

Analysedocument

Live performance

Michelle Broens | S21M | 07/06/2017

Inhoudsopgave

[Inleiding 2](#_Toc484612860)

[Begrippen 3](#_Toc484612861)

[Opdrachtgever 4](#_Toc484612862)

[Scenario 5](#_Toc484612863)

[Functionele eisen 6](#_Toc484612864)

[Niet-functionele eisen 9](#_Toc484612865)

[Use case 10](#_Toc484612866)

[Use case diagram 10](#_Toc484612867)

[Use case beschrijvingen 11](#_Toc484612868)

[User interface 13](#_Toc484612869)

[UI 13](#_Toc484612870)

[EER model 14](#_Toc484612871)

[EER bijlage 15](#_Toc484612872)

[Productdecompositie 16](#_Toc484612873)

# Inleiding

In dit document komen er een aantal verschillende onderwerpen aanbod die te maken hebben met de analysefasen van het maken van een applicatie over … De onderwerpen die aan bod komen zijn:

* Een lijst met begrippen: in deze lijst staan woorden die de lezer misschien niet begrijpt of misschien niet helemaal begrijpt.
* Opdrachtgever: in dit stuk kom een beschrijving over wie de opdrachtgever is en wat de opdrachtgever algemeen verwacht.
* Het scenario: een korte beschrijving over een persoon die door de ontwikkelde applicatie loopt.
* De functionele eisen: de eisen die het programma moet kunnen, bijvoorbeeld een gebruiker kan inloggen.
* De niet-functionele eisen: de eisen die niet te maken hebben met het programmeren van code zoals de functionele eisen, bijvoorbeeld de code moet herbruikbaar zijn.
* De use cases: de functionele eisen worden in een diagram gezet zodat het duidelijk wordt door wie deze functie uitgevoerd wordt. Deze functies worden in de use case beschrijving van start tot eind doorgelopen.
* De userinterface: hier wordt een schets gemaakt van hoe de applicatie er ongeveer uit gaat zien.
* Het EER model: een schets van hoe de structuur van het programma er in de database uit gaat zien.
* De productdecompositie: een schets van de verschillende onderdelen die opgeleverd worden.
* Planning: de planning van hoe dit project uitgewerkt wordt. Per onderdeel staat een tijd van wanneer er een onderdeel begonnen wordt en wanneer deze af is.

# Begrippen

|  |  |
| --- | --- |
| **Begrip** | **Uitleg** |
| Use case | In software engineering is een use case een lijst van acties die een persoon (actor) en het systeem uitvoert om een doel te bereiken, bijvoorbeeld het inloggen. |
| Userinterface | Hier praten de gebruiker en het systeem met elkaar. |
| Database | Een georganiseerde collectie van data in tabellen. |
| AFK | Een afkorting voor het woord afkorting. |
| MoSCoW | Een methode om aan te geven hoe belangrijk een requirement is, met de M voor must(moet), S voor shoulds(zou moeten), C voor could(zou kunnen) en W voor will not(wordt niet geïmplmenteerd). |
| BR | Boodschappen requirement |
|  |  |

# Opdrachtgever

De opdrachtgever van dit project is de stichting Voedselafval Reductie Eindhoven, VRE. De VRE is bezig met een project waarmee ze zelfstandig wonende studenten willen aanspreken, het gaat om de ontwikkeling van de applicatie BoodschApp.

Uit een enquête is gebleken dat veel studenten geen geld hebben om vaak buitenshuis te eten, maar ze willen wel graag vers en afwisselend eten. Het merendeel van de studenten in Eindhoven slaan op de maandag boodschappen in voor maandag tot en met donderdag omdat veel studenten op vrijdag teruggaan naar het ouderlijk huis. Veel studenten gaan naar een van drie supermarkten, welke supermarkt gekozen wordt hangt af van het budget van een bepaalde student. Het merendeel van de studenten vindt dat ze te lang bezig zijn met boodschappen doen dan nodig is omdat ze lang bezig zijn met het zoeken naar bepaalde producten, dit komt omdat in iedere supermarkt de positie van de producten anders is.

De applicatie die de VRE wil ontwikkelen is bedoeld om deze studenten het leven wat makkelijker te maken door ze te helpen met het inslaan van de juiste hoeveelheid boodschappen. Dit bespaart tijd, geld en vermindert het voedselafval.

# Scenario

De gebruiker opent de BooschApp. Daar kiest hij ervoor om op de knop “Nieuw boodschappenlijstje” te klikken. Hier wordt hij het “Lijst scherm” getoond.

Hij heeft nu een paar opties en hij kiest ervoor om op de knop “Product toevoegen” te klikken. Nu bevindt hij zich op het “Producten scherm”. Hier kan hij een aantal producten aanklikken en op de knop “Product toevoegen” klikken, dan worden de producten toegevoegd aan zijn boodschappenlijstje. Nu kiest hij ervoor om op de knop “Terug” te klikken. Hij bevindt zich nu weer op het “Lijst scherm”.

Hij heeft nog twee opties die hij niet eerder uitgevoerd heeft en kiest ervoor om op de knop “Gerecht toevoegen” te klikken. Hij wordt naar het “Gerechten scherm” gestuurd en ziet daar een lijst van alle standaard gerechten die op dat moment aanwezig zijn. Hij kiest ervoor om macaroni toe te voegen aan zijn boodschappenlijstje. Op zijn boodschappenlijstje komen dan de producten te staan die hij nodig heeft om macaroni te maken.

Hij is tevreden mijn zijn aankopen en wil graag een uitdraai van zijn boodschappenlijstje. Hij klikt nu op de knop “Bestelling afronden”. Hij wordt nu naar het “Bestel scherm” gestuurd. Hier ziet hij nog een keer zijn huidige selectie aan producten en gerechten. Als hij hier tevreden mee is klikt hij op de knop “Bestellen”, nu krijgt hij een uitdraai van zijn boodschappenlijstje gesorteerd op de loopvolgorde van de winkel.

Als hij nu nog een keer de applicatie opent zie hij dat de het veld onder “Vaker bestelde producten” nu inhoud heeft. Hij kan hierop klikken en wordt dan naar het “Bekende producten scherm” gestuurd. Hij kan hier de gewenste producten opnieuw aan zijn boodschappenlijstje toevoegen en door de stappen die eerder gedaan zijn lopen..

# Functionele eisen

Dit hoofdstuk gaat over de functionele requirements die geïmplementeerd gaan worden in de applicatie. Deze worden beschreven met een korte beschrijving, aangeduid hoe belangrijk deze requirement is met de MoSCoW analyse en een eventuele opmerking.

|  |  |  |  |
| --- | --- | --- | --- |
| **AFK** | **Requirement** | **MoSCoW** | **Opmerking** |
| BR1 | Bekende producten moeten aan het boodschappenlijstje kunnen worden toegevoegd | M | Dus producten die al vaker besteld zijn |
| BR2 | Voor een systeem bekend gerecht moeten alle bijbehorende producten aan een boodschappenlijst kunnen worden toegevoegd | M |  |
| BR3 | Het sorteren van het boodschappenlijstje gebeurt op loopvolgorde van de geselecteerde winkel | M |  |
| BR4 | Er moet een lijst van gerechten kunnen worden opgevraagd waar de combinatie voor weinig restjes zorgt | M |  |
| BR5 | Het boodschappenlijstje moet geëxporteerd kunnen worden naar een tekstbestand op loopvolgorde voor een geselecteerde winkel | M |  |
| BR6 | Nieuwe gerechten toevoegen, aanpassen en/of verwijderen vanuit de software. | S |  |
| BR7 | Aangeven dat alle producten op het boodschappenlijstje zijn gekocht en daarmee de producten van het boodschappenlijstje verwijderen (je mag ervan uit gaan dat alle producten bij alle supermarkten voldoende op voorraad zijn). | S |  |
| BR8 | Het opvragen van gerechten welke gecombineerd voor weinig restjes zorgen mag niet langer dan 2 seconden duren (ongeacht hoeveel gerechten er in de database bekend zijn). Na 2 seconden wordt het combineren beëindigd en worden de tot dan toe gevonden combinaties getoond. | S |  |
| BR9 | Ondersteuning in het database-ontwerp voor de houdbaarheid van producten voor een toekomstige versie van de software. | S |  |
| BR10 | Producten toevoegen, aanpassen en/of verwijderen vanuit de applicatie (direct in de database toevoegen is voor de eerste versie eigenlijk al goed genoeg) | C |  |
| BR11 | Winkels toevoegen, aanpassen en/of verwijderen (direct in de database toevoegen is voor de eerste versie eigenlijk al goed genoeg). | C |  |
| BR12 | Naast het aangeven dat alle producten op het boodschappenlijstje zijn gekocht ook de mogelijkheid om aan te geven dat een deel van de boodschappen is gekocht en van het boodschappenlijstje kan worden verwijderd. | C |  |
| BR13 | Van de lijst van gerechten welke gecombineerd voor weinig restjes zorgen worden alleen de 25 meest zuinige combinaties getoond | C |  |
| BR14 | Het zou mooi zijn als alvast per product de houdbaarheid kan worden aangegeven. Bij het combineren van gerechten hoeft dan alleen gekeken te worden naar de niet lang houdbare producten (de lang houdbare producten hoeven namelijk niet weg te worden gegooid). | C |  |
| BR15 | De product-categorie-mappen structuur wordt alleen in de database verwerkt en dus niet in de applicatie (code). | W |  |
| BR16 | Een historie van gekochte boodschappen hoeft niet te worden opgeslagen. | W |  |
| BR17 | Het is niet nodig de locatie op de looproute van producten in een winkel vanuit de software aan te kunnen passen (direct in de database aanpassen is voor de eerste versie goed genoeg). | W |  |
| BR18 | Gerechten zijn altijd eenpersoonsmaaltijden, er hoeft geen rekening gehouden te worden met gasten die komen eten. | W |  |

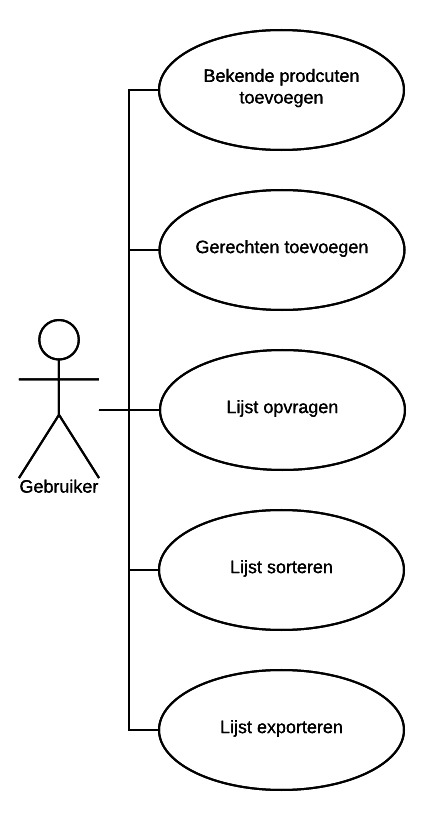
# Niet-functionele eisen

|  |  |  |  |
| --- | --- | --- | --- |
| **AFK** | **Requirement** | **MoSCoW** | **Opmerking** |
| NF1 | De applicatie bestaat maar voor één gebruiker, elke gebruiker heeft een aparte database | M |  |
| NF2 | De database moet ondersteuning bieden voor een (sub)categorieën structuur waar de producten in de toekomst geplaatst kunnen worden | M |  |

# Use case

In dit hoofdstuk komt het use case diagram en de beschrijvingen die bij het diagram horen.

## Use case diagram



## Use case beschrijvingen

|  |  |
| --- | --- |
| **Naam** | Bekende producten toevoegen |
| **Samenvatting** | De actor voegt producten/gerechten die vaker besteld zijn toe aan het boodschappenlijstje |
| **Actor** | De gebruiker |
| **Aanname** | Er zijn vaker al producten besteld |
| **Beschrijvingen** | 1. De actor start de applicatie en gaat naar het “Bekende producten scherm” 2. Het systeem kijkt in de database of de gebruiker al eerder een boodschappenlijstje geëxporteerd heeft 3. Het systeem laadt deze producten in op het bekende producten scherm |
| **Uitzondering** | N.v.t. |
| **Resultaat** | De bekende producten zijn aan het “Bekende producten scherm” toegevoegd |

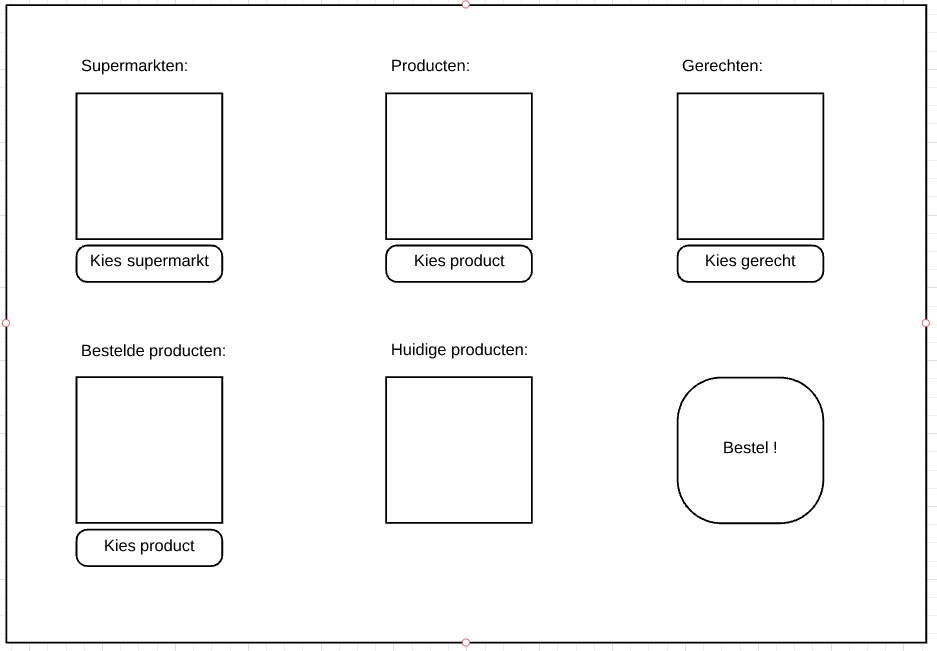
|  |  |
| --- | --- |
| **Naam** | Gerechten toevoegen |
| **Samenvatting** | Er wordt een gerecht gemaakt van de aanwezige producten |
| **Actor** | De gebruiker |
| **Aanname** | Er zijn een aantal producten toegevoegd aan het boodschappenlijstje |
| **Beschrijvingen** | 1. De actor bevindt zich op het “Lijst scherm” en klikt op de knop “Gerecht toevoegen” 2. Het systeem laat het “Gerechten scherm zien” met een lijst van alle standaard gerechten 3. De gebruiker klikt op een van deze gerechten en dan op de knop “Gerecht toevoegen” 4. Het systeem voegt dan alle producten toe aan het boodschappenlijstje |
| **Uitzondering** | N.v.t. |
| **Resultaat** | De actor heeft een gerecht aan de lijst toegevoegd |

|  |  |
| --- | --- |
| **Naam** | Lijst sorteren |
| **Samenvatting** | De boodschappenlijst wordt gesorteerd |
| **Actor** | De gebruiker |
| **Aanname** | Er zijn een aantal producten aanwezig die gesorteerd kunnen worden |
| **Beschrijvingen** | 1. De actor gaat naar het “Bestel scherm” 2. Het systeem gaat na bij welke winkel de gebruiker zich bevindt en sorteert op de bijbehorende looproute van de winkel |
| **Uitzondering** | N.v.t. |
| **Resultaat** | Het boodschappenlijstje van de gebruiker is gesorteerd |

# User interface

In dit hoofdstuk komen een paar designs voor hoe de applicatie er van de buitenkant uit komt te zien.

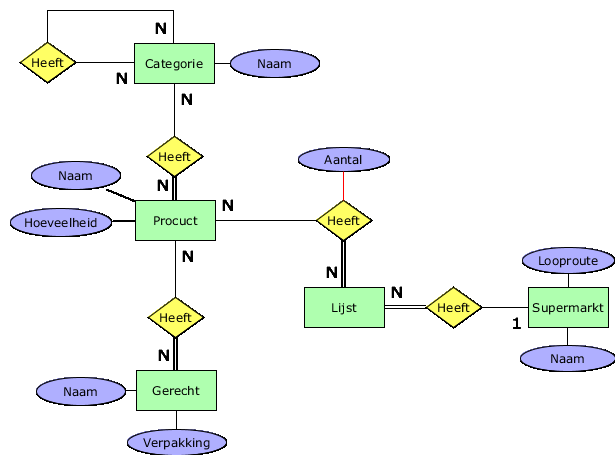
## UI



.

# EER model

In dit hoofdstuk komt het EER model samen met een bijlage om het model verder uit te leggen.



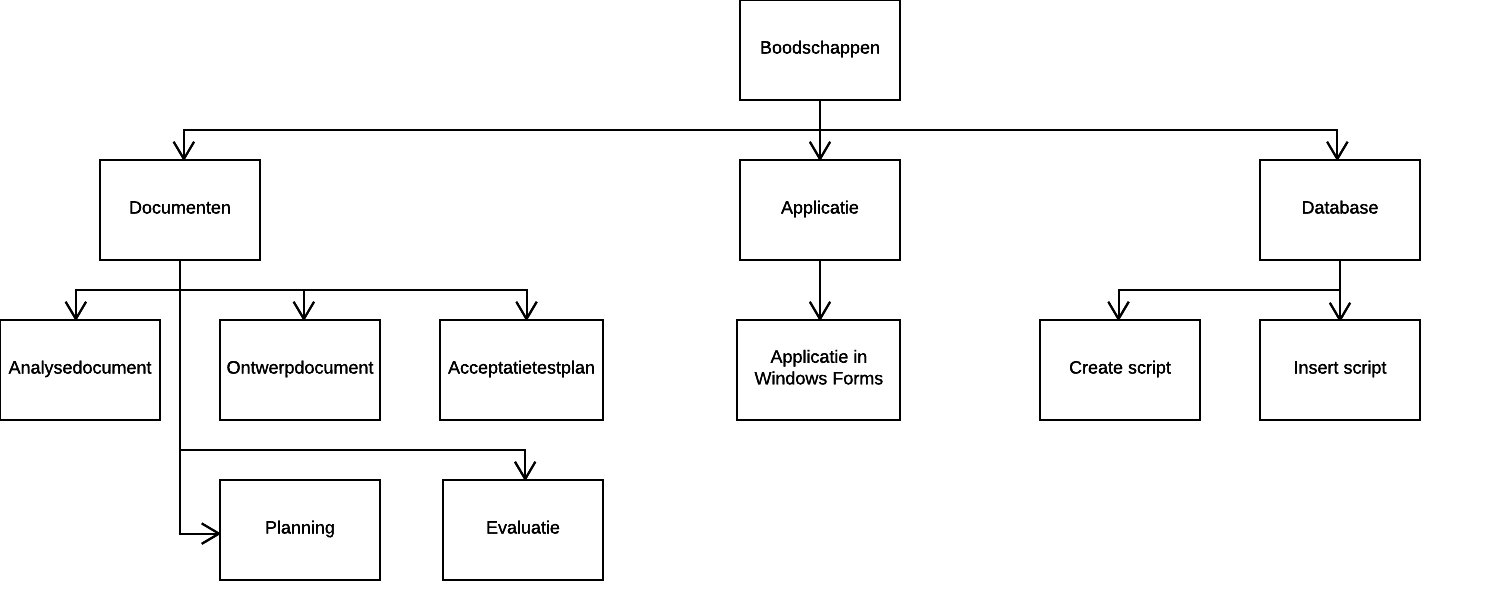
## EER bijlage

Een **Product** heeft een **Naam** en **Hoeveelheid** (125 ml, 400 gram etc.). Een **Product** heeft een **Categorie**, dus zuivel, vleeswaren en een **Categorie** kan ook een subcategorie hebben zo is rundvlees een subcategorie van vleeswaren. Een **Product** heeft verplicht een of meer categorieën omdat een product sowieso één categorie heeft, als deze onder een subcategorie valt zijn dit er meerdere.

Een **Product** kan een **Gerecht** hebben, een **Gerecht** is een verzameling van een aantal **Producten**. Een **Gerecht** heeft een **Naam** bijvoorbeeld macaroni. Een **Gerecht** heeft ook een **Verpakking**, dit is het percentage van hoe vaak een product gebruikt kan worden, stel voor kip heeft een percentage van 50%, wil dat zeggen dat een schaal kip twee keer gebruikt kan worden in een gerecht, dus als er twee gerechten zijn met kip 50%, kunnen er twee gerechten gemaakt worden van de ene schaal kip.

De **Lijst** entiteit is letterlijk het boodschappenlijstje, het **Aantal** dat bij de relatie staat is het aantal keer dat een **Product** op het lijstje staat. **Lijst** heeft ook één **Supermarkt** waar de boodschappen gedaan gaan worden.

# Productdecompositie

Dit hoofdstuk geeft je een beeld in welke delen het project opgedeeld wordt, bijvoorbeeld wat er allemaal opgeleverd moet worden.