ontwerpdocument

Live performance

Michelle Broens | S21M | 07/06/2017

Inhoudsopgave

[Inleiding 2](#_Toc484615179)

[Begrippen 3](#_Toc484615180)

[Architectuur 4](#_Toc484615181)

[Klassendiagram 5](#_Toc484615182)

[Klassendiagram bijlage 6](#_Toc484615183)

[Database ontwerp 7](#_Toc484615184)

[Het ontwerp 7](#_Toc484615185)

[Bijlage bij het database ontwerp 8](#_Toc484615186)

[Tabellen 9](#_Toc484615187)

[Categorie 9](#_Toc484615188)

[Prodcut 9](#_Toc484615189)

# Inleiding

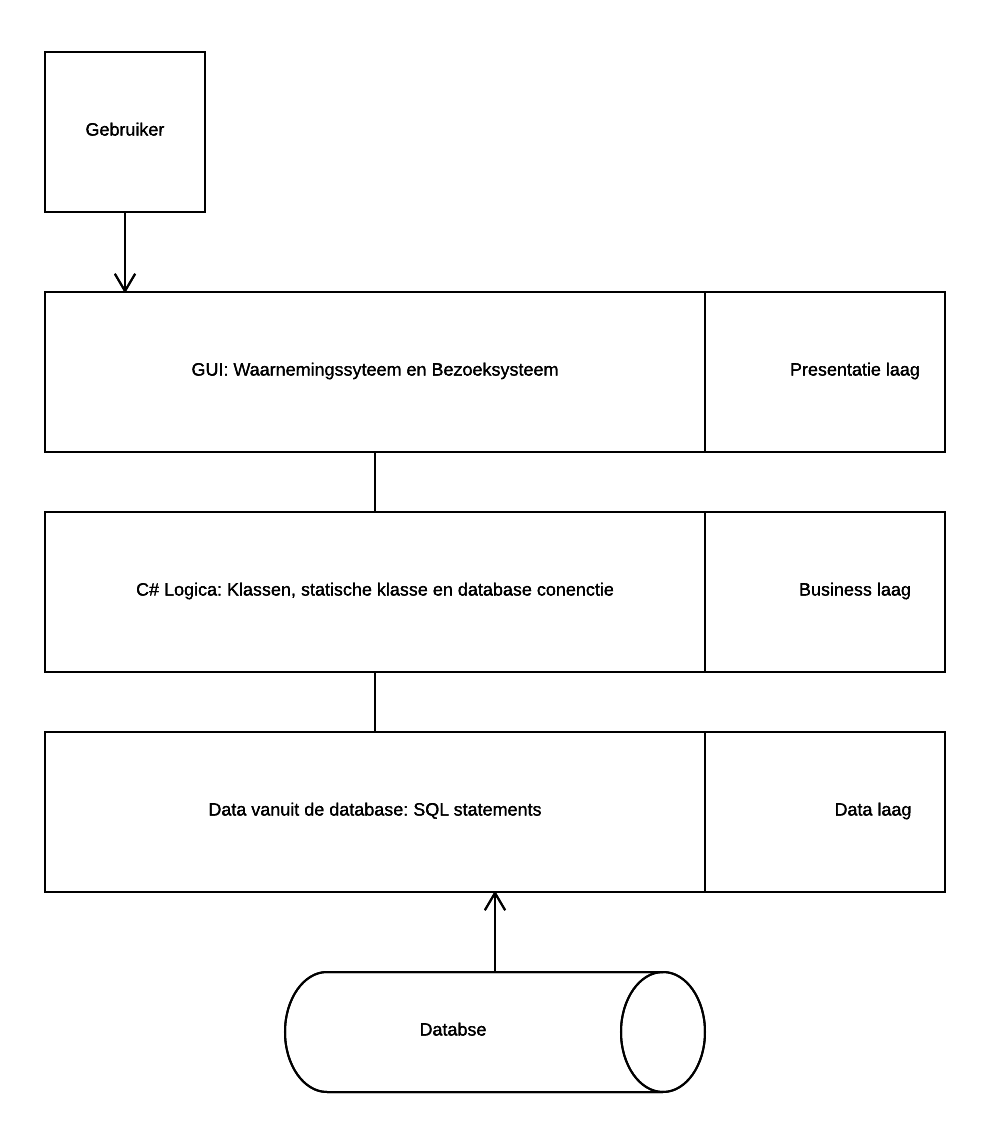
In dit document wordt er dieper ingegaan op de applicatie die in het document beschreven staat. Hier komen de meer gecompliceerde uitwerkingen die bijna meteen geïmplementeerd kunnen worden. Deze onderdelen zijn:

* Begrippen: dit is een lijst met begrippen hier staan woorden die de lezer misschien niet begrijpt of misschien niet helemaal begrijpt.
* Architectuur: dit geeft de structuur en communicatie van de applicatie weer in de vorm van een diagram
* Database ontwerp: hierin komt een diagram van de tabellen en attributen die in de database gezet worden, samen met een bijlage van dit diagram.
* Klassendiagram: dit geeft aan welke klassen, properties en methodes er in de applicatie gebruikt gaan worden.
* Infrastructuur: hier wordt in een diagram beschreven hoe de infrastructuur in elkaar zit, dus de server waar eventueel de database opstaat, de clients waarop de applicatie zou kunnen draaien.

# Begrippen

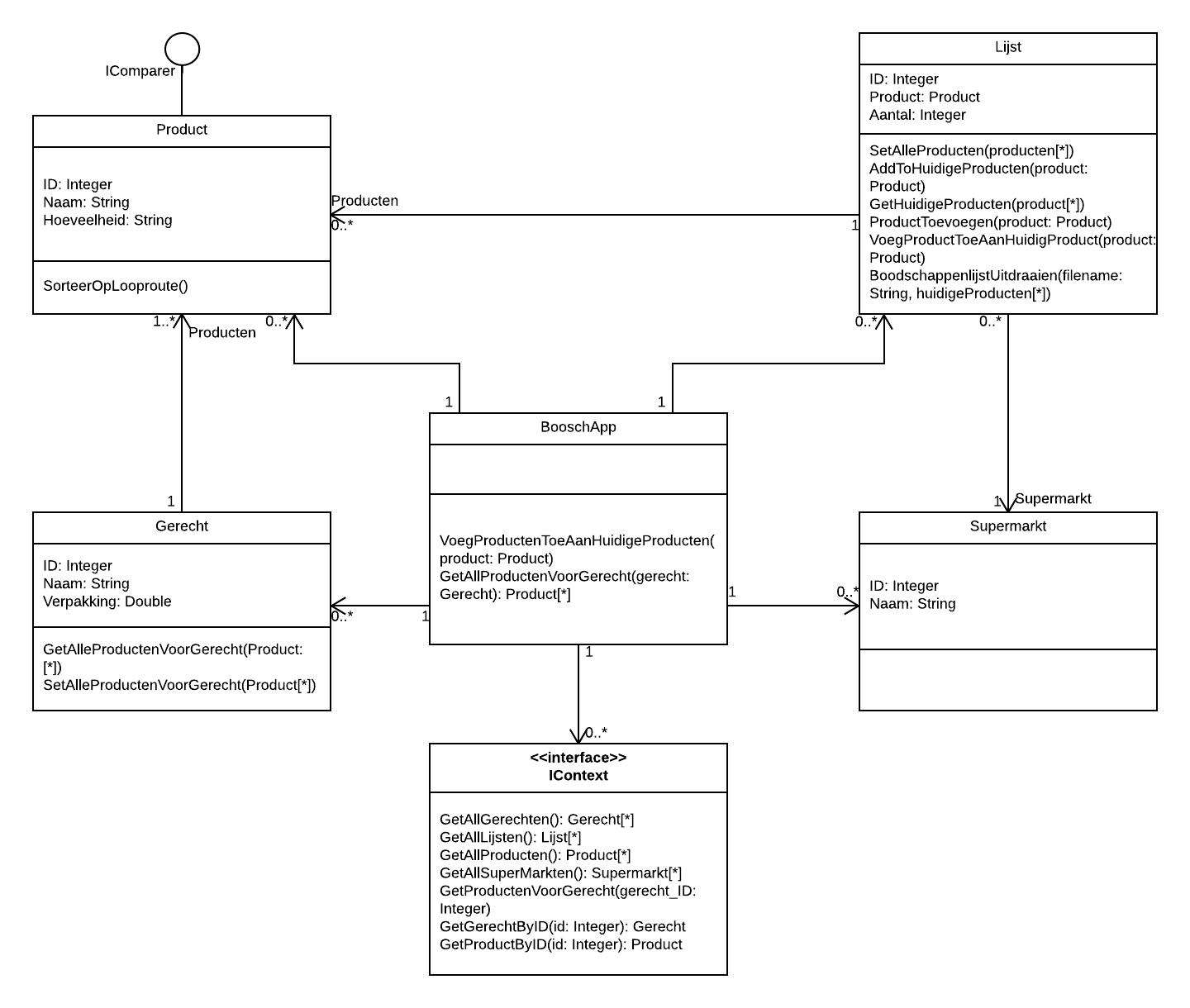
|  |  |
| --- | --- |
| **Begrip** | **Uitleg** |
| Property | In een property worden waardes opgeslagen om een privéveld buiten de klasse te kunnen gebruiken. |
| Methode | Een handeling die meerdere keren aangeroepen kan worden. |
| Infrastructuur | Een verzameling van voorzieningen die nodig is voor het transport van data. |
| Clients | Een client is een werkstation dus een pc of een laptop, die verbonden zit met de rest van de infrastructuur. |
| Query | Een query is een verzoek voor data/informatie van een tabel (of meerdere tabellen) aan de database. |
|  |  |
|  |  |

# Architectuur

De gebruiker logt in op het inlogscherm (presentatie laag) de gegevens worden doorgestuurd naar de business laag. Deze gegevens worden door middel van een query (data laag) naar de database gestuurd.

# Klassendiagram

Hier komen alle klassen die geïmplementeerd worden in de applicatie met onder het diagram nog een bijlage met een verdere uitleg.



## Klassendiagram bijlage

De klasse **BoodschApp** is de klasse die in de UI laag de andere klassen aanroept, deze roept daar de methodes aan die nodig zijn. Deze klasse implementeert ook de **MSSQLContext** klasse, deze klasse bevat de queries die naar de database geschreven worden, de methodes in deze klasse worden afgehandeld door middel van de **IContext** interface.

De klasse **Lijst** implementeert de **IComparer** interface. Deze interface zorgt ervoor dat de **Lijst** klasse gesorteerd kan worden. Het sorteren zal gebeuren op **Loopvolgorde** van de **Supermarkt**, ik heb ervoor gekozen dat dit altijd de alfabetische volgorde is.

De implementatie van de attributen worden allemaal private setters tenzij dit anders aangegeven is.

Verder ga ik nog gebruik maken van een MSSQL-database de interface **IContext** praat met de data laag.

# Database ontwerp

## https://documents.lucidchart.com/documents/9b84c53e-30f7-48d8-9196-3a402a0c2c47/pages/0_0?a=905&x=-3&y=-15&w=779&h=1025&store=1&accept=image%2F*&auth=LCA%20d3685d169a12c13e48020bf9f56855ec67faa71d-ts%3D1496755675Het ontwerp

## Bijlage bij het database ontwerp

De tabel **Categorie** kan nog een **Categorie** hebben, dit is dan de subcategorie van de categorie. De oplossing die ik geïmplementeerd heb bij het oplossen van de recursie is het toevoegen van een ware “parent”, de subcategorie heeft dus een categorie van een level hoger wat zijn “parent” is.  
Een **Categorie** kan bij nul of bij meerdere **Producten** horen, maar een **Product** kan maar een **Categorie** hebben. Ik heb hiervoor gekozen omdat anders je hetzelfde product steeds opnieuw te zien krijgt.

De tabel **Product** en **Gerecht** zijn gekoppeld door een koppeltabel. Ik heb hiervoor gekozen omdat je zo gemakkelijk kan zien welk gerecht welke producten nodig heeft. Het veld **Verpakking** in de koppeltabel is het percentage (in kommagetallen) voor hoe veel van één product gebruikt wordt in een gerecht.

Het veld **Hoeveelheid** in de tabel **Product** wil zeggen hoeveel liter/gram/stuks/porties het **Product** is. Een **Product** staat één keer op een **Lijst**, deze **Lijst** is een regel waar het **Product** op staat met het **Aantal** keer dat het **Product** gekocht wordt.

Een **Lijst** hoort bij één **Supermarkt**. Als er boodschappen gedaan worden moet dit allemaal bij één **Supermarkt** gedaan worden.

Ik heb ervoor gekozen dat de **Looproute** in de winkels op alfabetische volgorde is.

## Tabellen

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribuut** | **Type** | **Beschrijving** | **Verplicht** |
| ID | Integer |  | Ja |
| HoofdCategorie\_ID | Integer | Dit is het ID van de categorie die hier een niveau boven staat, bijvoorbeeld een biefstuk is rundvlees maar boven rundvlees zit nog de categorie vleeswaar | Ja |
| Naam | Varchar(20) |  | Ja |

### Categorie

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribuut** | **Type** | **Beschrijving** | **Verplicht** |
| ID | Integer |  | Ja |
| Naam | Varchar(20) |  | Ja |
| Hoeveelheid | Varchar(50) | Dit geeft aan wat de hoeveelheid is van een product, bijvoorbeeld 400 gram | Ja |

### Prodcut