price, og rækkerne er egentlige objekter, hvis data er opdelt i kolonnerne.
Det er vigtigt, at der ikke må være to ens rækker i samme tabel.
Tabeller forbindes gennem primary keys og foreign keys.

En primary key er en unik kolonne i en tabel, som har til formål at unikt identificere en række.
Enhver kandidatnøgle (kolonne der kan anses for at være unik) kan bruges som primary key.
Der kan kun være én primary key i en tabel. Oftest bruges ID'er som primary keys, som sættes til automatisk at tælle opad.

En foreign key er en kolonne i en tabel, der optræder som primary key i en anden tabel.
Gennem denne foreign key skabes der forbindelse mellem tabellerne, så de kan få adgang til hinanden.[4]

Fra tabel til normalisering:
Under designet af vores database opdagede vi, at vi flere steder stødte på mange-til-mange-forhold mellem tabeller. Dette kan ikke laves i en database, og vi var derfor nødsaget til at oprette et såkaldt linking table, som er en tabel der bruges til at linke to tabeller.

For eksempel kan vores campaigns indeholde flere PC'er, men samtidig kan flere PC'er optræde i forskellige campaigns. Dette ville hurtigt blive rodet, og man ville være nødt til at oprette en kolonne for hver PC i en campaign. For at løse dette problem oprettede vi en tabel imellem dem kaldet CampaignPC, som består af foreign keys til både campaign og PC.

På denne måde bindes en campaign og en PC sammen i CampaignPC-tabellen, så en campaign kan være bundet til flere PC'er og vice versa, uden at man skal oprette en kolonne hver gang.

Dette er en del af en teknik kendt som Normalisering.

[4] Connelly, Thomas og Begg, Carolyn - Database Solutions - a step by step guide to building databases, Addison-Wesley, september 22, 2003

Normalisering

Normalisering bruges til at lave kvalitetskontrol af tabeller og reducere mængden af redundant data, dvs. data der unødvendigt optræder mere end én gang. Normalisering bruges også til at reducere antallet af NULL-værdier, der tager plads i databasen uden information.

Inden for normalisering taler man typisk om de tre normalformer, som er:

1. Normal form: Primary key er defineret, og alle kolonner er atomic, dvs. single valued. Denne form forbyder kolonner med mere end én værdi.

2. Normal form: Samme som 1. Normal form bare uden partial dependency, dvs. når en kolonne er afhængig af en del af nøglen. Bruges mest når man taler om composite keys

3. Normal form: De øvrige former + ingen transitive dependency. Dette vil sige, at alle non-primary-kolonner kun kan findes igennem primary key-kolonnen.[5]

Implementering

Da vi var færdige med at designe og normalisere, gik vi i gang med at oprette selve databasen.

Til at oprette databasen benyttede vi SQL queries til at lave de tabeller vi havde brug for samt indsætte testdata i tabellerne.

Bl.a. havde vi brug for en Table of Users, som vi kunne bruge til at filtrere databasens indhold, så det kun er tilgængeligt for den bruger der har oprettet dataene.

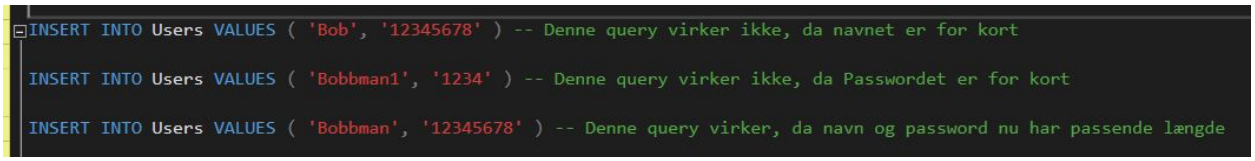
Et eksempel på nogle af vores tabeller kunne være UserTable og CampaignTable.
Ud over blot at lave tabellerne satte vi også nogle relevante constraints på for at sikre, at vores regler for brugere overholdes, og vi brugte UserId i UserTable som foreign key i CampaignTable, så campaigns senere har et UserId i sig, som vi kan bruge til filtrering.
Her ses et billede af den query, vi brugte til at lave vores UserTable og CampaignTable:

---

[5] https://fronter.com/easj/links/files.phtml/5c74f3d2be421.1225660000$50278981$/Resources/Course+materials/Software+Design/Sprint+2/Slides/normalization.pdf - brugt d. 23-05-2019

```
CREATE TABLE UserTable (
UserId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
UserName VARCHAR(20) NOT NULL UNIQUE,
UserPassword VARCHAR(30) NOT NULL,

CONSTRAINT NameLength CHECK(LEN(UserName) BETWEEN 4 AND 20),
CONSTRAINT PasswordLength CHECK(LEN(UserPassword) BETWEEN 6 AND 30),
)


CREATE TABLE CampaignTable(
CampaignId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
CampaignName NVARCHAR(60) NOT NULL,
CampaignDescription NVARCHAR(450),
UserId INT NOT NULL,

FOREIGN KEY (UserId) REFERENCES UserTable
)
```

Undervejs testede vi selve databasen vha. nogle queries. Formålet var at teste, at de regler vi oprettede holdt stik. Et par af disse kan ses herunder.



Scrum Review

Af Laura Vilen

I vores sprint fik vi indkaldt product owner til review på vores mock-ups og system design. Han kom med flere ændringer og features, han ønskede til programmet:

- En udvidet 'diceroller', som skulle have en del flere trin og funktioner, således at funktionen ville blive et brugbart og smart værktøj for brugeren.
- Det skal være muligt at ændre på rækkefølgen af kapitler i en kampagne, så programmet afspejler formålet, en vigtig kvalitet ved Game Masteren selv: hans evne til at tilpasse og ændre det planlagte spil.

Scrum Retrospektive

Under retrospective-mødet diskuterede vi vores planlægningsprocess og SCRUM metoden.
Vi havde alle en fornemmelse af ikke at være nået særligt langt med projektet; rigtig meget tid var gået med at etablere og planlægge, men uden at vi stod med nogen fremviselig del af vores projekt.
Vi måtte konkludere, at vores arbejde ikke havde været særligt iterativt, og at vi var begyndt ikke at følge SCRUM metoden.

Resultatet af Sprintet

Den første del af sprint 0 var klart en vigtig del af etableringen af vores projekt, men hen ad vejen begyndte vi at bevæge os væk fra iterativt arbejde og SCRUM metoden.
Ved at bygge databasen først samt oprette alle modelklasser og projekter, inden vi begyndte arbejdet med de enkelte user stories, fik vi lavet arbejde der ikke var nødvendigt endnu. Så vi endte med at frustrere os selv, da vi følte, vi spildte tid på forberedelse frem for at gå i gang med implementering af systemet.

Sprint 1

Af Laura Vilen

Sprint Planning

I sprint 1 udvalgte vi fire user stories på baggrund af, hvor mange story points vi regnede med at kunne færdiggøre i løbet af sprintet samt product owners prioritering i backloggen.
De blev:

1. Profil: Log In
2. Profil: Log Ud
3. Profil: Opret profil
4. Kampagner: Opret Kampagner

Der vil blive gået i detaljer med de nævnte user stories under sprint 2, hvor de blev færdiggjort. De er også beskrevet i backloggen. Se *Bilag #*

Derudover havde vi nogle tasks i forbindelse med design af vores main page og menu med navigering, som vi besluttede at kode i fællesskab.

Design Patterns

Af Rasmus Drotved

Design patterns er måder at strukturere sin kode på. Det forklares godt her:

"*Design Patterns are nothing but applications of those principles (Object-Oriented Design principles) in some specific and common situations, and standardizing some of those."*[6]

Nogle af de vigtigste Design Patterns[7] er:

1. Singletons
2. Factory method
3. Strategy
4. Observer

[6] https://www.codeproject.com/Articles/98598/How-I-explained-Design-Patterns-to-my-wife-Part - 26-05-2019 - 2. Passage under *What is a Design Pattern?*

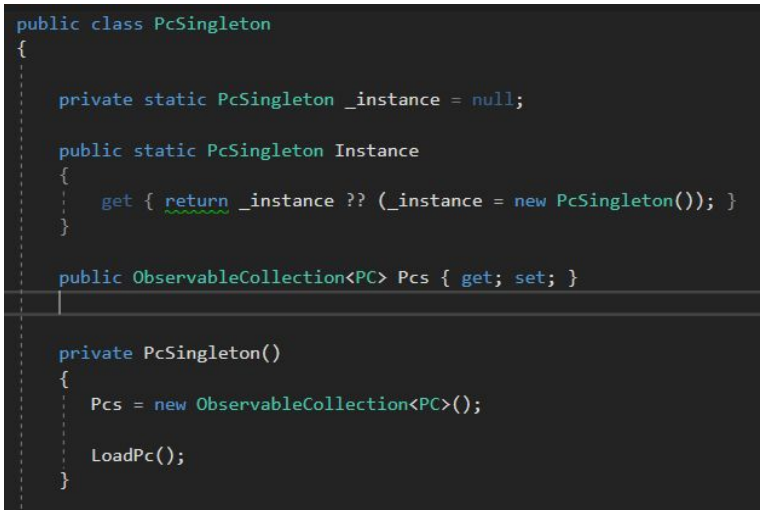
[7] https://medium.com/educative/the-7-most-important-software-design-patterns-d60e546afb0e?fbclid=IwAR1F1a599zdCtB5GiQ473GQoEJ3hWE8FI-dpV0w7J3jYaAOnSlwWTWtPGRA - 26-05-2019

5. Builder
6. Adapter
7. State

I vores projekt benytter vi flere af disse mønstre. Et par af dem ses nedenfor:

Singleton

I vores projekt opbevarer vi listen af PC'er i en singleton, PcSingleton, for at sikre os, at det ALTID er den samme liste af PC'er vi har fat i, fordi der kun kan oprettes én instans af klassen. Dette opnås ved at gøre constructoren private og kalde den gennem Set metoden i en property af klassen selv.



```
public class PcSingleton
{

    private static PcSingleton _instance = null;

    public static PcSingleton Instance
    {
        get { return _instance ?? (_instance = new PcSingleton()); }
    }

    public ObservableCollection<PC> Pcs { get; set; }


    private PcSingleton()
    {
        Pcs = new ObservableCollection<PC>();

        LoadPc();
    }
```

Observer

Observer design pattern går ud på at have et objekt, kaldet en *observer*, som observerer andre objekter, kaldet *subjects*. Man kalder det at observeren subscriper på et subject.
Når observeren bliver notificeret om at der sker en ændring med et af sine subjects, vil den foretage de handlinger eller opdateringer, der er defineret i koden.

En Observable Collection er en kollektion, som er et eksempel på en klasse der implementerer design pattern "observer".
Kollektionen implementerer nemlig interfacet INotifyCollectionChanged, som sørger for at når et objekt i kollektionen tilføjes eller slettes, gives der besked videre til alle observers der *subscriber* på den ObservableCollection.

I vores projekt benytter vi rigtig meget klassen ObservableCollection, da det er vigtigt at kollektioner af objekter der vises gennem UI'en, opdateres så snart der er lavet en ændring.
Vi implementerer også andre steder interfacet *INotifyPropertyChanged*, således at relevante objekter sender besked videre til sine *subscribers*, der kan foretage de nødvendige opdateringer. Dette gør vi ved at kalde metoden, OnPropertyChanged, fx i en property, således at de elementer der er bundet til propertien i UI'en, opdateres når dens værdi ændrer sig.

```
class NPCViewModel : INotifyPropertyChanged
```

```
#region INotifyPropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

[NotifyPropertyChangedInvocator]
protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}

#endregion
```

```
/// <summary> This is a collection of NPCs from the database. </summary>
public ObservableCollection<NPC> NPCs
{
    get => _npCs;
    set{ _npCs = value;
        OnPropertyChanged();
    }
}
```

Inheritance

Inheritance eller arv vil sige, at en klasse arver alle egenskaber og metoder fra "parent" klassen.

Under arbejdet med at designe og planlægge vores projekt indså vi, at mange af vores modelklasser indeholdt Name og Description-properties, så vi lavede oprindeligt en *Abstract class*, som indeholdt to properties: Name og Description, og som modelklasserne kunne arve fra.
Da vi oprettede vores database fik vi dog begået den fejl, at give attributterne "Name" og "Description" forskellige formater; Fx "UserName" eller "CampaignName" frem for at navngive attributterne på samme måde, som properties i vores abstract class.
Og da denne fejltagelse først gik op for os sent, valgte vi derfor at gå væk fra ideen om inharitance og i stedet omdøbe vores modelklassers properties.

GUI Design

I vores system vægter vi brugervenlighed højest. Det skal være nemt og intuitivt at navigere rundt og danne sig et overblik over systemet.
Vores forside danner rent kodemæssigt baggrund for alle andre sider i systemet, og det var derfor vigtigt for os at designe den så enkelt og overskueligt som muligt.

Mock up af MainPage og menuen

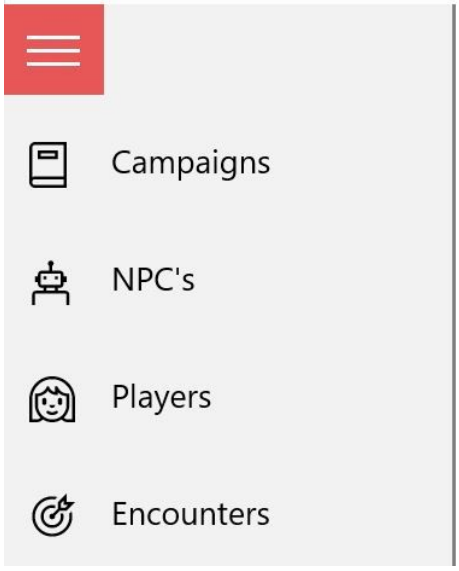
UI & UX

Af Kristian Hornbøll

Overordnet har vi forsøgt hele tiden at have Jacob Nielsens ten usability heuristics[8] i mente. Med heuristik menes, at der ikke nødvendigvis stræbes efter perfektion, men mere efter at finde en umiddelbar løsning, der kan få noget op at stå. Det går ud på at skabe og fastholde en forståelse mellem bruger og system, som ikke bliver for teknisk. Vi har for det meste fokuseret på de første 5.

Visibility of system status: Hvis der er noget der skal loades, må brugeren ikke være i tvivl om hvorvidt det er dét der sker, eller om systemet er brudt sammen. En typisk løsning kan være en lille tæller der går fra 0 til 100% eller en lille bar der udfyldes.

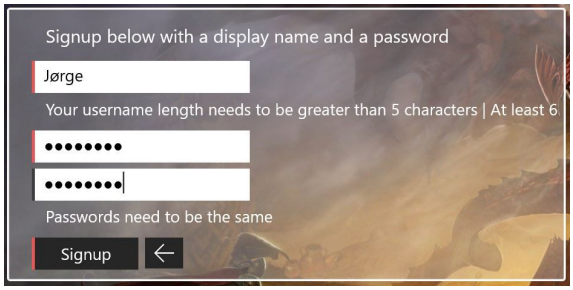
[8]Nielsen, Jacob: https://www.nngroup.com/articles/ten-usability-heuristics/ - 17/5-2019



Match between system and real world: Systemet skal kommunikere i et sprog, brugeren er bekendt med både i form af tekst og visuelt. Ikoner og grafik som afbilleder virkelige objekter er en enormt populær løsning og meget anvendt. Vi har brugt ikoner på mange af vores knapper, som repræsenterer det de leder henimod. I vores fejlhåndteringer har vi gjort meget ud af at informere brugeren om, hvad systemet kræver for at fungere, som det ses bl.a. ved oprettelse af ny bruger eller ved login med eksisterende bruger.

User control and freedom: Har du en bruger der tilfældigvis får trykket på noget ved en fejl, så sørg for at der er en udvej, enten en tilbage- eller luk-knap eller en besked der lige tjekker om det nu også var meningen. På den måde kan man hele tiden holde brugeren på rette vej. Vi har fx implementeret en besked der dukker op, når en bruger prøver at slette noget; der vælger man så ja eller nej.

Consistency and standard: Eftersom vi udvikler til Microsoft platform i form af et universal windows platform-projekt, har vi selvfølgelig brugt mange af de controls der er tilgængelige i xaml/visual studio, bl.a. Listviews, stackpanels og den velkendte burgermenu. Derudover har vi prøvet at holde os til et fast farveskema og ladet udseendet af de forskellige elementer være nogenlunde konsistent.



Error prevention: Vi har gjort rigtig meget ud af at sørge for, at programmet ikke lukker uventet ned, uanset hvad en bruger skulle finde på. Vi bruger try & catch, som går ud på at fange de fejl systemet smider, kaldet exceptions, samt message dialogs til at fange og formidle de fejlbeskeder der smides og give dem videre til brugeren. Typisk er indtastningsfejl ikke vist med exceptions, men som en fejlbesked i selve appen. Denne exception tyder på, at der ikke har været forbindelse til databasen.

Exception ved oprettelse af en spiller med et allerede eksisterende navn

De sidste 5 punkter hedder: Recognition rather than recall, Flexibility and efficiency of use, Aesthetic and minimalistic design, Help users recognize, diagnose and recover from errors, Help and documentation. Men vi har som sagt fokuseret mest på de første 5.

Scrum review

I slutningen af sprint 1 indkaldte vi Henrik Høltzer til reviewmøde på vores kode.
Ved mødet kom vi frem til, at det var en bedre idé at adskille vores UWP application og vores Web Api i to forskellige solutions (mere om det i sprint 2), så synkronisering med GitHub ville gå mere gnidningsfrit, da Web Api'en ofte skal udskiftes.
Derudover havde vi i løbet af ugen talt om, hvordan vi skulle håndtere billeder i databasen. Under reviewmødet blev der foreslået to forskellige håndteringsmuligheder, som blev nedskrevet og skal undersøges og diskuteres senere i programmets udvikling.

Scrum Retrospective

På retrospective-mødet diskuterede vi mest småting som småsludren i teamet, der til tider kunne løbe lidt af sporet, og som vi skulle være opmærksomme på at begrænse.
Ved det sidste møde havde vi måttet indse, at vi ikke fulgte SCRUM metoden, og vi havde følt os frustrerede over manglen på fremskridt med arbejdet.
I sprint 1 vendte vi tilbage til den agile og iterative måde at arbejde på, og herefter blev humøret bedre i gruppen, da vi begyndte at kunne se resultater af vores arbejde igen.
Under mødet kommenterede alle medlemmer også på det gode samarbejde i gruppen. Kommunikationen var åben og let, arbejdsmoralen god, og vores værdier og faglige niveauer syntes at stemme overens.

Resultatet af Sprintet

Under sprintet blev det meget tydeligt for os, at den iterative arbejdsstruktur var langt mere givende end det vi ellers var begyndt på ved en fejltagelse. Vi kunne langt bedre se resultatet af vores arbejde, hvilket også styrkede arbejdsmoralen.
Vi havde dog undervurderet vores tidsestimering af nogle opgaver, og derfor blev ingen af de påbegyndte user stories færdiggjort i dette sprint.

Sprint 2

Af Laura Vilen og Rasmus Drotved

Sprint Planning

I begyndelsen af sprintet havde vi allerede fire user stories, som var påbegyndt i tidligere sprint, men ikke færdiggjort. Se evt. *Sprint 1* for listen. Disse blev føjet til sprintloggen for sprint 2. Derudover udvalgte vi på baggrund af vores tidsestimering vha. story points tre yderligere user stories, som vi ville gå i gang med, når de første fire var færdige:

1. NPC: NPC-oversigt
2. PC: PC-oversigt
3. Kampagner: Kapiteloversigt

I dette sprint fik vi også oprettet vores web service i Azure.

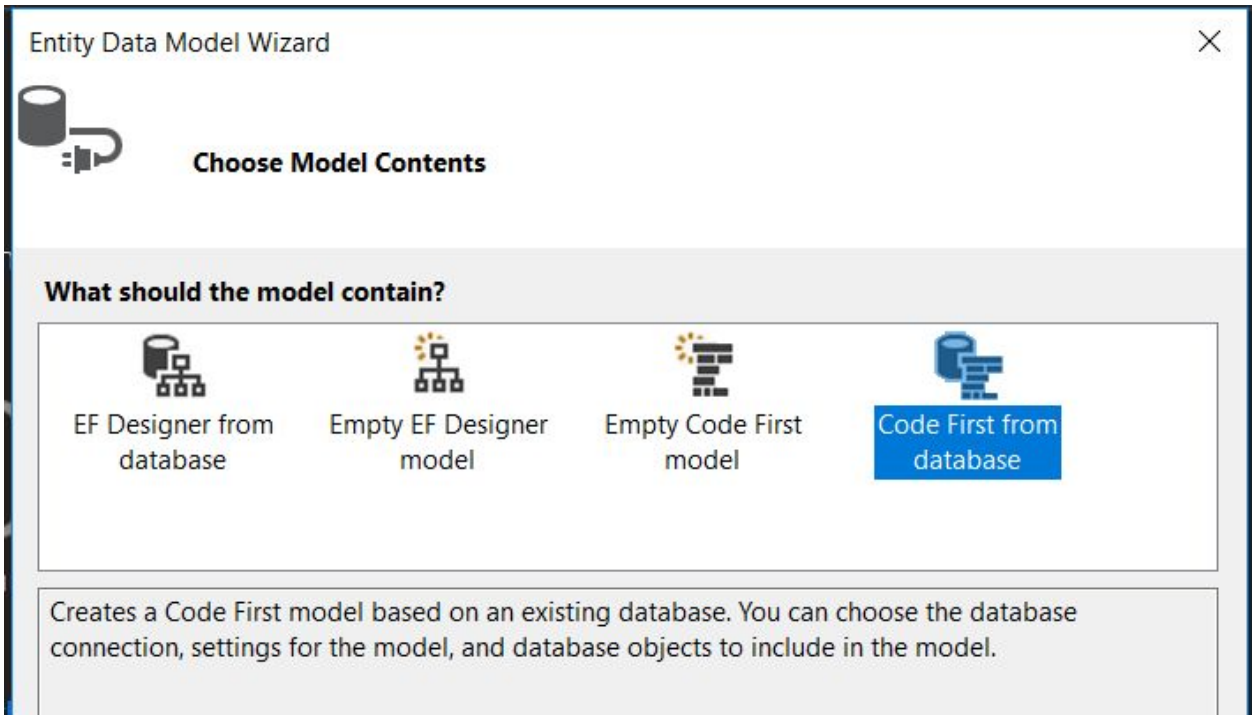
Web Service

Af Rasmus Drotved

Til at forbinde vores MVVM projekt med databasen benytter vi os af en projekttype kaldet ASP.NET Web Application. Vi valgte at lægge vores i sin egen solution adskilt fra selve UWP appen samt at lægge den og databasen op i Azure.

Vi starter med at oprette Web Application projektet med en Web Api.
Denne Api vil fungere som bindeled mellem vores program og selve databasen, da ASP.NET Web Api er designet til at gøre det lettere at håndtere HTTP requests.[9]

Når dette er oprettet, går vi ind og benytter os af ADO.NET DataModel, der er del af ORM (Object Relational Mapping) Frameworket Entity Framework. Denne datamodel lader os oprette klasser, der matcher indholdet i vores database automatisk. [10]
Derudover har datamodellen også adgang til databasens DBMS (Database Management System), der står for håndtering af database tabellerne.



[9] https://www.c-sharpcorner.com/article/create-simple-web-api-in-asp-net-mvc/ - 24-05-2019
[10] https://www.c-sharpcorner.com/article/introduce-entity-framework-with-ado-net-entity-data-model/ - 24-05-2019

Derefter er det tid til at oprette Web Api'ens controllers, der er klasser designet til at håndtere HTTP requests, som vi kommer til at modtage fra vores UWP app's Persistency Service.[11]

Når man opretter en controller, specificerer man også hvilken modelklasse, den skal fungere sammen med. Takket være Entity Frameworket kan man vælge en variant af controller, der opretter actions (controllerens public metoder)[12] og CRUD metoder automatisk.

Disse controllers fungerer ved at generere en API string, der holder adressen på objekter i databasen. Vha. denne string kan metoder kaldes på de rigtige tabeller eller elementer i tabellerne i databasen fra vores persistency service.

```
/// <summary> This field is used to hold the API string for Put and Delete methods. </summary>
private const string PutNDelete = "api/NPCs/";
```

Derefter starter vi programmet og API'en, der fører os til en webside, hvor vi kan tjekke, at der er hul igennem til databasen.

Forbindelse til MVVM

Som tidligere nævnt har vi en klasse i UWP appen, som står for at gemme al data fra vores app, kaldet Persistency Service - helt specifikt navngivet GenericdbPersistency. Persistens er også et lag i MVVM strukturen.
Vi besluttede os for at gøre klassen generisk for at gøre det muligt at genbruge dens CRUD metoder til flere forskellige objekttyper og dermed simplificere og effektivisere arbejdet.
Dette gjorde vi ved at erstatte alle datatyper for objektet vi har med at gøre med <T>, hvor T står for type. Vi tilføjede også <T> til klassens navn.
Herudover har vi gjort alle klassens metoder statiske, således at de kan kaldes uden først at skulle oprette et objekt af klassen.

Selve klassens metoder fungerer ved at sende en HTTP request til vores Web Api, som dirigerer requesten til den relevante controller, der så sørger for at interagere med databasen som ønsket.

Dette fungerer ved at vi kalder en af GenericDbPersistency's metoder, fx "Get"-metoden, der her specifikt er navngivet "GetObj()". I denne metode opretter vi først en HttpClientHandler, der

[11] http://heho-easj.dk/SWC2-2019f/SWC2Ugeplan2019f.html (uge 12-13) - 24-05-2019
[12] http://heho-easj.dk/SWC2-2019f/SWC2Ugeplan2019f.html (uge 12-13) - 24-05-2019

har ansvaret for de beskeder som HttpClient bruger[13], og en HttpClient, som er den klasse der står for at sende og modtage HTTP requests og responses[14]. HttpClienten bør oprettes i en "using" klammes parameter, da det vil betyde, at adgang til Web Api'en kun er mulig, når HttpClient'en er i brug. Vi sætter dernæst vores HttpClient's baseAddress til at pege på en ny Uri, der tager vores serverUrl, som er adressen til Web Api'en, som argument.

Dernæst kalder vi HttpClient'ens "GetAsync()" metode, som tager vores Api-streng som argument og gemmer resultatet af metoden i en variabel. Er dette en succes, gemmer metoden indholdet af responsen i en IEnumerable<T>, som føjes til en liste, som metoden returnerer til metodekaldet.

Selve metoden kører Async, hvilket vil sige, at metoden udføres parallelt med andre handlinger programmet foretager, og dermed behøver brugeren ikke vente på, at metoderne kører færdigt, før de kan foretage andre handlinger.
Nogle steder indeholder en metode nøgleordet "Await". Dette udtryk betyder, at man sætter programmet til at vente på, at det awaitede statement er færdigt, før det kører videre.

[13] https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclienthandler?redirectedfrom=MSDN&view=netframework-4.8 - 24-05-2019
[14] https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=netframework-4.8 - 24-05-2019

Dette billede viser vores GetObj() metode i vores GenericDbPersistency, som er den metode der henter data fra vores database.

```
namespace GameMastersTools.Persistency
{
    public class GenericDbPersistency<T>
    {
        private const string serverUrl = "https://gamemastertools3.azurewebsites.net";

        public static async Task<List<T>> GetObj(string api)
        {
            HttpClientHandler handler = new HttpClientHandler();
            handler.UseDefaultCredentials = true;

            using (HttpClient client = new HttpClient(handler))
            {
                client.BaseAddress = new Uri(serverUrl);

                try
                {
                    var response = client.GetAsync(api).Result;

                    if (response.IsSuccessStatusCode)
                    {
                        var obj = await response.Content.ReadAsAsync<IEnumerable<T>>();
                        return obj.ToList();

                    }

                    return null;
                }
                catch (Exception e)
                {
                    MessageDialogHelper.Show("ERROR, ERROR, ERORr", "Roor\n\n" + e.Message);
                    throw;
                }
            }
        }
```

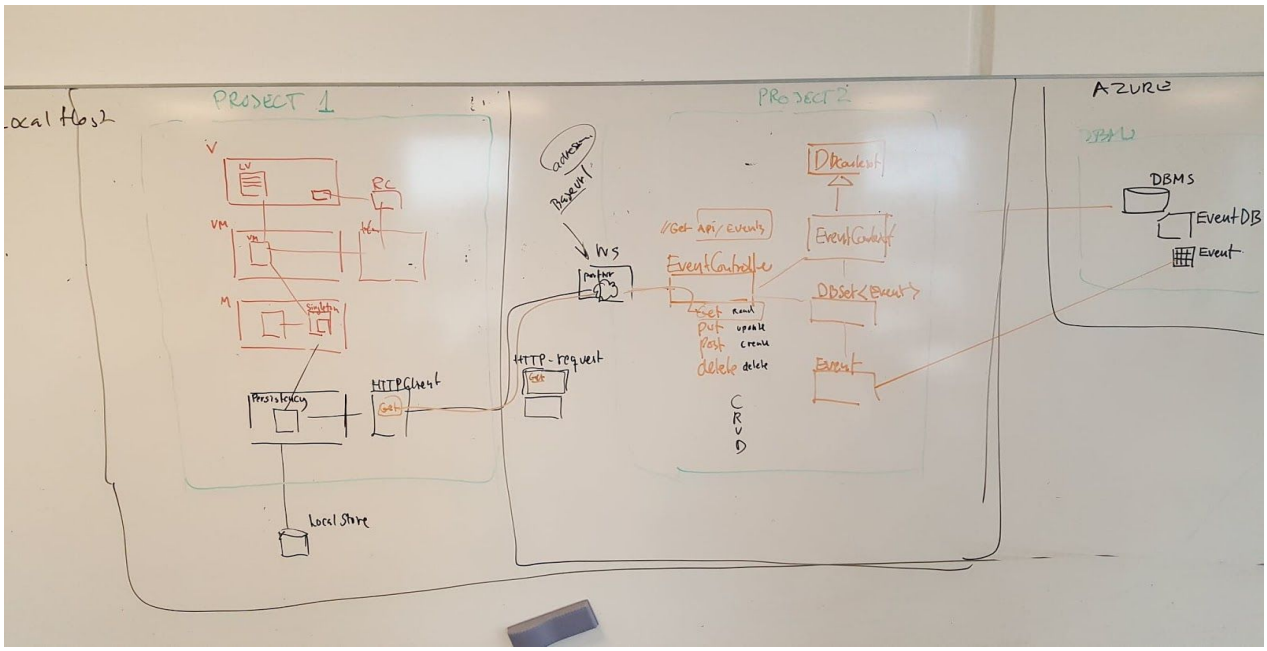
Metoden fungerer ved, at HttpClienten sender en HttpRequest til en “router”, som dirigerer requesten til den rette controller.

Requesten består af en Header, som indeholder tre dele: Url’en til vores Web Api, routerens portnummer og Api-strengen til at finde den specifikke Controller, vi leder efter.
Når routeren modtager requesten, bruger den Url’en til at identificere den rette Controller og kalder derefter den relevante Action.
Afhængigt af hvilken Action der kaldes, sker der forskellige ting i Datamodellen, som vi har kaldt DnDatabaseContext. I dette tilfælde henter den det DbSet, der matcher den ønskede dataType.

Datamodellen er en klasse, der arver fra klassen *Context* og indeholder forskellige DbSet[15] - som er en kollektion, der ikke kan indeholde dubletter - der kan indeholde instanser af de klasser, der autogenereres ved oprettelsen af datamodellen. Denne har så adgang til DBMS'en, der står for håndteringen af tabellerne i en database. Herved får vi altså adgang til tabeller i databasen og kan påvirke dem som ønsket.

Den forbindelse benyttes af persistens-klassen i vores UWP app til at modtage *response* fra databasen, når der hentes, sendes eller redigeres i objekter fra og til databasen.

Her ses et billede[16], der viser forbindelsen mellem databasen og MVVM:

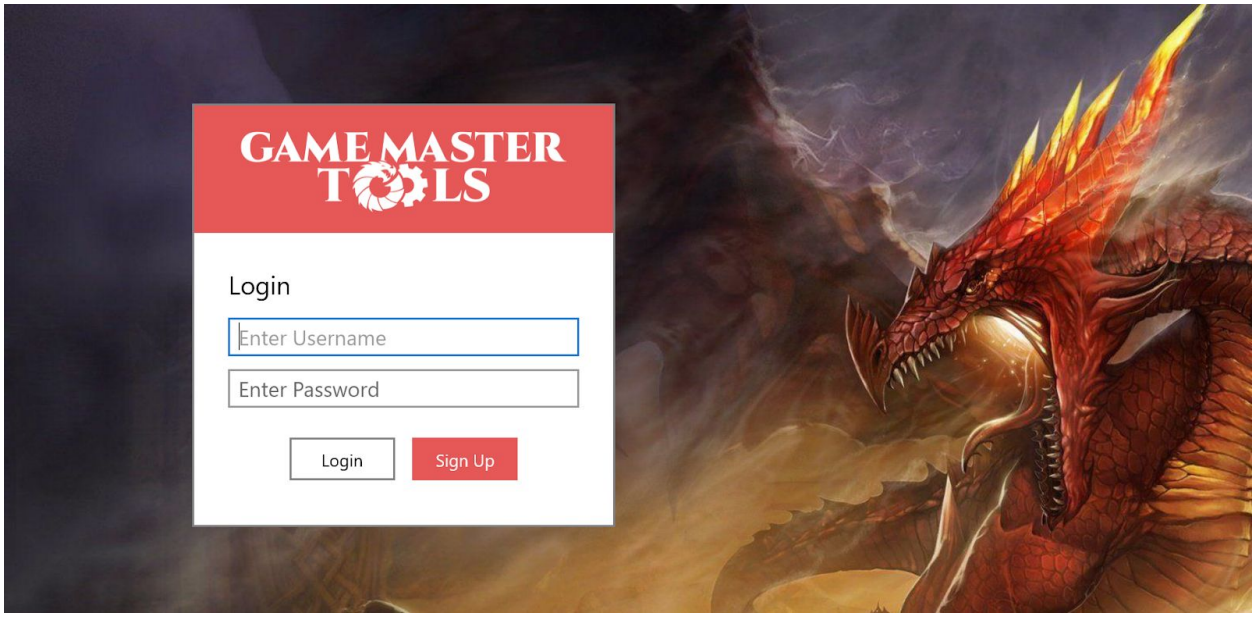


[15] https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbset-1?view=entity-framework-6.2.0 - 24-05-2019

[16] Billede taget fra Henrik Høltzers undervisning i Software Construction på 2. Semester - 15-03-2019

User story: Profil Login

Af Rasmus Drotved



Her vil vi gå i dybden med to user stories: Login og Logud. Disse blev udvalgt til at vise noget frem, som Rasmus har været ansvarlig for.
De lyder:

Login
Som bruger
Vil jeg gerne kunne logge ind
Så jeg kan gemme mine data privat

Acceptance Criteria:

- Ved Log ind tjekkes om
  Brugernavn og/eller password
  Eksisterer i brugerdatabasen.

- Skal udskrive fejlmeddelelse
  Ved forkert indtastning
  Såsom ugyldige tegn osv.

Logud
Som bruger
Vil jeg kunne logge ud af min profil
Så jeg kan afslutte min session

Acceptance Criteria:
- Brugeren logges ud

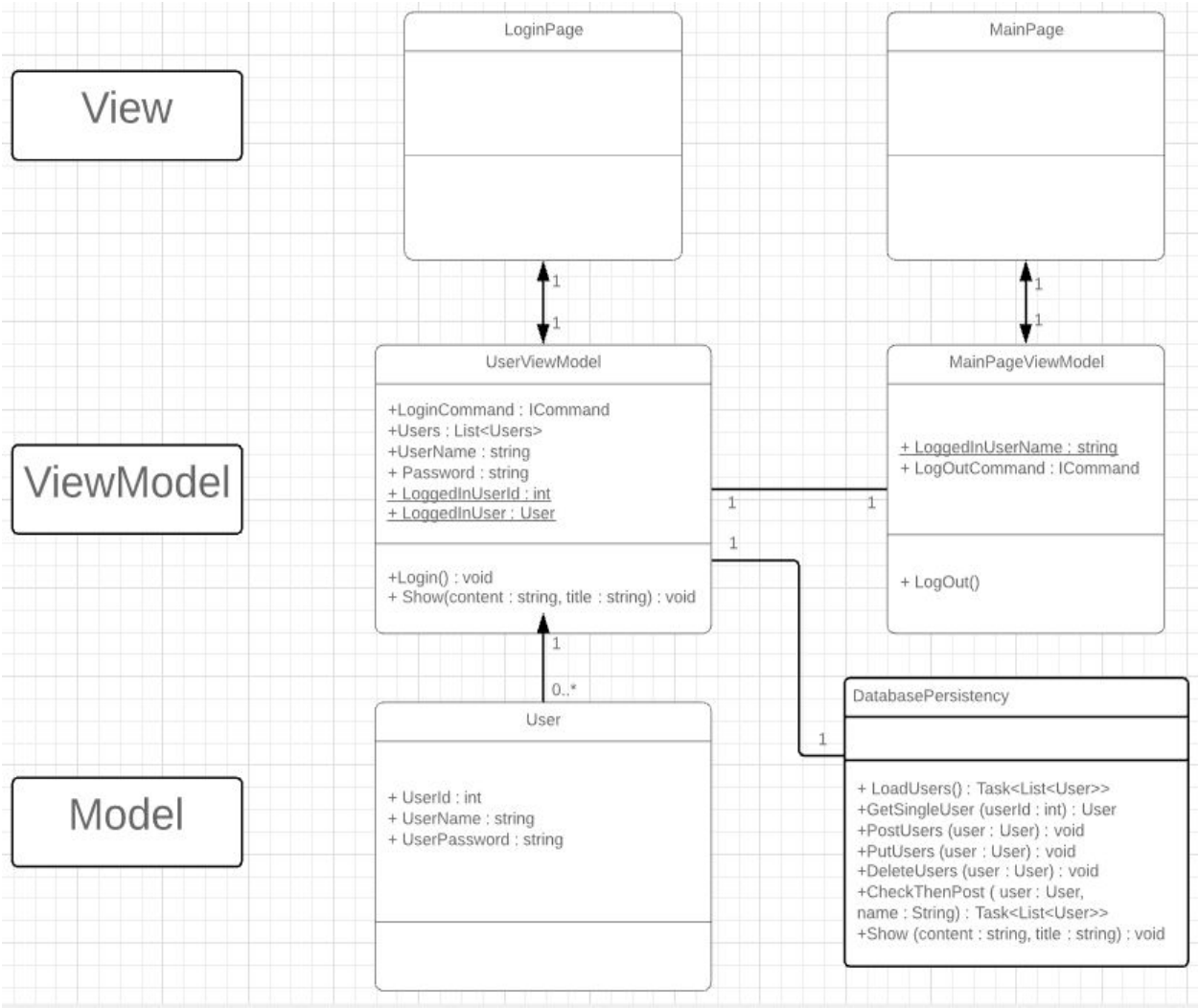
- Brugeren tages tilbage til log ind siden

Formålet er, at data i systemet skal holdes adskilt mellem brugerne, således at en bruger kun kan tilgå egne data og ingen andres. Til dette skal brugeren naturligvis kunne logge ind og ud af systemet.

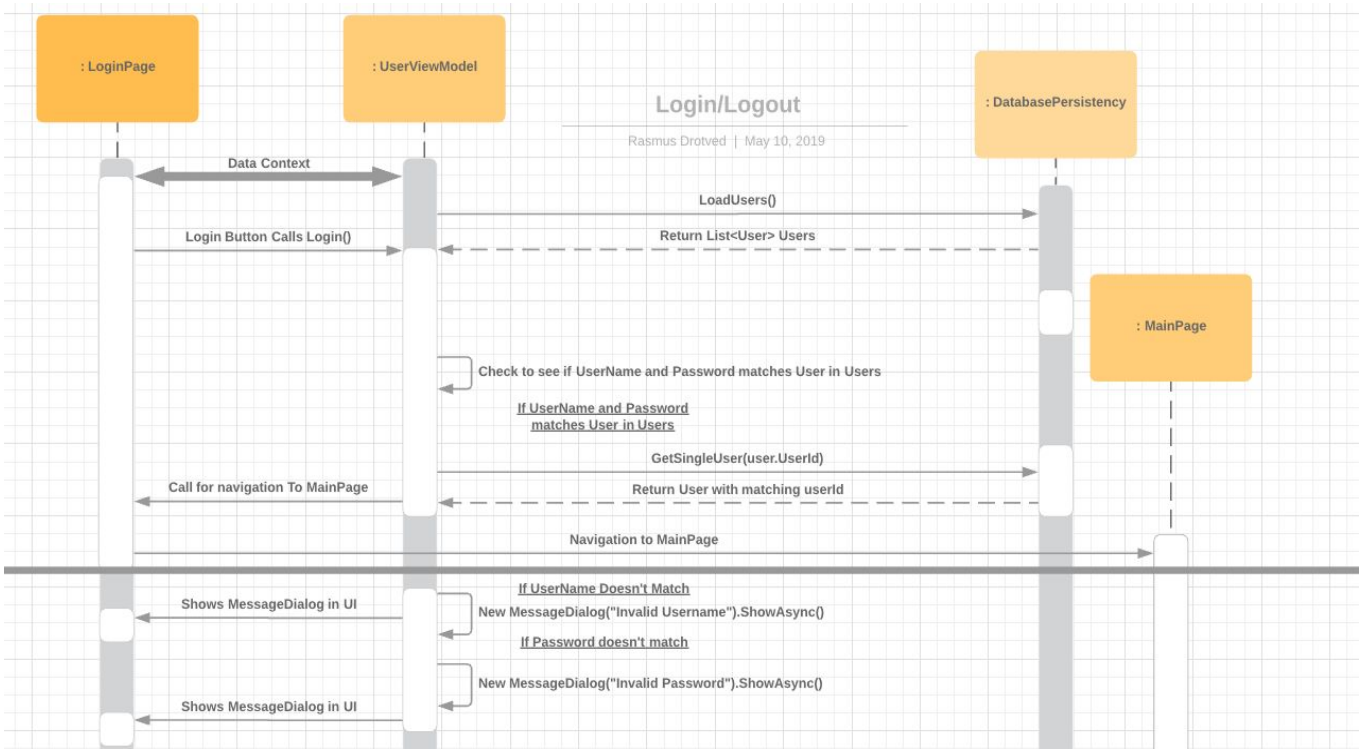
Design

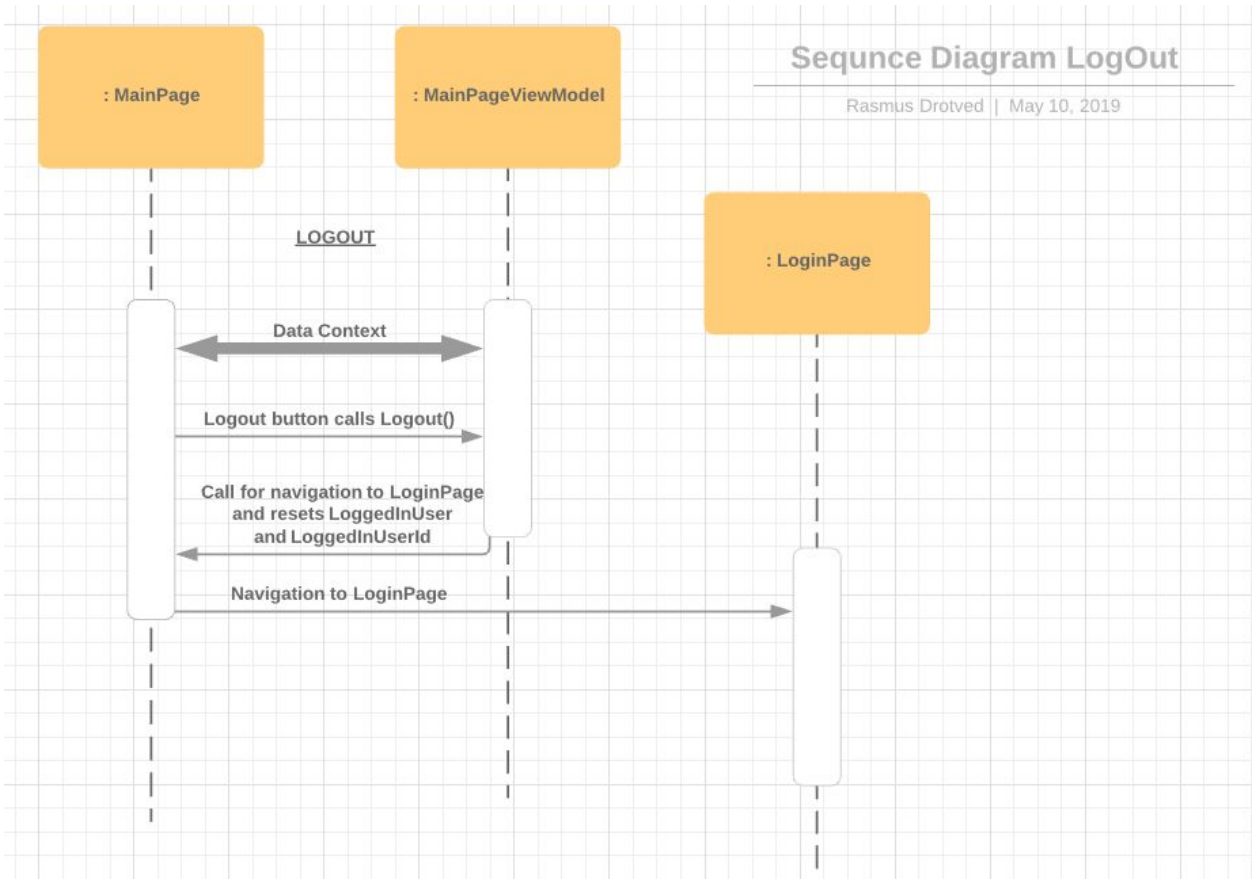
I forbindelse med denne user story blev der lavet et klassediagram for at danne et statisk billede af systemet samt et sekvensdiagram for henholdsvis Login og Logud-scenarierne for at vise et mere dynamisk billede af, hvad programmet gør.

Herunder ses vores UML klassediagram over Login og Logout systemet.



Herunder ses to sekvensdiagrammer: Et for Login og et for Logud.





Implementering

Login()

Funktionaliteten i Loginsystemet består i en sammenligning mellem det input brugeren indtaster i vores UI og de Userinformationer der ligger i vores database. Er der et match, tages brugeren videre. Hvis ikke, vises en fejlbesked der fortæller, om brugernavn eller password er forkert.

Herunder ses et billede af Login() metoden.

```
/// <summary>
/// This method checks if the inputted UserName and Password matches a users UserName and UserPassword in the database.
/// If it does the user is logged in, if not then an error message is shown.
/// Once a user has logged in it then stores the user as an object as well as stores the users UserId
/// </summary>
public void Login()
{   UserDoesNotExist = true;
    PasswordIsIncorrect = true;

    foreach (var user in Users)
    {
        if (UserName == user.UserName)
        {
            UserDoesNotExist = false;
            if (Password == user.UserPassword)
            {
                //Static ID for logged in User
                LoggedInUserId = user.UserId;
                PasswordIsIncorrect = false;

                //Returned User Object
                LoggedInUser =  DatabasePersistency.GetSingleUser(user.UserId);
                MainPageViewModel.LoggedInUserName = LoggedInUser.UserName;

                //Navigation (must be commented out when UnitTesting)
                Frame loginFrame = Window.Current.Content as Frame;
                if (loginFrame != null)
                { loginFrame.Navigate(typeof(MainPage)); }
                break;
            }
        }
    }
    if (UserDoesNotExist)
    { new MessageDialog("Invalid Username").ShowAsync();}
    else if (PasswordIsIncorrect)
    { new MessageDialog("Invalid Password").ShowAsync();}
    }
```

Når vores brugere indtaster deres brugernavn og password, opdateres de to properties UserName og Password i vores UserViewModel. Når brugeren så klikker på “Login”-knappen, kaldes vores Login() metode.

Det første der sker er, at de to bool’er, userDoesNotExist og passwordIsInCorrect, sættes til at være true.

Vores properties ses her:

```
public ICommand LoginCommand { get; set; }

/// <summary> This is a list of User objects that stores Users from the database.  </summary>
public List<User> Users { get; set; }

/// <summary> This UserName property is used to store the users input from the usernamebox from the login page. </summary>
public string UserName { get; set; }

/// <summary> This password property is used to store the users input from the passwordbox from the login page. </summary>
public string Password { get; set; }

/// <summary> This property is used to check whether a username exists in the database </summary>
public bool UserDoesNotExist { get; set; }

/// <summary> This property is used to check whether a password matches the password of the user in the database. </summary>
public bool PasswordIsIncorrect { get; set; }

/// <summary> This static LoggedInUserId stores the ID of the user who logs in. </summary>
public static int LoggedInUserId { get; set; }

/// <summary>  This property stores the user, who logs in, as a static object. </summary>
public static User LoggedInUser { get; set; }

public string BackgroundImage { get; set; }
```

Via en Foreach løkke sammenligner vi vores UserName og Password med user.UserName og user.UserPassword for hver bruger i Users, som er en liste af User-objekter, som vi får fra UserViewModellens konstruktør, der henter den fra databasen gennem DatabasePersistency.

```
public UserViewModel()
{
    LoginCommand = new RelayCommand(Login);
    Users = DatabasePersistency.LoadUsers().Result.ToList();
}
```

Hvis der er 1 match, sættes userDoesNotExist til false.
Dernæst tjekker vi, om Password og user.UserPassword matcher.
Hvis det gør, sættes passwordIsIncorrect til false.
Dermed må man konkludere, at brugeren eksisterer i databasen og frit kan tages videre.
Hvis UserName eller Password ikke matcher, vises en MessageDialog i UI, der fortæller brugeren hvilken del der var forkert.

Vi sætter LoggedInUserId til at være det samme som user.UserId og har dermed et statisk brugerId for den bruger der er logget ind, som vi kan bruge andre steder i programmet.

For at kunne bruge et objekt af typen User, rundt om i systemet, definerer vi en statisk property af typen User, som vi kalder LoggedInUser. Vi sætter værdien af denne property ved at hente vi et objekt fra vores database gennem DatabasePersistency. Dette gør vi ved at kalde metoden “GetSingleUser()” og giver det UserId, der er tilknyttet den bruger der er logget ind, som

argument til metoden. Herefter kalder vi “Set”-metoden på vores property til at sætte værdien til at være det hentede objekt.

Vha. denne bruger kan vi fx sætte LoggedInUserName i MainPageViewModel til at være lig vores LoggedInUser.UserName og vise brugerens navn i vores splitview i UI’en.

Endelig navigerer vi videre til vores MainPage ved at udskifte vores nuværende vindue med MainPagens frame.

LogOut()

Logout-systemet fungerer ved, at brugeren skal navigeres tilbage til LoginPage samt at properties, der holder på informationer omkring den bruger der loggede ind, skal nulstilles.

Dette fungerer ved, at når der klikkes på “Log Out”-knappen i vores splitview i UI’et, kaldes LogOut() metoden i MainPageViewModel.
Denne metode begynder med at sætte de to statiske properties i UserViewModel, LoggedInUser og LoggedInUserId, til henholdsvis null og 0. Derved er brugeren logget ud.

Herefter navigeres der på samme måde som i Login() metoden.

Herunder ses LogOut() metoden:

```
    /// <summary> This method logs the user out and returns the user to the login page </summary>
    public void LogOut()
    {
        //Logging User Out
        UserViewModel.LoggedInUser = null;
        UserViewModel.LoggedInUserId = 0;

        //Navigation
        Frame loginFrame = Window.Current.Content as Frame;
        if (loginFrame != null)
        {
            loginFrame.Navigate(typeof(LoginPage));
        }


    }
}
```

Test

Under implementeringen er dette system blevet testet manuelt for at sikre, at Login/logout-systemet overholder de regler, vi har sat for brugernavne og password, og hvor godt det kører.

Da systemet stod rimelig færdigt, kørte vi derudover nogle UnitTests for at være helt sikre.
Vi testede, om Login() metoden kunne kende forskel på korrekte og forkerte usernames og passwords samt om LogOut() metoden nulstillede de properties den skulle og derved loggede brugeren ud.

Et par eksempler på disse tests kan ses herunder.

```
private UserViewModel userViewModel = new UserViewModel();

[TestMethod]
public void UserNameTest()
{   // Test whether the system can recognise a correct Username
    userViewModel.UserName = "Hans";
    userViewModel.Password = "12345678";
    userViewModel.Login();
    Assert.IsFalse(userViewModel.UserDoesNotExist);   }

[TestMethod]
public void UserNameTest2()
{   // Test whether lower case or upper case letters influence the UserName,
    // as well as if the system checks if the password is correct.
    userViewModel.UserName = "hans";
    userViewModel.Password = "12345678";
    userViewModel.Login();
    Assert.IsTrue(userViewModel.UserDoesNotExist); }

[TestMethod]
public void PasswordTest()
{   // Test whether the method can recognise a correct password
    userViewModel.UserName = "Hans";
    userViewModel.Password = "12345678";
    userViewModel.Login();
    Assert.IsFalse(userViewModel.PasswordIsIncorrect); }

[TestMethod]
public void PasswordTest2()
{   // Test whether the method checks if the password is wrong.
    userViewModel.UserName = "Hans";
    userViewModel.Password = "123456789";
    userViewModel.Login();
    Assert.IsTrue(userViewModel.PasswordIsIncorrect); }
```

Resultat

Under implementeringen af disse to user stories, Login og Logout, stødte vi på et par ting til overvejelse, fx da vi skulle finde ud af, hvordan vi skulle håndtere den bruger der var logget ind. Overvejelsen gik på, om det smarteste ville være blot at gemme brugerens UserId i en static property, eller om det ville være bedre at hente selve brugeren fra databasen og gemme hele objektet i en static property.
Efter et par forsøg med det valgte vi simpelt hen at gøre begge dele.

Resultatet af disse user stories blev et udmærket Login/Logud-system, der med succes kan holde data fra forskellige brugere adskilt. Desuden er alle acceptance criteria overholdt.

Scrum review

I slutningen af sprintet indkaldte vi vejleder Henrik Høltzer til review på koden fra user stories: “Profil: Login” og “Profil: Logud”.
Under mødet blev der fundet en mere effektiv måde at loade brugeroplysninger på, som gjorde load af programmet hurtigere.
Herefter blev der kigget på Unit test, hvilket vi havde haft problemer med, men vi fandt ikke umiddelbart en løsning.

Der blev også indkaldt til møde med product owner, som fik fremvist programmet. Han var stort set tilfreds med vores GUI design, men havde et par ønsker til forbedring af layout og funktionalitet på Kampagneoversigts-siden.
Forslaget var enten at sørge for, at brugeren kan flytte kampagne-elementer rundt i gridviewet, eller kun få vist kampagner med navn og først ved klik på et element få fremvist kampagne-detaljer. Således ville oversigten blive mere overskuelig.
Dette blev tilføjet som task til user story “Kampagner: Kampagneoversigt”.

Scrum Retrospective

I slutningen af dette sprint var det efterhånden blevet klart, at tidsestimeringen af vores user stories var utilstrækkelig. Vi havde i høj grad undervurderet den nødvendige tid, de enkelte user stories ville tage at færdiggøre.
Vi mener, at dette skyldes flere ting: At vi ikke har taget ordentligt højde for tegning af diagrammer, test og unit test af user stories og rapportskrivning.
Vores fejlagtige estimering af tid resulterede også i, at nogle gruppemedlemmer følte sig presset under dette sprint, hvilket bidrog til en negativ stemning i teamet.
På mødet blev det derfor aftalt, at vi fremover begrænser scope for de enkelte sprints.

Resultatet af sprintet

Dette sprint har været det, hvor vi arbejdede mest efter SCRUM metoden. Vi var i første sprint meget langt fra konceptet, og i det næste var der nogle ting, vi havde brug for at kode i fællesskab for at genopfriske viden og blive enige om design.
I dette sprint har arbejdet været meget mere selvstændigt, og vi har dermed kunnet udnytte tiden mere effektivt.
De fire user stories, der blev valgt til sprintloggen i sprint 1, blev færdiggjort, og vi fik implementeret GUI design til programmet, som gør det elegant og overskueligt.

Vi undervurderede dog stadig tiden for implementering af vores user stories, og dermed nåede vi ikke at påbegynde de tre nye user stories, som vi udvalgte i begyndelsen af sprintet.

Sprint 3

Af Laura Vilen og Kristian Hornbøll

Sprint Planning

I de første to sprints måtte vi konkludere, at vores tidsestimering af vores user stories i langt de fleste tilfælde var meget i underkanten.
I dette sprint valgte vi derfor at gå væk fra vores tidsestimering vha. story points og i stedet nøjes med at udvælge tre user stories, en til os hver, som vi og vores product owner vurderede var vigtigst.

De tre var:

- NPC'er: NPC-oversigt
- PC'er: PC-oversigt
- Kampagner: Kapiteloversigt

I det følgende afsnit vil vi gå i dybden med design, implementering og test af én af disse user stories. For detaljer om de to andre henvises til bilag 6: Backlog.

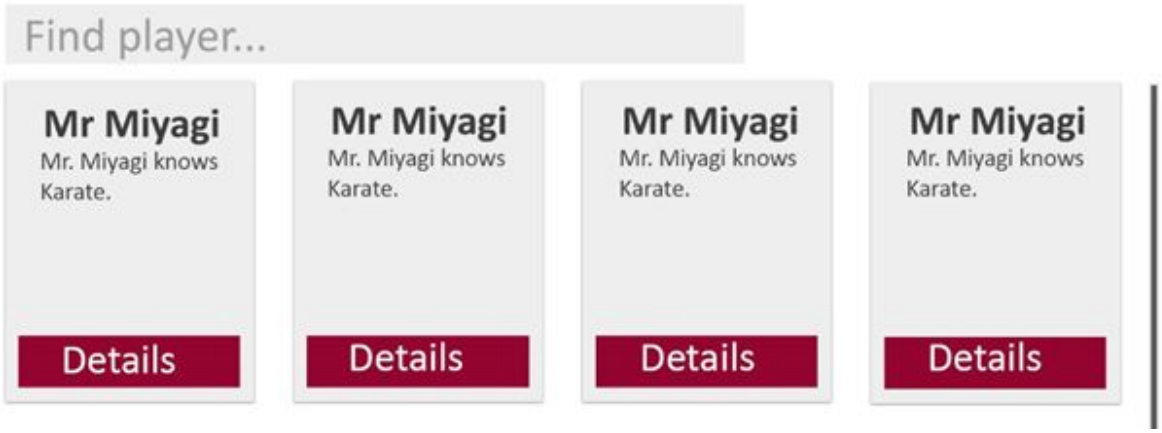
Userstory: PC-Oversigt

Af Kristian Hornbøll

"Som bruger vil jeg gerne have en oversigt over spillerkarakterer, så jeg bedre kan holde styr på plot og items, som er relevante for karakteren."

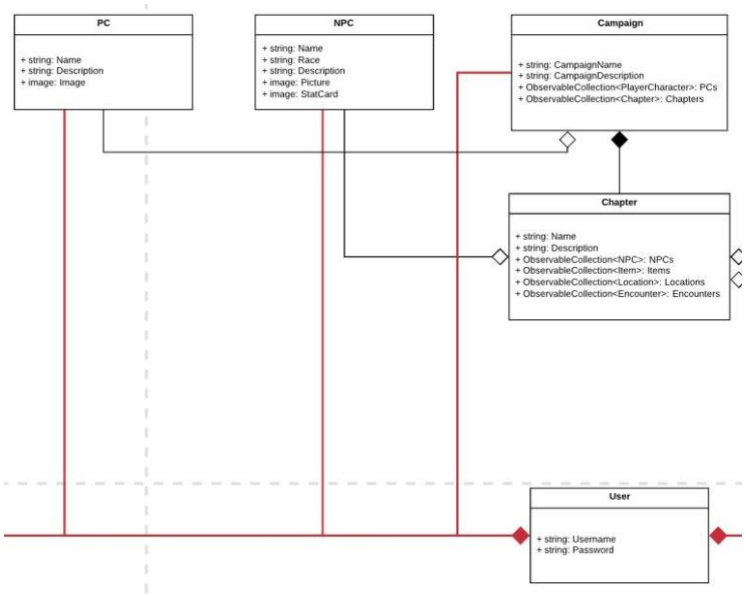
Spillerkarakteren er spillerens virtuelle figur. Som spilstyrer er det rart at have en lille idé om, hvad de figurer har af tilbehør, deres våben, ting, livspoint osv.

Design



Her er et par billeder af idéen, som er lavet med et billedprogram over den indledende player page plus et detaljevindue. Idéen er, at man efter navigering til siden skal se en liste af spillerkarakterer. Som standard vil alle spillere, som er tilknyttet en bruger, vises. Herfra kan man så søge på race, klasse, navn eller en kampagne for at se, hvilke karakterer der er tilknyttet en bestemt kampagne samt gå ind og rette i de tilknyttede informationer.

Domænemodellen



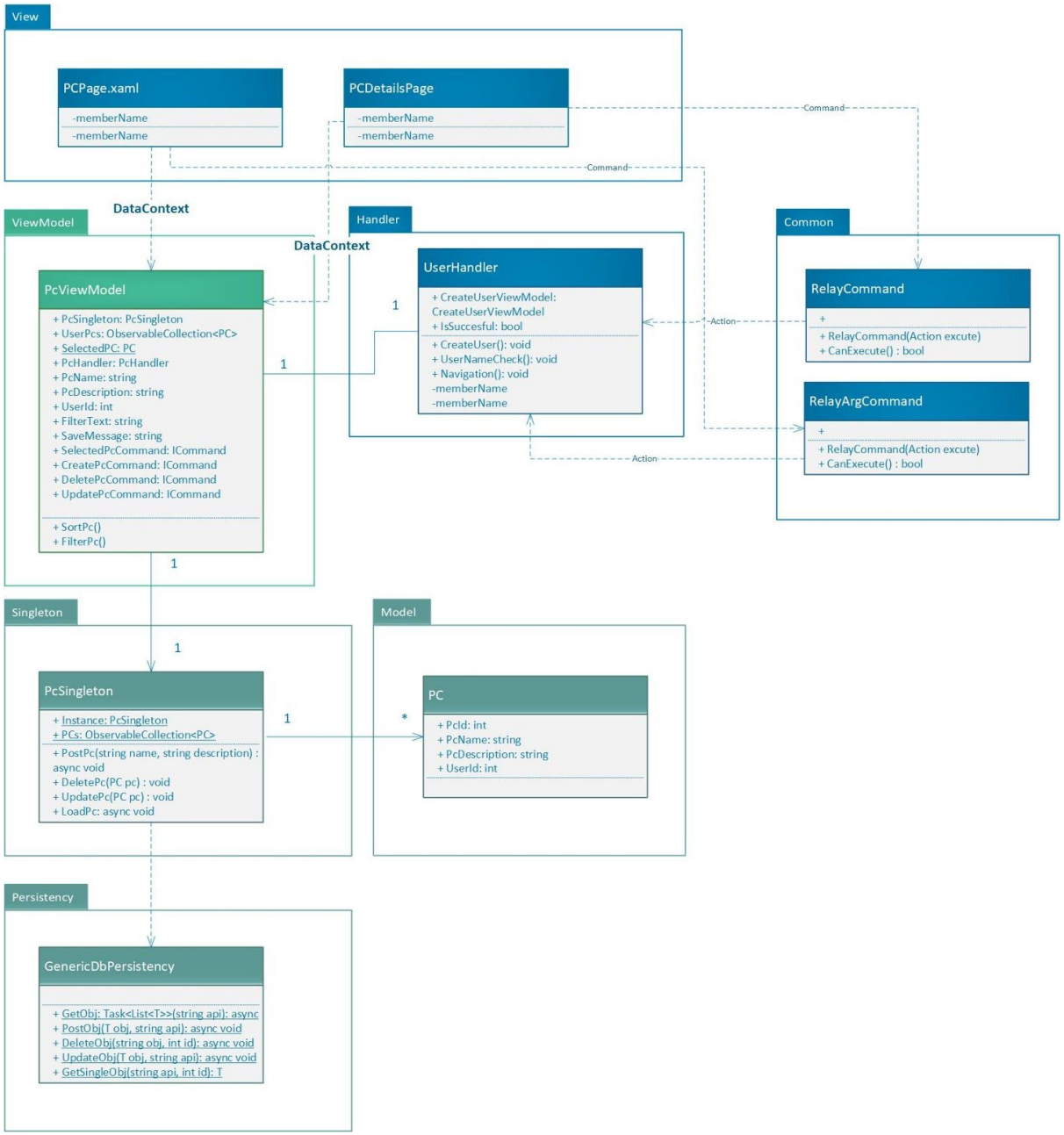
Ud fra vores indledende analyse/plan for projektet kan det i domænemodellen ses, at det er en bruger (User) der står for oprettelsen af en spillerkarakter (PC: Player Character), og derudover skal den enkelte spillerkarakter kunne knyttes til en bestemt kampagne.
En del af user story siger, at en spillerkarakter skal kunne tilknyttes en kampagne, hvori den skal kunne slettes igen, men uden at slette den fra den overordnede liste.

Jeg forestiller mig, at når en bruger opretter spillerkarakterer, vil de alle sammen, uanset bruger, tilgå den samme liste.

Med det i mente er min indledende plan at oprette en singleton, der indeholder en liste af spillerkarakterer samt nogle databasemetoder som kaldes, når der fx skal oprettes en ny spillerkarakter.

Klassediagram

Det endelige klassediagram for denne user story, som meget gerne skulle stemme overens med den indledende beskrivelse tidligere i afsnittet.



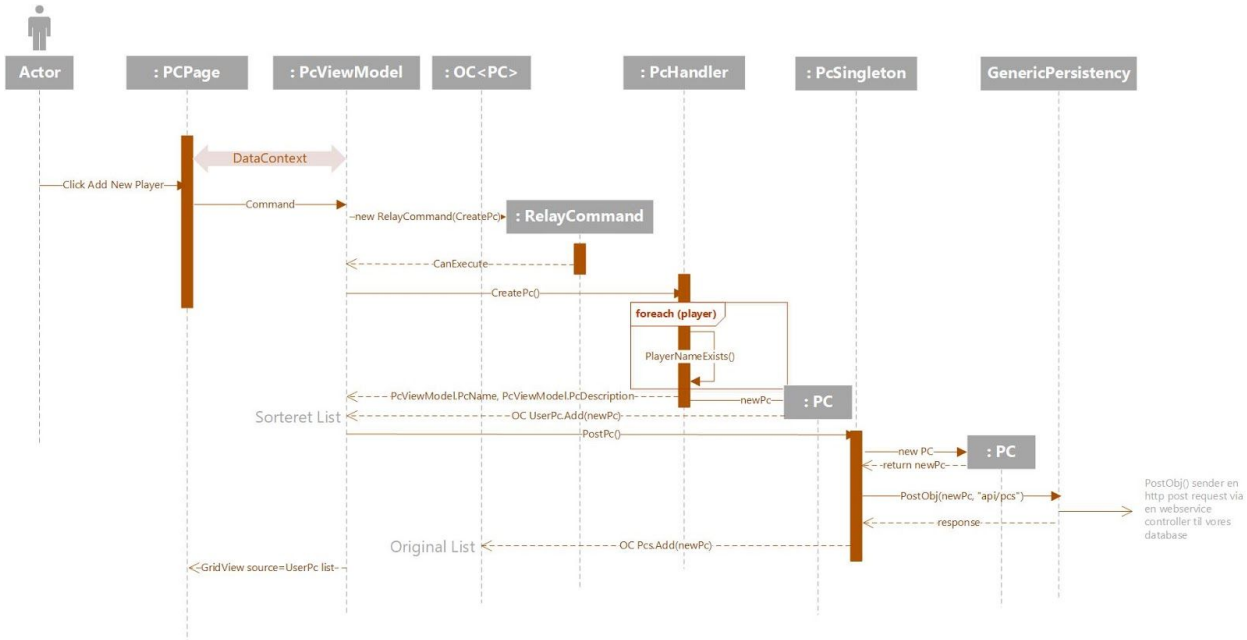
Implementering

Klikkes der på en knap i view, fx ”Add player”, sendes der via en ICommand i viewmodel en besked hen til en RelayCommand-klasse med metoden CreatePc som action. Dér finder RelayCommand-metoden ud af, om den kan køres. Hvis den kan, tjekkes der først i PcHandler klassen, om navnet der er indtastet findes i forvejen. Gør det ikke det, bliver det indtastede brugernavn og beskrivelse sendt videre til PostPc-metoden i PcSingleton, hvor der bliver instantieret en ny PC med disse oplysninger + det BrugerId der er logget ind med.

Så sendes der en anmodning til databasen gennem vores generiske persistens-metode PostObj(), og er den succesfuld og bliver føjet til databasen, tilføjes den også til vores ObservableCollection af spillerkarakterer.

```
public bool PlayerNameExists()
{   ≤ 4ms elapsed
    foreach (var player in PcViewModel.UserPcs)
    {
        if (PcViewModel.PcName == player.PcName)
        {
            throw new Exception( message: "Player name already exists");
        }
    }
    return true;
}
```

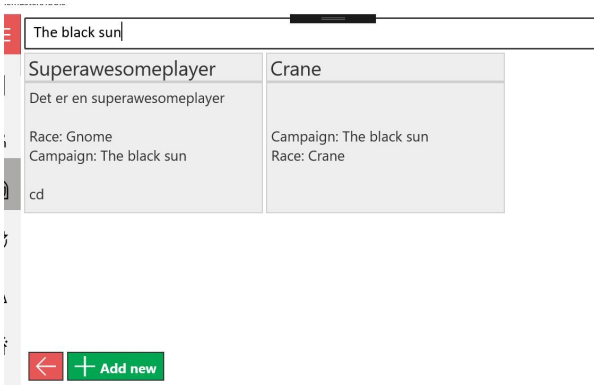
Sekvensdiagram af Create new Player



For kun at vise de spillerkarakterer der hører til den bruger som er logget ind, bliver der kørt en linq query på listen. Den går alle brugere igennem og tilføjer kun dem, som har det rigtige bruger-id.
På billedet vises både lambda-udtryk og den mere standard query syntaks.

```
public void SortPc()
{
    UserPcs = new ObservableCollection<PC>(PcSingleton.Pcs.Where(e => e.UserId == UserViewModel.LoggedInUserId));

    //var userPcs = from pc in PcSingleton.Pcs
    //    where pc.UserId == UserViewModel.LoggedInUserId
    //    select pc;


    //foreach (var pc in userPcs)
    //{
    //    UserPcs.Add(pc);
    //}

}
```

På samme måde har vi tilføjet en filtreringsfunktion, der gør det muligt at sortere listen efter navne eller andre specifikke ting fra beskrivelsen.




```
public void FilterPc()
{
    if (_filterText == null) _filterText = "";

    UserPcs = new ObservableCollection<PC>(PcSingleton.Instance.Pcs.Where(
        e =>
            e.UserId == UserViewModel.LoggedInUserId &&
            (e.PcName.ToLower().Contains(FilterText.ToLower()) ||
            e.PcDescription.ToLower().Contains(FilterText.ToLower()))
        ));
}
```

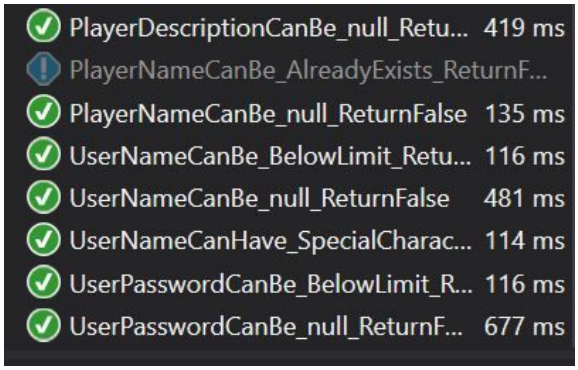
Tekstboksen som der søges i, er bundet til en property kaldet FilterText. Igen er der via en linq query først sørget for, at listen holdes korrekt i forhold til bruger-id. Derefter matches der mellem tekstboks og navn/beskrivelse, og hvis der er nogen af vores spillerkarakterer, der ikke indeholder det der filtreres efter, bliver de fjernet fra listen.

Test

For altid at kunne regne med at metoder nu også opfører sig som planlagt, når man begynder at ændre i dem eller andre steder i koden, laver vi unit test på en del af vores kode. Vi sætter nogle krav til vores metoder og tester så på de krav.

Unit Test af Create new Player: Her testes, om PcName kan være null. Inkluderet er også vinduet af resultatet af nogle af de andre tests.

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes
public void PlayerNameCanBe_null_ReturnFalse()
{
    // Arrange
    PcViewModel pcvm = new PcViewModel();
    // Add
    pcvm.PcName = null;
    pcvm.PcDescription = "A description";

    pcvm.PcHandler.CreatePc();

    // Assert
    Assert.IsFalse(pcvm.PcHandler.IsSuccesful);
}
```



Med unit tests konkluderet kan der sættes flueben ved den sidste del de tasks/kriterier, der har været til denne user story.

Scrum Review

I slutningen af sprintet kom Henrik Høltzer forbi for at vejlede os, og i den forbindelse kom vi frem til en yderligere specificering af kravene til vores rapport. Vi fandt ud af, at rapporten gerne skal indeholde:
- Dokumentation af vores kode
- Relevante kodeeksempler i rapporten
- Bevis på at vi håndterer exceptions ordentligt
- Et afsnit om arkitektur indeholdende ting som fx Generics
- Generelle billeder af selve programmet

Derudover fik Rasmus respons på sit sekvensdiagram og blev vist en bedre måde at lave dem på samt en henvisning til programmet Visio.

Scrum Retrospective

Ved sidste sprint havde gruppen talt om, at vi følte os meget pressede for at nå vores sprintlog og dermed fik skabt en stresset og negativ atmosfære.
Ved ikke at følge vores indledende tidsestimering og i stedet få begrænset scopet til noget simpelt og overskueligt under planlægning af sprint 3 mærkede vi hurtigt en forbedret stemning.
Det blev også bemærket, at gruppen havde fundet en god balance mellem hyggesnak og abejde. Vi var blevet langt mere fokuserede, men uden at det gik ud over den gode stemning i teamet.
Under dette sprint havde vi dog arbejdet meget selvstændigt på vores user stories og ikke programmeret eller diskuteret ret meget sammen, hvilket betød at flere i gruppen følte, at de manglede overblik over programmet som helhed.
Det blev derfor foreslået, at vi fremover bør have et afrundende møde sidst på dagen, hvor vi kan opdatere hinanden på design og implementering af vores user stories.

Resultatet af Sprintet

I dette sprint påbegyndte vi tre nye user stories. Vi havde på baggrund af de andre sprints været nødt til at revurdere vores tidsestimeringsmetode og forsøgt at begrænse den mængde arbejde vi skulle lave. Alligevel lykkedes det os ikke at blive færdig med alle de udvalgte user stories i dette sprint.
Vi mener dette skyldes et større fokus på rapportskrivning samt at de user stories vi har udvalgt - og vurderede som havende højest prioritering - rummer mange tasks og en stor arbejdsbyrde.

Men taget i betragtning at vi i løbet af dette sprint har fundet et mere stabilt arbejdstempo og føler, at vi er blevet mere effektive i vores arbejde, ser vi det ikke som et nederlag, at vi ikke er blevet helt færdige med de påbegyndte user stories.

Sprint 4

Af Laura Vilen

Sprint Planning

Sprint 4 var det sidste sprint vi havde til at færdiggøre vores projekt samt rapporten. Ved planlægning af sprintet blev det derfor besluttet, at hovedfokus skulle være på rapportskrivning samt at færdiggøre de user stories, der var påbegyndt, men ikke afsluttet, i sprint 3.
Det drejer sig om user story: “Kampagner: Kapiteloversigt” og “NPC: NPC oversigt”.

User story “Kampagner: Kapiteloversigt” og “PC: PC’er i Kampagne”

Ansvarlig: Laura Vilen

Dette afsnit går i dybden med User Story “Kampagner: Kapiteloversigt” og “PC: PC’er i Kampagne”, hvor implementering blev udført af Laura.

Kapiteloversigt

Som bruger
Vil jeg gerne kunne opdele mine kampagner i forskellige kapitler
Så det er mere overskueligt under spilgang

Acceptance Criteria:

- Kapitler skal kunne oprettes og slettes
- Kapitler skal gemmes i databasen
- Kapitler hører ind under en kampagne og skal have et unikt navn heri
- Brugeren skal vises en tydelig advarsel, før det er muligt at slette et kapitel

PC’er i Kampagne

Som bruger
Vil jeg gerne kunne tilknytte PC’er til en kampagne
Så jeg kun ser en liste over de PC’er, der er relevante for kampagnen, under spillet

Acceptance Criteria:

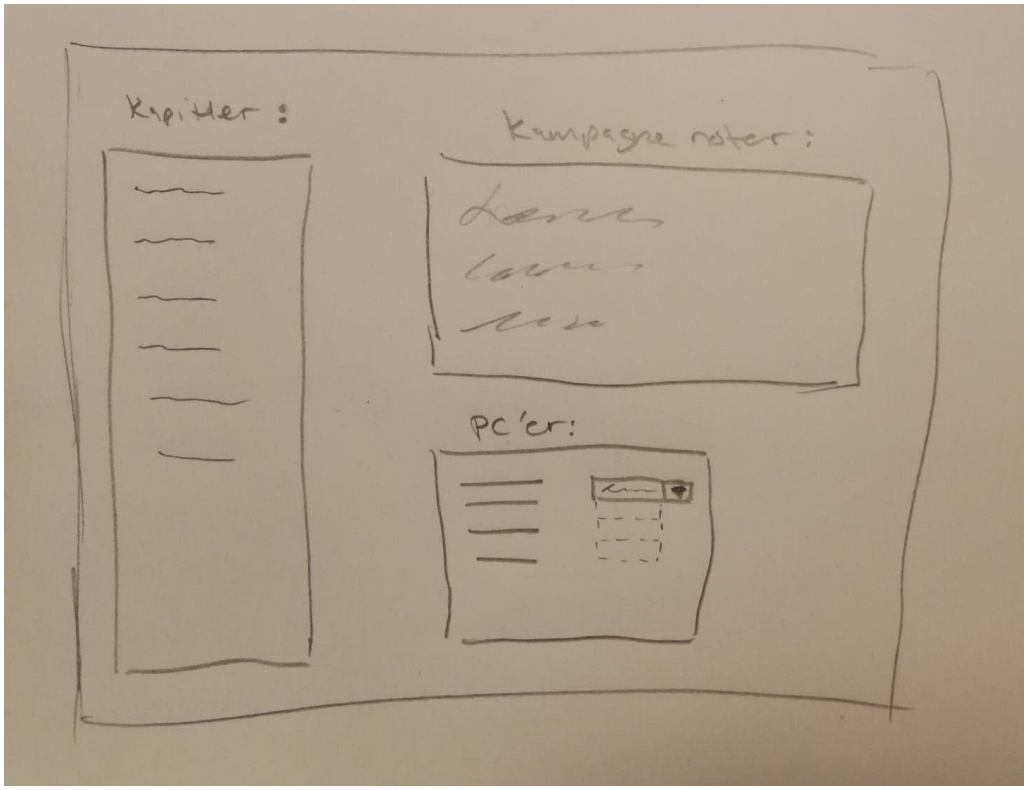
- Man skal kunne tilføje og fjerne PC'er fra en liste i kampagnen uden at slette PC'erne fuldstændigt

Design

Kampagneoversigten bygger delvist videre på en user story, der blev implementeret i et tidligere sprint: Kampagner: Kampagneoversigt. Denne user story vil der ikke blive gået i dybden med i denne rapport. For beskrivelse af user storien, sekvens- og klassediagram henvises der til bilag #, # og #.

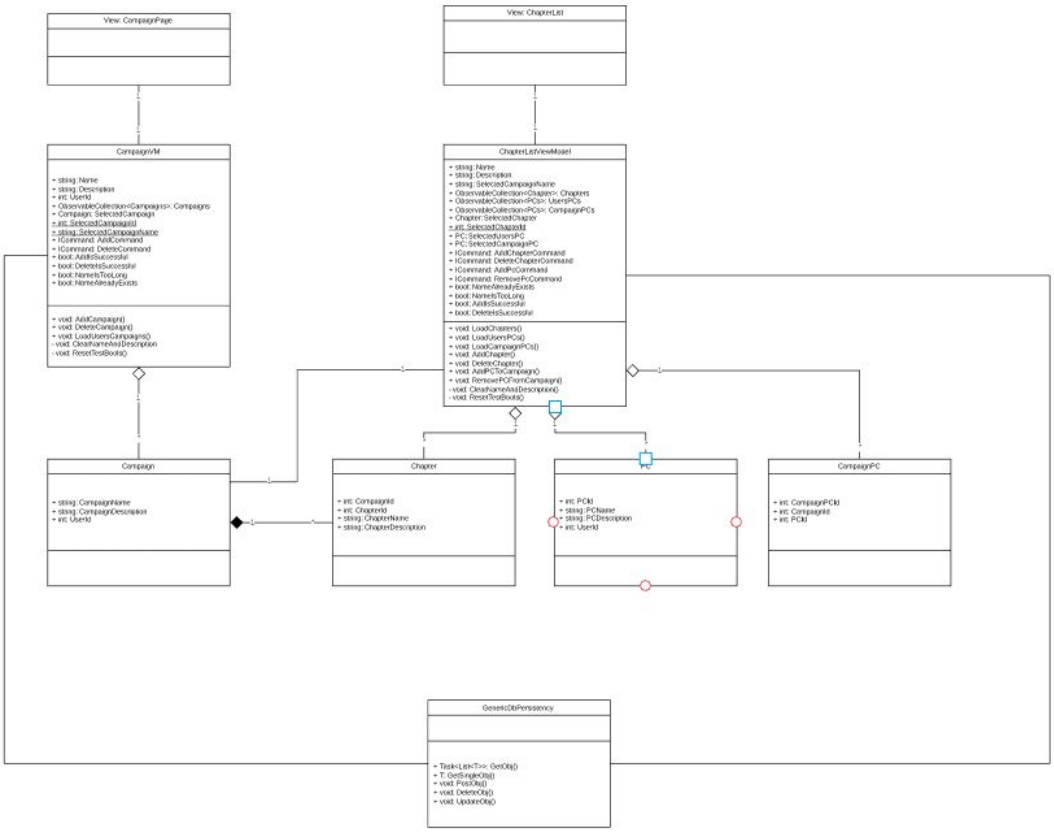
I programmet vises Chapter-klassen som en del af Campaign-klassen. En kampagne består af mange kapitler, og et kapitel er det element, der indeholder historie og detaljer for de enkelte spilgange.

En skitse af ChapterList-siden i designfasen:



Den ene halvdel af siden er dedikeret til detaljer om kampagnen og en liste af PC'er tilknyttet kampagnen, mens den anden halvdel indeholder en liste af kapitler samt mulighed for at oprette og slette kapitler og navigere videre til kapiteldetaljer for et valgt kapitel.

I et klassediagram ser det således ud:



Programmets CampaignPage er det view brugeren ser, når han vælger “Campaigns”-menuknappen. I dette view kan der vælges en kampagne, hvorfra der navigeres videre til en Kapiteloversigt.
Denne side er ChapterListPage, som indeholder en liste af kapitler, detaljer for den valgte kampagne og en liste af PC’er (playable characters) - altså spilleres karakterer - så brugeren har et overblik over de kapitler og spillere, der findes i en kampagne.

Den førnævnte sammenhæng vises mere tydeligt med et sekvensdiagram for sidens initialisering:

Når man vælger en kampagne fra viewet “CampaignPage”, vil man kunne navigere videre til det view der hedder “ChapterListPage”, som indeholder en oversigt over kapitler for den valgte kampagne. Dette view har datacontext til ChapterListViewModel, som ved initialisering loader

en liste af kapitler samt to seperate lister af PC'er - alle brugerens PC'er og en liste der er knyttet til kampagnen - gennem programmets persistency, der ved hjælp af webservice henter de relevante lister fra databasen.

Således ser initialiseringen af viewmodellen ud:

```
2 references | Laura Schlütter Vilen, 7 days ago | 1 author, 4 changes
public ChapterListViewModel()
{
    AddChapterCommand = new RelayCommand(AddChapter);
    DeleteChapterCommand = new RelayCommand(DeleteChapter);
    AddPcCommand = new RelayCommand(AddPCToCampaign);
    RemovePcCommand = new RelayCommand(RemovePCFromCampaign);
    SelectedCampaignName = CampaignVM.SelectedCampaignName;
    LoadChapters();
    LoadUsersPCs();
    LoadCampaignPCs();
    SelectedChapter = null;
    SelectedChapterId = 0;
    SelectedCampaignPC = null;

}
```

For at kunne knytte en PC til en eller flere kampagner har vi i databasen måttet lave et linking table mellem disse to tabeller: CampaignPC. Denne tabel indeholder tre integer værdier: En primær nøgle og to fremmednøgler, som peger på de førnævnte tabeller.
I vores program bruges denne klasse blot til at krydsreferere, hvilke PC'er der hører til hvilke kampagner, og muliggører at en PC kan tilhøre flere kampagner på samme tid.
Når der skal loades en liste af de PC'er der er tilknyttet kampagnen, bliver der således hentet en liste af klassen "CampaignPC", der bruges til at filtrere en liste af brugerens PC'er og tilføjer dem til en liste.
Der bliver dog også hentet den fulde liste af brugerens PC'er, således at brugeren fra oversigten kan føje flere spillerkarakterer til kampagnen.

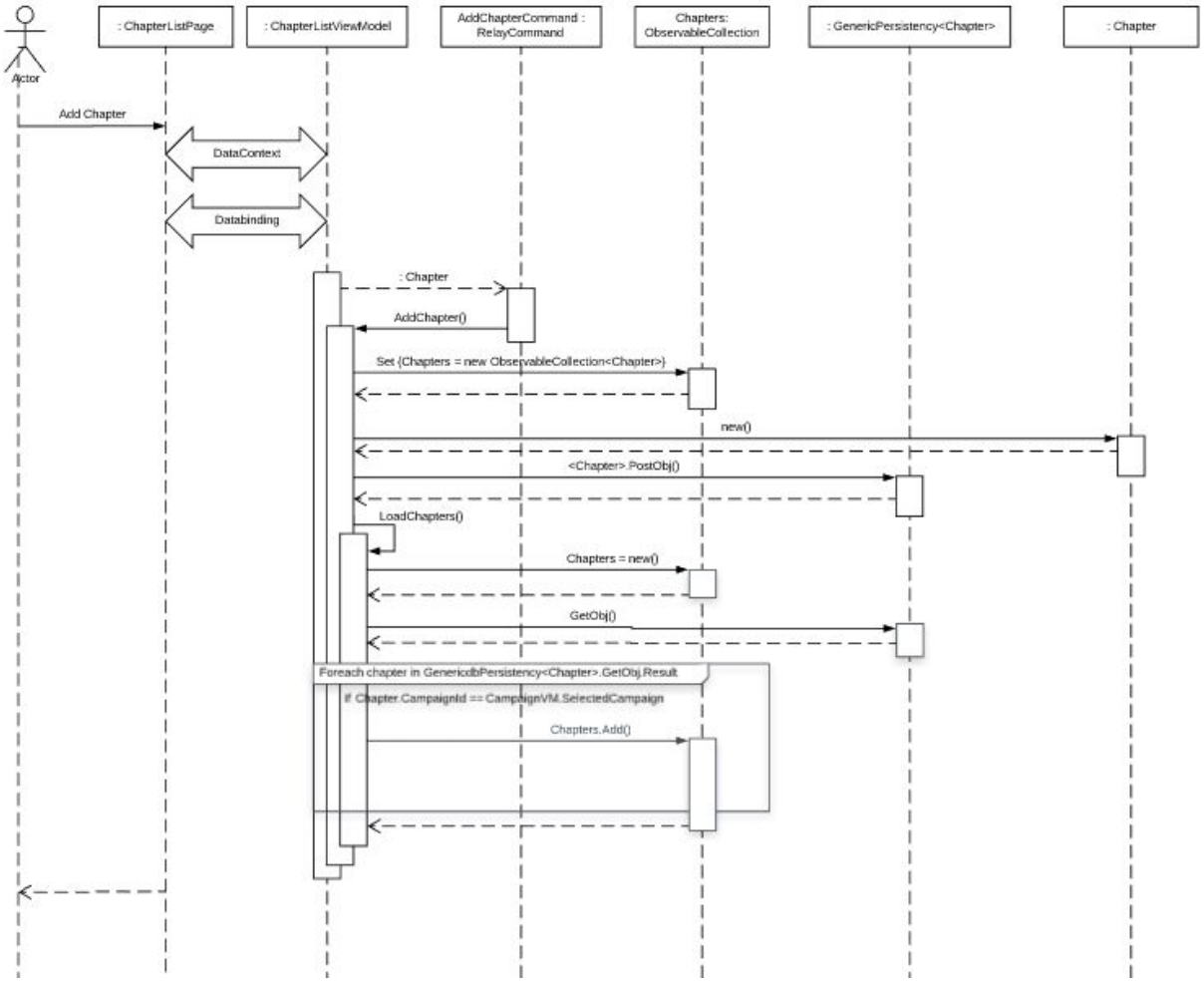
Load af kampagnespecifikke PC'er ser således ud:

```
3 references | Laura Schlütter Vilen, 8 days ago | 1 author, 1 change
public void LoadCampaignPCs()
{
    CampaignPCs = new ObservableCollection<PC>();

    foreach (var campaignPC in GenericDbPersistency<CampaignPC>.GetObj(api: "api/CampaignPCs").Result)
    {
        if (campaignPC.CampaignId == CampaignVM.SelectedCampaignId)
        {
            foreach (var pc in GenericDbPersistency<PC>.GetObj(api: "api/pcs").Result)
            {
                if (pc.PcId == campaignPC.PCId)
                {
                    CampaignPCs.Add(pc);
                }
            }
        }
    }
}
```

Metoden går således ind og henter en liste af CampaignPC fra databasen, og ved hjælp af denne liste laves der med et foreach-loop en filtrering af PC'er fra databasen. De PC'er der matcher kampagne-id'et, bliver tilføjet en liste, og denne liste vises i sidens view.

På ChapterList-siden er der fire forskellige funktioner, brugeren kan benytte sig af: Oprette kapitler, slette kapitler, tilføje PC'er til kampagnen og fjerne PC'er fra kampagnen.

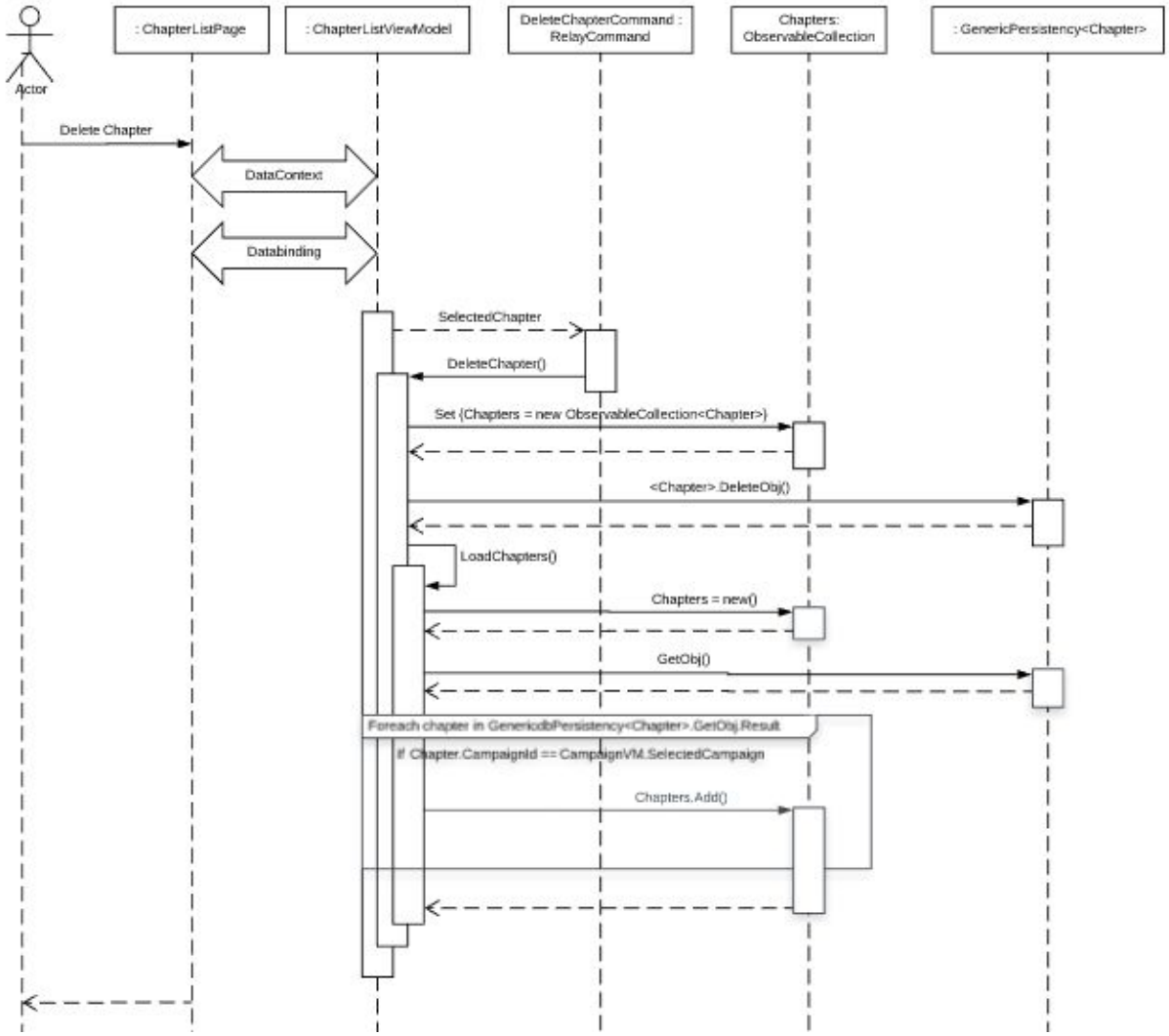
Opret kapitel:



Ved tilføjelse af et kapitel skal brugeren angive et navn. Herefter trykker brugeren på "add"-knappen, som er bundet til en RelayCommand, hvis funktion er AddChapter metoden defineret i viewmodellen.
Denne metode genererer et objekt af Chapter-klassen med det navn brugeren har angivet - så længe navnet ikke eksisterer i kampagnen i forvejen - og gennem persistency gemmes objektet i databasen.
Herefter bliver listen af kapitler loadet på ny fra databasen, så listen opdateres med det nye kapitel brugeren har skabt.

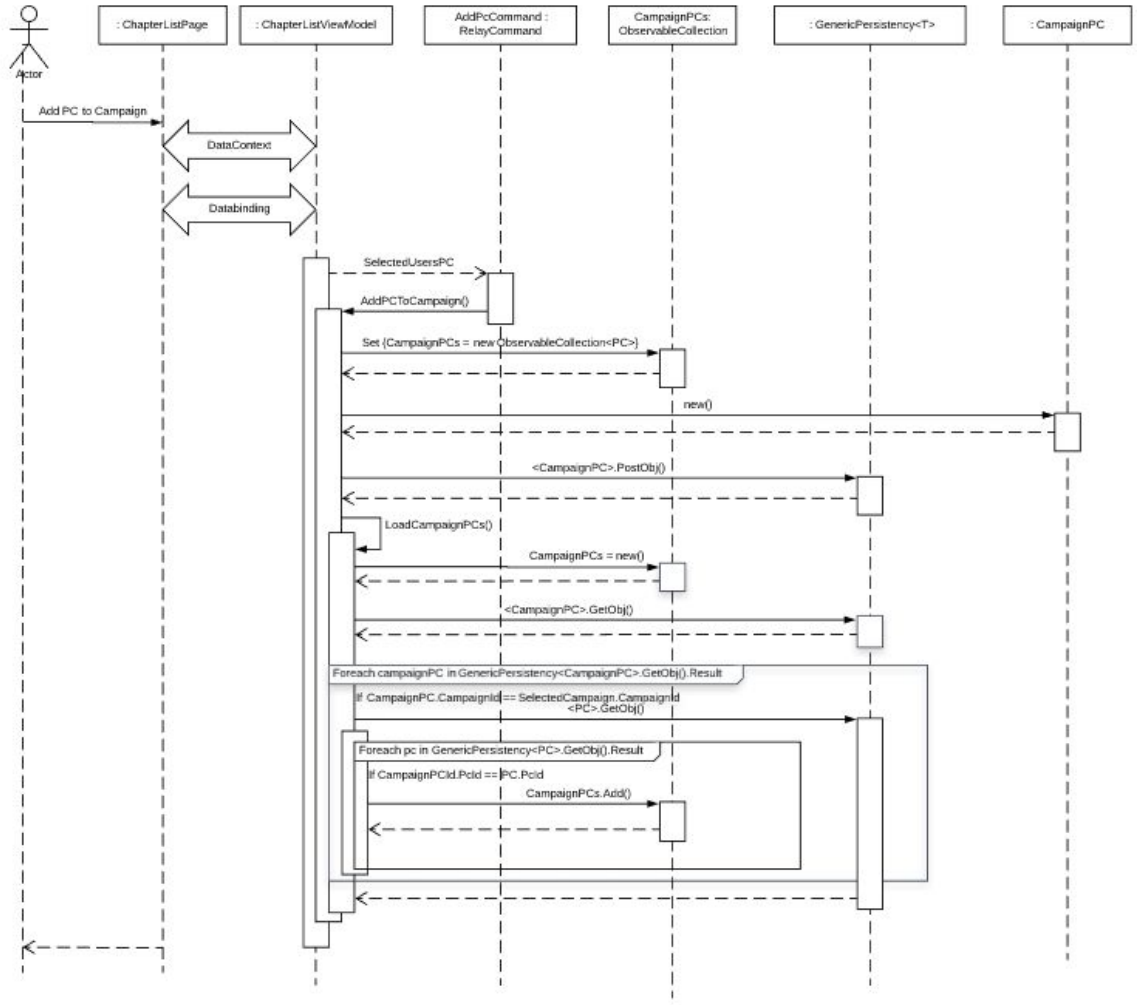
```
3 references | 0/1 passing | Laura Schlütter Vilen, 8 days ago | 1 author, 3 changes
public void AddChapter()
{
    NameAlreadyExists = false;
    CreateChapterIsSuccessful = false;

    foreach (var chapter in Chapters)
    {
        if (Name == chapter.ChapterName)
        {
            NameAlreadyExists = true;
        }
    }

    if (NameAlreadyExists == false)
    {
        GenericDbPersistency<Chapter>.PostObj(new Chapter(Name, Description, CampaignVM.SelectedCampaignId), api: "api/Chapters");
        LoadChapters();
        CreateChapterIsSuccessful = true;
    }

    else
    {
        ChapterListViewModel.MessageDialogHelper.Show(
            content: "You already have a campaign with this name. Please choose a unique name for your campaign.",
            title: "Invalid campaign name");
    }

    ClearNameAndDescription();
}
```

Slet kapitel:



Når brugeren vælger et kapitel og trykker på slet-knappen, vil han få en besked hvor han bliver bedt om at bekræfte sletningen.
Giver brugeren et bekræftende svar, vil en RelayCommand udføre en metode i viewmodellen, som kaldes Deletemetoden i persistency, og som sletter det valgte objekt fra databasen. Herefter bliver kapitellisten hentet på ny fra databasen og opdateres i viewet, så brugeren kan se at kapitlet er slettet.

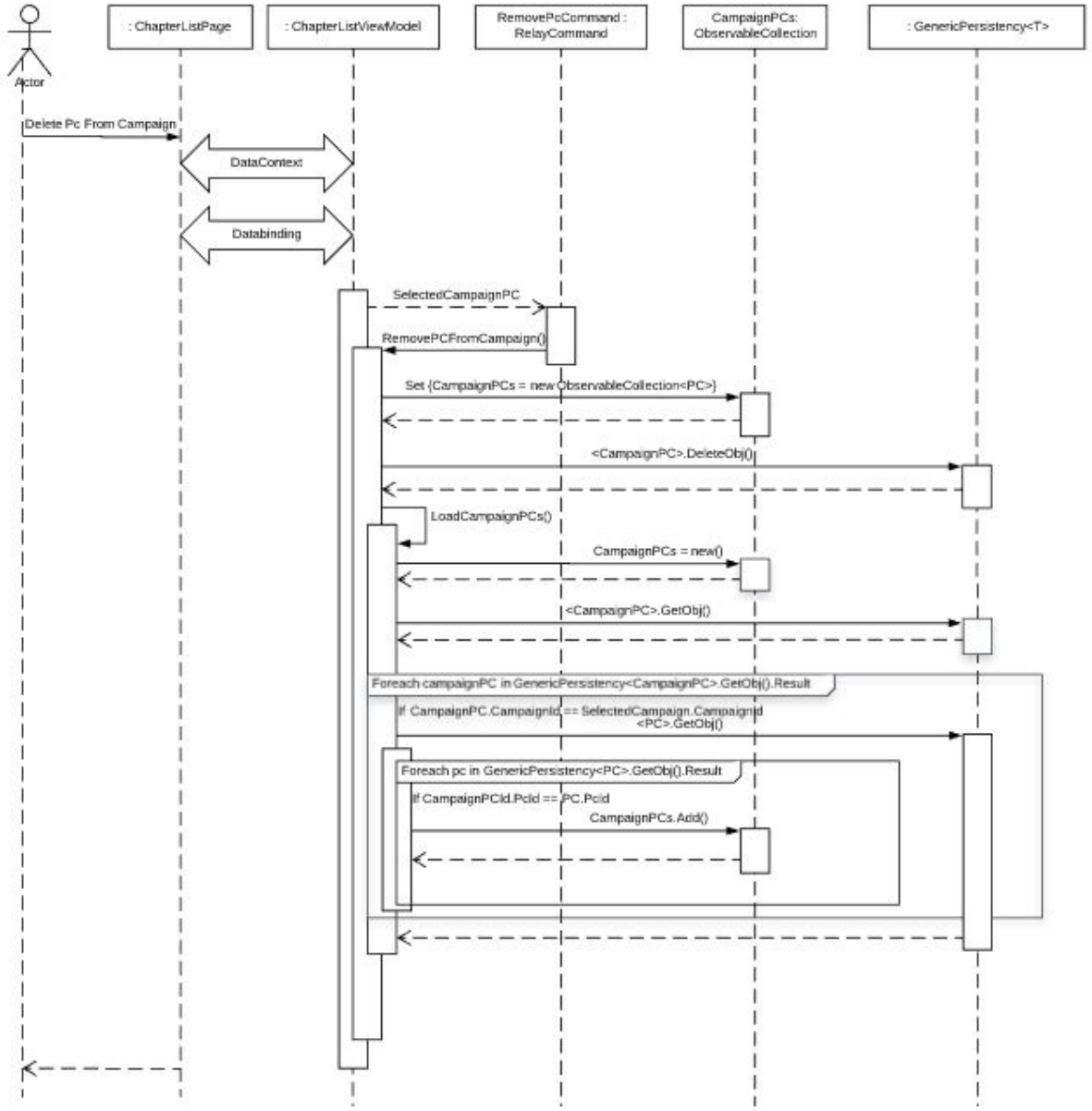
Tilføj spillerkarakterer til kampagne:



Brugeren kan tilføje PC'er til en kampagne ved at vælge fra en liste af brugerens PC'er. Klikker brugeren tilføj, kalder en RelayCommand AddPCToCampaign-metoden i viewmodellen.
Denne metode genererer et nyt objekt af CampaignPC-klassen med den valgte PC's Id og Id'et for den kampagne, brugeren er inde på, som argumenter.
Objektet bliver herefter sendt videre til persistency, som gemmer objektet i databasen.
Herefter loades listen over de PC'er, der er tilknyttet kampagnen, på ny og vises i viewet.

```
1 reference | Laura Schlütter Vilen, 8 days ago | 1 author, 1 change
public void AddPCToCampaign()
{
    if (SelectedUsersPC != null)
    {
        GenericDbPersistency<CampaignPC>.PostObj(new CampaignPC(CampaignVM.SelectedCampaignId, SelectedUsersPC.PcId), api: "api/CampaignPCs");
        LoadCampaignPCs();
        SelectedUsersPC = null;
    }

    else
    {
        MessageDialogHelper.Show( content: "Please select a PC to add", title: "No Pc selected");
    }

}
```

Fjern PC'er fra kampagne:



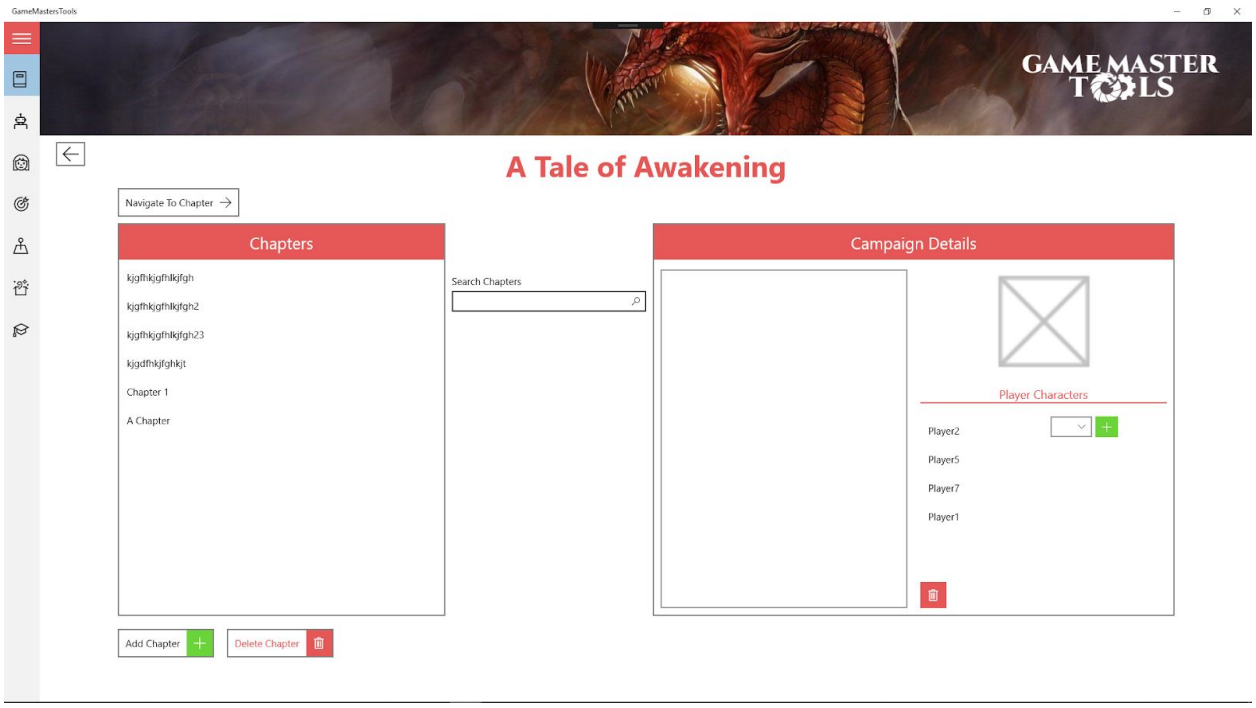
En bruger kan ud over at tilføje PC'er til en kampagne også fjerne dem igen. Når brugeren trykker på knappen til dette i viewet, vil den tilknyttede RelayCommand kalde RemovePCFromCampaign metoden i viewmodellen.
Denne metode sletter ikke et PC-objekt, men derimod et CampaignPC-objekt, som jo er den klasse der binder PC'er og Campaigns sammen. Dermed kan en PC altid tilføjes eller fjernes fra listen i en kampagne, men brugeren har stadig data på disse karakterer liggende til fremtidig brug.

Resultat

Implementering af kapitellisten for en kampagne var simpelt nok. Udfordringen i denne user story var at forbinde brugerens spillerkarakterer til forskellige kampagner uden at låse PC'en til en kampagne og dermed blive slettet fra brugerens data, hvis kampagnen skulle blive slettet.

Løsningen blev at lave linking tables i databasen mellem campaign og PC og i koden bruge objekter af CampaignPC til at filtrere en liste af PC'er. Denne måde at håndtere opgaven betyder, at vi kan oprette og slette objekter af CampaignPC uden at påvirke de andre tables, hvilket igen betyder at PC'er kan genbruges i forskellige kampagner.

Ved afslutning af sprintet ser den del af programmet, som user story "Kampagne: Kapiteloversigt" og "PC: PC'er i Kampagne" dækker over, således ud:



Følgende tasks og krav er opfyldt:

- Liste af kapitler
- Liste af PC'er for kampagnen
- Oprettelse og tilføjelse af kapitler og PC'er til en kampagne
- En kampagne kan ikke have to kapitler med samme navn
- Slet og fjern kapitler og PC'er med relevante advarsler inden slet for at minimere fejlklik
- Kapitler og kampagne-PC'er gemmes i databasen

Scrum Review

I slutningen af sprint 4 har vi indkaldt vejleder Jamshid Eftekhari til review af dele af vores rapport.
Her fik vi feedback på vores problemformulering, som blev uddybet og godkendt.
Derudover fik vi feedback på følgende afsnit af rapporten:

- Projektetablering: Projektbeskrivelse
- Vision Statement: Size it Up
- Virksomhedsanalyse
- Domænemodel og ER-diagram

Vi gennemgik også kort vores brugerflade i programmet, som vurderes som næsten færdigt.

Der blev desuden reflekteret over product owners rolle. Vi diskuterede muligheden for, at en af gruppemedlemmerne kunne have ageret product owner frem for Kasper Gregersen, som det har været svært at holde kontakt med og indkalde til møder.

Retrospective

Under retrospective diskuterede gruppen mest rapportskrivning. Der var en generel enighed om, at vi ikke havde været særligt gode til at undersøge kriterierne for rapporten tidligt i projektet, og derfor havde vi i dette sprint ekstra arbejde med at skrive rapport sammenlignet med andre sprints.

Resultatet af Sprintet

I dette sprint var vores hovedfokus at få færdiggjort igangværende user stories samt rapportskrivning
Vi har nået de tre user stories, der blev igangsat ved starten af sprint 3, og har efterhånden et godt basisprogram i forhold til vores målsætning. Vi har dog ikke haft mulighed for at påbegynde og færdiggøre nogen nye user stories grundet det ekstra arbejde med rapporten, som har været påkrævet denne uge.

Refleksion

I løbet af projektet er vi stødt på flere udfordringer og har måttet foretage justeringer og ændringer af vores produkt og arbejdsgang.

Den første faktor der er værd at tale om, er vores programs scope. Vi havde en lang featureliste og fik udarbejdet en meget stor backlog for programmet i etablerings- og designfasen. Vi besluttede os dog for at afgrænse scope til en mindre og mere fleksibel størrelse undervejs.
Det havde muligvis været fordelagtigt for teamet og projektet, hvis vi havde valgt at afgrænse projektet fra starten - altså kun udarbejdet de user stories vi regnede med at kunne nå - frem for at bruge tid og energi på at formulere tasks og krav til user stories, som vi vidste der alligevel ikke ville være tid til.

Derudover definerede vi i begyndelsen af projektet en tidsestimeringsmetode til at vurdere arbejdsbyrden for hver enkelt user story.
Her sagde vi, at 5 story points svarede til ca. en dags arbejde for et gruppemedlem, og efterfølgende vurderede vi hver enkelt user story med et antal story points for at bedømme, hvor meget af backloggen vi ville kunne nå i løbet af projektperioden på 5 uger.
Undervejs blev det dog tydeligt for gruppen, at vores estimater lå langt i underkanten, og vi nåede meget mindre, end vi havde regnet med i de enkelte sprint.
Frem for at revidere vores estimeringsmetode eller de estimater vi havde lavet af de enkelte user stories, valgte vi at gå bort fra denne måde at håndtere sprintloggen på.

En af årsagerne til at vi ikke valgte at revidere vores estimater, har højst sansynligt været den mængde tid der i så fald skulle bruges på opgaven - den mængde af user stories vi havde tilføjet til backloggen taget i betragtning.

En anden udfordring i vores arbejde med scrum har været mangel på kontakt med vores product owner.
I begyndelsen af projektet kontaktede vi Kasper Gregersen, som vi fik til at agere product owner. Dette gjorde vi på basis af hans erfaring med lignende IT-løsninger og med spillet, Dungeons & Dragons.
Hvad vi dog ikke havde taget ordentligt under overvejelse, er det faktum at Kaspers arbejdstider falder sammen med vores skoletid, og at han har mange andre opgaver han skal tage sig af i hverdagen. Det blev derfor hurtigt et problem at indkalde ham til reviewmøder og få markeret user stories ‘endeligt færdige’.

Efter undersøgelse af hvordan andre virksomheder håndterer lignende problemer, ved vi dog nu at der kunne være fundet en anden løsning.

Kasper var meget behjælpelig i designfasen af programmet samt til at komme med forslag til features, men hans skema var presset, og han var derfor ikke altid tilgængelig.
Det havde været en mulighed at lade et af teammedlemmerne agere product owner, som så havde kontakt til Kasper, der kunne agere kunde.
Således ville det også give mening, hvis det var én fra gruppen - som alle tre er undervist i hvordan man skriver user stories - som skrev og prioriterede user stories til backloggen, og gruppen ville ikke have været så afhængig af Kasper, men havde kunnet bruge ham som ressource i stedet.

Konklusion

Gennem dette projekt har vi villet undersøge, hvordan en agil og iterativ arbejdsproces kunne bruges til at udvikle et IT-system, der kan hjælpe Game Masters, der afholder D&D, med at gøre forberedelse og styring af spillet nemmere.
Til dette formål har vi benyttet os af arbejdsprocessen Scrum. Scrum har gjort det muligt at tilpasse og ændre vores løsning undervejs samt gjort det nemmere at overskue en større mængde arbejde ved at dele det op i mindre bidder.

I vores projekt har vi fulgt MVVM struktur for vores kode, hvilket har gjort det lettere og simplere at overskue vores arbejde. Desuden har vi fulgt nogle design patterns for at sikre kvaliteten af vores kode i form af at genbruge løsninger, som allerede eksisterer og virker, frem for at opfinde vores egne. Ud over at kunne genbruge kode gennem hele programmet har det gjort det meget mere overskueligt både for os og andre, som skulle kigge med.

Vi har arbejdet i UWP app for at have mulighed for at benytte vores program på flere forskellige platforme og dermed sikre alsidig brug af vores produkt. Det betyder, at vores kunder vil have flere muligheder for at tilgå deres data og dermed nemmere kunne administrere deres kampagner.

Desuden lagres brugernes data i en database, hvilket sikrer at man kan tilgå sine data overalt og på flere forskellige platforme, hvilket igen gør programmet nemt og attraktivt at bruge.

Brugerne skal have direkte adgang til at læse og redigere data, de selv har lagret i databasen, og der kan ofte være mange data lagret her. Derfor vurderede vi meget tidligt, at brugerfladens vigtigste kvaliteter var enkelthed og overskuelighed. Til dette formål benyttede vi de 10 heuristikker for god UI-udvikling, når vi kodede sider i xaml og opdelte data i forskellige kategorier.

Vi har arbejdet med et lille scope for vores program, som blot skulle være et eksempel på et program, der kan hjælpe med det administrative arbejde for en Game Master, og med de ovenstående værktøjer og metoder er det lykkedes os at fremstille et sådant produkt.

Litteraturliste

Rasmusson, Jonathan: The Agile Samurai: How Agile Masters Deliver Great Software, 2010 (Bog)

Connelly, Thomas og Begg, Carolyn - Database Solutions - a step by step guide to building databases, Addison-Wesley, september 22, 2003

https://fronter.com/easj/links/files.phtml/5c74f3d2be421.1225660000$50278981$/Resources/Course+materials/Software+Design/Sprint+2/Slides/normalization.pdf - brugt d. 23-05-2019

DnDBeyond: Basic rules: Introduction. Udgivet af Wizards of the Coast. Internetadresse: https://www.dndbeyond.com/sources/basic-rules/introduction - Besøgt d. 26.05.2019 (Internet)

DnDBeyond: Basic Rules. Udgivet af Wizards of the Coast. Internetadresse: https://www.dndbeyond.com/sources/basic-rules - Besøgt d. 26.05.2019 (Internet)

Windows.UI.Xaml.Controls. Udgivet af Microsoft. Internetadresse: https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls - Besøgt d. 16.05.2019 (Internet)

LucidChart. , : *UML Class Diagram Tutorial.* 21.07.2017. Internetadresse: https://www.youtube.com/watch?v=UI6lqHOVHic&ab_channel=Lucidchart - Besøgt d. 24.04.2019 (YouTube)

LucidChart. , : *How to Make a UML Sequence Diagram.* 27.08.2018. Internetadresse: https://www.youtube.com/watch?v=pCK6prSq8aw&t=455s&ab_channel=Lucidchart - Besøgt d. 07.05.2019 (YouTube)

Bell, Donald : The sequence diagram. I: *IBM.com*, 16.02.2004, s. 1. Internetadresse: https://www.ibm.com/developerworks/rational/library/3101.html - Besøgt d. 23.05.2019. (Artikel)

Laursen, Per: Object-Oriented Programming With C# - A Gentle Introduction. I: *EASJ Notes*, 07.04.2018, s. 175-193. Internetadresse: http://www.heho-easj.dk/SWC1-2018e/SWC%20Ugeplan.html - Besøgt d. 10.09.2018. (Artikel)

Create Simple Web API In ASP.NET Web API. Udgivet af C# Corner. Internetadresse: https://www.c-sharpcorner.com/article/create-simple-web-api-in-asp-net-mvc/ - Besøgt d. 24.05.2019 (Internet)

Introduce Entity Framework With ADO.NET Entity Data Model. Udgivet af C# Corner. Internetadresse: https://www.c-sharpcorner.com/article/introduce-entity-framework-with-ado-net-entity-data-model/ - Besøgt d. 24.05.2019 (Internet)

Software Construction - SWC 1.2 2C (uge 11-12). Udgivet af Henrik Høltzer. Internetadresse: http://heho-easj.dk/SWC2-2019f/SWC2Ugeplan2019f.html - Besøgt d. 24.05.2019 (Internet)

Software Construction - SWC 1.2 2C (uge 12-13). Udgivet af Henrik Høltzer. Internetadresse: http://heho-easj.dk/SWC2-2019f/SWC2Ugeplan2019f.html - Besøgt d. 24.05.2019 (Internet)

HttpClient Class. Udgivet af Microsoft. Internetadresse: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=netframework-4.8 - Besøgt d. 24.05.2019 (Internet)

HttpClientHandler Class. Udgivet af Microsoft. Internetadresse: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclienthandler?redirectedfrom=MSDN&view=netframework-4.8 - Besøgt d. 24.05.2019 (Internet)

DbSet Class. Udgivet af Microsoft. Internetadresse: https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbset-1?view=entity-framework-6.2.0 - Besøgt d. 24.05.2019 (Internet)

10 Usability Heuristics for User Interface Design. Udgivet af Nielsen Norman Group. Internetadresse: https://www.nngroup.com/articles/ten-usability-heuristics/ - Besøgt d. 17.05.2019 (Internet)

INVEST. Udgivet af Agile Alliance. Internetadresse: https://www.agilealliance.org/glossary/invest - Besøgt d. 15.05.2019 (Internet)

New to agile? INVEST in good user stories. Udgivet af Agile For All. Internetadresse: https://agileforall.com/new-to-agile-invest-in-good-user-stories/ - Besøgt d. 15.05.2019 (Internet)

Shubho, Al-Farooque: How I explained Design Patterns to my wife: Part 1. I: *The Code Project*, 09.08.2010, s. 1. Internetadresse: https://www.codeproject.com/Articles/98598/How-I-explained-Design-Patterns-to-my-wife-Part - Besøgt d. 26.05.2019. (Artikel)

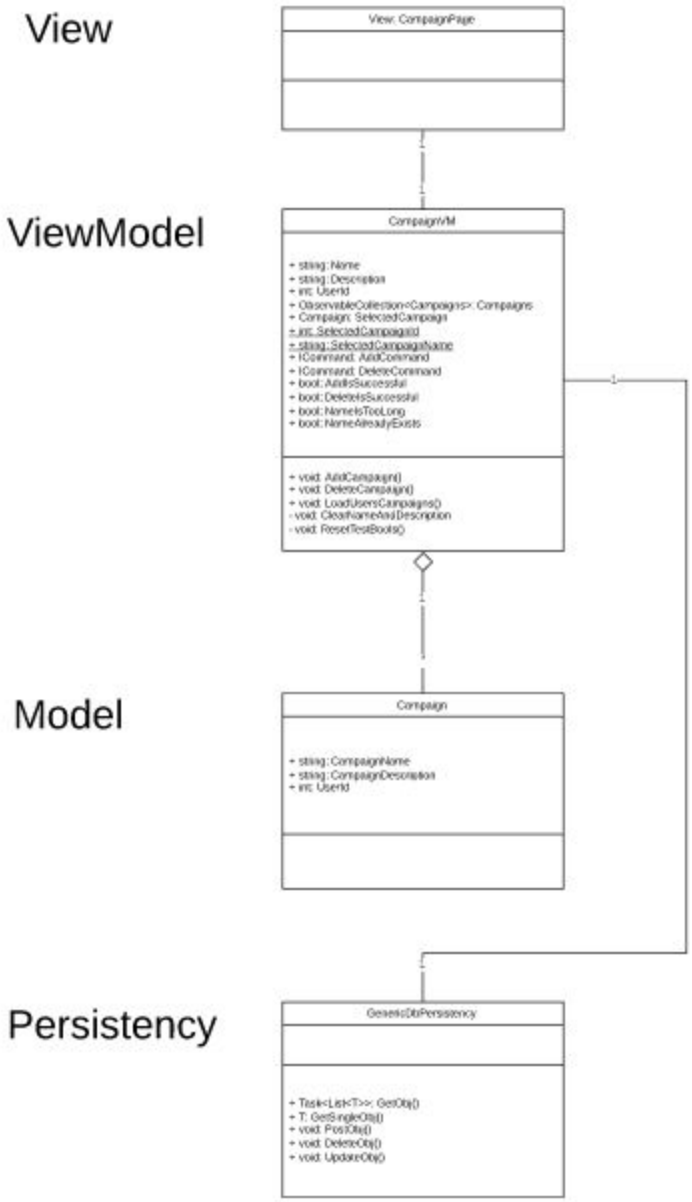
The 7 Most Important Software Design Patterns. Udgivet af Medium.com. Internetadresse: https://medium.com/educative/the-7-most-important-software-design-patterns-d60e546afb0e?fbclid=IwAR1F1a599zdCtB5GiQ473GQoEJ3hWE8FI-dpV0w7J3jYaAOnSlwWTWtPGRA - Besøgt d. 26.05.2019 (Internet)

Billede fra Henrik Høltzer's undervisning ang. Forbindelsen mellem MVVM, webservice og Database
Taget d. 15-03-2019

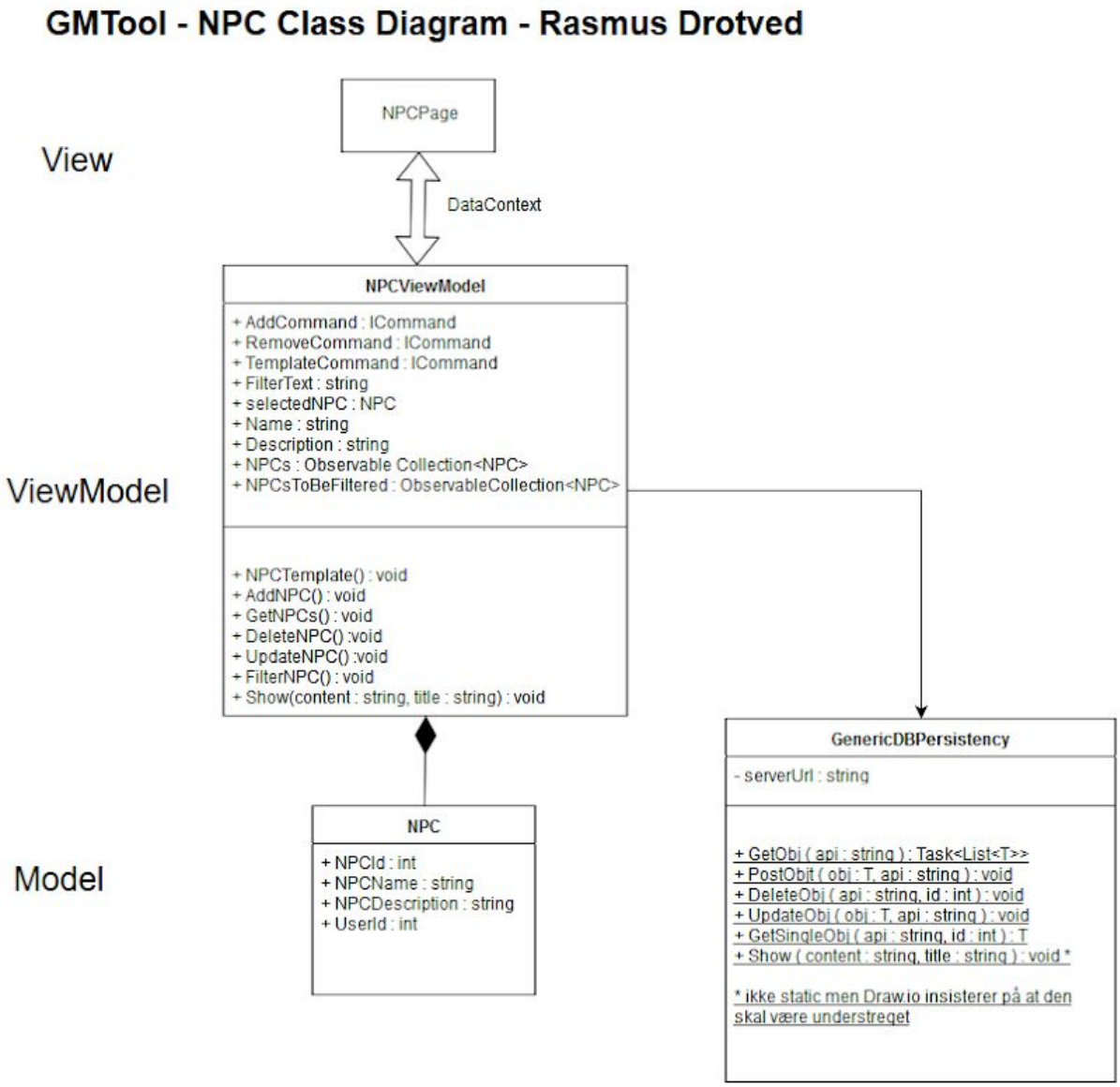
Bilag

Bilag 1: Klassediagram for User Story: “Kampagner: Kampagneoversigt”

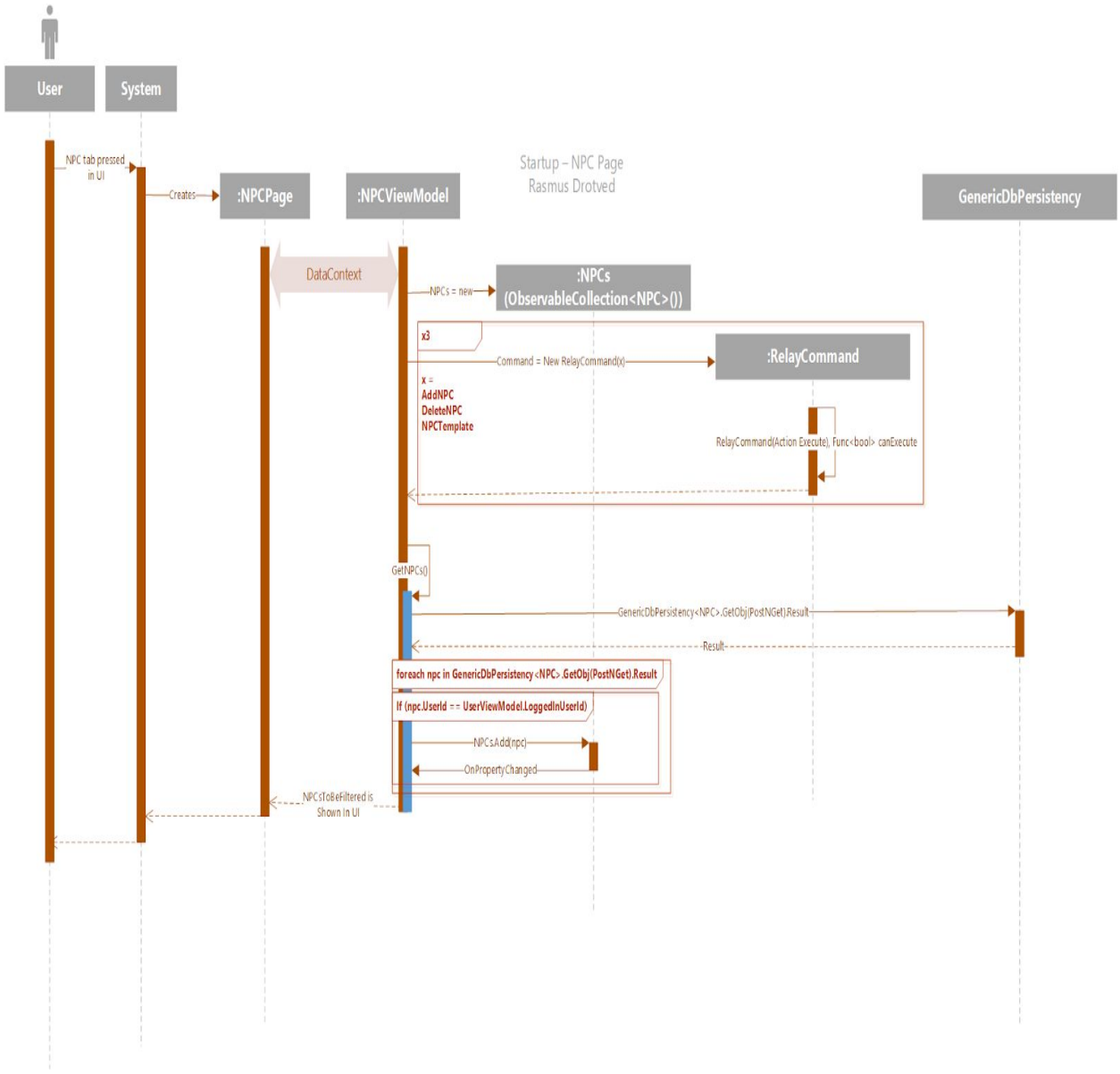
Ansvarlig: Laura Vilen



Bilag 2: Klassediagram for user story “NPC’er: NPC Oversigt”

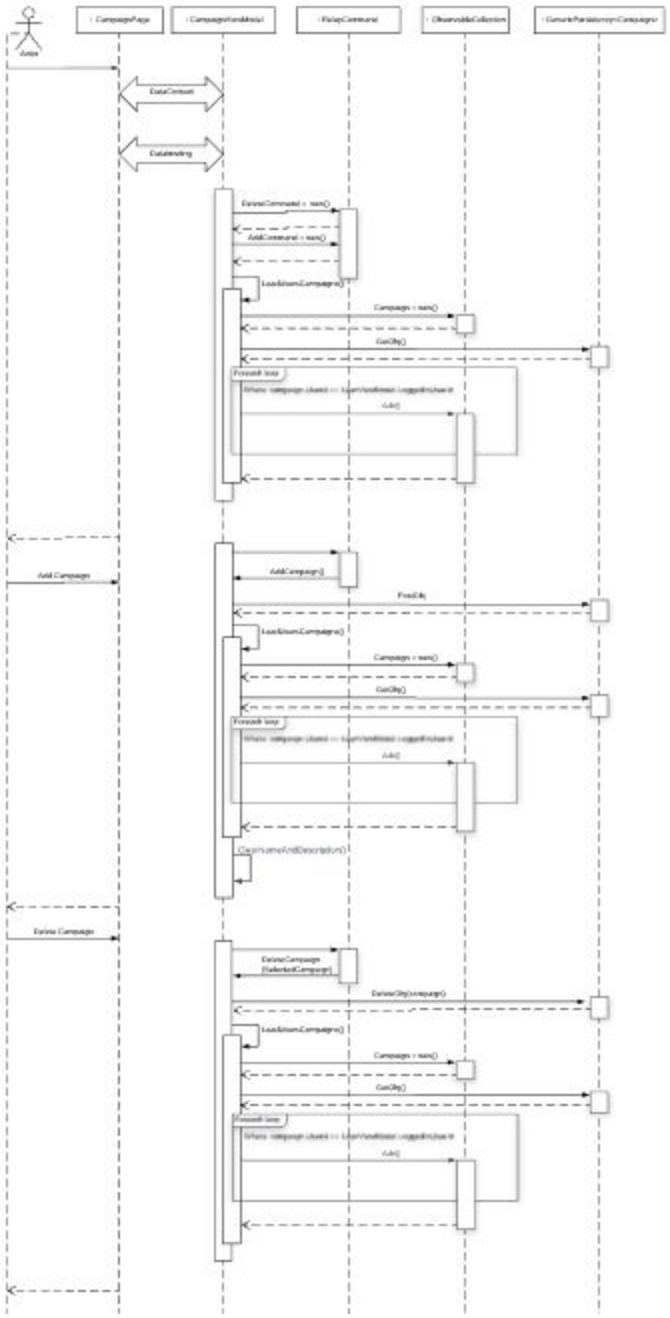


Bilag 3: Sekvensdiagram for opstart af NPCPage til user story “NPC’er: NPC oversigt”



Bilag 4: Sekvensdiagram for user story “Kampagner: Kampagne Oversigt”

Ansvalig: Laura Vilen



Bilag 5: Dagbog for sprint 0-4

Hvad vi har lavet op til projekt perioden:

- Business analysis - SWOT, Porters, Business Canvas
- Vision statement
- Mock ups
- Inception deck
- Entity relations
- Domæne Model
- Opsætning af GitHub
- Gruppekontrakt

Sprint 0

Tirsdag d. 23/04/19

To do:

- User Stories

Done:

- Udarbejdet user stories
- Mock designs
- Møde med product owner

Onsdag d. 24/04/19

To do:

- Gennemgå user stories igen
- Udarbejdet en backlog
- Sprintlog
- Database

Done:

- Forstået user stories bedre
- Begyndt på user stories

Torsdag d. 25/04/19

To do:

- Færdiggøre user stories
- Backlog

Done:

- Færdiggjort user stories
- Backlog

Til i morgen: Alle giver story points til hver user story.

5 point = 1 dags arbejde

Fredag d. 26/04/19

I går blev vi færdige med user stories
Vi begyndte at udarbejde backlog
Vi fik talt med product owner om prioritering af backlog

To do:

- Blive enige om story points
- Entity relations
- Database

Done:

- Story points til user stories
- Tilrettet user stories
- Domæne model

På tirsdag bør backlog være blevet prioriteret af PO. Så vil vi lave entity relations og gå i gang med database.

Sprint 1

Tirsdag d. 30/04/19

Sidste uge blev brugt på sprint 0.

Der blev lavet domæne model, user stories, backlog, mock ups. Derudover fik vi talt med product owner, som kom med feedback og forslag til nye features.

To do:

- Nyt Entity Relations diagram
- Opsætning af Aszure
- DATABASE

Done:

- User stories gennemset af Jamshid
- Entity Relations
- Opsat Azure

Onsdag d. 01/05/19

To do:

- Database
- Basic UI

Basic UI:
Opretter views. Beslutte standarder for farver, layout og skrifttyper. Lave commandbar og menu.

Hjælp til:
Entity Relations diagram
Azure opsætning
Maps/Database til billeder

Resultat af vejledning:

- Entity Relations diagram godkendt
- Azure opsætning er gjort korrekt
- Billeder i database kan gøres på forskellige måder. Undersøg og diskuter bedste løsning.

Done:

- Database sat op
- Basic UI - Splitview menu + view oprettet
- Basic MVVM - Modelklasser oprettet
- Fordelt ansvarsområder for user stories

Torsdag d. 02/05/19

To do:

- Webservice
- CRUD metoder
- User stories

Problemer:

- Laura skal bruge assistance til database
- Rasmus skal bruge hjælp til design

Fredag d. 03/05/19

Laura: Videre på campaign
Problemer:

Rasmus: Færdiggøre og opfylde acceptance criteria
Problemer: skal bruge data, for at teste et acceptance criteria

Kristian: Forsætte med opret profil. Unittest.
Problemer:

- Review: 10.30
- Retrospective
- Sprintplanning

Reviewmøde:

- Ny task: Adskille webservice og app i 2 forskellige github solutions

Der bliver talt om at adskille webservice og app'en i 2 forskellige solutions, så de to ikke er afhængige af hinanden når der synkroniseres med github.

Retrospective:

Kristian:

- Tilfreds med gruppen og samarbejdet.
- Laura og Rasmus har det med at køre ud af en tangent, og snakke om D&D hvilket kunne ryge lidt ud af en tangent.
- Regner med at når vi begynder langt mere individuelt arbejde, så vil det blive nemmere at holde de pauser vi har brug for, når vi har brug for det.

Laura:

- God gruppe. Vi er meget på samme faglige niveau, og kommunikationen fungerer godt.
- Vi har ikke haft aftalt pause, andet end frokost pause. Det har betydet at jeg ikke har følt jeg kunne tillade mig at tage pauser, når jeg havde brug for det.
- Forslag: Faste pauser.

Rasmus:

- Gruppen har været god at arbejde med. God stemning og arbejdsmiljø.
- Vi hjælper hinanden, fremfor at rette på hinanden.
- Begrænse småsludren. Synes vi har holdt småsludren indenfor en rimelig grænse, men hvis vi ikke er opmærksomme på det, kunne det godt blive en distraherende faktor fremover.

Vi brugte meget tid på forberedende arbejde: Database opsætning, webservice mv. frem for at gå i gang med user stories og implementere de dele vi skulle bruge, når vi skulle bruge dem.
Ikke særligt SCRUM eller agile måde at arbejde på.
Vi overdrev forberedelserne lidt.

Sprint Planning:

User stories til backlog:

- Profil: Log In
- Profil: Opret Profil
- Profil: Log ud
- Kampagner: Opret Kampagner
- Kampagner: Kapiteloversigt
- PC'er: PC-oversigt

Sprint 2

Tirsdag d. 07/05/19

Problemer:
Merge er gået galt - Rasmus og Kristian kigger på det.

To do:
Adskille WebService i egen solution

Sket:
Tekniske problemer
Arbejdede videre på user stories
Adskilt webservice i sin egen solution

Onsdag d. 08/05/19

Rasmus: Næsten færdig - finishing touches. Når færdig -> går videre til næste user story

Kristian: Fortsætter på user story.

Laura: Fortsætter på user story. Når færdig -> rapport.
Problem: Deletefunktion.

Torsdag d. 09/05/19

Rasmus:

- Færdig med implementering af 2 user stories i går
- Unittest i dag og færdiggøre rapportpunkt

Kristian:

- Næsten færdig med user story
- Rapport og unit test

Laura:

- Næsten færdig med user story, løst problem med delete
- Færdiggøre user story. Unittest og rapport

Fredag d. 10/05/19

Reviewmøde
Indkaldte Henrik til review af login/logud metoderne. Vi talte om det, han kom med et forslag der fiksede det, som Rasmus egentlig havde gjort fra start.
UnitTest kiggede vi i debug til vi i fællesskab fandt ud af at frame navigationen drillede i den. Henrik havde ikke umiddelbart en løsning.

Fremviste program for product owner. Han virkede tilfreds med UI-designet, men kom med et forslag:
At gøre 'Campaign' draggable, således at man kan sortere sine kampagner hvis man har mange, efter hvilke man spiller mest mv.
Ellers kunne han godt tænke sig at Campaign-boxen var mindre og blot indeholdt en overskrift, og så kunne man 'udvide' box'en så den viste beskrivelsen, hvis man trykkede på den.

Retrospective

Rasmus:

- Synes sprint 2 har kørt fint.
- Ikke lagt mærke til nogle ting som ikke kørte så godt.

Kristian:

- Føler at det bare kører i gruppen og med arbejdet.
- Synes at gruppen ind i mellem udtrykker at vi er mere pressede end jeg synes vi er og synes er rimeligt.

Laura:

- Synes det er godt vi er kommet i gang med at skrive en del mere rapport.
- Der er nogle ting i rapporten vi skal sørge for at få med.
- Mig og Rasmus lader os presse lidt udover hvad der er sundt. Forståelse for at det kan skabe en bestemt stemning i gruppen, der ikke er så rar. Vil forsøge at tone det ned og slappe lidt mere af.

Sprint 3

Tirsdag 14/05/19

Alle blev mere eller mindre færdige med user stories fra sprint 1.

Rasmus:
Regner ikke med at kunne arbejde på 100% kapacitet i dag.
Vil gå i gang med NPC-oversigt.

Kristian:
Ikke helt færdig med Opret profil.
Vi gå i gang med PC-oversigt, og springer nok lidt mellem de 2 user stories i dag.

Laura:
Vil gå i gang med Kapitel oversigt.

Onsdag 15/05/19

Rasmus:

- Sat UI op til NPC page
- Skal muligvis have kigget pån Load() funktion
- Vil prøve at få NPC page færdig i dag -> Tilføj, rediger og slet.

Kristian:

- Gik i gang med Unittest, lavet mindre ting på PC-page.
- Har problemer med unit test

Laura:

- Chapter side
- Linking tables og tilføjet PC'er til kampagne.

Torsdag 16/05/19

Rasmus:

- Mangler redigering af NPC'er + sortering
- Gå i gang med AutoSave user story

Kristian:

- Mere eller mindre færdig med rapport til PC user story
- Problemer med UnitTest -> snakke med Jamshid
- (Vil kigge på at fjerne specialtegn til brugernavn)

Laura:

- Næsten færdig med Chapter oversigt
- UnitTest på Campaign
- (Finpudse design)

Vi aftaler at tage retrospective i dag.
Kontakte Kasper i dag og aftale reviewmøde.

Fredag 17/05/19

Arbejde hjemme

Retrospektive:

Rasmus:

- Godt gruppearbejde. Der er plads til at have det sjovt, og en god atmosfære, men vi får stadig lavet en masse.

God balance

Kristian:

- Gruppen kommer godt ud af det med hinanden
- Gruppen er blevet bedre til at småsludre mindre

Laura:

- Synes at gruppen fik slappet mere af og stoppet med at stresse, efter vi snakkede om den stemning det afledte, ved sidste retrospektive
- Kunne godt tænke mig at kunne følge mere med i udviklingen af de user stories jeg ikke selv er involveret i: Holde mini-review sidst på dagen, hvor vi lige viser og kommer med feedback til hinanden.

Sprint 4

Tirsdag d. 21/05/19

Morgenmøde:

Rasmus:

- Næsten færdig med NPC oversigt
- Finder ud af hvad der skal laves efter sprint planning
- Problemer: Kigget på autosave funktionen, for at se om det var noget der kunne laves i dette sprint.
- Skrive om databasen og testing

Kristian:

- Næsten færdig med PC oversigt
- Mangler autosortering
- Problem: I tvivl om, hvorvidt task 'tilføje pc'er til kampagne' bør være i denne user story.
- Rettet lidt i rapporten. Skrevet lidt om INVEST (user stories) og heuristikker.

Laura:

- Mangler: Unittest, diagrammer, rapport til Kampagne Oversigt

Sprint planning:

Mere fokus på rapport!

- Kristians user story skal ind
- Laura skal skrive sin user story
- De rettelser Jamshid foreslå skal foretages
- Forbinde rapportafsnit ifbm. Analyse (SWOT, Porters) - Introduktionsafsnit
- Problemformulering
- Indledning og konklusion
- Sprint 3
- Stavefejl
- Skrive ansvalige på rapportafsnit
- Backlog
- Færdiggøre user stories fra sidste sprint: PC oversigt, Kapitel oversigt og NPC oversigt

Måske:

- Kapiteldetaljer
- Autosave

Onsdag d. 22/05/19

morgenmøde:

igår:
review ang. rapport, etableret hva rapporten mangler, hold retrospective

Kristian : INVEST afsnit i rapporten, rettede i inception deck.

Rasmus : arbejdede på diagrammer.

idag:
talt med henrik omkring requirements til produktet, samt UI. potentielt review

Kristian : færdiggøre diagrammer, lave diverse ting til rapporten.
Rasmus : færdiggøre diagrammer, uddybe login gennemgang, hvis færdig går igang med at teste databasen med queries

Review fra henrik:
requirements:
- dokumentation
- kode eksempler i rapporten
- exception håndtering
- afsnit om arkitektur samt relaycommands (husk delegate), generics + kode eksempler
- husk billeder af selve programmet

censor læser som udgangspunkt kun rapporten, derfor er det vigtigt at vise program til eksamen.

Fredag d. 24/05/19

Rasmus:
- Gennemlæst en del af rapporten
- Rapport om webservice
- Hvad er relaycommands etc

Kristian:
- Næsten færdig med PC-user story rapport del
- Fortsætte med ting fra rapport listen

Laura:
- Sekvensdiagrammer
- Masser af rapport!!!

Retrospective

Kristian:

- Enig i at vi kom sent i gang med at planlægge rapport og få overblik over denne

Rasmus:

- Laura og Rasmus havde en rigtig dårlig dag samme dag, torsdag. Det blev dog håndteret på en ordentlig måde, så det ikke gik ud over vores arbejde og effektivitet.
- Føltes som om vi er presset på rapport, men mener ikke at vi egentlig bør stresse så meget som vi gør

Laura:

- Føler sig lidt presset over rapportskrivning.
- Vi kunne have undersøgt noget før hvilke kriterier der var til rapporten, da det først var meget sent at vi fik et overblik over hvad der skulle være i den.

Mandag d. 27/05/19

Kristian:

- Skrevet om UML
- Rettet nogle afsnit
- Mere eller mindre færdig med de afsnit der er tildelt

Rasmus:

- Vil gerne have hjælp til Backlog
- Design patterns

Laura:

- Har nået afsnit der forklarer D&D, review og retrospective til flere sprint, Size it Up afsnittet og litteraturliste

To do i dag:

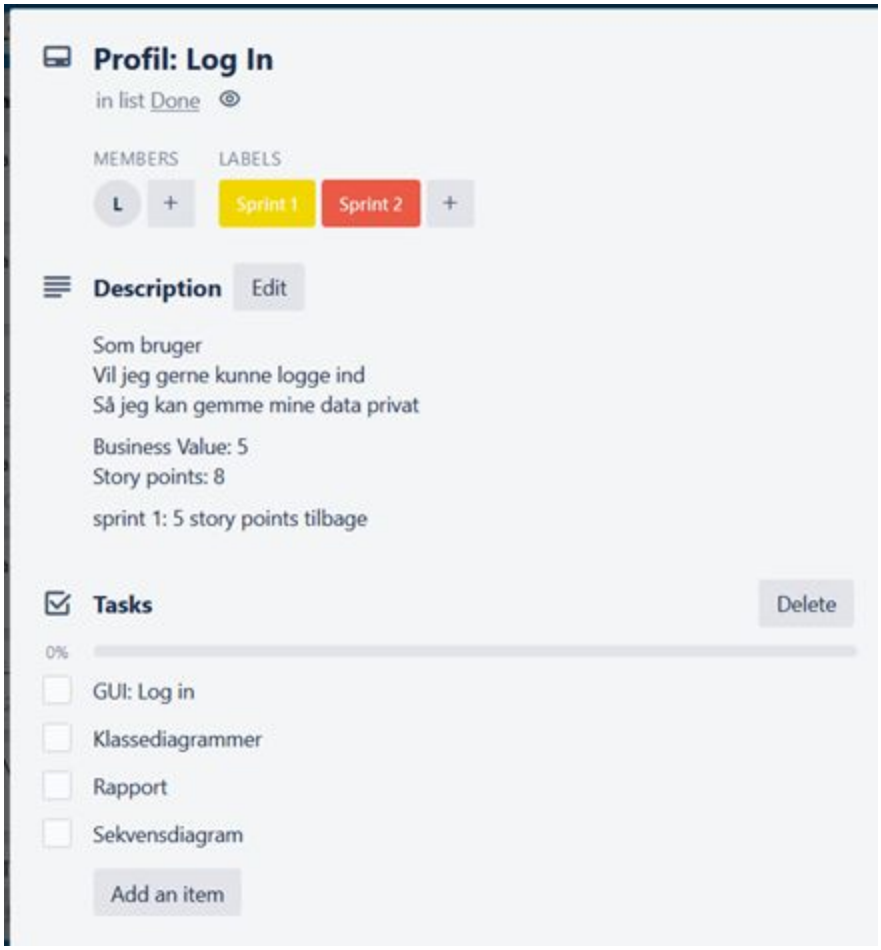
- Indledning
- Problemformulering
- Konklusion
- Sprint review 3
- Sprint resultat 3
- Sprint review 4
- Backlog bilag

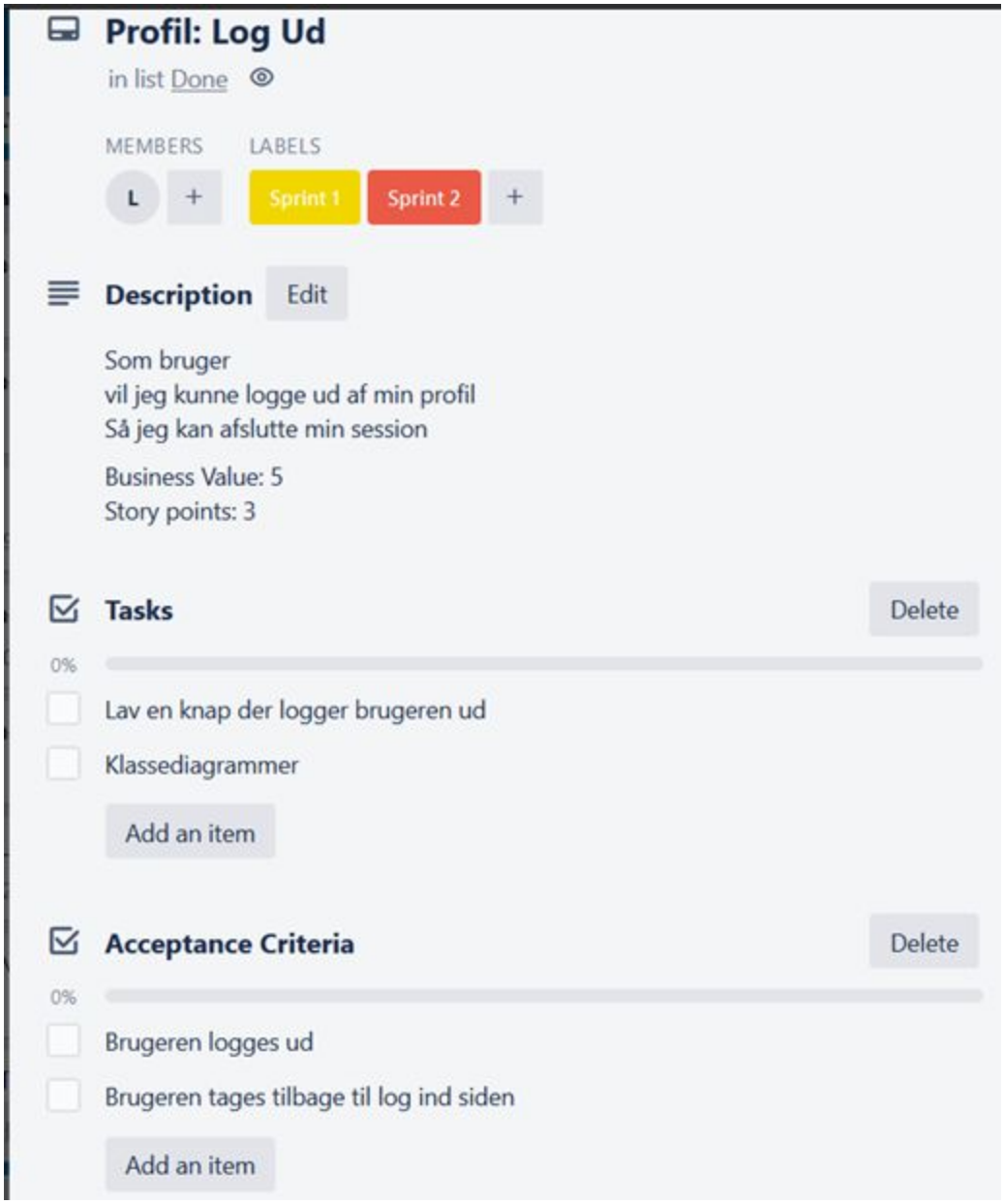
Doxygen

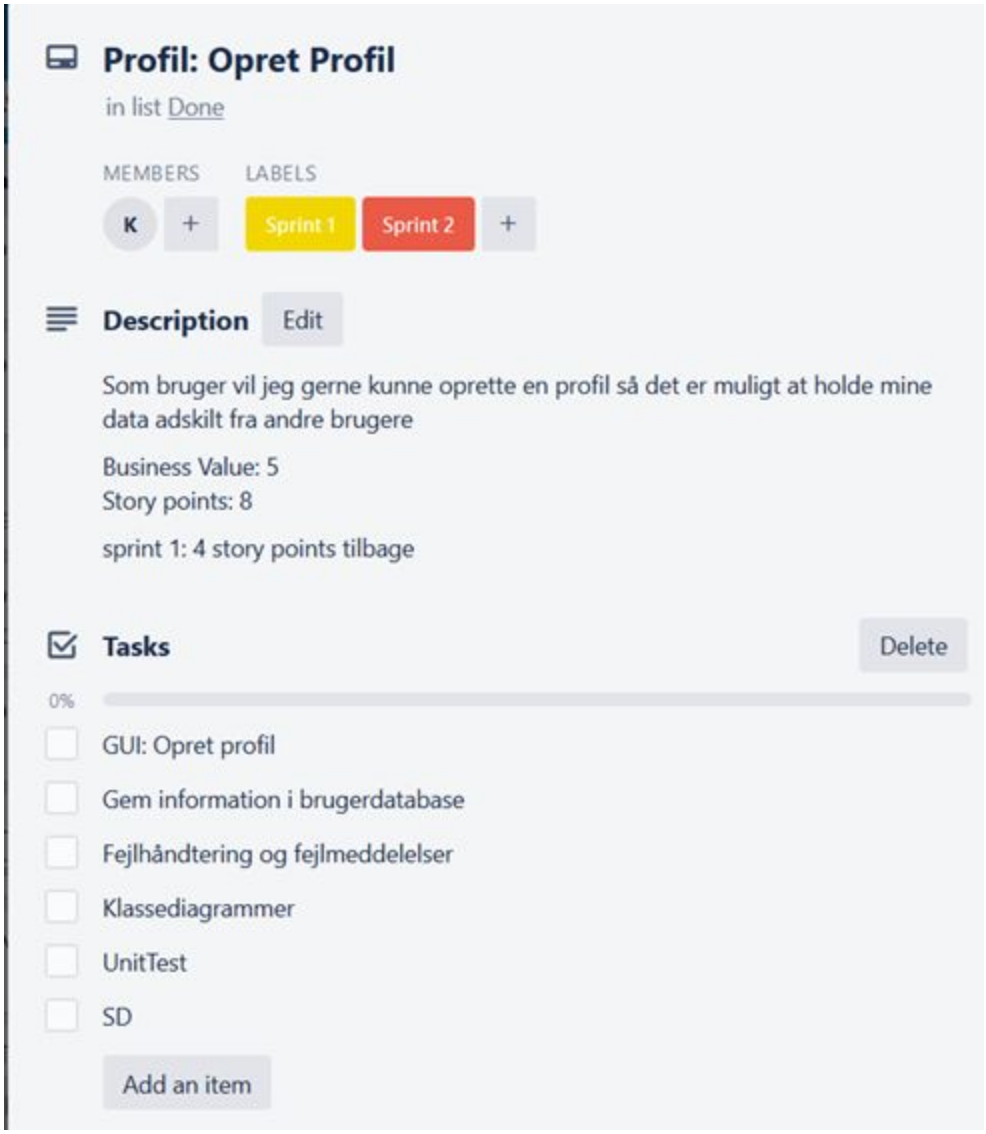
Review

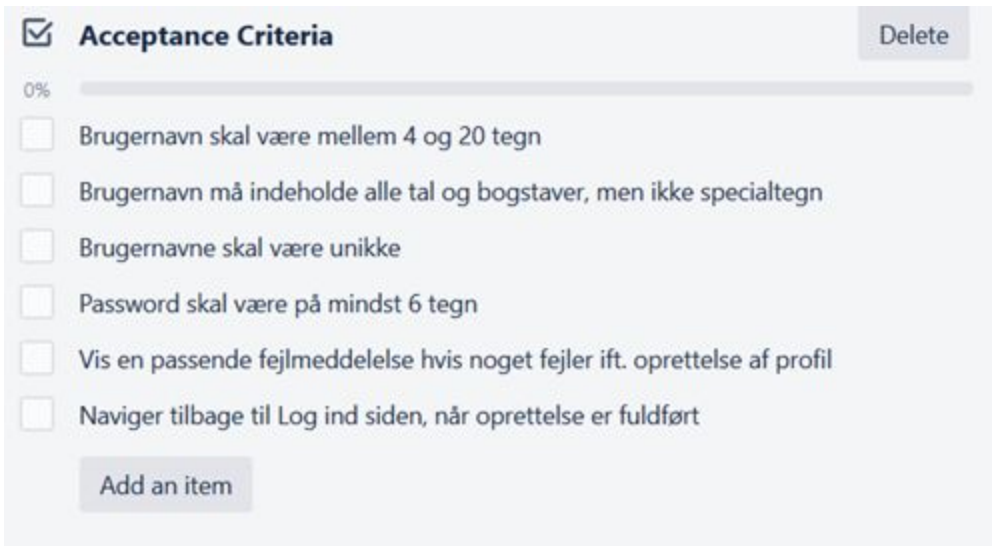
Bilag 6: Produkt Backlog

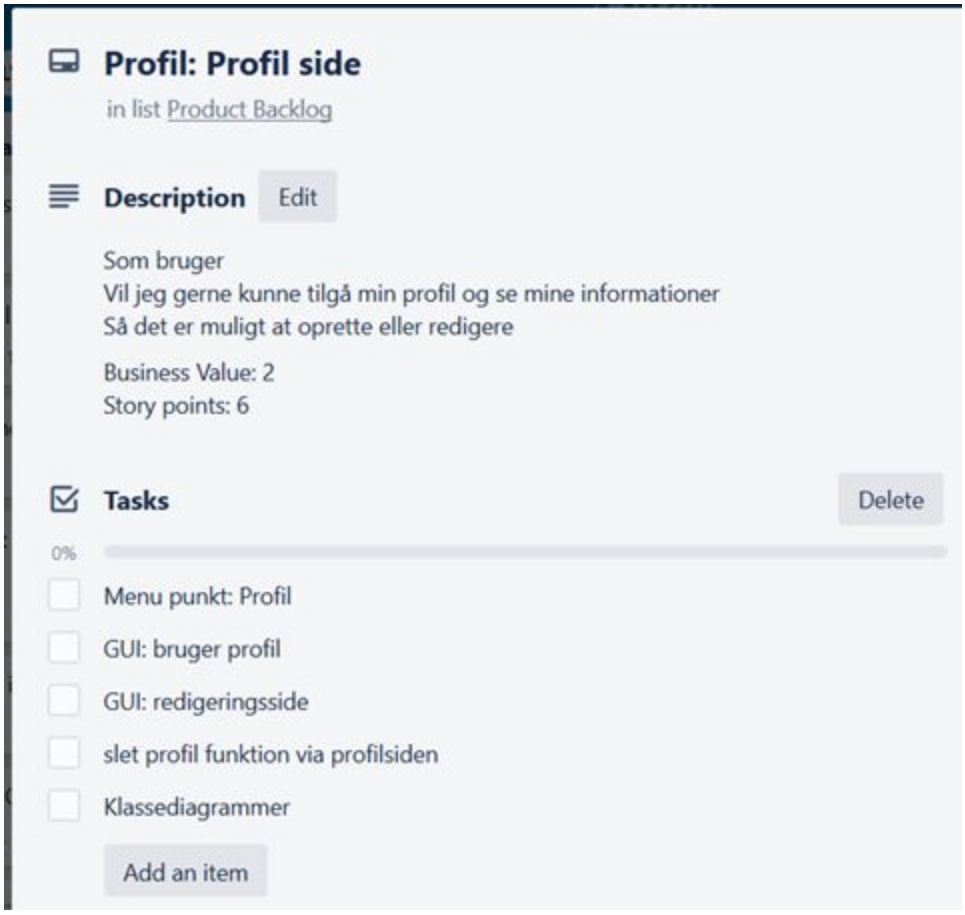
6.1 User Stories

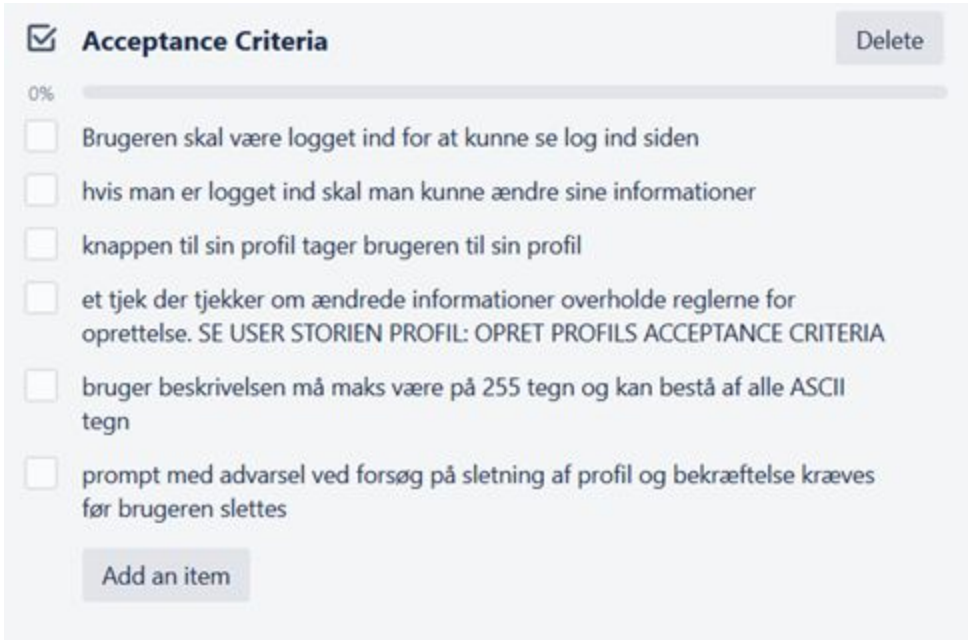



Profil: Log Ud
in list Done
MEMBERS
LABELS
L
Sprint 1
Sprint 2
Description
Edit
Som bruger
vil jeg kunne logge ud af min profil
Så jeg kan afslutte min session
Business Value: 5
Story points: 3
Tasks
Delete
0%
Lav en knap der logger brugeren ud
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Brugeren logges ud
Brugeren tages tilbage til log ind siden
Add an item

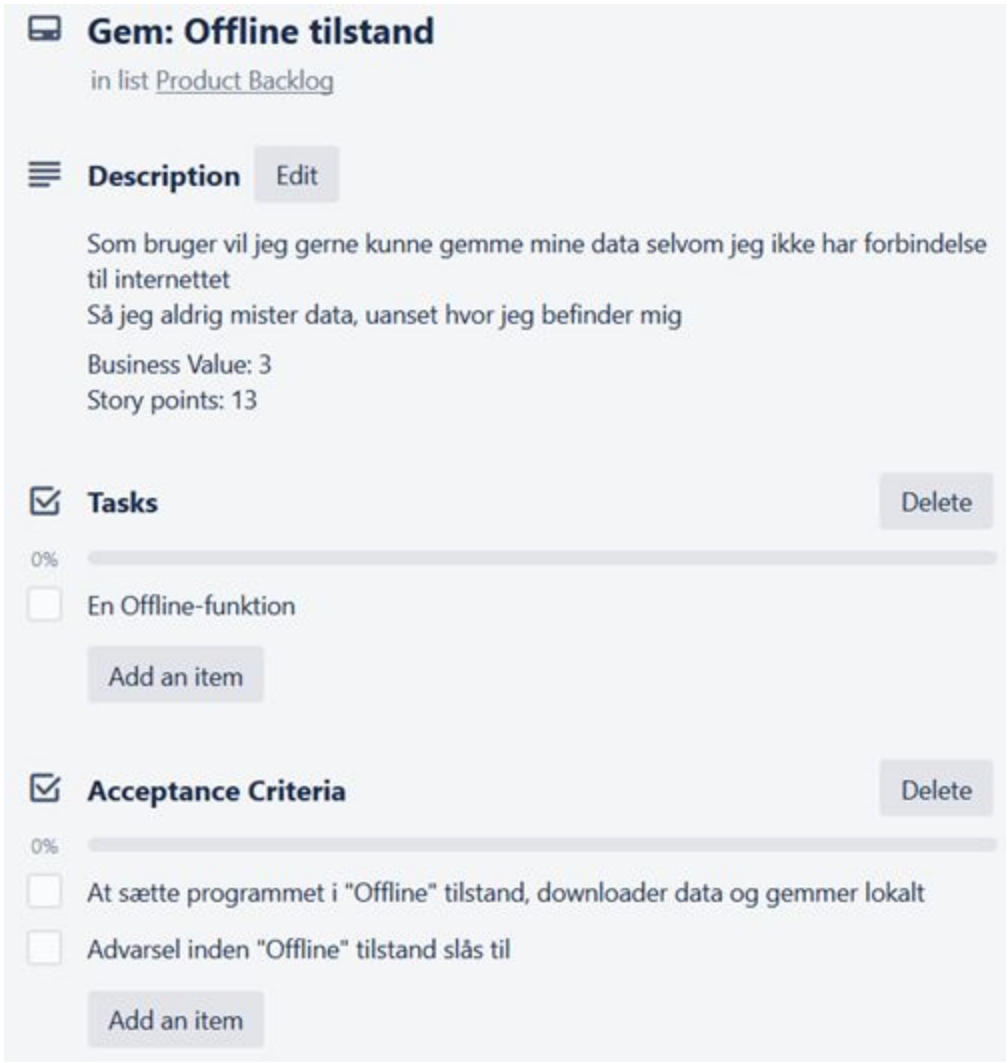

Profil: Opret Profil
in list Done
MEMBERS
LABELS
K
Sprint 1
Sprint 2
Description
Edit
Som bruger vil jeg gerne kunne oprette en profil så det er muligt at holde mine data adskilt fra andre brugere
Business Value: 5
Story points: 8
sprint 1: 4 story points tilbage
Tasks
Delete
0%
GUI: Opret profil
Gem information i brugerdatabase
Fejlhåndtering og fejlmeddelelser
Klassediagrammer
UnitTest
SD
Add an item

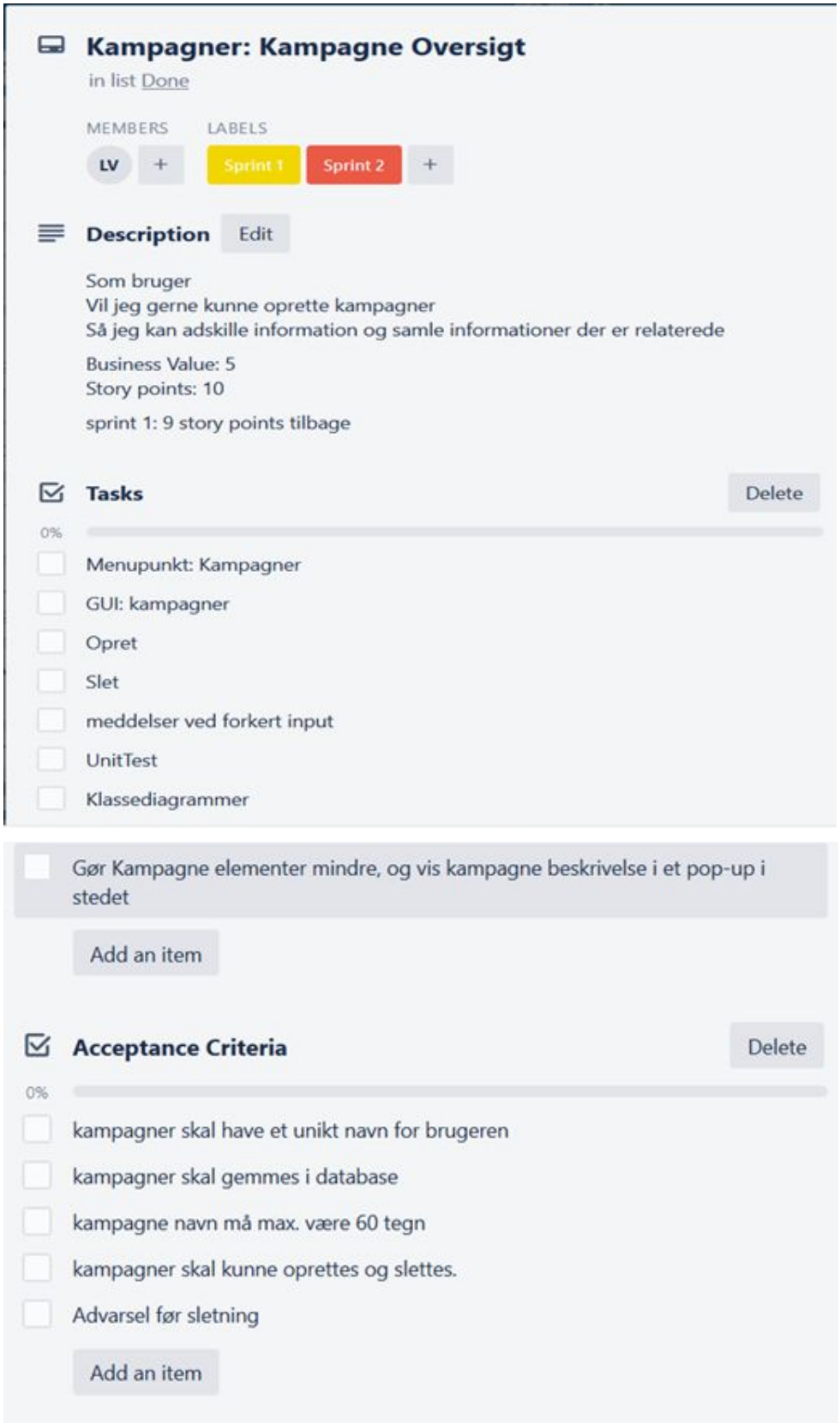

Acceptance Criteria
Delete
0%
Brugernavn skal være mellem 4 og 20 tegn
Brugernavn må indeholde alle tal og bogstaver, men ikke specialtegn
Brugernavne skal være unikke
Password skal være på mindst 6 tegn
Vis en passende fejlmeddelelse hvis noget fejler ift. oprettelse af profil
Naviger tilbage til Log ind siden, når oprettelse er fuldført
Add an item


Profil: Profil side
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne tilgå min profil og se mine informationer
Så det er muligt at oprette eller redigere
Business Value: 2
Story points: 6
Tasks
Delete
0%
Menu punkt: Profil
GUI: bruger profil
GUI: redigeringsside
slet profil funktion via profilsiden
Klassediagrammer
Add an item


Acceptance Criteria
Delete
0%
Brugeren skal være logget ind for at kunne se log ind siden
hvis man er logget ind skal man kunne ændre sine informationer
knappen til sin profil tager brugeren til sin profil
et tjek der tjekker om ændrede informationer overholde reglerne for oprettelse. SE USER STORIEN PROFIL: OPRET PROFILS ACCEPTANCE CRITERIA
bruger beskrivelsen må maks være på 255 tegn og kan bestå af alle ASCII tegn
prompt med advarsel ved forsøg på sletning af profil og bekræftelse kræves før brugeren slettes
Add an item


Gem: Autosave
in list Product Backlog
Description
Edit
Som bruger vil jeg gerne have at mine data bliver gemt automatisk
Så der ikke er fare for at jeg mister data
Business Value: 5
Story points: 8
Tasks
Delete
0%
En funktion der gemmer automatisk mens der arbejdes i fx en tekstboks
Add an item
Acceptance Criteria
Delete
0%
Brugerens data gemmes automatisk
Brugerens data gemmes lokalt hvis der ikke er internet
Klassediagrammer
Add an item


Gem: Offline tilstand
in list Product Backlog
Description
Edit
Som bruger vil jeg gerne kunne gemme mine data selvom jeg ikke har forbindelse til internettet
Så jeg aldrig mister data, uanset hvor jeg befinder mig
Business Value: 3
Story points: 13
Tasks
Delete
0%
En Offline-funktion
Add an item
Acceptance Criteria
Delete
0%
At sætte programmet i "Offline" tilstand, downloader data og gemmer lokalt
Advarsel inden "Offline" tilstand slås til
Add an item


Kampagner: Kampagne Oversigt
in list Done
MEMBERS
LABELS
LV
Sprint 1
Sprint 2
Description
Edit
Som bruger
Vil jeg gerne kunne oprette kampagner
Så jeg kan adskille information og samle informationer der er relaterede
Business Value: 5
Story points: 10
sprint 1: 9 story points tilbage
Tasks
Delete
0%
Menupunkt: Kampagner
GUI: kampagner
Opret
Slet
meddelser ved forkert input
UnitTest
Klassediagrammer
Gør Kampagne elementer mindre, og vis kampagne beskrivelse i et pop-up i stedet
Add an item
Acceptance Criteria
Delete
0%
kampagner skal have et unikt navn for brugeren
kampagner skal gemmes i database
kampagne navn må max. være 60 tegn
kampagner skal kunne oprettes og slettes.
Advarsel før sletning
Add an item


Kampagne: ændre rækkefølge på kapitler
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne ændre rækkefølgen på kapitler
Så jeg har mulighed for at tilpasse eventyret, hvis spillere springer i plottet
Business Value: 4
Story points: 10
Tasks
Delete
0%
Gør kapitler draggable
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Kapitler skal kunne flyttes rundt
Rækkefølgen må ikke autosorteres
Add an item

Kampagner: Kapiteloversigt

in list Done

MEMBERS LABELS

LV + Sprint 3 Sprint 4 +

Description Edit

Som bruger
Vil jeg gerne kunne opdele mine kampagner i forskellige kapitler
Så det er mere overskueligt under spilgang

Business Value: 5
Story points: 7

Tasks Delete

0%

- [ ] Liste af kapitler (under kampagner)
- [ ] mulighed for at oprette og slette kapitler
- [ ] GUI: chapter selection
- [ ] GUI: Kapiteldetaljer-side. (spilgangs siden) - Simpelt layout mv
- [ ] meddelelser ved forkert input
- [ ] Klassediagrammer
- [ ] UnitTest
- [ ] Sekvensdiagram
- [ ] Optional: Gør det muligt at ændre rækkefølge på elementer i kapitellisten ved at flytte dem rundt i GUI'en

Acceptance Criteria Delete

0%

- [ ] Kapitler kan oprettes og slettes
- [ ] kapitler skal kunne gemmes
- [ ] Kapitelnavn skal være unikt for kampagnen
- [ ] advarsel før slet

Add an item

PC'er: PC oversigt

in list Done

MEMBERS LABELS

K + Sprint 2 Sprint 3 +

Description Edit

Som bruger
Vil jeg gerne have en oversigt over spillerkarakterer
Så jeg bedre kan holde styr på plot og items som er relevant for karakteren

Business Value: 2
Story points: 8

Tasks Hide completed items Delete

100%

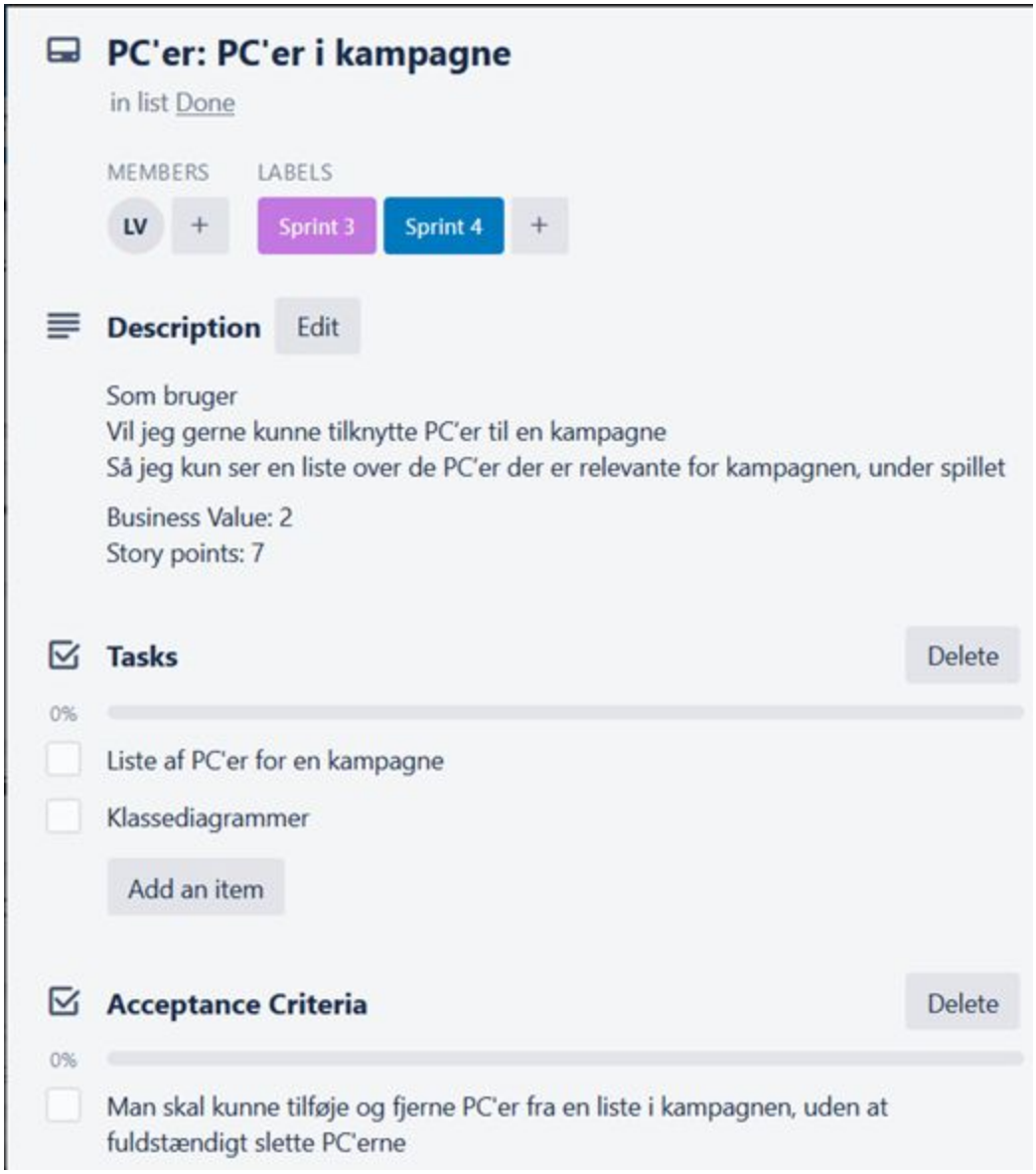
- ~~*Menupunkt: PC'er*~~
- ~~*Liste af PC'er*~~
- ~~*Mulighed for at oprette, redigere og slette PC'er*~~
- ~~*GUI: PC oversigt*~~
- ~~*GUI: PC detaljer*~~
- ~~*GUI: Oprettelses/Redigerings side*~~
- ~~*Sort: Autosortering efter navn*~~
- ~~*Meddelelse ved forkert input*~~
- ~~*Klassediagrammer*~~

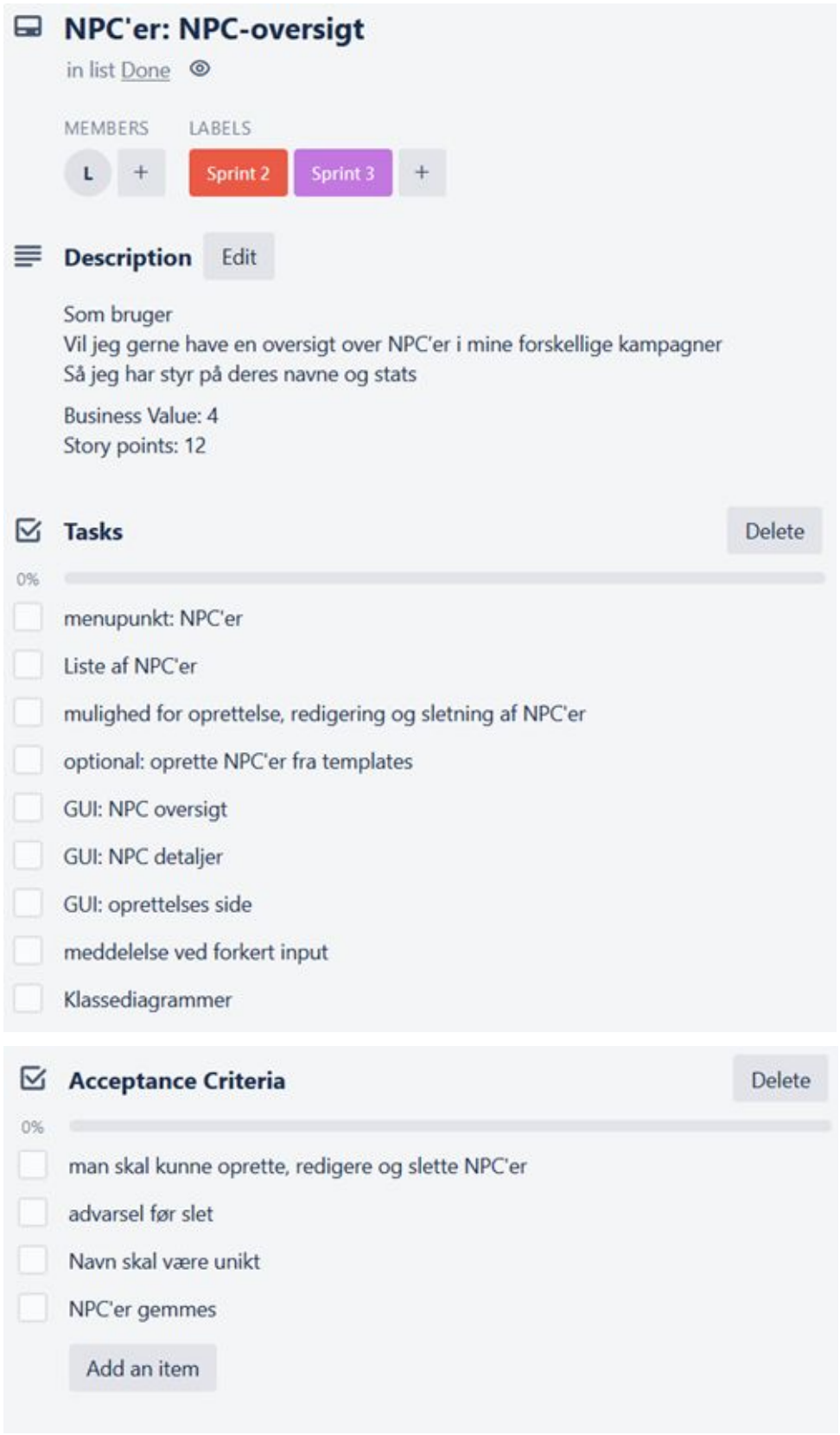
Acceptance Criteria Hide completed items Delete

100%

- ~~*Man kan oprette, redigere og slette PC'er*~~
- ~~*Man skal kunne gemme PC'er*~~
- ~~*PC navne skal være unikke*~~
- ~~*Advarsel før slet*~~
- ~~*Listen sorteres automatisk efter navn i alfabetisk rækkefølge*~~

Add an item


PC'er: PC'er i kampagne
in list Done
MEMBERS
LABELS
LV
Sprint 3
Sprint 4
Description
Edit
Som bruger
Vil jeg gerne kunne tilknytte PC'er til en kampagne
Så jeg kun ser en liste over de PC'er der er relevante for kampagnen, under spillet
Business Value: 2
Story points: 7
Tasks
Delete
0%
Liste af PC'er for en kampagne
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Man skal kunne tilføje og fjerne PC'er fra en liste i kampagnen, uden at fuldstændigt slette PC'erne


NPC'er: NPC-oversigt
in list Done
MEMBERS
LABELS
L
Sprint 2
Sprint 3
Description
Edit
Som bruger
Vil jeg gerne have en oversigt over NPC'er i mine forskellige kampagner
Så jeg har styr på deres navne og stats
Business Value: 4
Story points: 12
Tasks
Delete
0%
menupunkt: NPC'er
Liste af NPC'er
mulighed for oprettelse, redigering og sletning af NPC'er
optional: oprette NPC'er fra templates
GUI: NPC oversigt
GUI: NPC detaljer
GUI: oprettelses side
meddelelse ved forkert input
Klassediagrammer
Acceptance Criteria
Delete
0%
man skal kunne oprette, redigere og slette NPC'er
advarsel før slet
Navn skal være unikt
NPC'er gemmes
Add an item


NPC'er: NPC'er i kapitel
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne genbruge og oprette NPC'er i forskellige kapitler
Så jeg har nemmere adgang til mine NPC'ers stats
Business Value: 4
Story points: 7
Tasks
Delete
0%
Tilføj NPC'er til NPC liste i kapitler
Pop-up vindue med NPC noter og stats i kapitlet
Oprettelse og redigering af NPC'er fra kapitel fanen
Fjerne NPC'er fra kapitel uden at slette NPC'en fuldstændigt
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
NPC'er skal kunne tilføjes og fjernes fra en liste i kapitlet
Når en NPC fjernes fra et kapitel, slettes de IKKE fra systemet
Listen af NPC'er gemmes
Add an item


Encounters: Encounter oversigt
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne have en oversigt over encounters
Så jeg har overblik over de forskellige elementer i encounteret
Business Value: 4
Story points: 10
Tasks
Delete
0%
Menupunkt: Encounters
Liste af encounters
Mulighed for at oprette, redigere og slette encounters
GUI: encounter oversigt
GUI: encounter detaljer
GUI: oprettelses/redigerings side
Meddelelse ved forkert input
Klassediagrammer
Acceptance Criteria
Delete
0%
Man kan oprette, redigere og slette encounters
Man kan gemme encounters
Advarsel inden slet
Encounter navne skal være unikke
Add an item


Encounters: Encounters i kapitler
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne have en liste af encounters til hvert kapitel
Så jeg har styr på hvilke forhindringer spillerne kan møde og hvornår
Business Value: 4
Story points: 8
Tasks
Delete
0%
Liste af encounters på kapitelside
Tilføjelse og fjernelse fra liste fra den detaljerede kapitelside
Opret/rediger encounters fra listen på den detaljerede kapitelside
Pop-up med encounter detaljer fra den detaljerede kapitelside
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
man kan tilføje og fjerne fra listen i den detaljerede kapitel side
encounters bliver ikke fuldstændigt slettede fra systemet når man fjerner dem fra listen
Encounter detaljer vises som pop-up og navigerer derved ikke væk fra kapitlet
Add an item

Combat Tracker

in list Product Backlog

Description

Edit

Som bruger
Vil jeg gerne have en oversigt over tur og fjender
Så jeg nemmere kan holde styr på combat

Business Value: 4
Story points: 5 - Med optional: 15

Tasks

Delete

0%

- [ ] GUI: Combat tracker
- [ ] GUI: Knap i commandbar
- [ ] Optional: combat gemt i encounters
- [ ] Klassediagrammer

Add an item

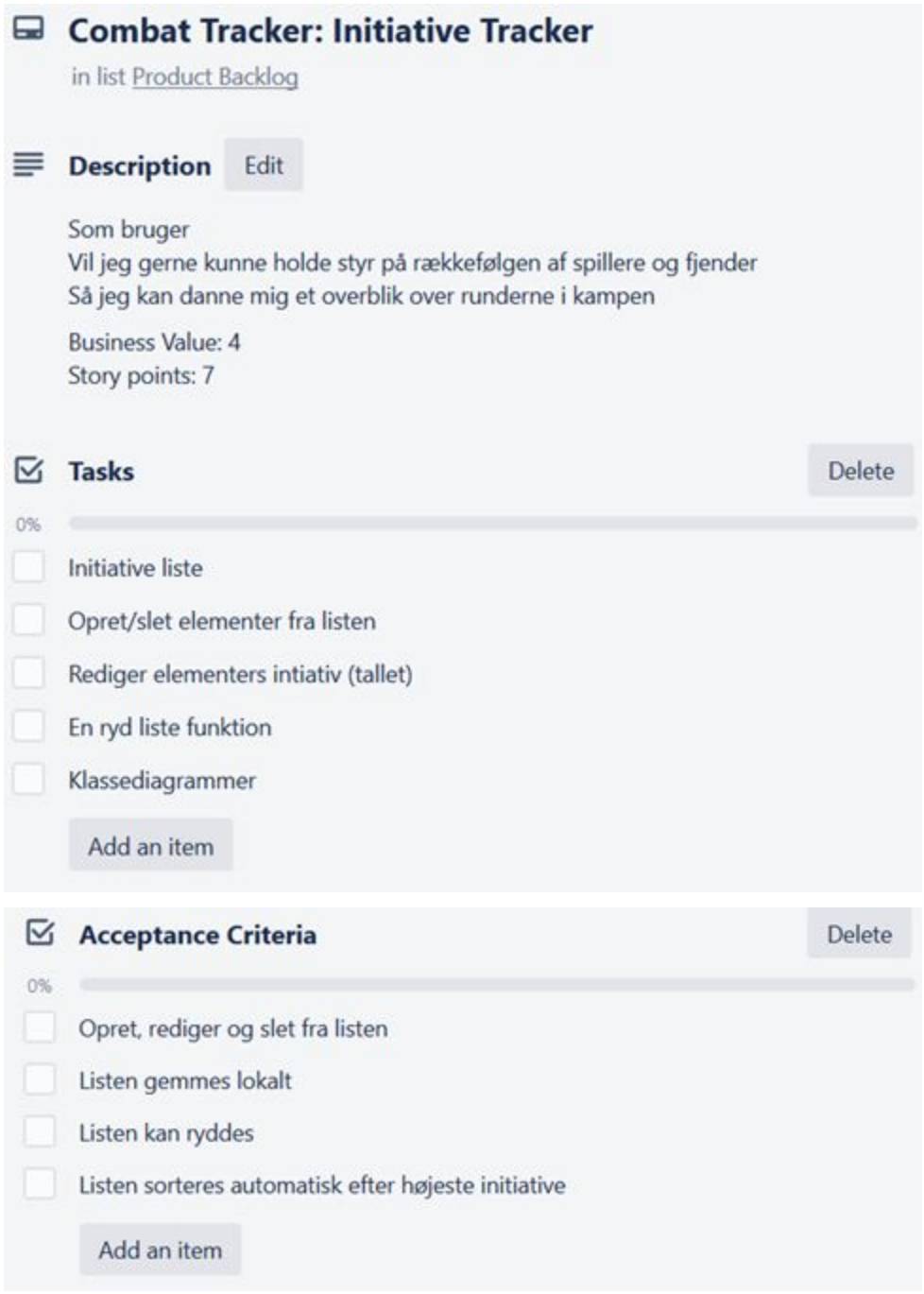
Acceptance Criteria

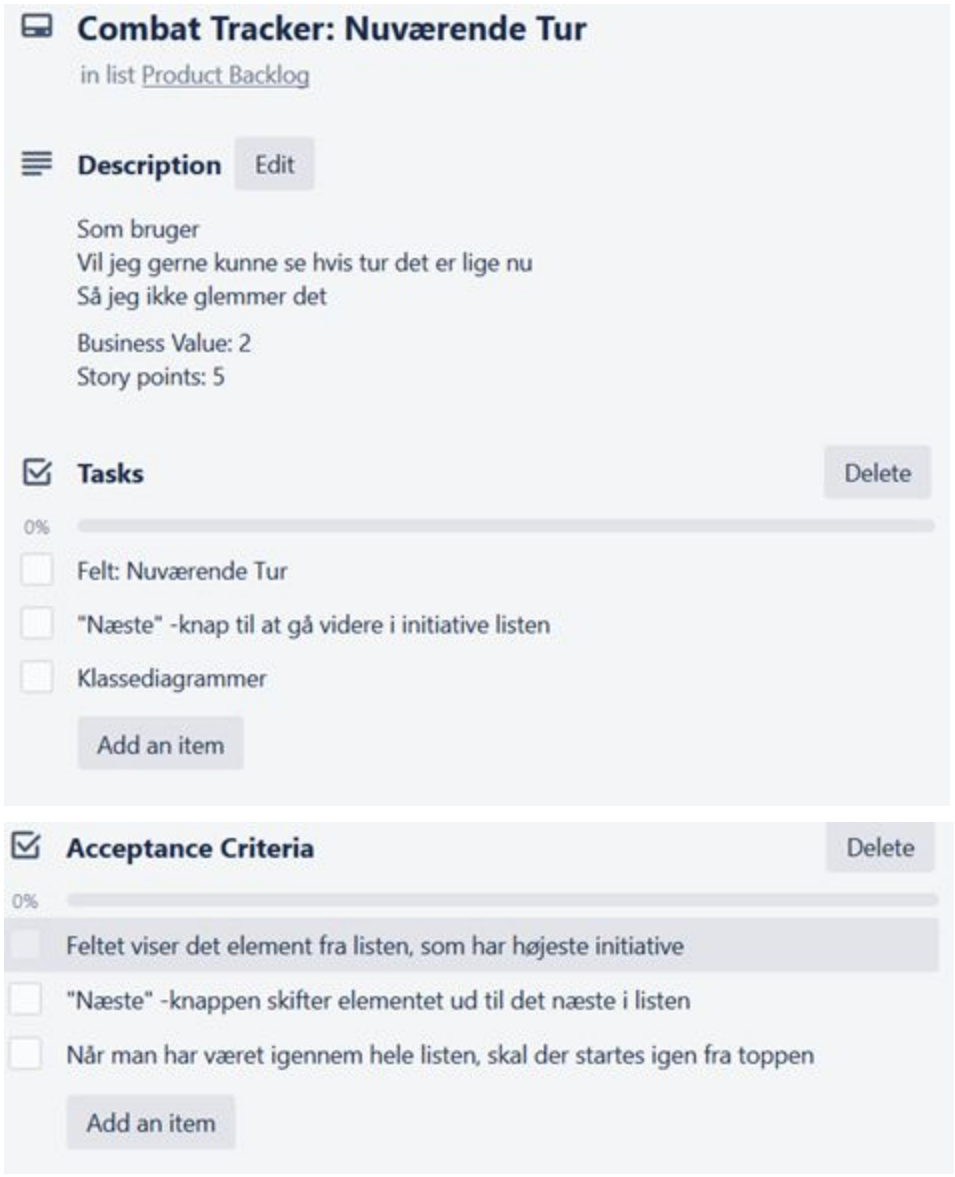
Delete

0%

- [ ] Knap i commandbar skal åbne et pop-up vindue med combat tracker
- [ ] Combat information skal gemmes
- [ ] Optional: Combat skal kunne gemmes under encounters
- [ ] Optional: Man skal kunne loade combat encounters fra en liste i vinduet

Add an item


Combat Tracker: Initiative Tracker
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne holde styr på rækkefølgen af spillere og fjender
Så jeg kan danne mig et overblik over runderne i kampen
Business Value: 4
Story points: 7
Tasks
Delete
0%
Initiative liste
Opret/slet elementer fra listen
Rediger elementers intiativ (tallet)
En ryd liste funktion
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Opret, rediger og slet fra listen
Listen gemmes lokalt
Listen kan ryddes
Listen sorteres automatisk efter højeste initiative
Add an item


Combat Tracker: Nuværende Tur
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne se hvis tur det er lige nu
Så jeg ikke glemmer det
Business Value: 2
Story points: 5
Tasks
Delete
0%
Felt: Nuværende Tur
"Næste" -knap til at gå videre i initiative listen
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Feltet viser det element fra listen, som har højeste initiative
"Næste" -knappen skifter elementet ud til det næste i listen
Når man har været igennem hele listen, skal der startes igen fra toppen
Add an item

Combat Tracker: Enemy Health Bars

in list Product Backlog

Description

Edit

Som bruger
Vil jeg gerne kunne se en oversigt over fjender, deres liv og AC samt redigere disse
Så jeg nemt kan holde styr på fjender under combat

Business Value: 4
Story points: 12

Tasks

Delete

0%

- [ ] Liste af enemies med hp og ac
- [ ] Tilføj/slet fra listen
- [ ] Redigere hp
- [ ] Visning af hvilke fjender der er døde
- [ ] En ryd funktion til at rydde listen
- [ ] Klassediagrammer

Add an item

Acceptance Criteria

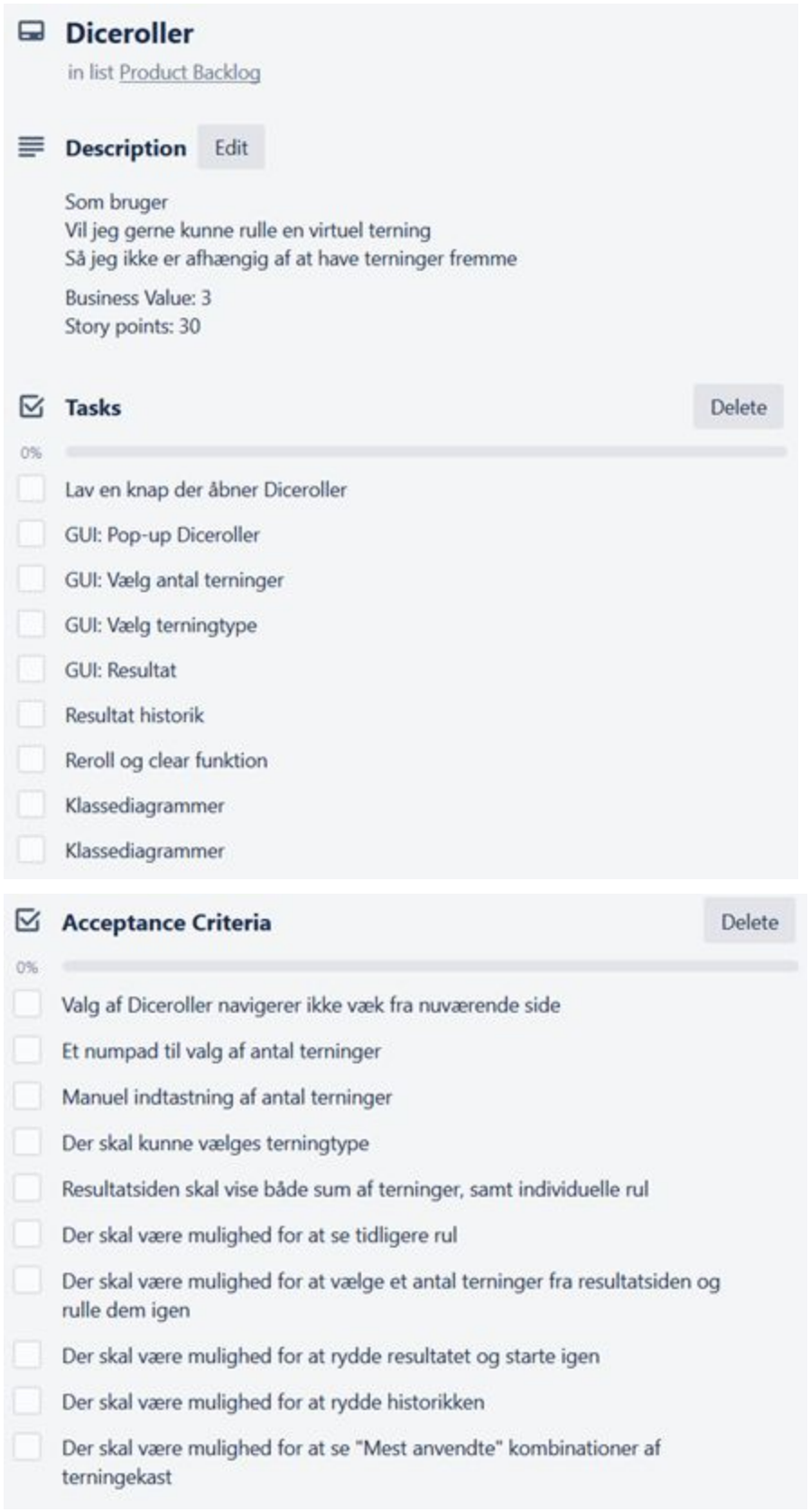
Delete

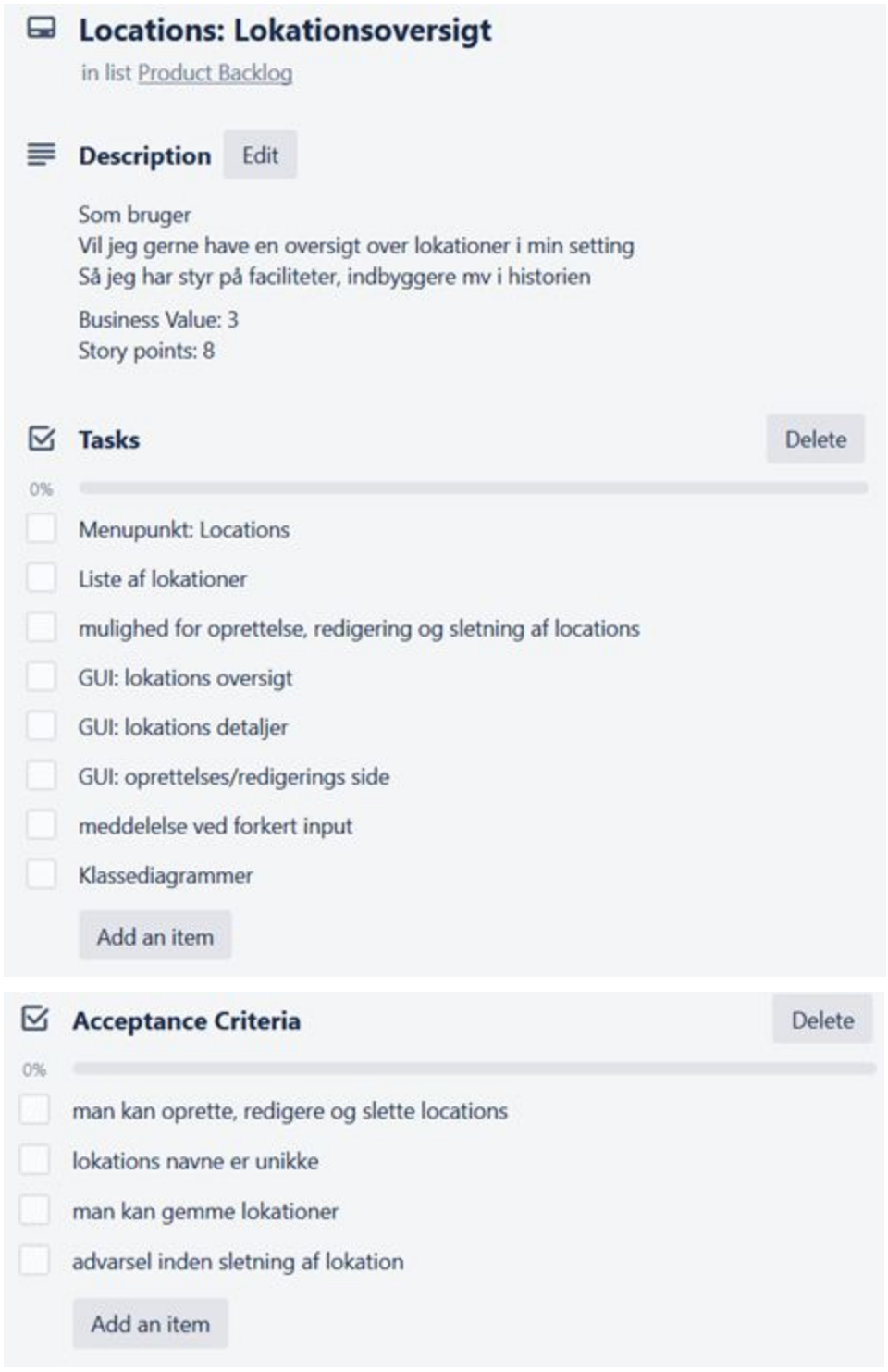
0%

- [ ] Opret og slet
- [ ] Hp skal være en tekstboks, som kan redigeres til en hver tid
- [ ] Når fjenders hp er 0 eller under, skal de markeres som døde
- [ ] Listen skal kunne ryddes

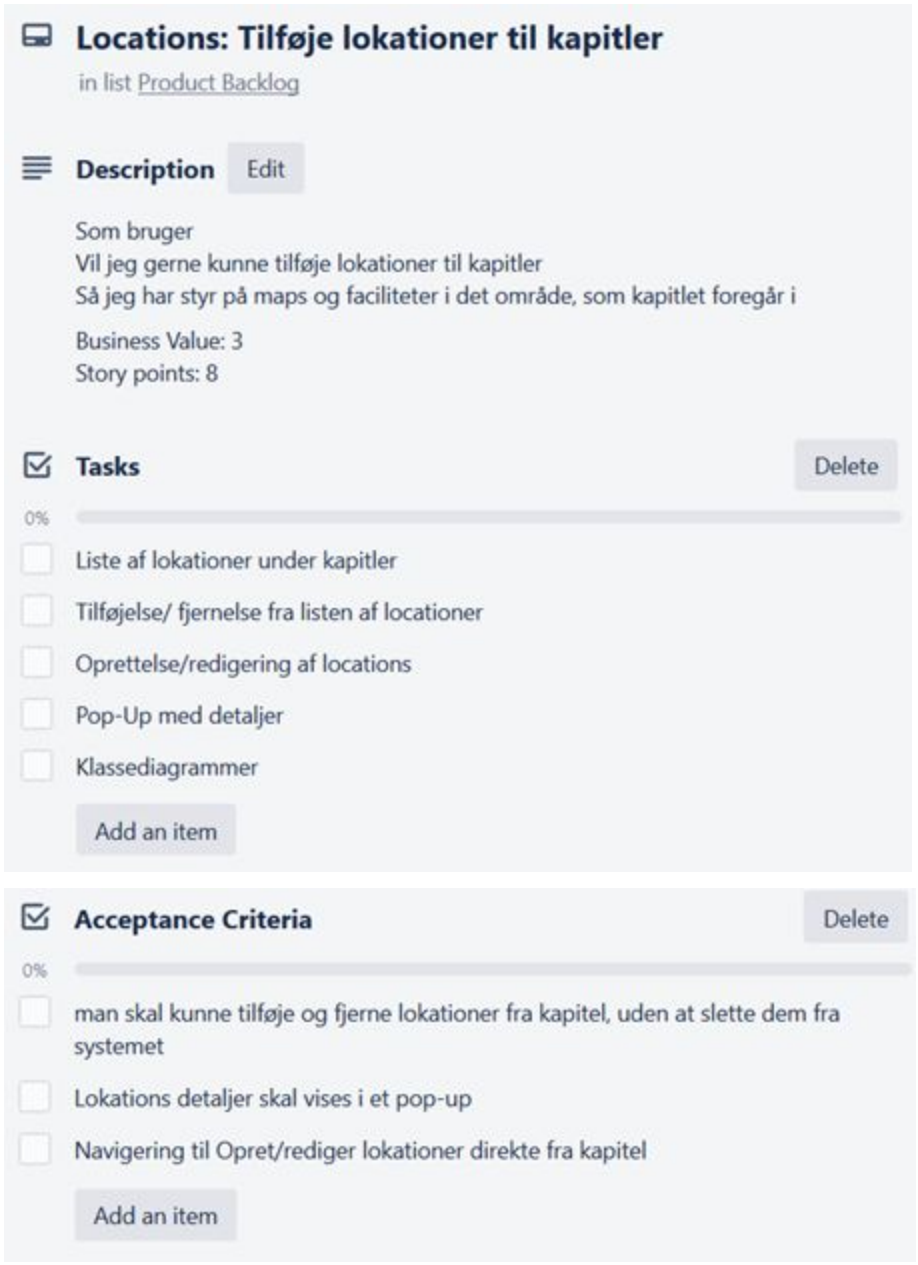
Add an item


Combat Tracker: Tilførsel af skade
in list Product Backlog
Description
Edit
Som bruger
vil jeg gerne kunne markere visse fjender og trække påført skade fra deres liv
så jeg ikke behøver regne hver fjendes liv ud manuelt, hver gang de tager skade
Business Value: 3
Story points: 6
Tasks
Delete
0%
Metode der trækker damage fra valgte fjenders HP
GUI: Apply Damage - knap
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Man skal kunne markere mere end én fjende
Funktionen skal automatisk opdatere fjenders hp
Add an item


Diceroller
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne rulle en virtuel terning
Så jeg ikke er afhængig af at have terninger fremme
Business Value: 3
Story points: 30
Tasks
Delete
0%
Lav en knap der åbner Diceroller
GUI: Pop-up Diceroller
GUI: Vælg antal terninger
GUI: Vælg terningtype
GUI: Resultat
Resultat historik
Reroll og clear funktion
Klassediagrammer
Klassediagrammer
Acceptance Criteria
Delete
0%
Valg af Diceroller navigerer ikke væk fra nuværende side
Et numpad til valg af antal terninger
Manuel indtastning af antal terninger
Der skal kunne vælges terningtype
Resultatsiden skal vise både sum af terninger, samt individuelle rul
Der skal være mulighed for at se tidligere rul
Der skal være mulighed for at vælge et antal terninger fra resultatsiden og rulle dem igen
Der skal være mulighed for at rydde resultatet og starte igen
Der skal være mulighed for at rydde historikken
Der skal være mulighed for at se "Mest anvendte" kombinationer af terningekast


Locations: Lokationsoversigt
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne have en oversigt over lokationer i min setting
Så jeg har styr på faciliteter, indbyggere mv i historien
Business Value: 3
Story points: 8
Tasks
Delete
0%
Menupunkt: Locations
Liste af lokationer
mulighed for oprettelse, redigering og sletning af locations
GUI: lokations oversigt
GUI: lokations detaljer
GUI: oprettelses/redigerings side
meddelelse ved forkert input
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
man kan oprette, redigere og slette locations
lokations navne er unikke
man kan gemme lokationer
advarsel inden sletning af lokation
Add an item


Locations: Maps
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne oprette maps til lokationer
Så jeg har bedre får et overblik over lokationen
Business Value: 3
Story points: 10
Tasks
Delete
0%
Liste af billeder
Tilføj og slet billeder fra listen
Frem- og tilbageknap til at scrolle gennem billeder
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Der skal kunne tilføjes flere maps til 1 lokation
Man skal kunne scrolle igennem billederne
Tilføjelse og sletning af maps
Navngivelse af billeder
Add an item


Locations: Tilføje lokationer til kapitler
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne tilføje lokationer til kapitler
Så jeg har styr på maps og faciliteter i det område, som kapitlet foregår i
Business Value: 3
Story points: 8
Tasks
Delete
0%
Liste af lokationer under kapitler
Tilføjelse/ fjernelse fra listen af locationer
Oprettelse/redigering af locations
Pop-Up med detaljer
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
man skal kunne tilføje og fjerne lokationer fra kapitel, uden at slette dem fra systemet
Lokations detaljer skal vises i et pop-up
Navigering til Opret/rediger lokationer direkte fra kapitel
Add an item

Items: Itemliste

in list Product Backlog

Description

Edit

Som bruger
Vil jeg gerne have en oversigt over items
Så jeg kan få overblik over hvilke items jeg har til rådighed og hvad de kan

Business Value: 3
Story points: 10

Tasks

Delete

0%

- [ ] Menupunkt: Items
- [ ] Item Liste
- [ ] Mulighed for oprettelse, redigering og sletning af Items
- [ ] GUI: item oversigt
- [ ] GUI: item detaljer
- [ ] GUI: oprettelses side/flyout
- [ ] meddelelse ved forkert input
- [ ] Klassediagrammer

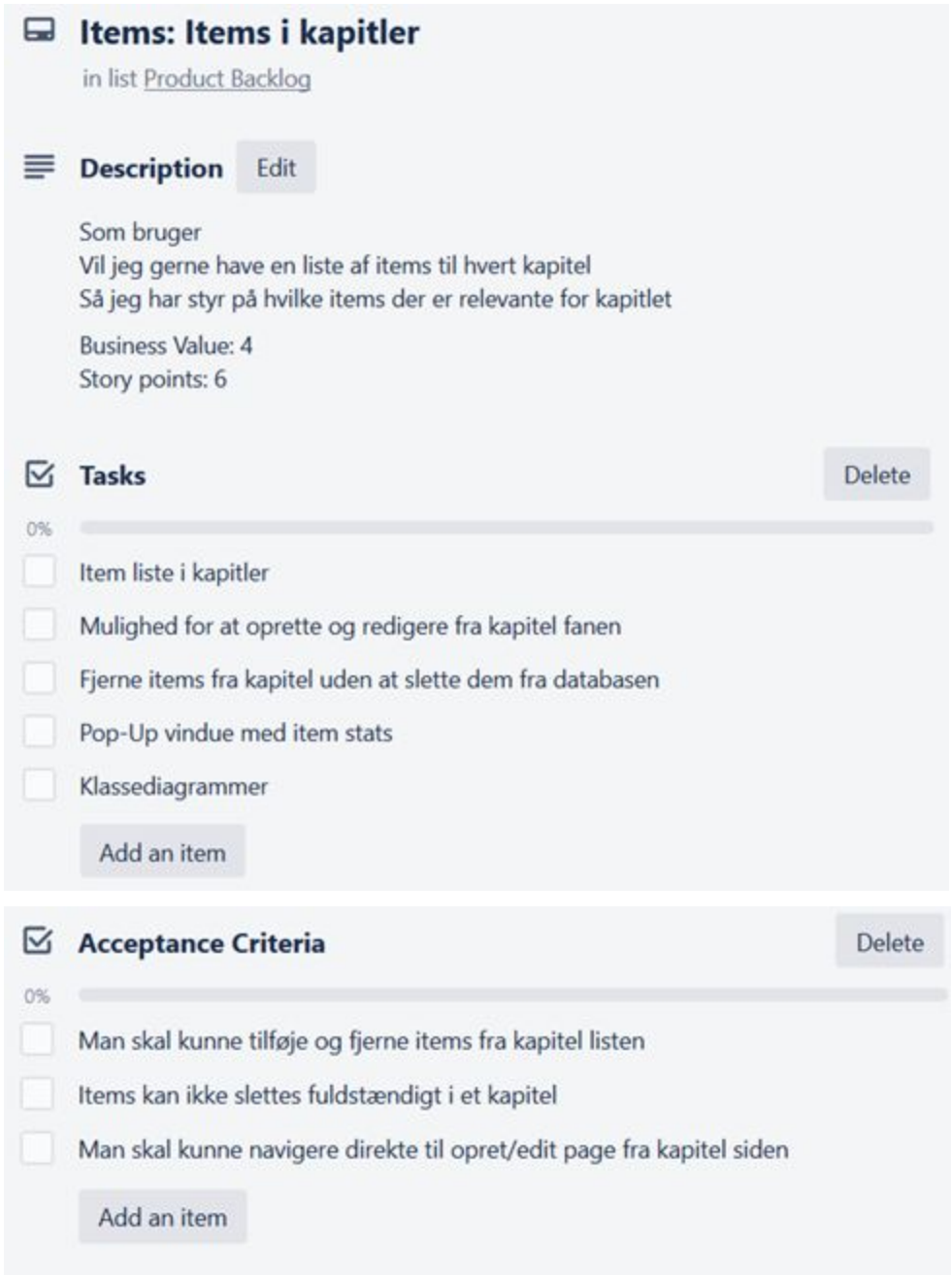
Acceptance Criteria

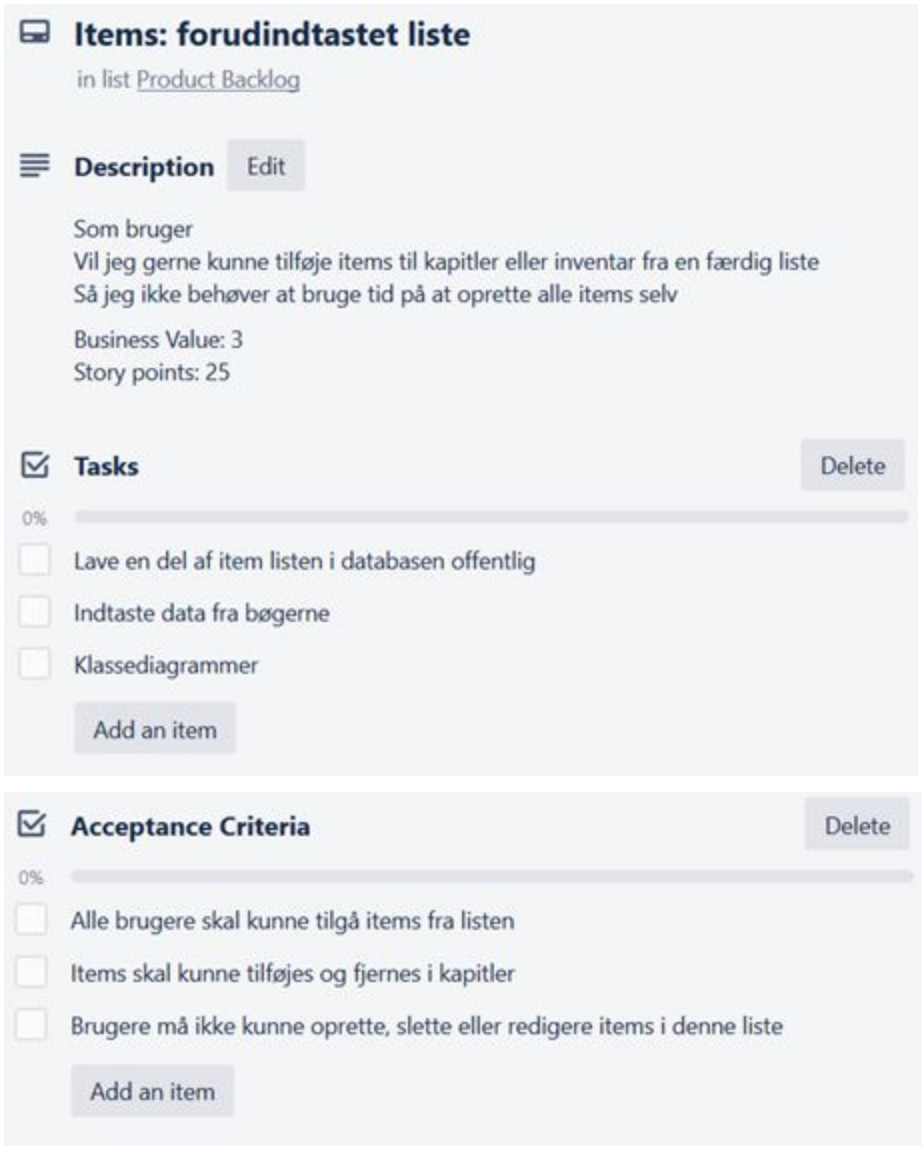
Delete

0%

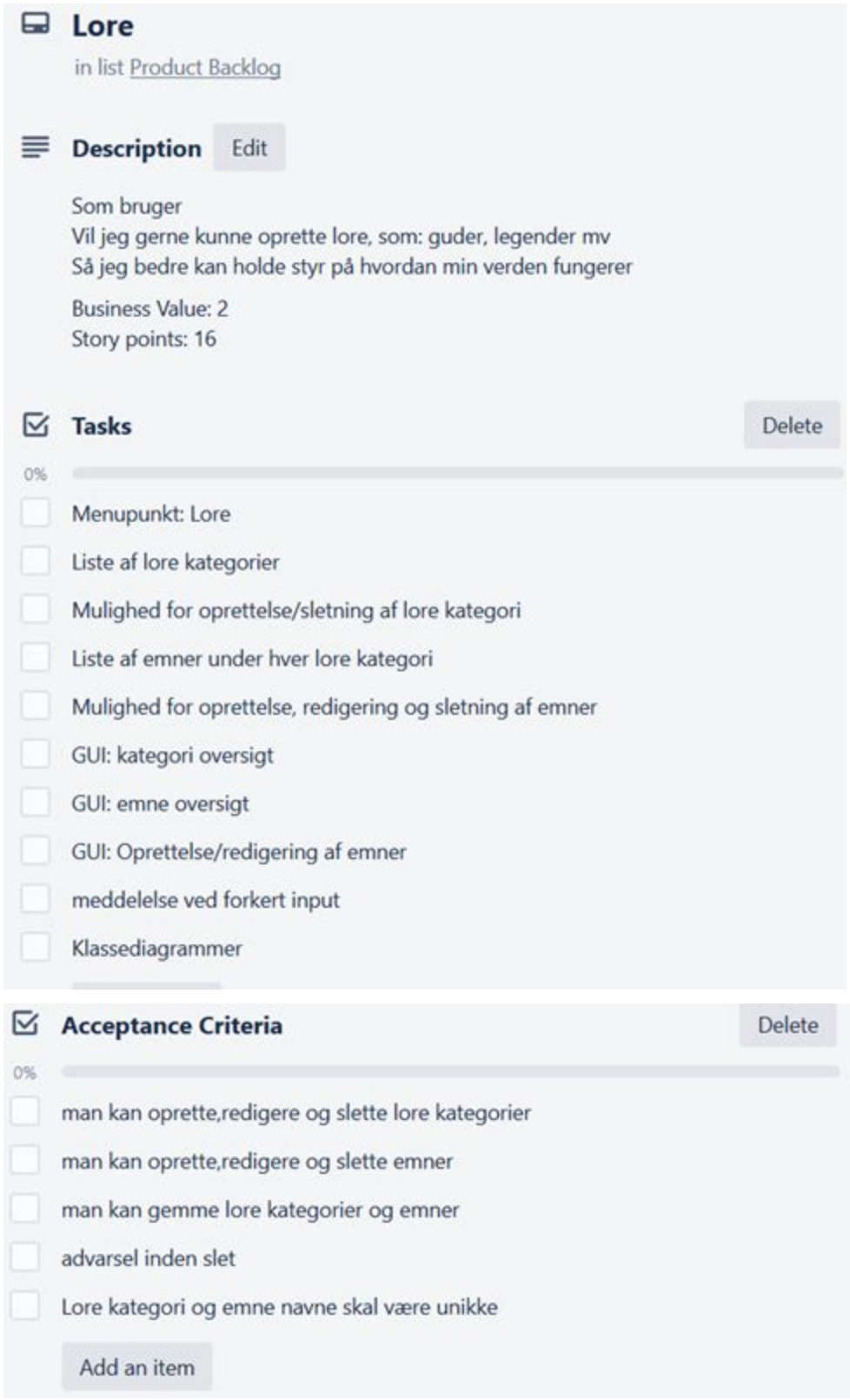
- [ ] Man skal kunne oprette, redigere og slette items
- [ ] advarsel inden slet
- [ ] items skal unikt navn
- [ ] Items skal kunne gemmes

Add an item


Items: Items i kapitler
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne have en liste af items til hvert kapitel
Så jeg har styr på hvilke items der er relevante for kapitlet
Business Value: 4
Story points: 6
Tasks
Delete
0%
Item liste i kapitler
Mulighed for at oprette og redigere fra kapitel fanen
Fjerne items fra kapitel uden at slette dem fra databasen
Pop-Up vindue med item stats
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Man skal kunne tilføje og fjerne items fra kapitel listen
Items kan ikke slettes fuldstændigt i et kapitel
Man skal kunne navigere direkte til opret/edit page fra kapitel siden
Add an item


Items: forudindtastet liste
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne tilføje items til kapitler eller inventar fra en færdig liste
Så jeg ikke behøver at bruge tid på at oprette alle items selv
Business Value: 3
Story points: 25
Tasks
Delete
0%
Lave en del af item listen i databasen offentlig
Indtaste data fra bøgerne
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
Alle brugere skal kunne tilgå items fra listen
Items skal kunne tilføjes og fjernes i kapitler
Brugere må ikke kunne oprette, slette eller redigere items i denne liste
Add an item

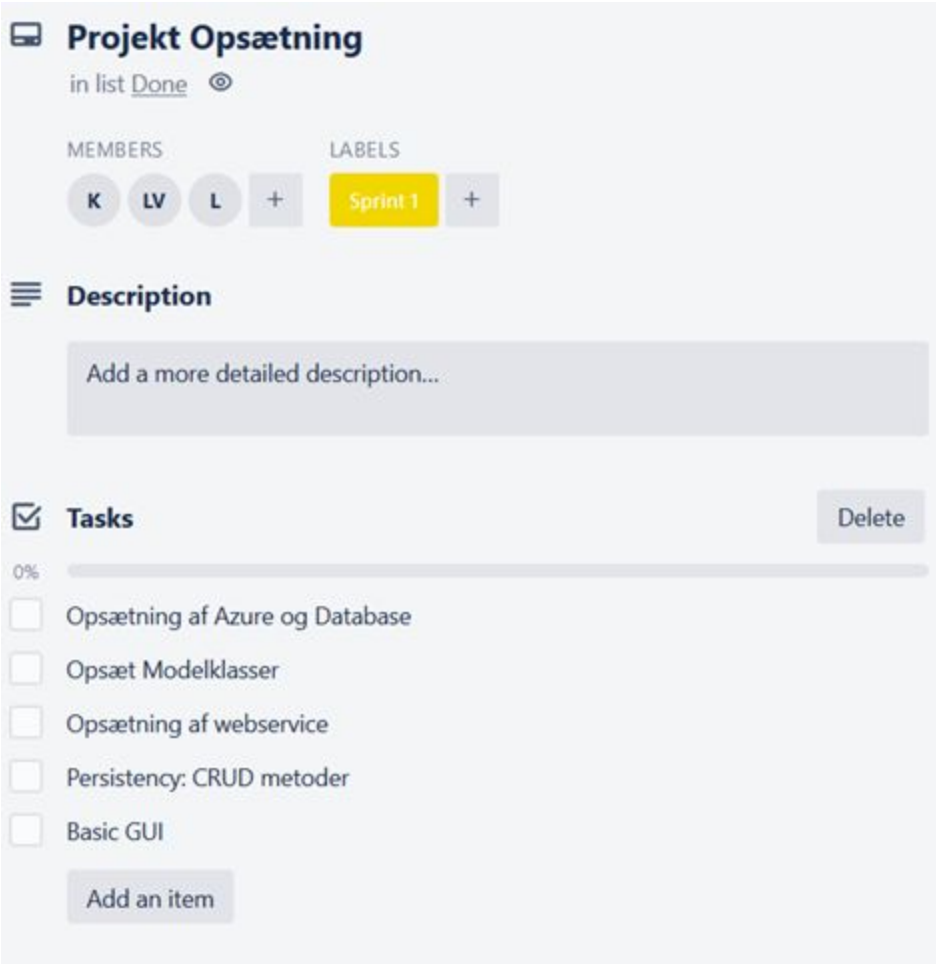

Items: Inventory
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne tilføje items til en NPC eller PC's inventar
Så jeg ved hvem der holder hvilke items, og hvilket loot der kan fås fra NPC'er
Business Value: 2
Story points: 7
Tasks
Delete
0%
NPC'er og PC'er har en liste kaldet inventory
Mulighed for at tilføje og fjerne items i Inventory
Add an item
Acceptance Criteria
Delete
0%
Der kan tilføjes og fjernes items i Inventory
Når et items fjernes fra Inventory, slettes det ikke fuldkomment
Der skal kunne navigeres til Opret/redigering af items direkte fra inventory
Add an item


Lore
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne oprette lore, som: guder, legender mv
Så jeg bedre kan holde styr på hvordan min verden fungerer
Business Value: 2
Story points: 16
Tasks
Delete
0%
Menupunkt: Lore
Liste af lore kategorier
Mulighed for oprettelse/sletning af lore kategori
Liste af emner under hver lore kategori
Mulighed for oprettelse, redigering og sletning af emner
GUI: kategori oversigt
GUI: emne oversigt
GUI: Oprettelse/redigering af emner
meddelelse ved forkert input
Klassediagrammer
Acceptance Criteria
Delete
0%
man kan oprette,redigere og slette lore kategorier
man kan oprette,redigere og slette emner
man kan gemme lore kategorier og emner
advarsel inden slet
Lore kategori og emne navne skal være unikke
Add an item

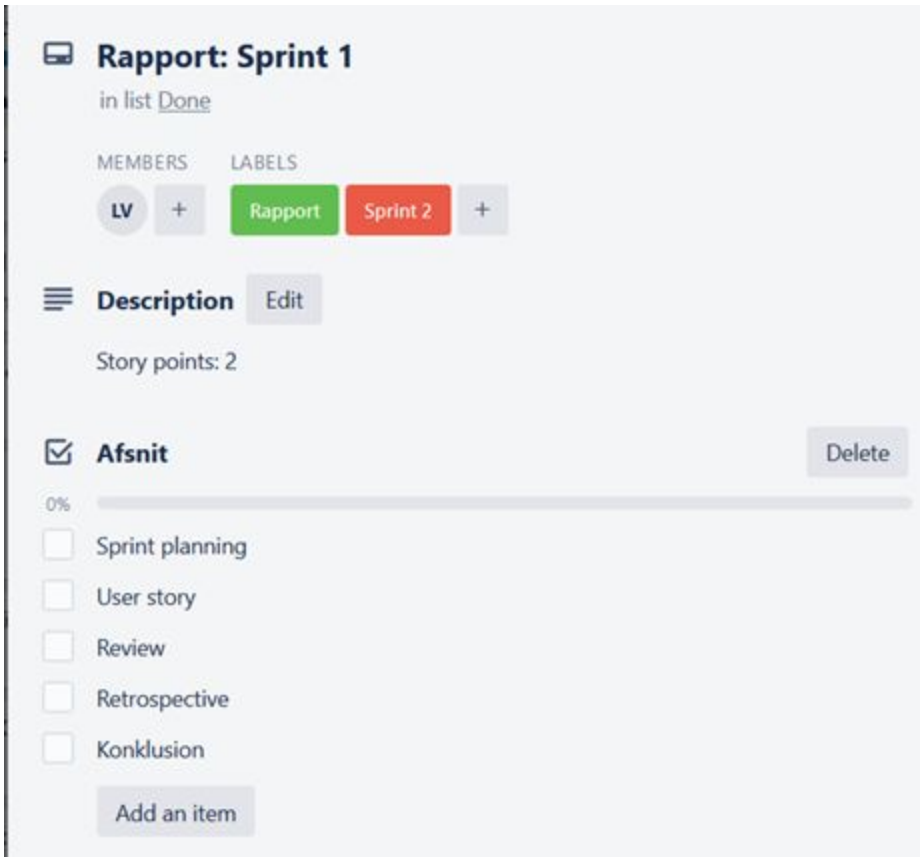

Filtrering
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne tilføje tag på forskellige elementer i programmet
Så jeg nemmere kan danne mig overblik og søge efter fællesnævnere
Business Value: 4
Story points: 18
Tasks
Delete
0%
Alle relevante kategorier i programmet skal have en liste af tags
Samme kategorier skal kunne filtreres efter tags
Brugeren skal kunne oprette, redigere og slette sine egne tags
Søgefunktion
Klassediagrammer
Add an item
Acceptance Criteria
Delete
0%
De relevante klasser har en liste af tags
Lister i GUI kan filtreres efter valgt tag
En bruger opretter selv tags
En bruger kan slette tags
En bruger kan lave søgninger og filtrere en liste således
Add an item


Formatering
in list Product Backlog
Description
Edit
Som bruger
Vil jeg gerne kunne formatere teksfelter
Så jeg nemmere kan danne mig et overblik over mine noter
Business Value: 5
Story points: 6
Tasks
Delete
0%
GUI: formateringsbar
Add an item
Acceptance Criteria
Delete
0%
Der skal kunne laves fed, kursiv og understreget skrift, samt punkt opsætning
Add an item

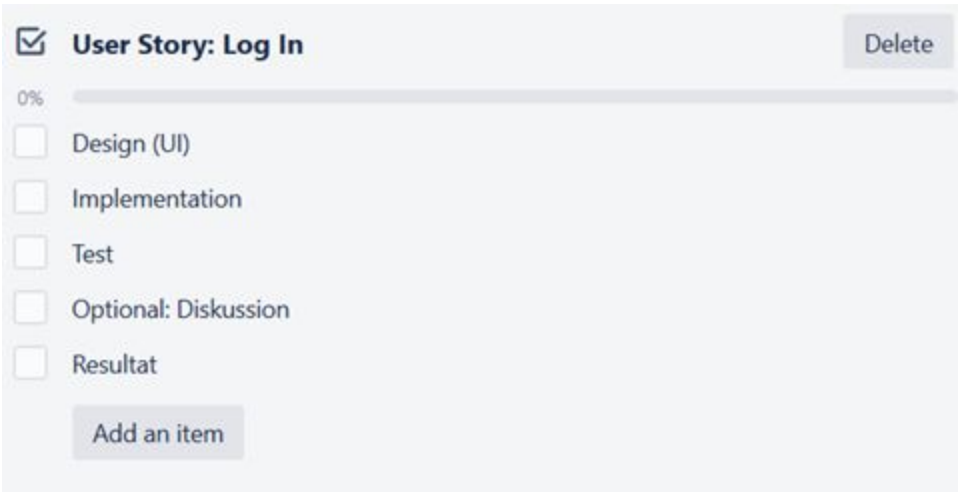
6. 2: Tasks og Rapport



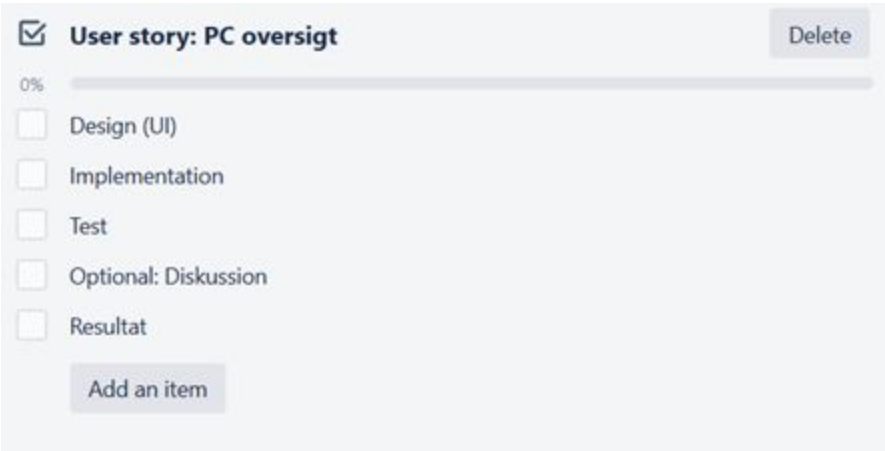

Rapport: Sprint 0
in list Done
MEMBERS
LABELS
LV
Rapport
Sprint 2
Description
Edit
Story points: 2
Tasks
Delete
0%
Beskrivelse
Review
Retrospective
Konklusion
Add an item

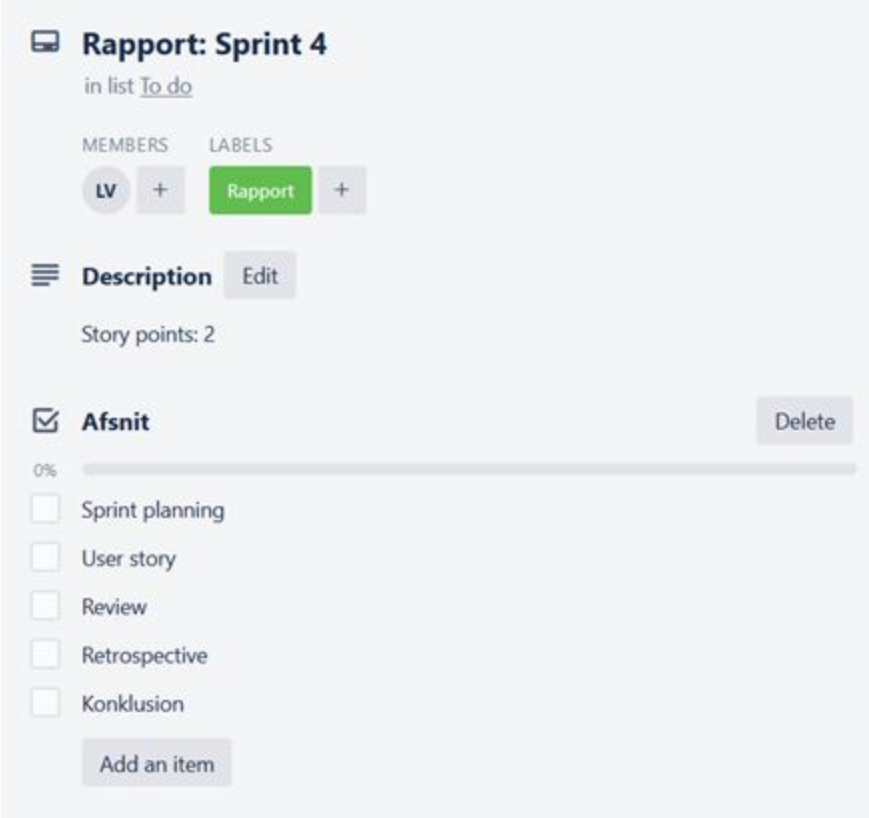

Rapport: Sprint 1
in list Done
MEMBERS
LABELS
LV
Rapport
Sprint 2
Description
Edit
Story points: 2
Afsnit
Delete
0%
Sprint planning
User story
Review
Retrospective
Konklusion
Add an item


Rapport: Sprint 2
in list Done
MEMBERS
LABELS
LV
L
Rapport
Sprint 3
Sprint 4
Description
Edit
Story points: 2
Afsnit
Delete
0%
Sprint planning
User story
Review
Retrospective
Konklusion
Add an item


User Story: Log In
Delete
0%
Design (UI)
Implementation
Test
Optional: Diskussion
Resultat
Add an item

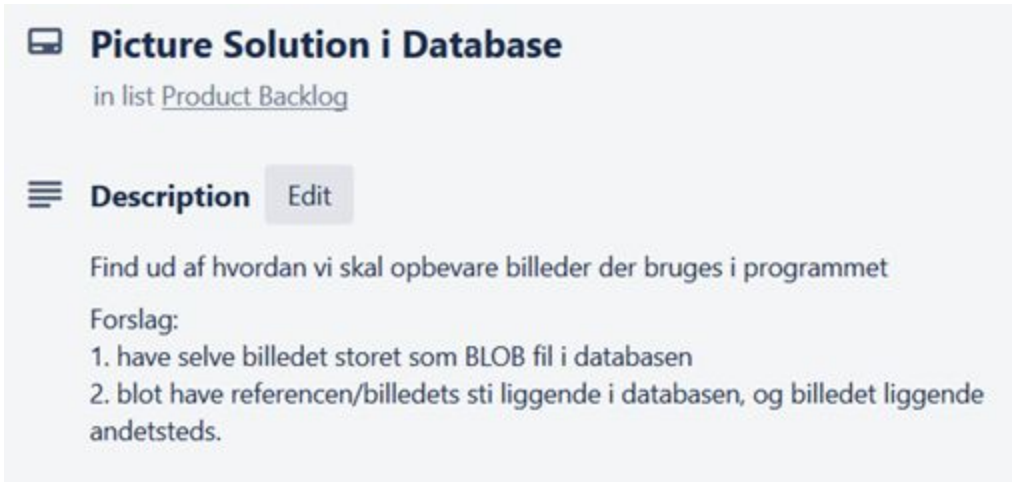

Rapport: Sprint 3
in list To do
MEMBERS
LABELS
K
LV
+
Rapport
+
Description
Edit
Story points: 2
Afsnit
Delete
0%
Sprint planning
User story
Review
Retrospective
Konklusion
Add an item

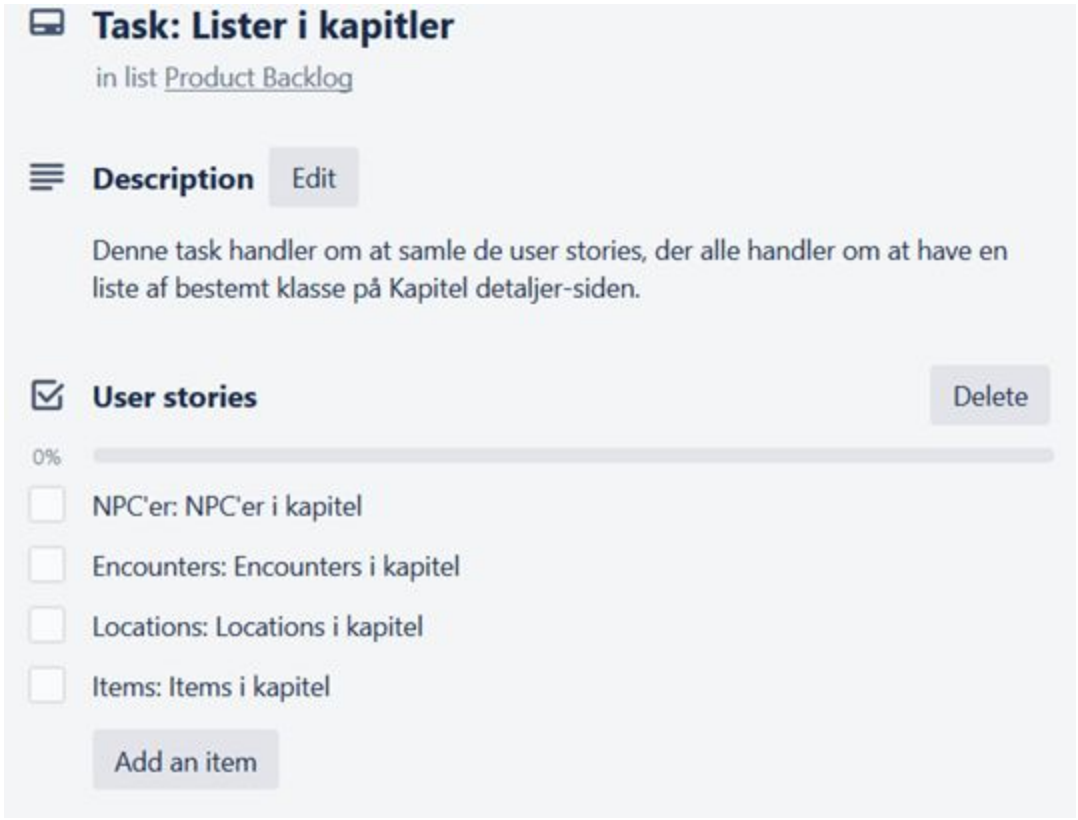

User story: PC oversigt
Delete
0%
Design (UI)
Implementation
Test
Optional: Diskussion
Resultat
Add an item


Rapport: Sprint 4
in list To do
MEMBERS
LABELS
LV
Rapport
Description
Edit
Story points: 2
Afsnit
Delete
0%
Sprint planning
User story
Review
Retrospective
Konklusion
Add an item


User Story: Kapitel oversigt
Delete
0%
Design + UI
Implementation
Test
Resultat
Add an item


Webservice som seperat projekt
in list Done
Description
Edit
Lav seperat projekt til webservicen som uploades som seperat github repository og skal ligge i cloud


Picture Solution i Database
in list Product Backlog
Description
Edit
Find ud af hvordan vi skal opbevare billeder der bruges i programmet
Forslag:
1. have selve billedet storet som BLOB fil i databasen
2. blot have referencen/billedets sti liggende i databasen, og billedet liggende andetsteds.


Task: Lister i kapitler
in list Product Backlog
Description
Edit
Denne task handler om at samle de user stories, der alle handler om at have en liste af bestemt klasse på Kapitel detaljer-siden.
User stories
Delete
0%
NPC'er: NPC'er i kapitel
Encounters: Encounters i kapitel
Locations: Locations i kapitel
Items: Items i kapitel
Add an item