Thijs de Rooij & Taoufik Sadaane

Brabotica 2.0 - Versie 0.9

Proftaakdocument

AVD Back-end development en databases

Inhoudsopgave

[Versiehistorie 2](#_Toc67578466)

[1 Algemene beschrijving 3](#_Toc67578467)

[2 Aanpak 4](#_Toc67578468)

[2.1 Planning 4](#_Toc67578469)

[2.2 Afspraken 5](#_Toc67578470)

[2.3 Eisen en wensen 5](#_Toc67578471)

[3 Functionele benodigdheden 7](#_Toc67578472)

[3.1 Algemeen 7](#_Toc67578473)

[3.2 Gebruikerssysteem 7](#_Toc67578474)

[3.3 Ordersysteem 8](#_Toc67578475)

[3.4 Productensysteem 9](#_Toc67578476)

[4 Entity Relationship Diagram (ERD) 10](#_Toc67578477)

[4.1 Toelichting ERD 11](#_Toc67578478)

[5 Relationele Database 12](#_Toc67578479)

[6 Database realisatie (DDL) 13](#_Toc67578480)

[6.1 Structuur tabel Address 13](#_Toc67578481)

[6.2 Structuur tabel Category 14](#_Toc67578482)

[6.3 Structuur tabel Discount 14](#_Toc67578483)

[6.4 Structuur tabel Order 15](#_Toc67578484)

[6.5 Structuur table Product 15](#_Toc67578485)

[6.6 Structuur tabel ProductCategoryJunction 16](#_Toc67578486)

[6.7 Structuur tabel ProductOrderJunction 16](#_Toc67578487)

[6.8 Structuur tabel Role 17](#_Toc67578488)

[6.9 Structuur tabel User 17](#_Toc67578489)

[7 Beveliging (DCL) 18](#_Toc67578490)

[8 Onderhoud 19](#_Toc67578491)

[9 Installatie 20](#_Toc67578492)

[10 Uitwerkingen (DML) 21](#_Toc67578493)

[10.1 Gebruikerssysteem 21](#_Toc67578494)

[10.2 Ordersysteem 21](#_Toc67578495)

[10.3 Productensysteem 21](#_Toc67578496)

[Bijlage 1: Compleet overzicht van de DDL 22](#_Toc67578497)

# Versiehistorie

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Datum** | **Auteur** | **Aanpassingen** |
| 0.1 | 14-02-2021 | Thijs de Rooij | Opzet van het initiële document |
| 0.2 | 22-02-2021 | Thijs de Rooij | Hoofdstuk 2 aangevuld en hoofdstuk 3 toegevoegd |
| 0.3 | 23-02-2021 | Thijs de Rooij & Taoufik Saadane | Hoofdstuk 3 aangevuld en hoofdstuk 4 toegevoegd |
| 0.4 | 01-03-2021 | Thijs de Rooij | Hoofdstuk 6 toegevoegd |
| 0.5 | 08-03-2021 | Thijs de Rooij | Hoofdstuk 6 aangepast |
| 0.6 | 11-03-2021 | Thijs de Rooij | Hoofdstuk 4 en 6 aangepast |
| 0.7 | 13-03-2021 | Thijs de Rooij & Taoufik Saadane | Hoofdstuk 4, 5 en 6 aangepast |
| 0.8 | 15-03-2021 | Thijs de Rooij | Bijlage 1 toegevoegd en hoofdstuk 6 aangepast |
| 0.9 | 25-03-2021 | Thijs de Rooij & Taoufik Saadane | Hoofdstuk 7, 8, 9 & 10 toegevoegd |
|  |  |  |  |

# 1 Algemene beschrijving

De database applicatie van Brabotica 2.0 moet het mogelijk maken voor klanten en medewerkers om op een makkelijke manier gebruik te maken van de webshop van Brabotica 2.0. De volgende subsystemen zijn aanwezig in de database applicatie:

1. Een gebruikerssysteem waarin klanten zich kunnen aanmelden en waar ze kunnen inloggen;
2. Een productensysteem waarin klanten producten kunnen bestellen en medewerkers producten kunnen toevoegen, wijzigen en verwijderen;
3. Een ordersysteem waarin medewerkers orders kunnen toevoegen, wijzigen en verwijderen.

In het schema hieronder zijn de verschillende acties weergegeven en wie toegang hebben tot deze acties.

Diagram

Description automatically generated

Figuur : Schema algemene beschrijving

# 2 Aanpak

In dit hoofdstuk staat beschreven hoe deze proftaak wordt uitgewerkt en wat nodig is om deze proftaak tot een goed einde te brengen. Het eerste deelhoofdstuk beschrijft de planning van de proftaak. In het tweede deelhoofdstuk staan de afspraken die gemaakt zijn binnen de proftaak. Het laatste deelhoofdstuk beschrijft de eisen en wensen welke gesteld zijn aan de proftaak.

## 2.1 Planning

In de planning staat begrepen wat elke week gebeurt om de proftaak tot een succesvol einde te brengen. De planning is opgedeeld in weken om een concreet overzicht te geven welke taken wanneer worden verricht.

|  |  |  |
| --- | --- | --- |
| **Week** | **Lesstof** | **Proftaak werkzaamheden** |
| 1 | Inleiding relationele datapersistentie | Inlezen proftaakomschrijving |
| 2 | Gegevensmodelering | Vormen groepen en bepalen eerste opzet documenten |
| 3 | *Vakantie* | Thijs: Maken eerste opzet proftaakdocument |
| Taoufik: Maken eerste opzet ERD en opzetten Github repository |
| 4 | P&OC | Thijs & Taoufik: Opzetten database met bijbehorende tabellen |
| 5 | Data integriteit  Werken met SQL | Thijs: Uitbreiden document met gemaakte diagrammen |
| Taoufik: ERD en DDL verbeteren |
| 6 | Fysiek ontwerp  Triggers, Constraints & Procedures  Over indices | Thijs & Toaufik: Werken aan het ERD, aan het relationele model en aan de DDL |
| 7 | P&OC | Thijs: Opstellen van dummy data ten behoeve van de database en hier een SQL query van maken |
| Toaufik: DML opstellen en uitwerken in Sql queries |
| 8 | Statistische functies  Joins  Normaliseren  PHP & MySQL | Thijs: DCL opstellen in uitwerken in queries. Onderhoudshandleiding schrijven |
| Taoufik: DML verder uitwerken en eventuele verbeteringen doorvoeren |
| Thijs & Taoufik: Installatiehandleing schrijven en het document nalopen op fouten ten behoeve van het assessment |
| 9 | *Assessments* |  |
| 10 | *Assessments* |  |

## 2.2 Afspraken

Binnen de proftaak dienen een afspraken gemaakt te worden om ervoor te zorgen dat het eindproduct van gewenste kwaliteit is. In de volgende tabel is te vinden welke afspraken zijn gemaakt. Als nieuwe afspraken worden gemaakt wordt de tabel aangevuld.

|  |  |
| --- | --- |
| **Afspraak** | **Motivatie** |
| Engels is de voertaal in de code en de verschillende modellen | Dit is makkelijker, omdat belangrijke sleutelwoorden in de code en in de modellen ook Engelstalig zijn |
| Namen van tabellen zijn altijd in het enkelvoud | Dit is de standaard conventie |
| Namen van tabellen zijn altijd met een kleine letter | Om geen verwarring te veroorzaken met waarden in de tabellen |
| Namen van kolommen zijn altijd in het enkelvoud | Kolommen geven informatie over een enkele eigenschap van een entiteit |
| Namen van kolommen zijn altijd met een kleine letter | Om geen verwarring te veroorzaken met waarden in de tabellen |
| Programma om diagrammen en modellen te maken is Draw.io | Dit programma is gratis en toegankelijk voor iedereen |
| Voor het beheren van code wordt Github gebruikt | Github is gratis en is een handige manier om alles overzichtelijk te houden |

## 2.3 Eisen en wensen

|  |  |  |
| --- | --- | --- |
| **Nummer** | **Specificatie** | **Toelichting** |
| **Gebruikerssysteem** | | |
| 1 | Gebruikers moeten een account kunnen aanmaken | Om orders met producten succesvol naar klanten te kunnen sturen moeten gebruikers in staat zijn om een account aan te maken |
| 2 | Gebruikers moeten de gegevens van hun account aan kunnen passen | Dit moet voor klanten mogelijk zijn omdat de gegevens van een klant kunnen veranderen. Denk hierbij bijvoorbeeld aan een verhuizing |
| 3 | Gebruikers moeten hun account kunnen verwijderen | Het kan voorkomen dat klanten niet meer voornemens zijn om producten te kopen bij Brabotica 2.0, of ze willen niet meer dat hun gegevens bewaart worden |
| **Ordersysteem** | | |
| 1 | Het moet voor klanten mogelijk om een order toe te voegen aan het systeem | Wanneer een klant iets besteld heeft moet de database uitgebreid worden met de desbetreffende order |
| 2 | Klanten moeten een order kunnen annuleren | Het kan voorkomen dat de klant zich bedenkt na het plaatsen van een order. Deze zou dan geannuleerd moeten kunnen worden. Belangrijk is dat alleen orders van de desbetreffende klant geannuleerd kunnen worden |
| 3 | Medewerkers moeten orders van klanten kunnen inzien | Dit is noodzakelijk, want om een order succesvol te kunnen laten verlopen is het voor een medewerker nodig om te zien wat er besteld is |
| 4 | Medewerkers moeten orders kunnen toevoegen | Het kan soms voorkomen dat een klant niet in staat is om een order te plaatsen via het internet. Hierdoor is het voor een medewerker noodzakelijk om handmatig een order te kunnen invoeren |
| 5 | Medewerkers moeten een order kunnen wijzigen | Het kan voorkomen dat bepaalde producten niet voorradig zijn en een vervangend product geleverd moet worden. In dat geval moet de medewerker de order kunnen wijzigen |
| 6 | Medewerkers moeten een order kunnen verwijderen | Soms kan het voorkomen dat een order geplaatst is en niet aan de betalingsverplichting is voldaan. In dit geval moet een medewerker de order kunnen verwijderen |
| **Productensysteem** | | |
| 1 | Klanten moeten producten aan kunnen passen | Dit moet niet direct mogelijk zijn, maar de voorraad van een product dient aangepast te worden wanneer een klant een order plaatst |
| 2 | Medewerkers moeten producten aan kunnen passen | Het komt voor dat producten veranderen. Dit houdt in dat bijvoorbeeld kenmerken van producten kunnen wijzigen |
| 3 | Medewerkers moeten producten kunnen toevoegen | Het assortiment van Brabotica 2.0 verandert regelmatig en wordt steeds groter. Dit houdt in dat medewerkers nieuwe producten moeten kunnen toevoegen |
| 4 | Medewerkers moeten producten kunnen verwijderen | Sommige producten verdwijnen uit het assortiment en dienen dusdanig verwijdert te worden uit de database |

# 3 Functionele benodigdheden

Aan de proftaak zijn een aantal eisen en wensen gesteld. Om hieraan te voldoen zijn een aantal scenario’s opgesteld. Deze scenario’s zijn gebaseerd op een aantal acceptatiecriteria. In dit hoofdstuk zijn de verschillende scenario’s van alle subsystemen in kaart gebracht.

## 3.1 Algemeen

De database draait op een MySQL-server. Deze server is nodig om de database van Brabotica 2.0 functioneel en draaiend te houden. De toegewezen administrator van Brabotica 2.0 is verantwoordelijk voor het onderhoud van zowel de database als de MySQL-server.

**Scenario 1: Het beheren van de database  
User story:** De administrator kan de database aanpassen aan de hand van de geprefereerde tool **Scenario:** Het aanpassen van de database **Given:** De administrator opent de geprefereerde tool **When:** De administrator heeft een tabel uit de database geselecteerd **And:** De data in deze tabel kan worden aangepast **Then:** Deze wijzigingen worden doorgevoerd in de tabel

## 3.2 Gebruikerssysteem

**Scenario 1: Het aanmaken van een account  
User story:** De klant kan een account aanmaken op de website van Brabotica 2.0 **Scenario:** De klant vult zijn/haar gegevens in in de velden van de registratiepagina **Given:** De klant bevindt zich op de registratiepagina **When:** De klant heeft zijn/haar gegevens ingevuld in de velden **And:** De klant heeft nog geen account welke bestaat in de database **Then:** Het nieuwe account wordt opgeslagen in de database

**Scenario 2: Het aanpassen van een account  
User story:** De klant kan zijn/haar gegevens aanpassen op de website van Brabotica 2.0 **Scenario:** De klant vult de aan te passen gegevens in in de daarvoor bestemde velden **Given:** De klant bevindt zich op de wijzigingspagina van Brabotica 2.0 **When:** De klant heeft nieuwe gegevens ingevuld **And:** De klant heeft een bestaand account bij Brabotica 2.0 **Then:** De gegevens van de klant worden bijgewerkt in de database

**Scenario 3: Het verwijderen van een account  
User story:** De klant kan zijn/haar account verwijderen op de website van Brabotica 2.0 **Scenario:** Het verwijderen van een klant uit de database **Given:** De klant bevindt zich op de verwijderpagina van Brabotica 2.0 **When:** De klant heeft aangegeven dat hij/zij zijn/haar account wilt verwijderen **And:** De klant heeft een bestaand account bij Brabotica 2.0 **Then:** Het account van de klant wordt verwijderd

## 3.3 Ordersysteem

**Scenario 1: Het aanmaken van een order in de database door een klant  
User story:** De klant kan een order plaatsen in het systeem **Scenario:** De klant rondt de bestelling af en bestelt de door hem/haar gekozen producten **Given:** De klant bevindt zich op de betaalpagina van Brabotica 2.0 **When:** De klant succesvol betaald heeft **Then:** De order wordt geplaatst in de database

**Scenario 2: Het annuleren van een order door een klant  
User story:** Het is voor een klant mogelijk om een order te annuleren **Scenario:** De klant is niet tevreden over zijn/haar bestelling en wilt de order annuleren **Given:** De klant bevindt zich op de orderoverzichtpagina van Brabotica 2.0 **When:** De klant op de knop ‘Order annuleren’ heeft gedrukt en dit bevestigd heeft **And:** De order bestaat in de database **Then:** De door de klant gekozen order wordt geannuleerd en verwijdert uit de database

**Scenario 3: Het inzien van een order door een medewerker  
User story:** Een medewerker wilt een order inzien om te kijken welke producten bij de bestelling horen **Scenario:** De medewerker navigeert naar het orderoverzicht en selecteert de desbetreffende order **Given:** De medewerker bevindt zich op de orderoverzichtpagina **When:** De medewerker de juiste order heeft geselecteerd **And:** De order bestaat in de database **Then:** De geselecteerde order wordt getoond aan de medewerker

**Scenario 4: Het toevoegen van een order door een medewerker  
User story:** Een medewerker kan een order toevoegen aan het systeem wanneer dit voor een klant niet mogelijk is **Scenario:** De medewerker vult de order in in de database **Given:** De medewerker bevindt zich op de orderoverzichtpagina **When:** De medewerker de order heeft toegevoegd **And:** De medewerker is ingelogd als administrator **Then:** De order wordt toegevoegd aan de database

**Scenario 5: Het wijzigen van een order door een medewerker  
User story:** Een medewerker kan een order in het systeem wijzigen **Scenario:** De medewerker verandert de gegevens van een order **Given:** De medewerker bevindt zich op de overzichtspagina van een order **When:** De medewerker heeft wijzigingen aangebracht aan de order **And:** De medewerker is ingelogd als administrator  
**Then:** de order wordt gewijzigd in de database

**Scenario 6: Het verwijderen van een order door een medewerker  
User story:** Een medewerker kan een order verwijderen uit het systeem **Scenario:** De medewerker klikt op de ‘Verwijderen’ knop om een order te verwijderen  
**Given:** De medewerker bevindt zich op de overzichtpagina van een order **When:** De medewerker op de ‘Verwijderen’ knop heeft geklikt en dit bevestigd heeft **And:** De medewerker is ingelogd als administrator **Then:** De geselecteerde order wordt verwijderd uit het systeem

## 3.4 Productensysteem

**Scenario 1.1: Een klant moet een product kunnen aanpassen (voorraad)  
User story:** De klant kan gegevens van een product aanpassen op de website van Brabotica 2.0 door producten te bestellen **Scenario:** De klant heeft producten besteld en de voorraad van deze producten wordt automatisch bijgewerkt op de website van Brabotica 2.0 **Given:** De klant heeft succesvol een order geplaatst **When:** De klant is ingelogd met zijn/haar account **Then:** De voorraden van de producten worden bijgewerkt in de database

**Scenario 1.2: Een klant moet een product kunnen aanpassen (review)  
User story:** De klant kan gegevens van een product aanpassen op de website van Brabotica 2.0 door een review te plaatsen **Scenario:** De klant heeft een review geplaatst en de reviews van deze producten worden automatisch bijgewerkt op de website van Brabotica 2.0 **Given:** De klant heeft succesvol een review geplaatst **When:** De klant is ingelogd met zijn/haar account **Then:** De review van het product wordt bijgewerkt in de database

**Scenario 2: Het wijzigen van een product door een medewerker  
User story:** De medewerker past de gegevens van een product aan in het systeem **Scenario:** De medewerker past de gewenste gegevens aan van een product in de database  
**Given:** De medewerker bevindt zich in de overzichtpagina van het product **When:** De medewerker heeft aanpassingen aangebracht aan het product **And:** De medewerker is ingelogd als administrator **Then:** De gegevens van het product worden bijgewerkt in de database

**Scenario 3: Het toevoegen van een product door een medewerker  
User story:** De medewerker voegt een product toe aan de database  
**Scenario:** De medewerker voegt het gewenste product toe aan de database zodat deze te bestellen is voor klanten  
**Given:** De medewerker bevindt zich op het productenoverzicht **When:** Het product bestaat nog niet in de database **And:** De medewerker is ingelogd als administrator **Then:** Het product wordt toegevoegd aan de database

**Scenario 4: Het verwijderen van een product door een medewerker  
User story:** Een medewerker kan een product verwijderen uit het systeem **Scenario:** De medewerker klikt op de ‘Verwijderen’ knop om een product te verwijderen  
**Given:** De medewerker bevindt zich op de overzichtpagina van een product **When:** De medewerker op de ‘Verwijderen’ knop heeft geklikt en dit bevestigd heeft **And:** De medewerker is ingelogd als administrator **Then:** Het geselecteerde product wordt verwijderd uit het systeem

# 4 Entity Relationship Diagram (ERD)

Diagram

Description automatically generated

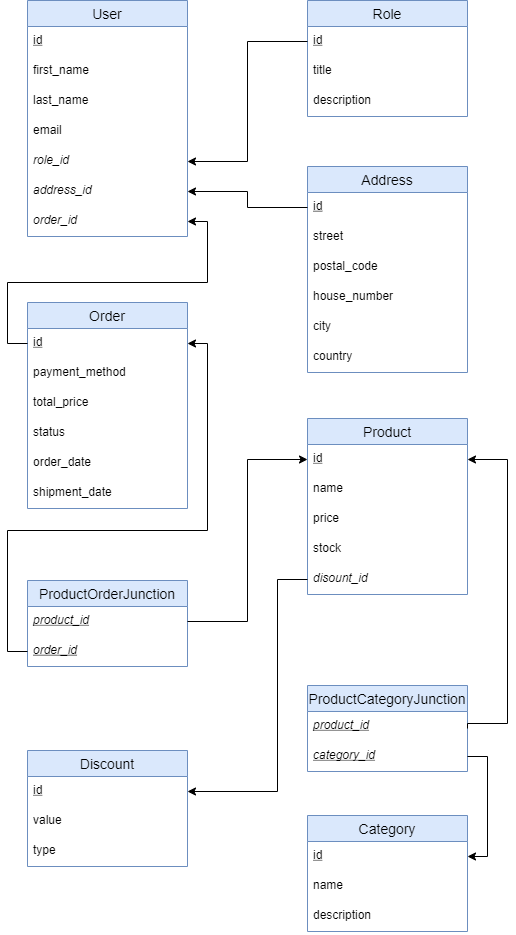
Figuur : ERD van Brabotica

## 4.1 Toelichting ERD

Omdat alleen een afbeelding niet voldoende is wordt in dit hoofdstuk toegelicht wat de verschillende entiteiten zijn in de databse en wat de onderlinge relaties zijn van deze entiteiten.

|  |  |
| --- | --- |
| **Tabel** | **Toelichting** |
| Address | Elk Address kan meerdere users hebben, omdat het voor kan komen dat klanten woonzaam zijn op hetzelfde adres. |
| Category | Category is onderdeel van een product. Een Category heeft een enkel Product of geen. Het kan voorkomen dat bij voorbaat een nieuwe Category wordt gemaakt voor nieuwe producten. |
| Discount | Discount is onderdeel van een product. Een Discount kan een of meerdere producten hebben. Omdat dezelfde Discount onderdeel kan zijn van meerdere producten. |
| Order | Een Order kan een enkele of meedere producten bevatten. Om een Order te kunnen plaatsen moeten er producten in de order zitten. Tevens heeft elke Order een enkele User. Een User moet een order plaatsen. Daarom is dit verplicht |
| Product | Een Product kan geen of meerdere orders hebben. Tevens kan een Product geen of een enkele Discount hebben. Als laatste kan een Product een enkele of meerdere categories hebben. |
| Role | Een Role kan een enkele of meerdere users hebben. |
| User | Elke User heeft een enkel adres. Hiervoor is bewust gekozen in plaats van dat er gebruik wordt gemaakt van bijvoorbeeld een afwijkend factuuradres. Elke User kan ook maar een enkele Role hebben. Tevens kan een User geen of meerdere orders hebben. |

# 5 Relationele Database



Figuur : Relationeel model van Brabotica

# 6 Database realisatie (DDL)

De DDL is noodzakelijk om een database op te zetten en in te richten. Voordat de verschillende CREATE-queries worden aangeroepen wordt eerst gecontroleerd of de tabellen al bestaan binnen de database. Dit gebeurt door middel van het volgende statement: DROP TABLE IF EXISTS <tabelnaam>. Een overzicht van de volledige DDL-query staat in Bijlage 1.

Om de tabellen op te zetten is gebruik gemaakt van de volgende eigenschappen:

|  |  |
| --- | --- |
| **Eigenschappen** | **Omschrijving** |
| NOT NULL | Het is hier niet mogelijk om een NULL waarde in te geven. Dit zorgt ervoor dat dit een verplicht veld is |
| INT | Kan alleen worden ingevuld met een geheel getal |
| VARCHAR | Kan ingevuld worden door alle soorten karakters |
| AUTO\_INCREMENT | Wanneer een waarde wordt aangemaakt waar deze eigenschap aan vast hangt, dan neemt de waarde automatisch toe met 1 |
| DEFAULT | Als er bij het invoeren van een nieuwe regel niks is ingegeven in de tabel wordt de waarde/tekst achter DEFAULT ingevuld |
| DATE | Dit veld kan alleen worden ingevuld met een datum |
| LONGTEXT | Dit veld kan worden ingevuld met tekst |
| DECIMAL | Dit veld wordt ingevuld met een niet geheel getal |
| PRIMARY KEY | Dit is de primary en unieke key van een tabel. Waarden in de tabel kunnen worden herkent door deze waarde. |
| FOREIGN KEY | De foreign key verwijst naar een primary key van een andere tabel. De foreign key wordt gebruikt om twee waarden in feite aan elkaar te koppelen. |

## 6.1 Structuur tabel Address

CREATE TABLE `address`

(

    `id`           int(11) NOT NULL AUTO\_INCREMENT,

    `street`       varchar(255) NOT NULL,

    `postal\_code`  varchar(255) NOT NULL,

    `house\_number` varchar(5)   NOT NULL,

    `city`         varchar(25)  NOT NULL,

    `country`      varchar(255) NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuw Address de ‘id’ automatisch ophogen.
* ‘street’ is van het type varchar met een maximale grootte van 255.
* ‘postal\_code’ is van het type varchar met een maximale grootte van 255.
* ‘house\_number’ is van het type varchar met een maximale grootte van 5, omdat letters kunnen voorkomen in een huisnummer.
* ‘city’ is van het type varchar met een maximale grootte van 25.
* ‘country’ is van het type varchar met een maximale grootte van 255.

## 6.2 Structuur tabel Category

CREATE TABLE `category`

(

    `id`         int(11) NOT NULL AUTO\_INCREMENT,

    `name`       varchar(255) NOT NULL,

    `desciption` longtext     NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuwe Category de ‘id’ automatisch ophogen.
* ‘name’ is van het type varchar met een maximale grootte van 255.
* ‘description’ is van het type longtext, omdat een varchar van 255 vaak niet voldoende is.

## 6.3 Structuur tabel Discount

CREATE TABLE `discount`

(

    `id`    int(11) NOT NULL AUTO\_INCREMENT,

    `type`  varchar(10)   NOT NULL,

    `value` decimal(5, 2) NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuwe Discount de ‘id’ automatisch ophogen.
* ‘type’ is van het type varchar met een maximale grootte van 10.
* ‘value’ is van het type decimal met een grootte van 5 cijfers en maximaal 2 cijfers achter de komma.

## 6.4 Structuur tabel Order

CREATE TABLE `order`

(

    `id`             int(11) NOT NULL AUTO\_INCREMENT,

    `payment\_method` varchar(255)   NOT NULL,

    `total\_price`    decimal(10, 2) NOT NULL,

    `status`         varchar(255)   NOT NULL,

    `order\_date`     date           NOT NULL,

    `shipment\_date`  date DEFAULT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuwe Order de ‘id’ automatisch ophogen.
* ‘payment\_method’ is van het type varchar met een maximale grootte van 255.
* ‘total\_price’ is van het type decimal met een grootte van 10 cijfers en maximaal 2 cijfers achter de komma.
* ‘status’ is van het type varchar met een maximale grootte van 255.
* ‘order\_date’ is van het type date.
* ‘shipment\_date’ is van het type date. En is standaard null, dit komt omdat een order niet altijd meteen verzonden wordt.

## 6.5 Structuur table Product

CREATE TABLE `product`

(

    `id`          int(11) NOT NULL AUTO\_INCREMENT,

    `discount\_id` int(11) DEFAULT NULL,

    `name`        varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

    `price`       decimal(10, 0)                          NOT NULL,

    `stock`       int(11) NOT NULL,

    PRIMARY KEY (`id`),

    FOREIGN KEY (`discount\_id`) REFERENCES `discount` (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuw Product de ‘id’ automatisch ophogen.
* ‘discount\_id’ is de vreemde sleutel in de tabel. Deze verwijst naar de id van de Discount tabel. Tevens is deze waarde van het type int.
* ‘name’ is van het type varchar met een maximale grootte van 255.
* ‘price’ is van het type decimal met een grootte van 10 cijfers en maximaal 2 cijfers achter de komma.
* ‘stock’ is van het type int

## 6.6 Structuur tabel ProductCategoryJunction

CREATE TABLE `product\_category`

(

    `product\_id`  int(11) NOT NULL,

    `category\_id` int(11) NOT NULL,

    PRIMARY KEY (`product\_id`, `category\_id`),

    FOREIGN KEY (`category\_id`) REFERENCES `category` (`id`)

ON DELETE CASCADE,

    FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`)

ON DELETE CASCADE

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘product\_id’ is de unieke identifier van de tabel. Tevens is deze waarde de vreemde sleutel en verwijst naar het bijbehorende Product uit die tabel.
* ‘category\_id’ is de unieke identifier van de tabel. Tevens is deze waarde de vreemde sleutel en verwijst naar het bijbehorende Category uit die tabel.

## 6.7 Structuur tabel ProductOrderJunction

CREATE TABLE `order\_product`

(

    `order\_id`   int(11) NOT NULL,

    `product\_id` int(11) NOT NULL,

    PRIMARY KEY (`order\_id`, `product\_id`),

    FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`)

ON DELETE CASCADE,

    FOREIGN KEY (`order\_id`) REFERENCES `order` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘order\_id’ is de unieke identifier van de tabel. Tevens is deze waarde de vreemde sleutel en verwijst naar het bijbehorende Order uit die tabel.
* ‘product\_id’ is de unieke identifier van de tabel. Tevens is deze waarde de vreemde sleutel en verwijst naar het bijbehorende Product uit die tabel.

## 6.8 Structuur tabel Role

CREATE TABLE `role`

(

    `id`          int(11) NOT NULL AUTO\_INCREMENT,

    `title`       varchar(255) NOT NULL,

    `description` longtext,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuwe Role de ‘id’ automatisch ophogen.
* ‘title’ is van het type varchar met een maximale grootte van 255.
* ‘description’ is van het type longtex.

## 6.9 Structuur tabel User

CREATE TABLE `user`

(

    `id`         int(11) NOT NULL AUTO\_INCREMENT,

    `role\_id`    int(11) DEFAULT NULL,

    `address\_id` int(11) DEFAULT NULL,

    `first\_name` varchar(255) NOT NULL,

    `last\_name`  varchar(255) NOT NULL,

    `email`      varchar(255) NOT NULL,

    PRIMARY KEY (`id`),

    FOREIGN KEY (`role\_id`) REFERENCES `role` (`id`),

    FOREIGN KEY (`address\_id`) REFERENCES `address` (`id`)

) ENGINE=InnoDB;

De structuur van de tabel is als volgt opgebouwd:

* ‘id’ is de unieke identifier van de tabel. Is van het type int en zal bij het aanmaken van een nieuwe User de ‘id’ automatisch ophogen.
* ‘role\_id’ is de vreemde sleutel van de tabel. En verwijst naar de bijbehorende Role uit die tabel.
* ‘address\_id’ is de vreemde sleutel van de tabel. En verwijst naar de bijbehorende Address uit die tabel.
* ‘first\_name’ is van het type varchar met een maximale grootte van 255.
* ‘last\_name’ is van het type varchar met een maximale grootte van 255.
* ‘email’ is van het type varchar met een maximale grootte van 255.

# 7 Beveliging (DCL)

# 8 Onderhoud

# 9 Installatie

# 10 Uitwerkingen (DML)

## 10.1 Gebruikerssysteem

## 10.2 Ordersysteem

## 10.3 Productensysteem

# Bijlage 1: Compleet overzicht van de DDL

CREATE DATABASE IF NOT EXISTS `brabotica`;

USE `brabotica`;

SET FOREIGN\_KEY\_CHECKS=0;

DROP TABLE IF EXISTS `role`;

DROP TABLE IF EXISTS `address`;

DROP TABLE IF EXISTS `user`;

DROP TABLE IF EXISTS `category`;

DROP TABLE IF EXISTS `discount`;

DROP TABLE IF EXISTS `product\_category`;

DROP TABLE IF EXISTS `product`;

DROP TABLE IF EXISTS `order\_product`;

DROP TABLE IF EXISTS `order`;

CREATE TABLE `role`

(

    `id`          int(11) NOT NULL AUTO\_INCREMENT,

    `title`       varchar(255) NOT NULL,

    `description` longtext,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

CREATE TABLE `address`

(

    `id`           int(11) NOT NULL AUTO\_INCREMENT,

    `street`       varchar(255) NOT NULL,

    `postal\_code`  varchar(255) NOT NULL,

    `house\_number` varchar(5)   NOT NULL,

    `city`         varchar(25)  NOT NULL,

    `country`      varchar(255) NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

CREATE TABLE `user`

(

    `id`         int(11) NOT NULL AUTO\_INCREMENT,

    `role\_id`    int(11) DEFAULT NULL,

    `address\_id` int(11) DEFAULT NULL,

    `first\_name` varchar(255) NOT NULL,

    `last\_name`  varchar(255) NOT NULL,

    `email`      varchar(255) NOT NULL,

    PRIMARY KEY (`id`),

    FOREIGN KEY (`role\_id`) REFERENCES `role` (`id`),

    FOREIGN KEY (`address\_id`) REFERENCES `address` (`id`)

) ENGINE=InnoDB;

CREATE TABLE `category`

(

    `id`         int(11) NOT NULL AUTO\_INCREMENT,

    `name`       varchar(255) NOT NULL,

    `desciption` longtext     NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

CREATE TABLE `discount`

(

    `id`    int(11) NOT NULL AUTO\_INCREMENT,

    `type`  varchar(10)   NOT NULL,

    `value` decimal(5, 2) NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

CREATE TABLE `product\_category`

(

    `product\_id`  int(11) NOT NULL,

    `category\_id` int(11) NOT NULL,

    PRIMARY KEY (`product\_id`, `category\_id`),

    FOREIGN KEY (`category\_id`) REFERENCES `category` (`id`) ON DELETE CASCADE,

    FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `product`

(

    `id`          int(11) NOT NULL AUTO\_INCREMENT,

    `discount\_id` int(11) DEFAULT NULL,

    `name`        varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

    `price`       decimal(10, 0)                          NOT NULL,

    `stock`       int(11) NOT NULL,

    PRIMARY KEY (`id`),

    FOREIGN KEY (`discount\_id`) REFERENCES `discount` (`id`)

) ENGINE=InnoDB;

CREATE TABLE `order\_product`

(

    `order\_id`   int(11) NOT NULL,

    `product\_id` int(11) NOT NULL,

    PRIMARY KEY (`order\_id`, `product\_id`),

    FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE,

    FOREIGN KEY (`order\_id`) REFERENCES `order` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `order`

(

    `id`             int(11) NOT NULL AUTO\_INCREMENT,

    `payment\_method` varchar(255)   NOT NULL,

    `total\_price`    decimal(10, 2) NOT NULL,

    `status`         varchar(255)   NOT NULL,

    `order\_date`     date           NOT NULL,

    `shipment\_date`  date DEFAULT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB;

SET FOREIGN\_KEY\_CHECKS=1;