Hovedopgave plan

# Forskellige faser at projektet

**CSV Loading:**

1. Loade salgs filer (slsrpt), lager filer(invrpt) og produkt information (pricat) som csv
   1. Sæt en mappe man ligger filer ind i, og så læser den alle filerne igennem. Hvis der er flere en en, kører den hele processen flere gange.
2. Check for delimiter, data typer, header navne osv.
   1. Header navne er til at verificere hvilken fil vi har.
   2. Data typer er for at sikre at der ikke kommer fejl senere.
   3. Delimiter for at vide hvad vi skal splitte på

**Data håndtering:**

1. gruppere data efter hvert product
   1. da filerne kommer til at indeholde flere forskellige produkter, giver det ikke mening at sortere på størrelse når produkterne er blandet. Så vi ”sortere” først på produktet. Vi kan gøre dette med at bruge den samme sorting algoritme til at sortere på deres EAN eller Stylecolorint osv. Derefter sortere vi størrelser så det kommer i en den rækkefølge.
2. sortere størrelser og tæl hvor mange forskellige der er
   1. output filen kommer til at have andre koloner end input filen har. En af disse kommer til at være hvor mange af en størrelse der er. Man kunne se på om det ville give mening at have et felt der tæller alle af det slags produkt.
3. lav udregninger på hvor mange penge værd der er indlæst (som ekstra check og hurtigere udregning senere)
   1. input filen indeholder et ”pris” felt. Vi ganger den pris med hvor mange produkter der af den slags. Nok af per. Størrelse. Felterne kunne være de samme som GoFact bruger, turnover, grossprofit, costprice, VAT

**Returnering af data:**

1. Giv kunden/GoFact en rapport om hvilken data der er, og eventuelt hvad gik galt, og hvad det var så de kan fixe det og prøve igen.
   1. Rapporten er en CSV eller JSON-fil eller noget lign. Der indeholder resultatet. Det er et summary, som kombinerer samme slags produkter i en, og tæller hvor mange der er af den slags.

# Filstruktur

Programmet har 3 ”faser”, så det giver mening at splitte det op i de faser. Måden jeg vil gøre dette er at lave mapper som indeholde de forskellige klasser som indgår i den fase. Jeg kunne splitte det op i namespaces også, men det bliver ikke stort nok til at det vil blive svært at finde en fil. Og de forskellige faser har ikke så meget med hinanden at gøre, så namespaces funktion til at hjælpe brugeren med flere klasser der hedder det samme kommer ikke til at være et problem.

De 3 faser er; Indlæsning af filer, data håndtering, skrivning til filer. Hvis jeg skriver nogle custom hjælpemetoder som de forskellige faser alle bruger, bruger jeg en 4. mappe som jeg ligger de fælles metoder ind i.

# LoadAllFiles

To load the file from CSV into C#, I’ll just read each row, and add the value to a string list. Then from there I can parse the values and so on. But to get the data type and what it is, I’ll need to have another list that contains the names of the headers. And as long as the order of the data list and the header list is the same, then I can look at the index of the list item, compare it to the name in the list of header names, and use that to infer what data it is.

# ParseStringsToValues

For at oversætte strings til værdierne, skal jeg kigge på navne fra headerne og oversætte dem til den type de er. Jeg kunne genbruge den måde jeg brugte til min CM hos GoFact. En switch case der looper igennem hver string i headervalues, og så tager det index i listen vi er nået til, og bruger den samme index i rows, til at oversætte hvilken værdi det skal være.

# SortingBySize

Dette indeholder hele min smartsort jeg lavede hos GoFact. Jeg skal lige læse det igennem og få det til at virke med mit nye system. Samt omskrive hvad dens input er, og hvordan man kalder den da den før var en extensions metode.

# Common Methods

**CSV\_DBO:**

Hver af disse klasser kommer til at være alt data fra en CSV fil. For at dette virker skal der være en liste af et andet objekt der hedder rows. På Rows objectet skal der være en liste der indeholder værdierne per kolonne.

# TODO:

* Læse smartSort igennem igen for at forstå hvordan den virker igen
  + Ændre på smartSort så at den passer til projektet.
  + Gennemtænk hvad dens output skal være
    - Skal den tælle og sortere? Eller sortere alle først, så tælle dem bagefter?
* Færdiggøre CanonicalModel med de forskellige felter der skal være i den endelige CSV fil struktur.
* I program filen er der en lang kommentar hvad der skal gøres projekt-wide