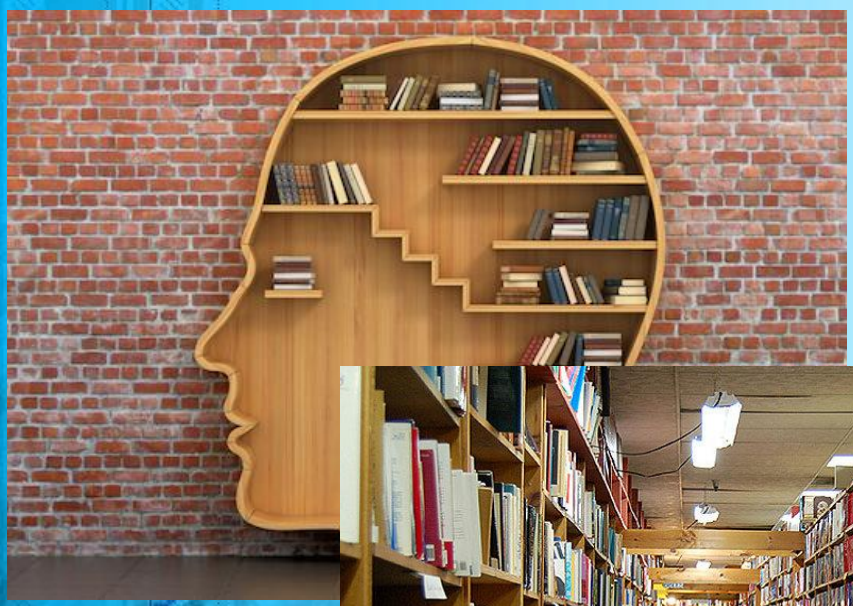
Eventuele aanpassingen functioneel en technisch ontwerp

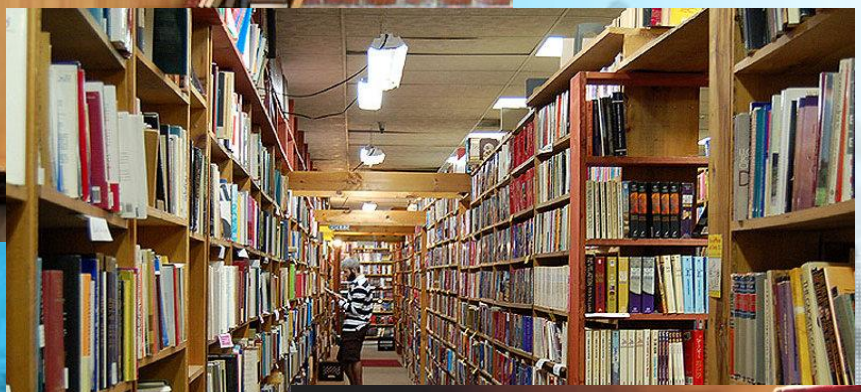
Naam: **Giovanni Koolhoven**

Klas: **Ib3A**

Assessment: **Uitgeverij het boekje**

Periode: **3.4**





INHOUD

FUNCTIONEEL ONTWERP

USECASE

In mijn Use Case diagram miste ik nog een Use Case: “Factuur openen”.

Deze tabel hoort hierbij:

Element	Beschrijving
Naam use case:	Factuur openen
Actor:	Winkelmedewerker, Assistent
Korte beschrijving:	De actor kan voor administratie/als de klant het aangeeft, een factuur openen.
Precondities:	De actor moet ingelogd zijn. De actor moet bestelrechten hebben.
Postcondities:	De actor heeft een factuur geopend.
Stroom van gebeurtenissen:	1. De actor klikt op bestelling waar hij de factuur van wilt. 2. De actor klikt op het factuur icoontje.

De Use Case “bestelling afronden” is niet meer van toepassing, omdat dat automatisch wordt gedaan als het boek op voorraad is. Als het boek niet op voorraad is wordt nu gekozen om de bestelling te wijzigen.

De Use Case “bestelling wijzigen” is daarom ook veranderd. Er kan nu alleen gekozen om de status te wijzigen i.p.v. alle velden. Hier heb ik voor gekozen omdat anders een gedoe zou zijn met eventuele leverdatums en boek prijzen die zouden veranderen (en omdat ik nogal krap met de tijd was). Doormiddel van de status aan te passen kan de bestelling worden afgerond als er eventuele boeken niet op voorraad zouden zijn.

ERD

In mijn Erd zijn 3 tabellen anders

BESTELREGEL: hier is de prijs weggehaald. Eerst wilde ik de prijs in mijn bestelregel bewaren maar dat was niet netjes omdat je dubbele waarden opslaat, en de prijzen van de bestelling staan toch in de factuur.

MEDEWERKER: hier heb ik de eigenschap “zichtbaar” toegevoegd. Net als bij de boeken wordt een medewerker onzichtbaar gezet i.p.v. helemaal verwijderd.

KLANT: hier heb ik de eigenschap “zichtbaar” toegevoegd. Net als bij de boeken wordt een klant onzichtbaar gezet i.p.v. helemaal verwijderd.

WIREFRAMES

In de Wireframes die in mijn functioneel ontwerp zaten heb ik eigenlijk alleen het bestelling aanmaken scherm anders gemaakt in mijn applicatie. Eerst had ik in gedachten dat de medewerker de besteldatum en leverdatum zou kunnen kiezen. Maar dat is onlogisch omdat besteldatum altijd de datum op de dag zelf zou zijn, en leverdatum is dat ook of 3 dagen later.

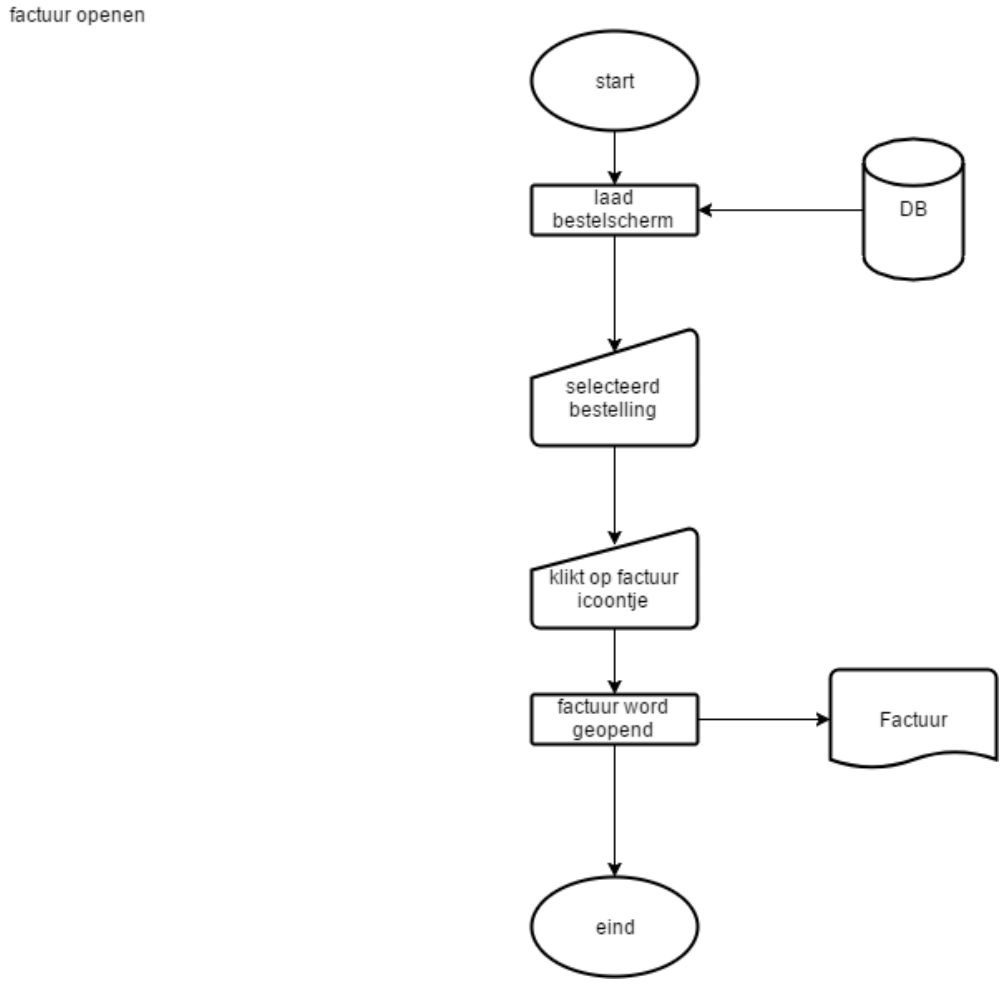
Ik heb dus nu een grote datagrid gepakt waardoor er meer plek is voor de boeken waar je uit kunt kiezen.

SYSTEEMSTROOMDIAGRAM

In het systeemstroomdiagram is de “Bestelling afronden” niet meer van toepassing

En het systeemstroomdiagram “bestelling wijzigen” wijzigt dat alleen de status gewijzigd kan worden doormiddel van het klikken op het status icoontje.

Er is een nieuwe systeemstroomdiagram bijgekomen, factuur openen:



TECHNISCH ONTWERP

FUNCTIONEEL MODEL

In mijn functioneel model zijn dezelfde dingen veranderd als in het Erd.

Hier is bijgekomen dat bij “Boeken” de “beschrijving” varchar(MAX) is geworden. Dit was makkelijker aangezien een beschrijving groter kan zijn dan de eerder aangegeven 200.

Ook had was ik de relatie tussen uitgeversectoren en boeken vergeten te tekenen. Dit is net als in het Erd: een uitgeversector mag een of meerdere boeken bevatten. En een boek moet bij een uitgeversector horen.

UITGEWERKTE SCHERMEN

In het boekscherm, had ik een fout gemaakt dat de linker tabel niet de medewerkers maar de uitgeversectoren moest bevatten. In mijn applicatie is dit wel netjes verwerkt.

In het bestelscherm is hetzelfde als bij de Wireframe. Het bestelling aanmaken scherm is aangepast.

CLASS DIAGRAMMEN

In de **entity classes** zijn alleen sommige classes iets aangepast doordat de ERD is veranderd. Zie ERD

In de **overige classes** is sectorDb niet meer van toepassing.

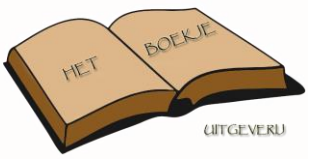
In de boekDb staan nu 2 extra methodes (overzichtSector(string naam) en overzichtSectoren(string data)) om een sector en om meerdere sectoren op te halen. Ook staat er nog een methode overzichtBestelling() in boekDb. Hier krijg je een overzicht van alle boeken in een bestelling.

In bestellingDb zijn 2 methodes bijgekomen: overzichtKlant(int id) en GetLaatsteBestelling(). overzichtKlant(int id (klant id)) om een overzicht te krijgen van alles bestellingen van een klant. En GetLaatsteBestelling() om de laatste bestelling uit de DB te halen.

In bestellingregelDb is een methode bijgekomen: overzichtBestelling(Bestellingregel data). Hierdoor krijg je een overzicht van alle bestelregels van een bestelling.

In klantDb zijn 2 methodes bijgekomen: overzichtBestelling(int id) en GetLaatsteKlant(). overzichtBestelling(int id (bestelId) om de klant te krijgen die bij de bestelling hoort. GetLaatsteKlant() om de laatste klant uit de DB te halen.

In LoggingDb is een methode bijgekomen: overzichtMedewerker(int id). overzichtMedewerker(int id (medewerkerId)) haalt alle loggings op van een medewerker.



In MedewerkerDb is een methode bijgekomen: overzichtLogging(int id). overzichtLogging(int id (loggingId)) geef de medewerker terug die bij de logging hoor.

Tenslotte is er in UitgeverDb ook een extra methode toegevoegd: overzichtUitgever(string isbn_nummer). Deze geeft de uitgever van een boek terug.

Deze methodes zijn vooral gemaakt dat als er geselecteerd wordt op de andere tabel er ook gefilterd word op de tabel ernaast.

Voorbeeld: als je klikt op een klant in de linker tabel zie je in de rechter tabel alle bestellingen die hoort bij die klant. Ik kwam er later achter dat dit erg handig was en er goed uitzag als ik dit zo zal filteren, daarom zijn deze methodes later gekomen.

Tevens had ik nog een kopieer fout gemaakt bij de overige classes. Elke methode staat daar op private, maar dat is public geworden aangezien t anders moeilijk is om deze methodes überhaupt aan te spreken.

Het leek me makkelijk om een class te hebben die verschillende eigenschappen valideert. Daarom heb ik een class toegevoegd met custom validaties.

Deze class hoort daarbij: