Projekt Genspil

Skrevet af:

- Martin Herskind Laursen.
- Mette Åse Petersen.
- William Bøgh Christensen.
- Anders Holm.
- Peter Ilsted Bech.

Hold- eller klasse: DMOoF24 – Team 10.

Lærer eller vejleder:

Programmering: Thomas Tjellesen, Diaa Zobair Shollar.

Systemudvikling: Lene Vestergaard Andersen og Anja Birkelund.

Uddannelse: Datamatiker Online

Aflevering: 14/4 klokken 21:00

Figur liste:

Abstract / resumé

Dette projekt sigter mod at udvikle et effektivt system til lagerstyring for Genspil, en virksomhed der sælger brugte brætspil. Virksomheden oplever udfordringer med deres nuværende system der består af en Word-liste over lagerbeholdningen. Dette er ineffektivt, uoverskueligt, og besværligt at vedligeholde. Projektet vil fokusere på at løse disse udfordringer ved at tilbyde funktioner såsom registrering og opdatering af spil på lager, mulighed for søgning efter spil baseret på forskellige kriterier, håndtering af forespørgsler på spil, mulighed for at tilføje nye spil til systemet og individuel prissætning af hvert spil baseret på dets stand. Derudover vil systemet muliggøre udskrivning af en lagerliste til lageroptælling. Fokusset vil være på brugervenlighed og funktionalitet.

Indledning

Brætspil har altid været et populært produkt med evnen til at samle familie og venner til mange timers sjov underholdning, men i en verden, der i stigende grad domineres af digital underholdning, har markedet for fysiske brætspil skulle tilpasse sig. Dette har for mange virksomheder været en svær overgangsperiode. Selvom efterspørgslen for digitale underholdnings produkter er markant steget, eksisterer der stadig et stort marked for fysiske brætspil. Derfor er nødvendigheden for tilpasning og optimering af IT-system essentielt for at forblive konkurrencedygtige på markedet. Projektet har dermed til formål at skabe et nyt funktionelt IT-system til virksomheden Genspil. Projektet omfatter udover et funktionelt IT-system også en række relevante modeller og use cases.

Problemformulering

Genspil, en virksomhed der handler med brugte brætspil, mangler et effektivt system til at håndtere deres lagerstyring. Den nuværende metode med en Word-liste er ineffektiv, uoverskuelig og besværlig at vedligeholde. Der er behov for et system, der giver et bedre overblik over lagerbeholdningen og spillets stand, samt mulighed for at søge på spil ud fra forskellige kriterier som genre, antal spillere, pris osv. Desuden skal systemet kunne håndtere forespørgsler på spil, herunder registrere hvilke spil der er efterspurgt og af hvem. Der er også behov for at kunne lægge nye spil ind i systemet, også selvom de ikke er på lager, samt valgmulighed for at individuelt prissætte hvert spil baseret på dets stand. Et krav fra virksomheden er adgang til at udskrive en lagerliste til lageroptælling, der kan sorteres både efter navn og genre. Systemet skal være brugervenligt og funktionelt, hvor funktionalitet vægter højere end æstetik.

Afgrænsning

Systemet vil primært fokusere på at løse Genspils behov for en mere effektiv lagerstyring af deres brugte brætspil. Dette inkluderer funktioner som:

1. **Overblik over lageret:**
    - Skabe et centralt sted, hvor alle tilgængelige brætspil kan ses.
    - Mulighed for at se, hvilke spil der er tilgængelige i lageret, og hvilke der er reserveret eller bestilt.
2. **Spilinformation:**
    - Gemme detaljeret information om hvert spil, herunder titel, genre, antal spillere, stand, pris og eventuelle andre relevante oplysninger.
    - Mulighed for at tilføje nye spil til systemet, også hvis de ikke er på lager, men der er efterspørgsel efter dem.
3. **Søgefunktionalitet:**
    - Mulighed for at søge efter spil baseret på forskellige kriterier, såsom genre, antal spillere, stand, pris og navn.
4. **Forespørgsler og reservationer:**
    - Registrere forespørgsler fra kunder om bestemte spil.
    - Mulighed for at se, hvilke spil der er forespurgt på, og hvem der har lavet forespørgslen.
5. **Prissætning:**
    - Mulighed for at indstille individuelle priser for hvert spil baseret på dets stand.
6. **Udskrivning af lagerliste:**
    - Mulighed for at udskrive en lagerliste til lageroptælling.
    - // Sorteringsmuligheder efter spilnavn, genre og andre relevante kriterier.
7. **Brugervenlighed:**
    - Systemet skal være intuitivt og let at bruge for medarbejdere.

- Funktionalitet skal prioriteres over æstetik.

Projektet vil ikke omfatte en analyse eller implementering af multisøgekriterier i det nye lagerstyringssystem.

Derudover har teamet grundet tidspres, kun formået at tilføje en automatisk sortering af stand, for hver liste over fysiske eksemplarer, som bliver instantieret ved tilføjelse af nyt eksemplar.

Projektstyring

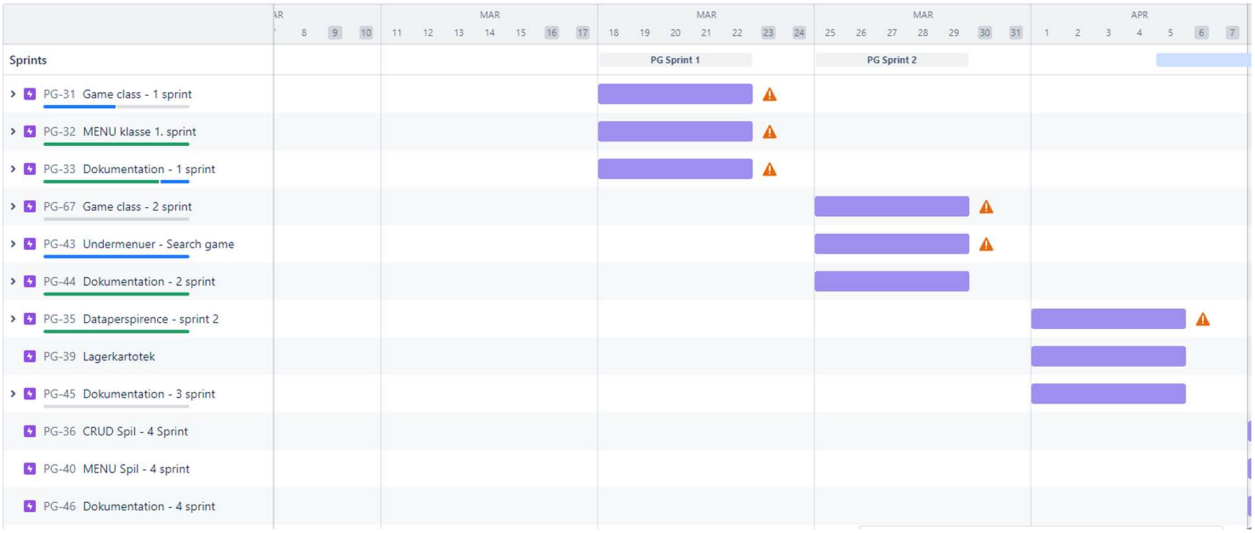
Scrum-Framework:

Scrum-Framework er anvendt til udviklingen af systemet, herunder sprintplanlægning, sprint gennemgang og retrospektiv.

Jira er blevet benyttet for at skabe overblik over de forskellige events der sker ved brug af Scrum.

Scrum Planning:

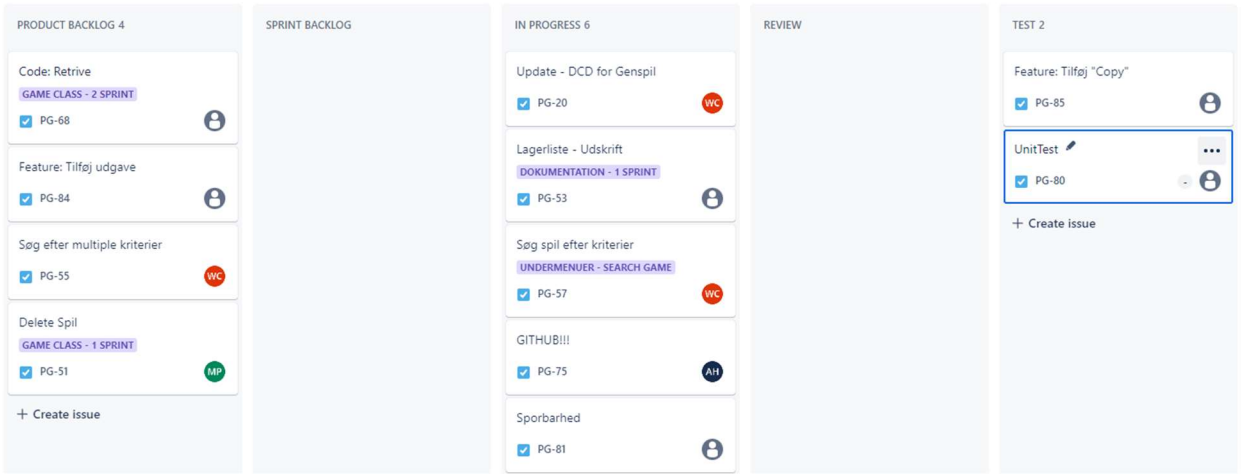
Nedstående figur viser tidsplanen som gruppen har prøvet at forholde sig til igennem hele projektet. Hver uge er et sprint og efter hver slutning vil der blive holdt retrospektive møde.



Figur 1. Scrum Timeline

Sprint Backlog:

Her er et uddrag fra en af de sprint som gruppen har arbejdet ud fra. Ved hjælp af Jira kan disse opgaver flyttes til det stadie hvor de høre til i processen.



Figur 2: Scrum sprint backlog

Scrum retrospective:

Her er et eksempel på et Scrum meeting som gruppen har planlagt i start sprintet.

I "Blokade"-kolonnen er der skrevet, grundet mangel på viden af persistens, at denne del af opgaven ikke vil kunne udføres på daværende tidspunkt.

"Hvad har jeg gjort"-kolonnen viser at der er udført nogle af opgaverne til sprint 1 så de vil kunne sættes i done, i sprintet boardede.

Til sidst kan vi se en, "Hvad vil vi gøre til næste gang"-kolonne, der viser hvad vær gruppemedlem har til opgave til næste samling.

Scrum meeting - sprint 1

Page Properties

Date	18-03-24
Team	10
Participants	Alle

Retrospective

Blokade	Hvad har jeg gjort	Hvad vil vi gøre? til næste gang
Mangle på viden - persistens	• Create spil • Basic menu • pseudo kode til update spil • Scrum planning	• Martin: Update spil • Peter: create spil • Mette: Use Cases • Anders: Remove - code

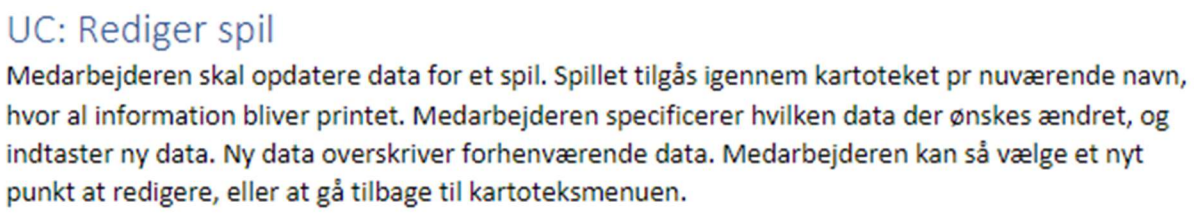
Figur 3. Retrospektive

Kravindsamling:

Kravsamling er den proces der anvendes til at undersøge og forstå de præcise krav, som projektet har brug for. Projektet har i kravindsamling haft opmærksomheden rettet mod hvilke krav Genspil har efterspurgt til det nye system. I dette projekt er der kun anvendt en kilde, nemlig Genspil, til indsamling af krav.

High Level Design

Use Case:

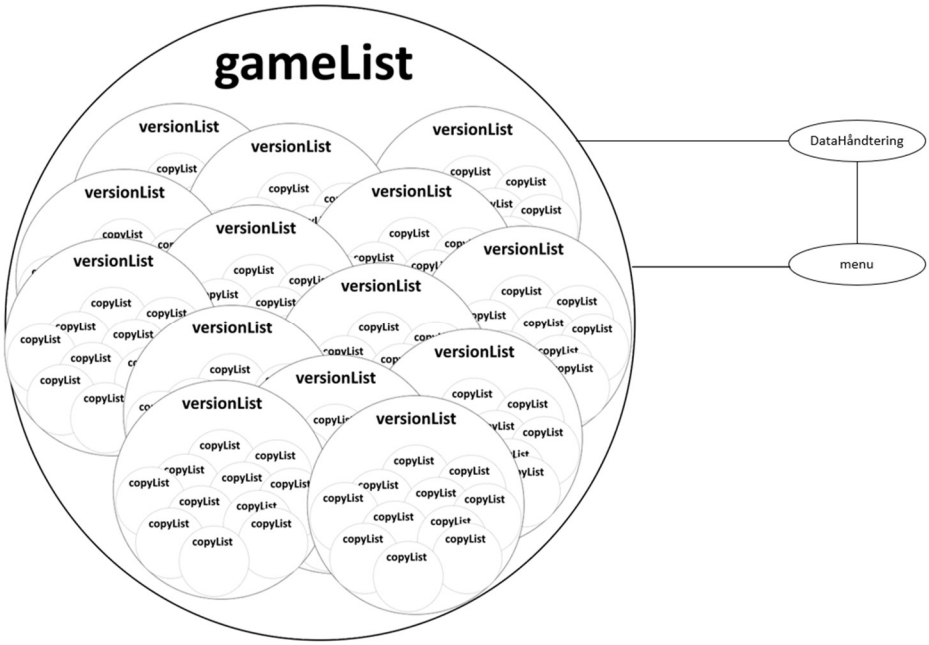


Figur 4. Use case: Rediger spil

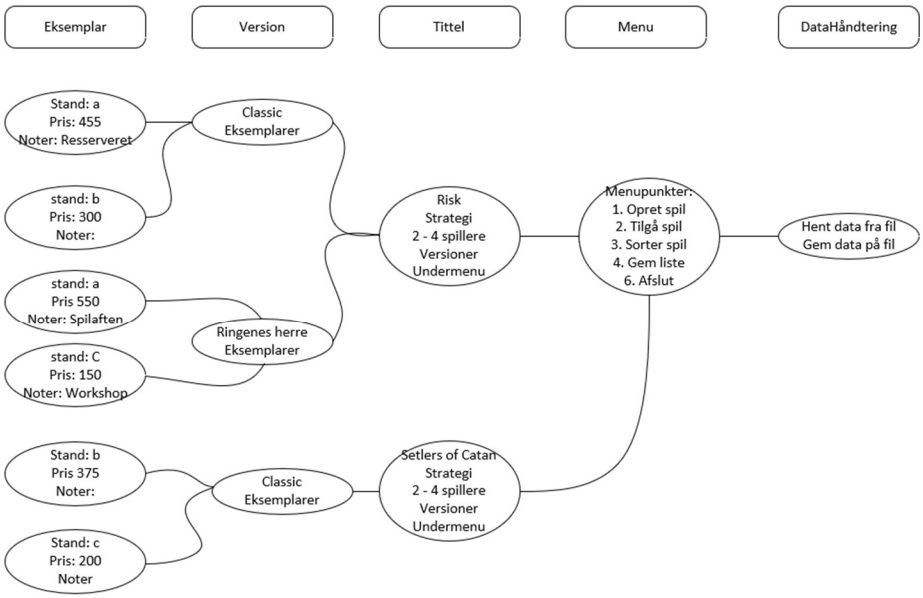
Figur 4 viser en af projektets mange use cases, hvor de resterende kan findes i bilags sektionen. Use casene var nogle af de første elementer lavet i projektet, og udgøre grundstenene for meget af resten af projektet.

Object Model:

Denne object model viser projektets tanke med inception. Her kan ses at gameList'e indeholder versionList som indeholder copyList. Dette er gjort som en tanke at et spil kan have flere versioner og at man kan have flere kopier af hver version.



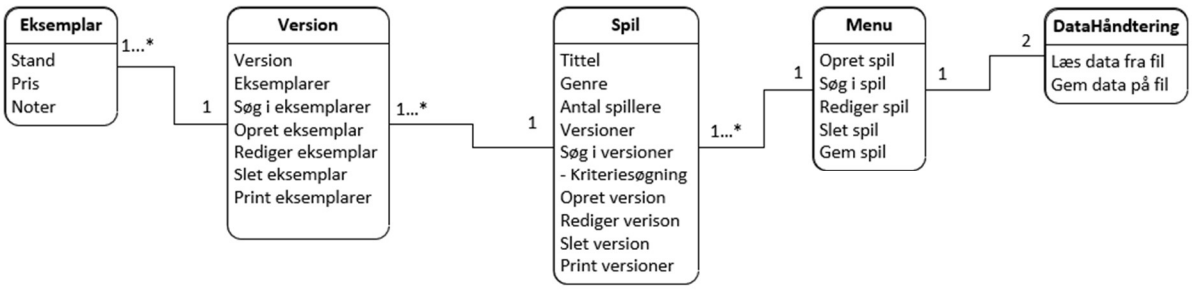
Figur 5. Object model: Inception



Figur 6. Object model: Original

I denne objektmodel er projektet som helhed blevet udformet med en omfattende forståelse af dets formål og funktionalitet. Hvert enkelt objekt er designet til at udføre specifikke opgaver og bidrage til projektets mål. Der er lagt vægt på at sikre, at objekterne arbejder sammen og kommunikerer sammen for at opnå ønskede resultater. Denne objektmodel tjener som et fundament, der gør det muligt at struktureret projektet.

Domænemodel:

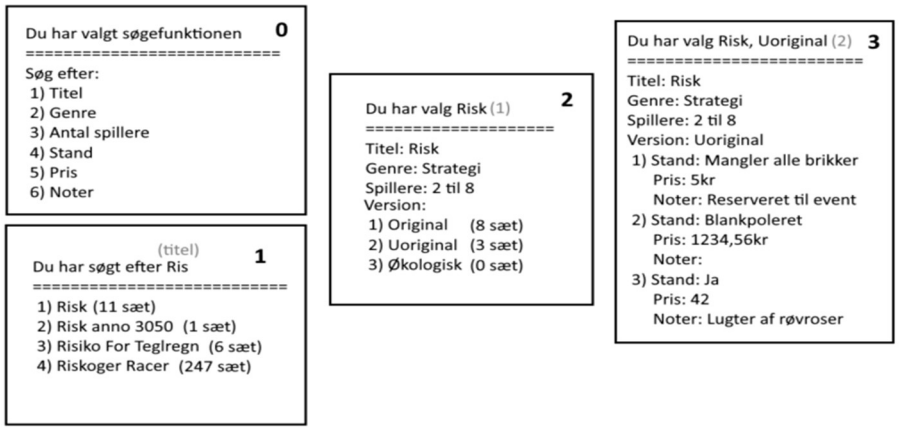


Figur 7. Domænemodel.

I figur 7 ses projektets domænemodel. Modellen har 5 klasser. Spil klassen, Menu klassen, Version klassen, Eksemplar klassen og Datahåndtering klassen. Alle 5 klasser har en række diverse attributter, der er defineret af deres klasse. Domænemodellen illustrer også et overblik over klassernes associeringer. Associeringen er repræsenteret som en linje mellem klasserne. Desuden er klassernes associering også anvist med deres multiplicity. Denne domænemodel har bidraget til at danne overblik, samt retningslinje for projektets aktuelle programmering.

Wireframes

Wireframesne giver visualisering af systemets brugergrænseflade og funktionalitet. Derudover skaber det klarhed til hvor projektet gerne skal bevæge sig imod. Nedstående Figur (Figur 8: Wireframes) viser tilgangen til søgefunktionen(frame0). Bruger får fremvist hvilke søgningskriterier der er. Derefter er søgning på titel er valgt(frame1). Efterfølgende vil brugeren blive præsenteret for versionerne for det tilgåede spil(frame2). Når versionen så er valgt, kan bruger se de forskellige sæt med stand, pris og, noter(frame3).



Søgnings kriterier, input, og output:
- Søgt efter "Strategi"
1) Risk, Strategi

- Søgt efter "Spillere 5" (Højere end minimum, lavere end maximum)
1) Risk, 2 til 8

- Søgt efter "Stand A" (Minimum input)
1) Risk, A

- Søgt efter "Pris 100,00kr" (Maximum input)
1) Risk, 123kr

- Søgt efter "Noter Mette" (String input)
1) Risk, Reserveret af Mette P

Figur 8. Wireframe

Low Level Design

DCD - Design class diagram:

DCD'en er et dynamisk diagram, der løbende er blevet opdateret og tilpasset under projektet. Formålet med dette er at sikre diagrammet repræsenterer systemets design og krav. Diagrammet giver udviklingsteamet struktur, samt et overblik over systemets klasser og deres relationer.

I projektets DCD er der identificeret og defineret de nødvendige klasser og deres relationer baseret på kravene og analysen af Genspils behov. Klasserne repræsenterer de forskellige konceptuelle enheder i systemet, såsom spil, brugere, forespørgsler og lagerstyring.

Nedenfor er en kort beskrivelse af nogle af de vigtigste klasser i vores DCD:

Game class indeholder GameVersion class som indeholder GameCopy Class. Dette bliver refereret til inception i teksten, da det er gruppens ordbrug om konceptet.

Game Class:

- Denne klasse indeholder variablerne for Titel, Genre, Min og Max spillere, samt instantieringen af GameVersion, hvor variablerne Version, Stand, Pris og Noter bliver overført.
- Dette er den yderste skal i vores inception koncept. *Se figur 5. Object model: Inception*

GameVersion Class:

- Denne klasse indeholder variablen for version, samt hele listen af hver enkelt digital repræsentation, af hvert eksemplar af den givne titel. Hvert eksemplar bliver herefter tildelt variablerne for Stand, Pris og eventuelt noter.
- Dette er det næste step i vores inception, hvor Game Class indeholder mange GameVersion'er. *Se figur 5. Object model: Inception*

GameCopy Class:

- Denne klasse er hvor hvert eksemplar af det enkelte spil, og version af et spil, med oplysninger som stand, pris og noter.
- Disse spil er sæts af versionerne.
- Dette er det sidste step i inception. Her vil der eksistere mange GameCopy'er i en version. *Se figur 5. Object model: Inception*

Menu Class:

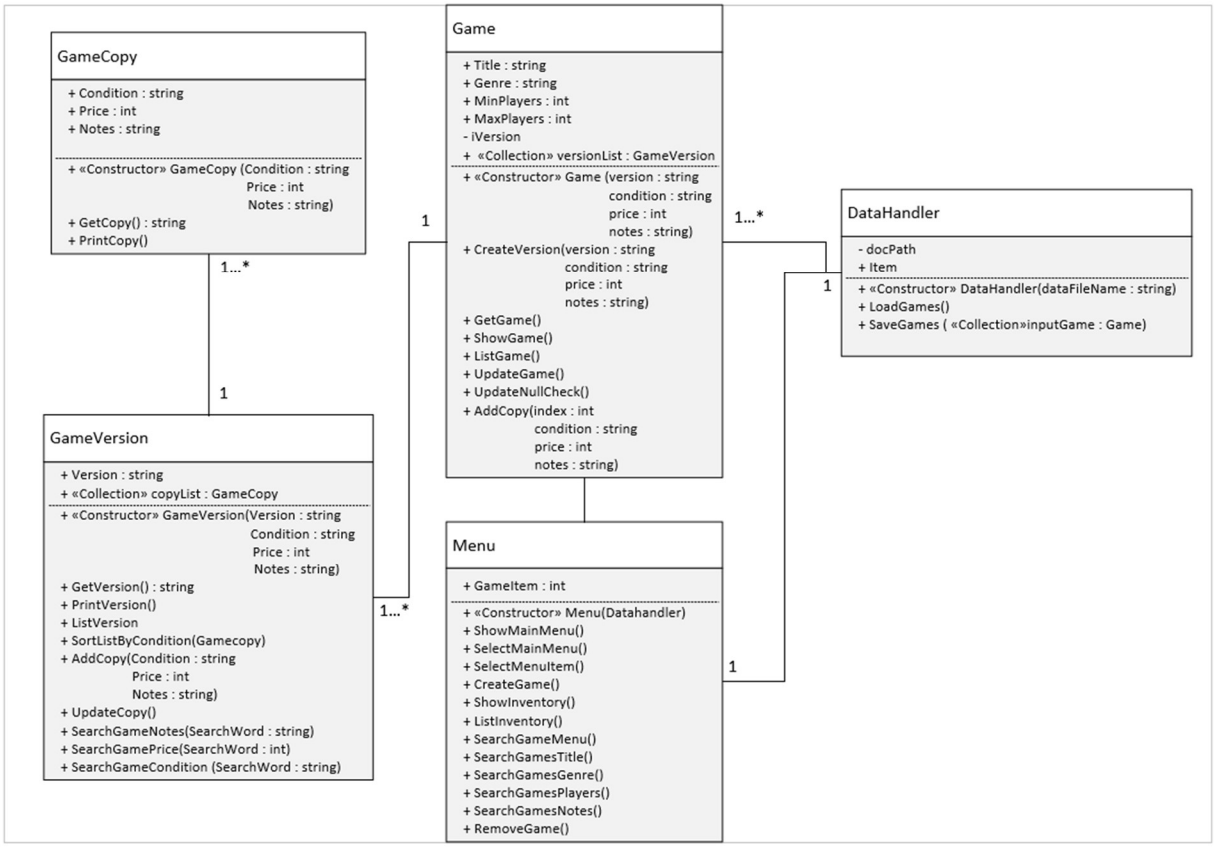
- Denne klasse styrer menuen i systemet og fremviser hvordan brugeren kan navigere rundt i funktionerne. Derudover indeholder den "søg menu" hvor brugeren hvor brugeren kan søge under enkelte kriterier. Den har også funktioner som oprette eller fjerne et spil.

DataHandler Class:

- Denne klasse håndterer indlæsning og lagring af spildata til og fra en fil. Den vil også indlæse og gemme spil fra en tekstfil.

Program Class:

- Denne klasse indeholder metoden **Main()**, som er indgangspunktet til programmet. Det opretter en instans af **Menu**-klassen og starter hovedmenuen.



Figur 9. Design model diagram

Pseudokode: Udkast til visionen for Game – klassen

---

erklær variabler

- titler
- udgave
- genre
- antal spillere
- stand
- antal
- pris
- note

erklær metoder

- metode til at oprette spil
- metode til at printe spil
- metode til at printe spilkartotek (titel, udgave, antal)
- metode til at rette titel
- metode til at rette udgave
- metode til at rette genre
- metode til at rette antal spillere
- metode til at rette stand
- metode til at rette antal
- metode til at rette pris
- metode til at rette noter

Under design af første vision skrev teamet pseudokode for hver enkelt klasse, for at give overblik over, hvad der skulle være, samt opbygning af menuer og undermenuer.

Programming

```
414     void SearchGamesNotes(string searchWord)
415     {
416         Console.WriteLine($"Resultatet af din søgning: '{searchWord}'");
417         bool found = false;
418         foreach (Game game in gameList)
419         {
420             for(int i = 0; i < game.versionList.Count; i++)
421             {
422                 for (int j = 0; j < game.versionList[i].copyList.Count; j++)
423                 {
424                     if (game != null && game.versionList[i].copyList[j].Notes.IndexOf(searchWord, StringComparison.OrdinalIgnoreCase) >= 0)
425                     {
426                         Console.WriteLine($"{game.Title}, {game.versionList[i].Version}, {game.versionList[i].copyList[j].GetCopy()}");
427                         found = true;
428                     }
429                 }
430             }
431 
432         }
```

Figur 10. SeachGamesNotes

I dette kodeeksempel er der taget udgangspunkt i metoden SearchGamesNotes() hvor den vil tilgå noterne til de enkelte eksemplarer af hver spil.

1. Først tjekker den om objektet er null.
2. Derefter vil den gennemgå hver spil i gameList.
3. Efterfølgende vil den gennemgå hver version i versionList.
4. Til slut vil den så gå ned i det sidste lag for at gennemgå hver copy for at finde noten som brugeren søger.

```
2 references | MHLa, 1 day ago | 2 authors, 5 changes
public void SaveGames(List<Game> inputGame)
{
    DateTime date = DateTime.Now;

    using (StreamWriter sw = new StreamWriter(Path.Combine(docPath, DataFileName), false))
    {
        sw.WriteLine($"Lagerliste Genspil - {date}");   // Using full to check saving functionality, but ToShortDateString is good too
        //sw.WriteLine($"Lagerliste Genspil - {date.ToShortDateString()}");
        foreach (Game game in inputGame)
        {
            sw.WriteLine(game.GetGame());
            for (int i = 0; i < game.versionList.Count; i++)
            {
                sw.WriteLine(game.versionList[i].GetVersion());
                for (int j = 0; j < game.versionList[i].copyList.Count; j++)
                {
                    sw.WriteLine(game.versionList[i].copyList[j].GetCopy());
                }
            }
            sw.WriteLine("-");  // Special spacing used for loading. Also is more pleasing within the stock list
        }
    }
```

Figur 11. Datahandler

Dette kodeeksempel tager udgangspunkt i metoden SaveGames(), hvor listen af spil bliver løbet igennem, og for hver titel, bliver fundet versionerne, og de enkelte eksemplarer af spillene, og lister det hele op i tekstdokumentet efter:

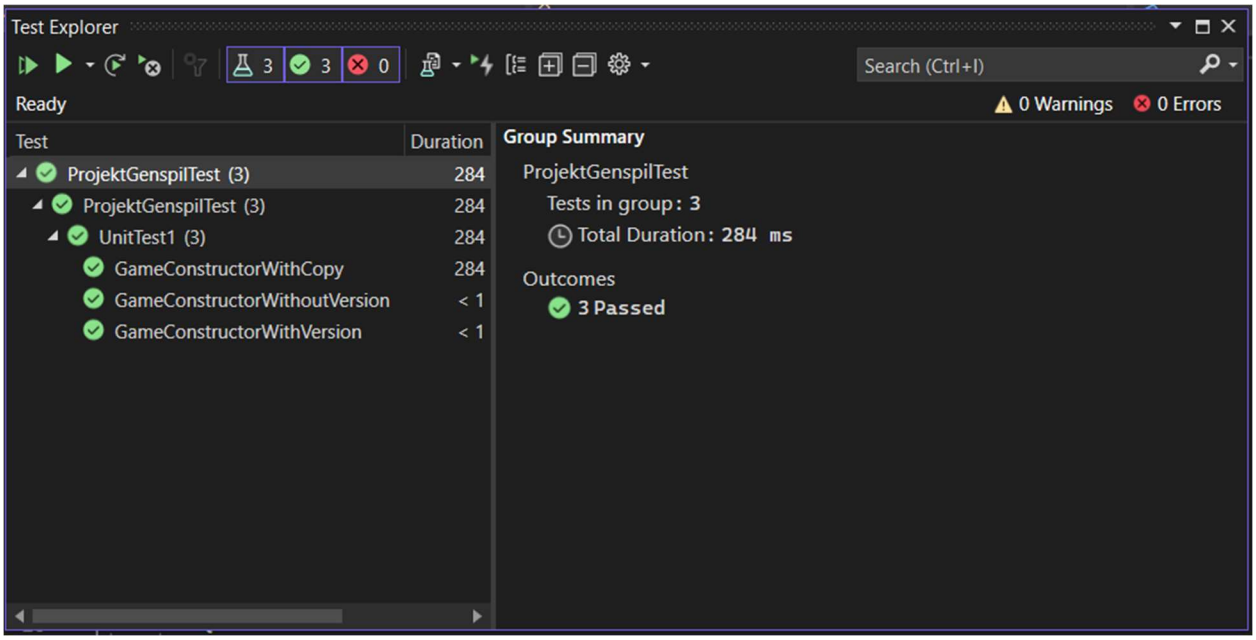
- Titel, genre, min og max players.

- Versioner.
  - Individuelle eksemplar, med stand, pris og evt. Noter.

Test

UnitTest

Under udarbejdelsen af Inception konceptet, har teamet valgt at programmere en række tests, for at teste funktionaliteten af oprettelsen af et fuldt spil, ned igennem lagene i vores objekter, samt metoderne for oprettelsen af nye versioner, og eksemplarer, i deres respektive "lag".



Figur 12. UnitTest

Diskussion og vurdering

Projektet Genspil havde til mål at udvikle et system til lagerstyring for virksomheden Genspil. Gennem projektet blev der arbejdet på at analysere og løse de udfordringer, som virksomheden stod overfor med deres nuværende manuelle lagerstyringssystem baseret på en Word-liste.

Hovedformålet med projektet var derfor at skabe et system, der kunne tilbyde bedre overblik over lagerbeholdningen samt mulighed for at søge efter spil baseret på forskellige kriterier som genre, antal spillere, pris osv. Ved at lave funktioner som registrering og opdatering af spil, individuel prissætning af hvert spil og udskrivning af en lagerliste, blev disse udfordringer løst.

En central del af projektets arbejde var at designe et designclassdiagram (DCD), der gav overblik over systemets struktur og relationer mellem klasserne. DCD'en gav et klart overblik over systemets arkitektur og var afgørende for en implementering af systemet. Desuden blev der under udviklingsprocessen anvendt Scrum-frameworket, hvilket gav en projektstyrings form med løbende tilpasninger og forbedringer. Jira har som udgangspunkt været værktøj til projektstyring, bidraget til overblik og struktur over projektets forløb i starten. Der har dog været tidspunkter efter påskeferien hvor projektstyringen faldt fra hinanden og overblikket var intet sted at finde. Ud fra denne erfaring har gruppen forstået vigtigheden af at bruge scrum og Jira som værktøj, samt nødvendigheden af statusmøder, for at få alle ender til at mødes.

En vigtig del af projektet var også testfasen, hvor der blev udarbejdet en række unit tests for at sikre at projektet kom i mål. Disse tests bidrog til at finde fejl og rette mulige fejl og manglende kode, og skal i fremtidige projekter spille en større rolle.

Når der reflekteres over projektets forløb, kan gruppen se tilbage på omstruktureringer hvor der har været v1, v2, v3 og til sidst v-inception. Dette kunne have være omgået ved en bedre kommunikation, en klarhed for visionen og at have dokumentation i orden fra tekst til diagram til programmering. Dette ses som en erfaring og ikke et nederlag i processen. I sidste ende står gruppen med et produkt som opfylder kravene.

Konklusion

Projekt Genspil havde til formål, at udvikle samt kode et nyt funktionelt it-system til kunden Genspil. På baggrund af dette formål kan der konkluderes, at bearbejdelsen af projektets DCD gav overblik over systemets klasser og deres relationer. Ud fra DCD'et kan der konkluderes at de primære klasser i vores it-system var Game Class, GameVersion Class, GameCopy Class, Menu Class, DataHandler Class og Program Class. Projektet kan også konkludere, at det færdigkodet system kan håndtere forespørgsler på spil, herunder registrere hvilke spil der er efterspurgt og af hvem. Desuden kan systemet også lægge nye spil ind i systemet, også selvom de ikke er på lager, samt mulighed for at individuelt prissætte hvert spil baseret på dets stand.

Derudover kan der konkluderes at projektets brug af inception, som kan ses via gameList'e indeholder versionList som indeholder copyList. Har gjort sådan så et spil kan have flere versioner og at man kan have flere kopier af hver version. Der kan også konkluderes, at brugen af wireframes har gjort det nyttigt at visualisere systemets brugergrænseflade og validere designbeslutninger. Til sidst kan der konkluderes, at projektets team har programmeret en række teste. Testene har testet funktionaliteten af oprettelsen af et fuldt spil, ned igennem lagene i vores objekter, samt metoderne for oprettelsen af nye versioner, og eksemplarer, i deres respektive "lag".

Bilag

Pseudo – menu:

Erklær metode for at fremvise menu

- Ryd konsol
- print titel (“Genspil”)
- print kartotek (titel - undertittel / antal)
- print første punkt - Søg spil
- print andet punkt - Ret spil
- print tredje punkt - Opret spil
- print fjerde punkt - Slet spil
- print femte punkt - Note til spil

Erklær metode til at vælge menupunkt

- erklær loop (true)
    - modtag input fra konsol
    - opvej input mod valgmuligheder
        - første valgmulighed - kald søge metode fra klasse Spil
        - anden valgmulighed - kald ret spil metode fra klasse spil
        - tredje valgmulighed - kald opret metode fra klasse spil
        - fjerde valgmulighed - kald slet metode fra klasse spil
        - femte valgmulighed - Kald note metode fra klasse spil (til reservation/udlån)
        - else - print “forkert input, vælg ovenstående”

Pseudo – Undermenu:

Instantier metode for undermenu

- Ryd konsol
- print titel (“Genspil”)
- print kartotek (titel - undertitel / antal)

- print første punkt - (x) Søg spil
    - print første punkt (1) Søg efter titel
    - print andet punkt (2) søg efter udgave
    - print tredje punkt (3) søg efter genre
    - print fjerde punkt (4) søg efter flere ting
- print andet punkt - (x) Ret spil
- print tredje punkt - (x) Opret spil
- print fjerde punkt - (x)Slet spil
- print femte punkt - (x) Note til spil

Use Case 2:

UC: Søg efter spil

En kunde kommer med en forespørgsel på et bestemt spil, hvor kunden er ikke helt sikker på, hvilken version af spillet der ønskes. Derefter søger medarbejderne efter dette spil inde i systemet.

Use Case 3:

UC: Opret spil

Medarbejder modtager nyt spil, som skal oprettes i systemet. Medarbejderen tilgår systemet, og indtaster først Navn på spillet, og derefter informationer som Version, Årgang, Antal spillere, Genre, Antal, stand og pris. Det nye spil gemmes herefter i systemet, og i den Dataperspirente fil.

Use Case 4:

UC: Lagerliste

En medarbejder kan ikke finde et spil i butikken, derfor ønsker denne medarbejder at få vist en lagerliste. Det gøres ved at tilgå systemet. Først skal medarbejderen tilgå menu og få udskrevet en lagerlist.

Use Case 5:

UC: Udskrift af lagerliste

En medarbejder kan ikke finde et spil i butikken. Derfor ønsker han at få en lagerliste udskrevet. Derfor skal han tilgå systemet, eftersom lagerlisten indeholder alle butikkens spil, derefter kan han tjekke op på, hvad der er gået galt.

UnitTests – Kode:

```
using Projekt_Genspil_v._2;

namespace ProjektGenspilTest
{
    [TestClass]
    0 references | Peter Ilsted, 45 minutes ago | 1 author, 1 change
    public class UnitTest1
    {
        Game g1, g2, g3;

        [TestInitialize]
        0 references | Peter Ilsted, 45 minutes ago | 1 author, 1 change
        public void Init()
        {
            g1 = new Game("Risk", "Classic", "Strategi", 2, 6, "a", 300, "Resserveret");
            g2 = new Game("Cluedo", "Den bedste version", "familie", 2, 5);
            g3 = new Game("Kalaha", "Familie", 1, 2);
        }

        [TestMethod]
        0 references | Peter Ilsted, 45 minutes ago | 1 author, 1 change
        public void GameConstructorWithoutVersion()
        {
            Assert.AreEqual("Spil: Kalaha -- Genre: Familie -- Spillere: 1 til 2", g3.GetGame());
        }
        [TestMethod]
        0 references | Peter Ilsted, 45 minutes ago | 1 author, 1 change
        public void GameConstructorWithVersion()
        {
            Assert.AreEqual("Spil: Cluedo -- Genre: familie -- Spillere: 2 til 5", g2.GetGame());
            Assert.AreEqual("Version: Den bedste version", g2.versionList[0].GetVersion());
        }
        [TestMethod]
        0 references | Peter Ilsted, 45 minutes ago | 1 author, 1 change
        public void GameConstructorWithCopy()
        {
            Assert.AreEqual("Spil: Risk -- Genre: Strategi -- Spillere: 2 til 6", g1.GetGame());
            Assert.AreEqual("Version: Classic", g1.versionList[0].GetVersion());
            Assert.AreEqual("Stand: a -- Pris: 300 -- Noter: Resserveret", g1.versionList[0].copyList[0].GetCopy());
        }
    }
}
```