Rapport Lab 4 Version 3 – Örjan Andersson (alfoande100)

1. Antaganden

- Data på produkter sparas/läses från en fil på CSV format och UTF8. Format på CSV filen är följande: Type,Id,Name,Price,Author,Platform,Genre,Format,Language,PlayTime,Stock

- Data på order sparas/läses från en fil på CSV format och UTF8. Format på filen är följande: OrderId,OrderDate,ProductId,Name,Type,Quantity,Price

- Har valt att inte använda en FileOpenPicker för bättre upplevelse. När programmet startar upp första gången så ser den efter en produktfil lagrat under ApplicationData.Current.LocalFolder. Om filen finns så är det denna filen som används för att hämta och lagra produkter till. Om filen inte finns så läser man in en ”startfil” som innehåller de produkter som är specificerade i uppgift 4. Har valt godtycklig tillgänglighet antal på varje produkt för att ha något att starta med. Dessa produkter kopierar över till en fil på ApplicationData.Current.LocalFolder så att användaren slipper själv att ta ställning till var denna filen finns. På samma plats kommer sålda produkter att lagras till. Varje order lagras i en CSV fil också. Det är viktigt att man inte manuellt går in i dessa filer när programmet är i gång. Var filerna finna kan man se på en egen tabb i lösningen som heter Info. Programmet kan ”nollställas” genom att alla filerna som finns under fil ApplicationData.Current.LocalFolder tas bort.

- Antar att det är viktigast för butiken att ha filter på sina produkter för effektiv försäljning. Här kan man bygga ut funktionalitet senare om man vill.

- Antar att en retur av produkt sker i lagersystemet

- Antar att ett kvitto enbart skrivs ut för senaste köpet. Det finns med information i lagrad order så man kan lösa detta senare om man vill.

- För att flytta produkter fram och tillbaka mellan butik och korg så använder man knappar och inte dubbelklick

- Man har 1 generell produkt med begränsad information om vad som skall lagras. Har man behov för mer detaljer för en produkt så måste koden tillpassas till det. Just nu är det tillpassat för bok, video och spel. Kommer det in en ny produkt som inte är bok, video eller spel kan man fortfarande använda produkt.

- Sortering av produkter är inte tagit hänsyn till. De kommer bara i den ordning som man har lagrat dem i CSV. Nya produkter kommer alltid sist i listan.

- Vad som är nyttig information om produkt är satt till att i listor visar det jag menar är mest viktigt och gemensamt. Om man vill ha alla detaljer om en produkt måste man se på en produkt på lager genom att klicka på den.

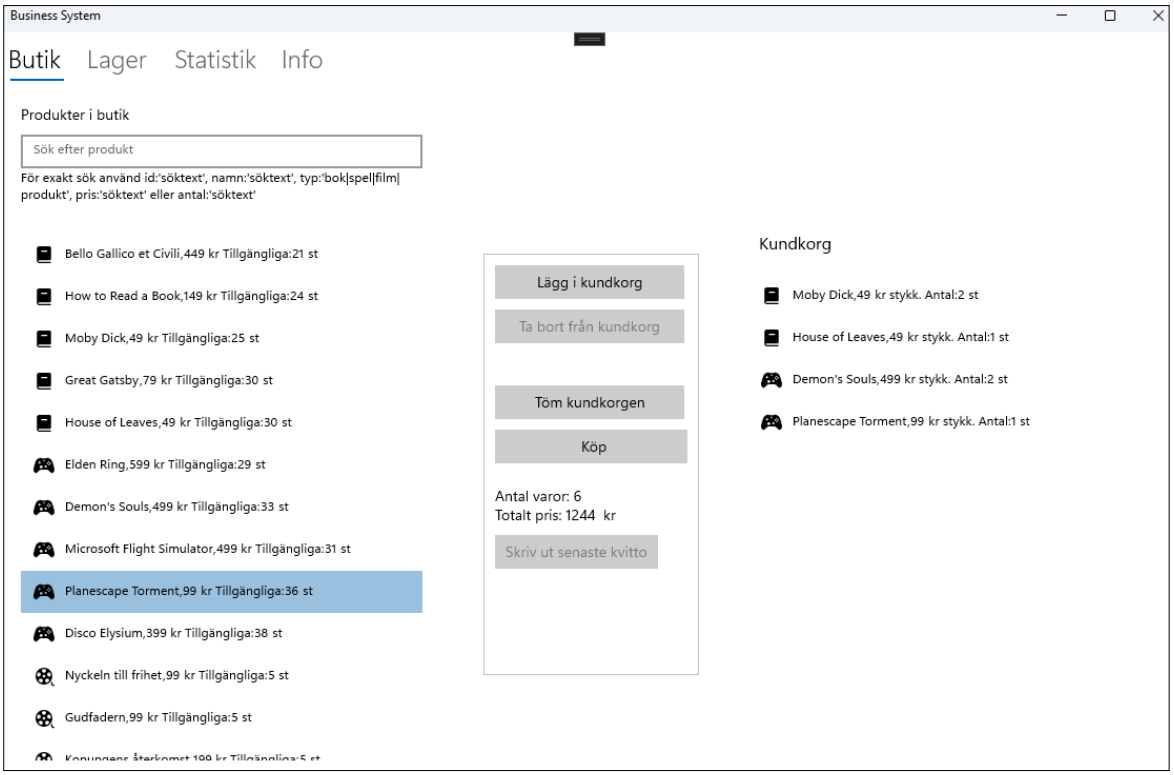
- Har lagt till ikon för mer visuellt se vad det är för typ av produkt

- Produkt id, pris, antal och speltid är inte högre än int datatyp

2. Översikt

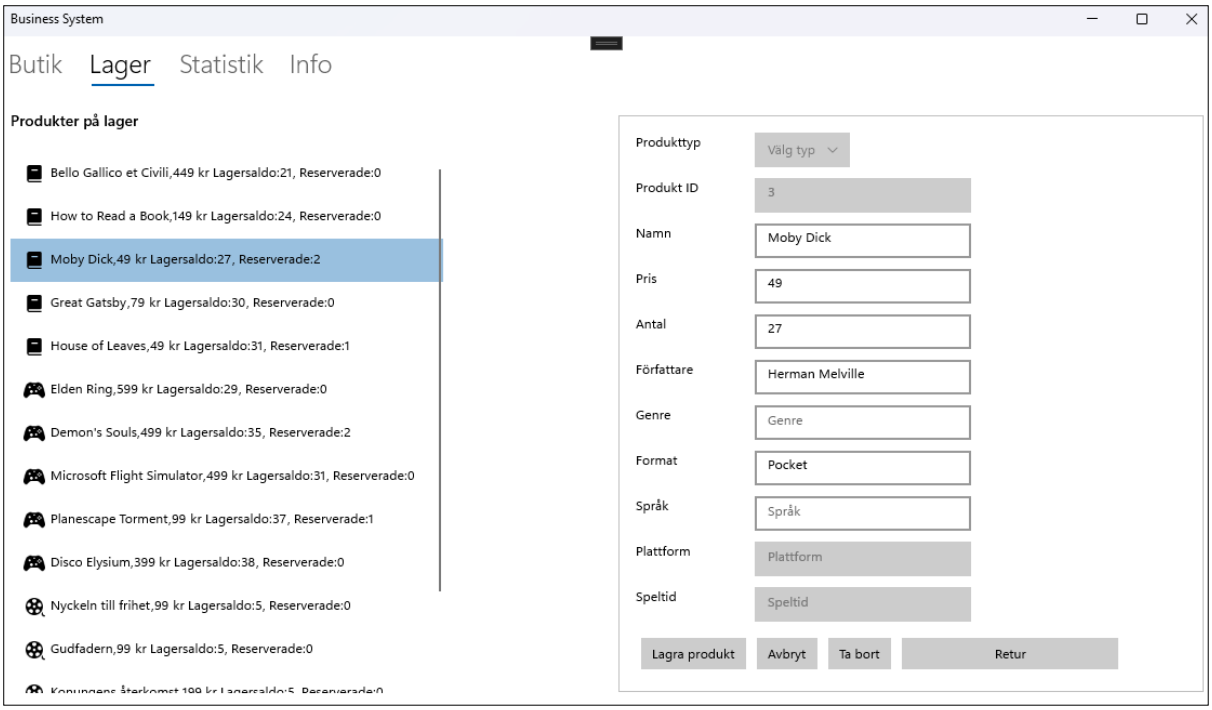
Applikation har 4 olika tabbar. Butik, Lager, Statistik och Info.

I **butiken** kan en säljare söka och välja bland tillgängliga produkter. Man kan utföra fritextsök eller söka på några utvalda egenskaper. En vald produkt kan läggas till (och tas bort) en kundkorg. Säljare kan välja att ta bort 1 och 1 produkt åt gången eller alla på en gång med knappen töm kundkorg. Man får en fråga om man önskar att tömma kundkorgen men när man tar bort en produkt sker det inte en fråga, då uppdateras enbart antal eller så plocka produkten bort från listan om antal är 0. Listorna på produkter i butiken uppdateras med tillgängliga varor respektive antal som kunden valt när man trycker på knapparna. Man kan inte trycka dubbelt på en produkt för att flytta den mellan butiksöversikt och kundkorg, där måste knappar alltid användas. Det visas också en översikt på totalt antal varor i kundkorgen med totalt pris. Om en vara uppdateras på lager så kommer också priset att uppdateras i kundkorg om en kund inte hunnit att köpa den ännu. När säljare har lagt in alla varor som kunden skall ha, avslutat säljare genom att trycka på knappen köp. Man får då en fråga om man är klar att genomföra köpet. Efter man har sålt produkterna har man möjlighet att skriva ut kvitto på de senaste sålda produkterna. Detta kvitto ändras efter varje köp, det lagras alltså inte en lista med kvitton någonstans. För varje köp som genomförs så lagras det information i databas (CSV) om produkterna som sålts. Denna information blir tillgänglig på tabben Statistik. När man avslutar applikationen så lagras nya och uppdateras produkterna i databas (CSV fil).

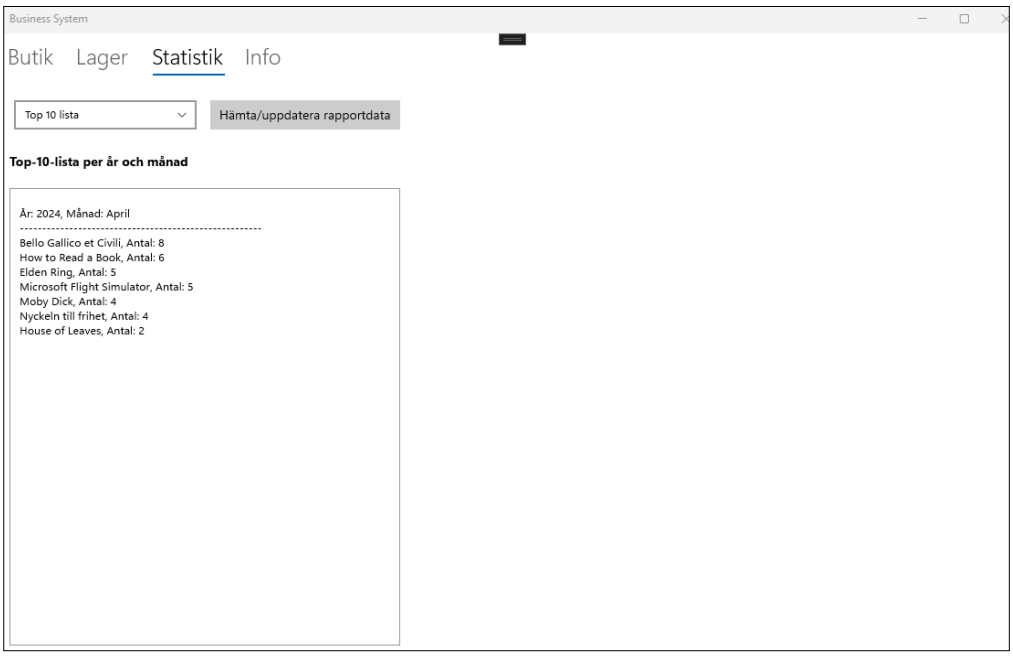


På **Lager** kan man ändra, lägga till och ta bort produkter. Här kan man också göra en retur på en produkt som kommer tillbaka från en kund. Översikten på produkter på lager är vald att inte ha filtrerad på sök, det görs enbart i kassan. För att välja en produkt så trycker man på den i listan. En vald produkt kan då uppdateras på några av de fält som är tillgängliga per typ (har generella produkter, bok, film och spel). En existerande produkt kan också tas bort, där användare får en fråga om man skall ta bort produkten om det finns flera än 0. Eventuella produkter som lags i kassan

försvinner också om man tar bort produkten. Man kan också registrera en retur av varan här, vilket gör att per retur så uppdateras tillgängligheten med 1. Om man har valt en produkt och trycker avbyt så får man en skärmbild där man kan lägga in en ny produkt. Beroende på vilken typ man väljer (produkt, bok, film, spel) så får man tillgång till olika antal fält att fylla i information.



På **Statistik** kan man få information om sålda produkter. Efter eventuella köp så måste man välja att uppdatera data om man redan har valt att se på en rapport. Det finns en rapport för Topp 10 sålda per år och månad, samt en rapport över total försäljning.



På **Info** står det var datafiler är lagrade om man har behov för att göra något manuellt med dessa. Denna tillför enbart information för den som skall testa eller om man får problem med datafilerna

eller vill starta om på ”nytt”. För att rätta upp i datafilerna manuellt måste man ha god kunskap om struktur och lagra dessa på rätt format. Det är viktigt att man inte är inne och ändrar i filerna samtidigt som man använder programmet för att undvika att man ”låser” filerna. Om det sker kommer programmet inte fungera.

3. Beskrivning av applikation och klasser

Kort beskrivning om struktur av applikation

Applikationen använder av 2 st Pages. En för själva applikation (MainPage) och en för att skriva ut kvitto (PrintReceiptPage).

Hela flytet för applikationen sker mer eller mindre inne i MainPage. Denna är en kombination med GUI i xaml för att dela upp den i 4 olika tabbar som är beskrivet i översikten.

Olika event för knappar och tryck på listor sker inn i MainPage.cs. Det finns 3 olika listor för produkter samt 3 olika variabler som håller vald produkt för samma lista. Detta är listor för butik, kundkorg och lager. Det är alltid listan för lager som är gällande och de andra listorna tar den som en utgångspunkt. Beroende på vad som är valt i applikationen och vad som finns tillgänglig för varje produkter så slår man på synlighet eller möjligheten att trycka på en knapp.

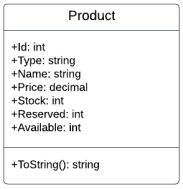
För att minska koden inne i MainPages.cs har det tagits fram hjälpklasser för att flytta ut lite av koden till mer logiska grupper och hantering av kod på en plats. Detta är olika helper, repositories och extensions.

All data som läses/skrives till en databas lagras i en CSV fil på UTF 8 format. Orginalfilen som används första gången när man inte har några filer på ApplicationData.Current.LocalFolder ligger under Repositores/Data/products_initial_data.csv

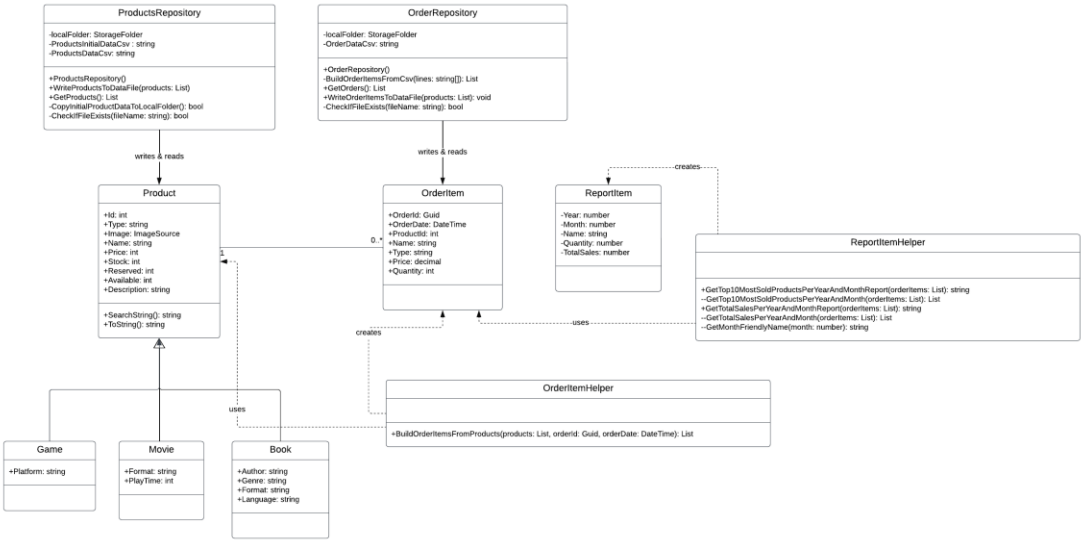
Ordrar skrivs kontinuerligt till datafiler vid köp. Uppdatering av datafiler för produkter sker enbart när man stänger applikationen.

För en mer detaljerade information om klasser och vilka metoder och vad de gör i detalj så kan man se i källkoden. Detta är på engelska i källkoden. Ett alternativt är också att generera dokumentation baserat på källkod för att inte ha duplikat att beskrivning. Denna beskrivning visar det mest översiktliga klasserna med kort beskrivning. I klassdiagrammen nedanför har jag inte tagit med själva Applikationen utan mer fokuserat på klasserna runt om. Har också några enum och extensions metoder som inte är med i klassdiagrammen.

Det första klassdiagrammen som togs fram innan start av kodning såg ut så här:



Det slutliga klassdiagrammen ser ut så här:



Product.cs

Innehåller information som är unik för en produkt. Den innehåller också några metoder/egenskaper som kan överstyras av klassar som ärver från denna. Exempel på detta är bild, beskrivning, sökträng och type.

Game.cs

Ärver från Producs. Innehåller information som är unik för ett spel.

Movie.cs

Ärver från Produc. Innehåller information som är unik för en film.

Book.cs

Ärver från Produc. Innehåller information som är unik för en bok.

ProductsRepository.cs

Används för att läsa och skriva produkter till och från CSV fil. Den ser också till att kopiera startdata för att lagra och sätta upp ApplicationData.Current.LocalFolder.

OrderRepository.cs

Används för att läsa och skriva en order (efter köp i kundkorg) till och från CSV fil.

OrderItemHelper.cs

Används av OrderRepository.cs för att bygga upp en order på rätt format som har hämtats från CSV

OrderItem.cs

Lagrar information om ett köp. Det genereras en GUID och datum som är unik per köp. Det lagras en rad per produkt med antal vi ett köp.

ReportItem.cs

Innehåller information om en rapport som skall visas. Denna byggs upp dynamiskt med information från OrderItems och ReportHelper.cs

ReportHelper.cs

Hjälper till att bygga upp rapporter som skall visas.

4. Problem

Problem som uppstod var att få Observable listor att bli synliga och synkas mellan olika views för produkter. Det löses genom att man måste ha olika listor för de olika views + att man implementerar en INotifyPropertyChanged för att GUI så skall få reda på att något är ändrat.

Utskrift till printer/pdf var väldigt tidkrävande och var tvungen att prova många varianter. Det löses genom att tillpassa ett exempel som hittades på nätet. Tittade också på dokumentation som Microsoft men denna var väldigt omfattande och innehöll mycket som man inte behövde. Plockade fram en egen variant baserad på det infon som fanns tillgänglig.

Rapporter och försök att slå samman dessa var tidskrävande. Provat många varianter med LINQ för att försöka komma fram till önskat resultat. Löses genom att sitta med debug i Visual Studio, göra om IEnumerable resultat till listor för att kunna se på resultat i debugger. Delade upp koden mer för att se vad som lagrats efter gruppering och sortering.

Generellt GUI byggande var väldigt tidkrävande och svårt att veta vilka komponenter som var mest effektiva för att lösa alla uppgifterna. Här kan det jobbas mer för att göra dem flexibla på storlekar av fönster. Löses genom att hittade varianter som vad ”god nog” för en användare. Mer fördel om man får lite upplärning första gången.

5. Sammanfattning

Har lärt mig att jobba mer med GUI och jobba med tankegång att ha information som skall vara synkad på många olika platser. Har också fått med mig att utskrift är ganska svårt i UWP men hittar en variant som fungerar. Många saker i applikationen som sök, rapporter, utskrift av kvitto, översikt på tidigare köp, dubbelklick på produkter skulle kunna ha byggts på ordentligt men har valt att implementera det som krävs för att få en MVP som kan värdera. En applikation kan ju alltid byggas vidare på.

Projektet har tagit längre tid än antagit, mycket på grund av jobb med listor som skall uppdateras samt utskrift som var komplicerad.

Estimerad tidsåtgång:

Kassa: 20-30 timmar
Utskrift: 20-30 timmar (hade trott 1 timme max)
Lager: 20-30 timmar
Statistik: 5 timmar
Info: 1 timme

Dokumentation: 1 dag

6. Referenslista och bilagor

Referenser till källor finns dokumenterat i källkoden.

Bilagor Lab4_Version3.zip