

SilentDisco

Beheerdokument

Auteurs: Thomas Daane, Marcel Forman, Bart Frijters & Wesley Geboers

Datum: 10-02-2022

Plaats: Breda

Jaar: 2022

Contents

1. Algemeen	1
2. Aanpak	2
2.1 Planning	2
2.2 Afspraken.....	2
3. Functional requirements	3
3.1 Algemeen.....	3
3.2 Webshop	3
3.3 Klantenportaal.....	4
3.4 Medewerkersportaal.....	5
4. Entity Relationship Diagram (ERD).....	9
4.1 Toelichting ERD.....	9
4.1.1 Employees	9
4.1.2 Roles	9
4.1.3 Products.....	10
4.1.4 ProductLogs	10
4.1.5 Users.....	10
5. Relational diagram (RD)	11
5.1 Toelichting RD.....	11
5.1.1 Employees	11
5.1.2 Roles	11
5.1.3 Employee-Roles	11
5.1.4 Products.....	12
5.1.5 ProductLogs	12
5.1.6 Users.....	12
5.1.7 OrderHeaders	12
5.1.8 OrderLines	12
6. Database realisatie (DDL)	13
6.1 Tables.....	13
6.1.1 Structuur table Employees	13
6.1.2 Structuur table Roles	14
6.1.3 Structuur table Employees-Roles	15
6.1.4 Structuur table Products	16
6.1.5 Structuur table ProductLogs.....	17
6.1.6 Structuur table Users.....	18
6.1.7 Structuur table OrderHeaders.....	19

6.1.8 Structuur table OrderLines	20
6.2 Triggers	20
6.2.1 Trigger LineTotalPrice	20
6.2.2 Trigger UpdateTotalPriceHeader	21
6.3 Views	21
6.3.1 View Top10ProductsAmount	21
6.3.2 View Top10ProductsPrice	22
6.3.3 View RevenuePerMonth	23
6.3.4 View Invoices	23
6.4 Stored Procedure	24
6.4.1 Profielgegevens tonen	24
6.4.2 Status bestelling bekijken	24
6.4.3 Bestelling details bekijken	25
6.5 Stored Function	25
6.5.1 users_updateName	25
6.5.2 Toon jaren in dienst medewerker	26
7. Security (DCL)	27
7.1 Admin	27
7.2 system	27
7.3 wachtwoorden	27
8. Maintenance	28
8.1 Instructie backup	28
8.2 Instructie analyze table	28
8.3 Instructie Check table	28
8.3 Instructie optimize table	28
9. Installatie	29
9.1 Installatie handleiding	29
9.1.1 Installatiebestand uitpakken	29
9.1.2 Installatie via de command line	29
9.1.3 Importeren via phpMyAdmin	29
9.1.4 Query handmatig kopiëren en plakken via phpMyAdmin	30
9.2 Insert data	30
9.2.1 Insert Employees	30
9.2.2 Insert Users	31
9.2.3 Insert Roles	31
9.2.4 Insert Employees-Roles	31

9.2.5 Insert Products	32
9.2.6 Insert ProductLogs.....	32
9.2.7 Insert OrderHeaders.....	32
9.2.8 Insert OrderLines	33
10 Uitwerking (DML)	34
10.1 Edit user.....	34
10.2 Show orders.....	35
10.3 Show order details.....	35
10.4 Edit products	35
10.5 Delete user role	36

1. Algemeen

In dit document zal de database voor StilteAUBv gedocumenteerd worden. StilteAUBv heeft momenteel een promotionele website die omgebouwd zal worden naar een webshop. Om de webshop goed te laten functioneren zal er een database gebouwd en gekoppeld worden.

Vanuit de webshop is het voor de klant mogelijk om bestellingen te plaatsen en bestaande orders in te zien. Dit betekent dat de database hierop voorbereid moet zijn. De database zal daardoor data vastleggen van bijvoorbeeld artikelen, bestellingen en klanten.

Ook zal de webshop voorzien worden van een medewerkersportaal waarmee artikelen aangemaakt en gewijzigd kunnen worden. Ook zal het vanuit hier mogelijk zijn om de gemaakte bestellingen in te zien en van status te veranderen. Omdat niet alle data beschikbaar mag zijn voor alle medewerkers zullen er rollen en rechten gemaakt worden. De database zal om deze reden ook data vastleggen voor medewerkers en de rollen van die medewerkers.

Met dit in gedachte zal het vervolg van dit document de database structuur verder beschrijven. Zo zullen er ERD en RD-diagrammen gemaakt worden, zullen de query's gedocumenteerd worden en zal de security en onderhoud verder beschreven worden.

2. Aanpak

2.1 Planning

Om dit project in goede banen te kunnen leiden is er een globale planning gemaakt. De verschillende stappen zijn beschreven met de daar bijhorende verantwoordelijke en start- en einddatum. De planning is hieronder te zien.

Beschrijving	Verantwoordelijke	Start	Eind
Start Periode 3		02-02-2022	
Beheersdocument – 1 Algemeen	Thomas Daane	09-02-2022	17-02-2022
Beheersdocument – 2 Aanpak	Wesley Geboers	09-02-2022	17-02-2022
Beheersdocument – 3 Functional requirements	Team	09-02-2022	17-02-2022
Beheersdocument – 4 ERD	Team	09-02-2022	17-02-2022
Beheersdocument – 5 Relational model	Team	09-02-2022	17-02-2022
Beheersdocument – 6 Database realization	Team	18-02-2022	10-03-2022
Beheersdocument – 7 Security	Marcel Forman	18-02-2022	10-03-2022
Beheersdocument – 8 Maintanance	Bart Frijters	18-02-2022	10-03-2022
Github inrichten	Wesley Geboers	17-02-2022	18-02-2022
Database inrichten	Team	18-02-2022	10-03-2022
webbased frontend inrichten	Team	10-03-2022	17-03-2022
Oplevering			27-03-2022

2.2 Afspraken

Voor het schrijven van dit document, versiebeheer en query's zijn er als team een aantal afspraken gemaakt, met het doel om uniform te werk te gaan. Hieronder zijn de verschillende afspraken beschreven.

Definitie	Toelichting
Versiebeheer	Versiebeheer zal plaatsvinden op GitHub.
Taal	De tabellen, kolommen etc. worden geschreven in het engels.
Query's	<ul style="list-style-type: none">• Sleutelwoorden in hoofdletters. Denk hierbij aan SELECT en WHERE.• Tabelnamen in meervoud.• Kolomnamen in enkelvoud.• Tekst, datum en tijd in enkele aanhalingstekens.• Getallen zonder aanhalingstekens.
Modellen	De modellen worden getekend op Draw.io

De URL naar GitHub: <https://github.com/wgeboers/stilteaubvDatabase>

3. Functional requirements

3.1 Algemeen

Scenario 1: Registreren

User story: De gebruiker heeft de mogelijkheid om een account aan te maken voor het inzien van zijn of haar bestellingen en de daarbij horende statussen.

Scenario: De gebruiker bezoekt de site, heeft nog geen account en beslist een account aan te maken.

Given: De gebruiker heeft nog geen account en drukt op de knop registreren.

When: De gebruiker heeft op de knop registreren gedrukt waarna er een formulier verschijnt.

And: De gebruiker vult de benodigde gegevens in.

Then: De gebruiker drukt op opslaan waarna de gegevens worden opgeslagen in de tabel `Users`.

Scenario 2: Inloggen

User story: De gebruiker kan indien hij of zij een account heeft inloggen op de website om zo zijn of haar bestellingen en de daarbij horende statussen in te zien.

Scenario: De gebruiker bezoekt de site en wilt de status van zijn of haar bestelling weten.

Given: De gebruiker heeft een bestaand account en drukt op de knop inloggen.

When: De gebruiker heeft op de knop inloggen gedrukt en vult zijn of haar inloggegevens in.

And: De site haalt toont de profiel pagina van de gebruiker. Deze gegevens worden aan de hand van het account opgehaald uit de database en getoond op de pagina.

Then: De gebruiker drukt op de knop bestellingen waarna de bestellingen van de ingelogde gebruiker worden opgehaald uit de database en getoond op de pagina bestellingen.

3.2 Webshop

Scenario 1: Producten toevoegen aan winkelmandje

User story: De gebruiker kan op de website 1 of meerdere producten aan zijn of haar winkelmandje toevoegen om later te bestellen.

Scenario: De gebruiker bezoekt de site en ziet een product die hij of zij wilt bestellen.

Given: De gebruiker ziet een product die hij of zij wilt bestellen, vult een aantal in en drukt op de knop toevoegen.

When: De gebruiker heeft een aantal ingevuld en op de knop toevoegen gedrukt.

Then: De producten zullen tijdelijk worden opgeslagen in de cache en toegevoegd aan het winkelmandje.

Scenario 2: Bestellen

User story: De gebruiker heeft 1 of meerdere producten toegevoegd aan het winkelmandje en kan deze producten nu bestellen.

Scenario: De gebruiker heeft 1 of meerdere producten aan het winkelmandje toegevoegd en wilt deze gaan bestellen.

Given: De gebruiker heeft alle gewenste producten en aantallen ingevuld en drukt op de knop bestellen.

When: De gebruiker heeft op de knop bestellen gedrukt en krijgt een popup te zien met alle producten die besteld worden en het totaal bedrag. De gebruiker kiest een betaal methode en drukt op de knop bestellen.

And: De gebruiker betaald via de ideal pagina.

Then: De gebruiker word doorverwezen naar een pagina waarop word aangegeven dat de bestelling is gelukt. Gelijker tijd word de bestelling opgeslagen in de tabellen `OrderHeaders` en `OrderLines`.

3.3 Klantenportaal

Scenario 1: Profielgegevens wijzigen

User story: De gebruiker heeft de mogelijkheid om zijn of haar profielgegevens te wijzigen in het klanten portaal.

Scenario: De gebruiker heeft een bestaand account en wilt zijn of haar gegevens wijzigen.

Given: De gebruiker heeft al een account, is ingelogd en drukt op de knop profiel.

When: De gebruiker heeft op de knop profiel gedrukt waarna de data van de ingelogde gebruiker word opgehaald vanuit de database en getoond kan worden op de profiel pagina.

And: De gebruiker drukt op de knop profielgegevens wijzigen, past de gewenste gegevens aan en drukt op opslaan.

Then: De gewijzigde gegevens zullen worden opgeslagen in de table `Users`. Er zal op dat moment een update plaatsvinden op de record die bij het ingelogde account hoort.

Scenario 2: Overzicht bestellingen en bestelling details bekijken

User story: De gebruiker heeft de mogelijkheid om een overzicht van de bestellingen op te roepen en per bestelling kunnen de details worden bekeken.

Scenario: De gebruiker heeft een bestaand account en wilt zijn of haar bestellingen bekijken.

Given: De gebruiker heeft al een account, is ingelogd en drukt op de knop profiel.

When: De gebruiker heeft op de knop *overzicht bestellingen* geklikt.

And: De gebruiker op een bestelling die aanwezig is in het overzicht.

Then: Het overzicht toont alle bestellingen, indien aanwezig en na het klikken op een bestelling worden de details van die bestelling getoond.

3.4 Medewerkersportaal

Scenario 1: Medewerker aanmaken

User story: IT medewerkers hebben de rechten om accounts voor medewerkers van StilteAUBv aan te maken.

Scenario: De IT medewerkers heeft een bestaand account, de benodigde rechten en wilt een account aanmaken voor een medewerker van StilteAUBv.

Given: De IT medewerker heeft een bestaand account, is onderdeel van de rol IT, zit in het medewerkersportaal en drukt op de knop account aanmaken.

When: De IT medewerker heeft op de knop account aanmaken gedrukt waarna de site een formulier opent voor het aanmaken van de account.

And: De gebruiker vult de benodigde gegevens in en drukt op de knop opslaan.

Then: De gegevens zullen worden opgeslagen in de tabel `Employees`.

Scenario 2: Rechten wijzigen

User story: IT medewerkers met de juiste rechten hebben de mogelijkheid om de rechten van medewerker accounts aan te passen. Dit doet hij of zij door een account toe te voegen aan de gewenste rol in het medewerkersportaal.

Scenario: Een medewerker binnen StilteAUBv gaat voortaan werken voor de afdeling operations en heeft daarvoor andere rechten nodig. De IT medewerker wilt deze rechten toepassen in het medewerkersportaal.

Given: De IT medewerker heeft een bestaand account, bezit de juiste rechten, zit in het medewerkers portaal, kiest de gewenste gebruiker en drukt op de knop wijzigen.

When: De IT medewerker heeft op de knop wijzigen gedrukt waarna het profiel van het gekozen account getoond word.

And: De IT medewerker past de rol van het account aan naar operations.

Then: De gewijzigde gegevens zullen worden opgeslagen in de tabel `Employees`. Er zal op dat moment een update plaatsvinden op de record die van toepassing is.

Scenario 3: Account deactiveren

User story: IT medewerkers met de juiste rechten hebben de mogelijkheid om een medewerkers account te deactiveren. Dit doet hij of zij door het account op inactief te zetten in het medewerkersportaal

Scenario: Een medewerker binnen StilteAUBv gaat uit dienst waardoor het account gedeactiveerd moet worden. De IT medewerker wilt het account deactiveren in het medewerkersportaal.

Given: De IT medewerker heeft een bestaand account, bezit de juiste rechten, zit in het medewerkers portaal, kiest de gewenste gebruiker en drukt op de knop wijzigen.

When: De IT medewerker heeft op de knop wijzigen gedrukt waarna het profiel van het gekozen account getoond word.

And: De IT medewerker zit het vinkje uit bij actief.

Then: De gewijzigde gegevens zullen worden opgeslagen in de tabel `Employees`. Er zal op dat moment een update plaatsvinden op de record die van toepassing is.

Scenario 4: Artikelen toevoegen

User story: Een operations medewerker heeft de mogelijkheid om artikelen/producten toe te voegen aan de webshop.

Scenario: StilteAUBv wilt een nieuw product aanbieden op de webshop. De operations medewerker wilt dit product toevoegen aan de webshop.

Given: De operations medewerker heeft een bestaand account, de juiste rechten, zit in het medewerkers portaal en drukt op de knop artikelen toevoegen.

When: De operations medewerker heeft op de knop artikelen toevoegen gedrukt.

And: De site toont een formulier en de operations medewerker vult de benodigde gegevens in.

Then: De gegevens worden opgeslagen in de tabel `Products`. Ook word er een record aangemaakt in de tabel `ProductLogs`.

Scenario 5: Artikelen wijzigen

User story: Een operations medewerker heeft de mogelijkheid op de gegevens van een artikel/product te wijzigen in het medewerkersportaal. Denk hierbij aan het wijzigen van voorraad, prijs, naam en omschrijving.

Scenario: Er zijn producten geleverd waarna de voorraad op de site gewijzigd moet worden. De operations medewerker wilt dit aanpassen in het medewerkersportaal.

Given: De operations medewerker heeft een bestaand account, de juiste rechten, zit in het medewerkers portaal en drukt op de knop wijzigen naast het gewenste artikel.

When: De operations medewerker heeft op de knop wijzigen gedrukt.

And: De site toont een formulier die vooraf is ingevuld met gegevens van het gekozen artikel. Deze gegevens worden opgehaald vanuit de database en getoond in het formulier. De operations medewerker past de voorraad aan en drukt op opslaan.

Then: De gegevens worden opgeslagen in de tabel `Products`. Ook word er een record aangemaakt in de tabel `ProductLogs` op de wijzigingen van de producten bij te kunnen houden.

Scenario 6: Top 10 verkochte producten op basis van hoeveelheid

User story: Een operations medewerker is verantwoordelijk voor de inkoop van producten en moet bepalen welke hoeveelheid moet worden ingekocht.

Scenario: Voor de hoeveelheid te bestellen artikelen is het van belang om te weten welke producten het meest worden verkocht op basis van aantallen.

Given: De operations medewerker heeft een bestaand account, de juiste rechten en zit in het medewerkers portaal.

When: De operations medewerker drukt op de knop *Top 10 verkochte artikelen*.

And: Kiest daarna voor *op basis van aantal*.

Then: De top 10 van verkochte producten op basis van aantallen per product wordt getoond aan de gebruiker.

Scenario 7: Top 10 verkochte producten op basis van bedrag

User story: De directie is verantwoordelijk voor het bepalen van de strategie voor de organisatie en heeft hier informatie voor nodig over de verkoop.

Scenario: De directie heeft een overzicht nodig van de top 10 verkochte producten op basis van de omzet.

Given: De directie medewerker heeft een bestaand account, de juiste rechten en zit in het medewerkers portaal.

When: De operations medewerker drukt op de knop *Top 10 verkochte artikelen*.

And: Kiest daarna voor *op basis van omzet*.

Then: De top 10 van verkochte producten op basis van de totale verkoopopbrengst per product wordt getoond aan de gebruiker.

Scenario 8: Omzet op basis van jaar en maand

User story: De afdeling administratie is verantwoordelijk voor de boekhouding en heeft hier informatie over de omzet voor nodig.

Scenario: Een medewerker van de administratie heeft een overzicht nodig van omzet per jaar onderverdeeld per maand.

Given: De administratie medewerker heeft een bestaand account, de juiste rechten en zit in het medewerkers portaal.

When: De administratie medewerker drukt op de knop *Toon omzetoverzicht*

Then: Het overzicht toont de omzet per jaar onderverdeeld in maanden met per maand de omzet.

Scenario 9: Jaren in dienst van een medewerker

User story: De afdeling administratie is verantwoordelijk het plannen en organiseren van evenementen voor een medewerker wanneer deze een bepaald aantal jaar in dienst is.

Scenario: Een medewerker van de administratie heeft een overzicht van het aantal jaren in dienst van de medewerkers.

Given: De administratie medewerker heeft een bestaand account, de juiste rechten en zit in het medewerkers portaal.

When: De administratie medewerker drukt op de knop *Toon aantal jaren in dienst*

Then: Het overzicht toont per medewer

Scenario 10: Overzicht bestellingen per klant

User story: De afdeling administratie is verantwoordelijk voor de boekhouding en heeft hier informatie over de bestellingen per klant voor nodig.

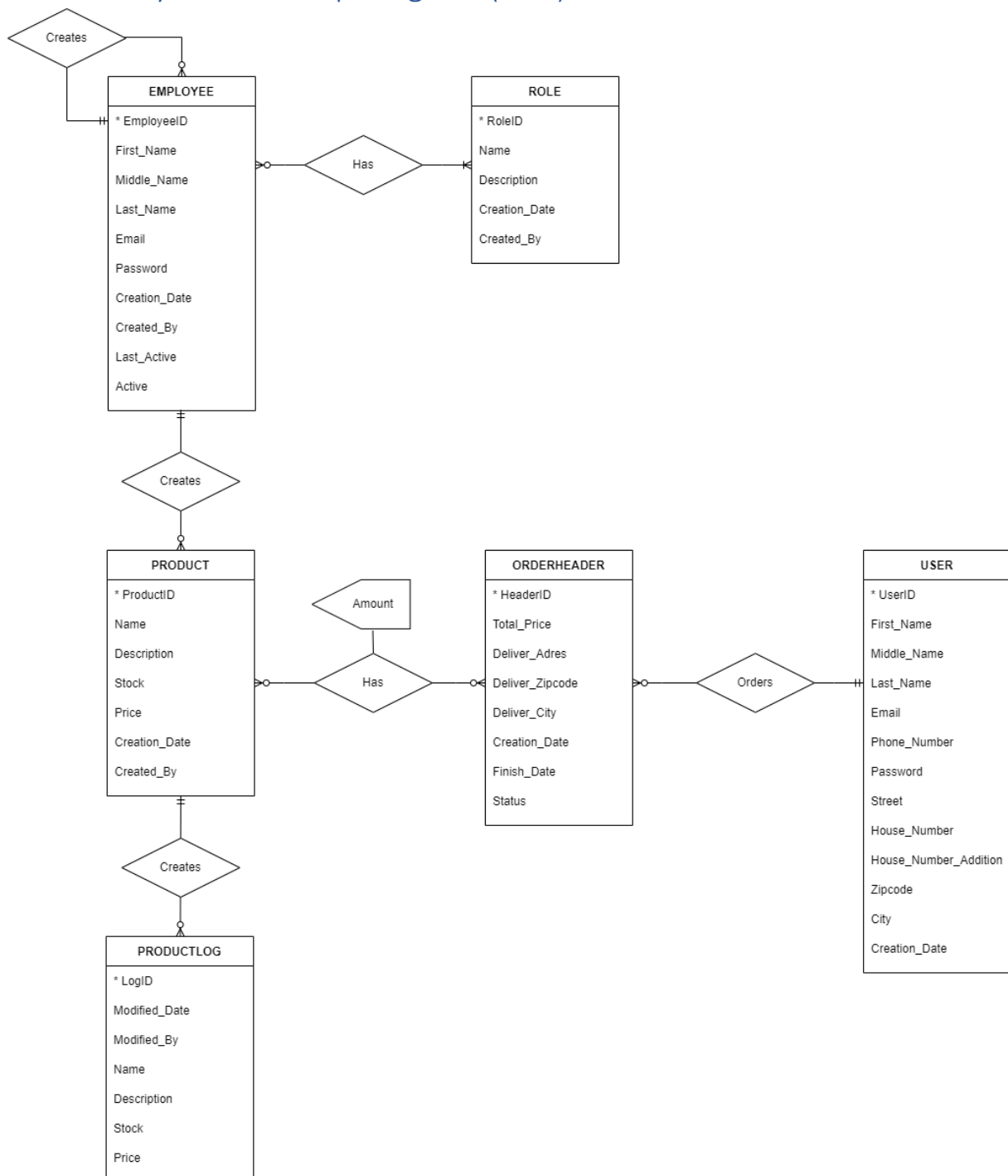
Scenario: Een medewerker van de administratie heeft een overzicht nodig van de bestellingen met bedrag per klant.

Given: De administratie medewerker heeft een bestaand account, de juiste rechten en zit in het medewerkers portaal.

When: De administratie medewerker drukt op de knop *Toon facturen per klant*

Then: Het overzicht toont per klant de bestelling en het bedrag.

4. Entity Relationship Diagram (ERD)



4.1 Toelichting ERD

4.1.1 Employees

Medewerkers van StilteAUBv hebben de mogelijkheid om gebruik te maken van het medewerkersportaal, hiervoor hebben zij echter wel een account voor nodig. De entiteit 'Employees' zullen deze gegevens opslaan en zo toegang geven tot het medewerkersportaal.

4.1.2 Roles

Als een medewerker een account heeft en dus bestaat in de entiteit 'Employees' heeft de medewerker ook nog een rol nodig. Deze rollen worden opgeslagen in de entiteit 'Roles'. Elke

employee heeft tenminste 1 of meerdere rollen. Deze rollen bepalen uiteindelijk de rechten van een medewerker in het medewerkersportaal.

4.1.3 Products

De webshop zal uiteraard producten nodig hebben. Deze producten worden opgeslagen in de entiteit `Products`.

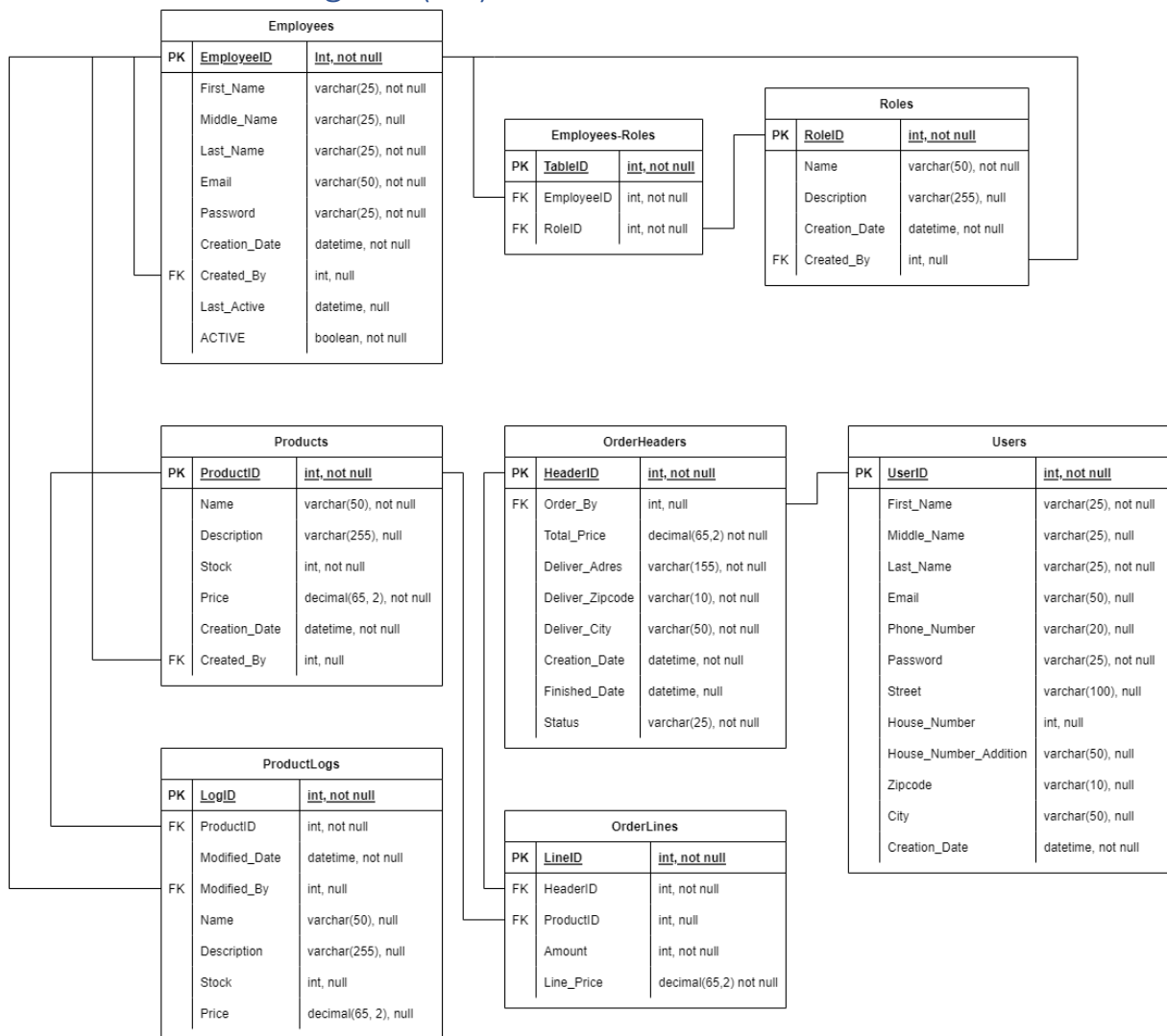
4.1.4 ProductLogs

Producten kunnen gewijzigd worden door de operations medewerkers van StilteAUBv. Deze wijzigingen dienen gelogd te worden voor eventuele rapportages op de instroom en uitstroom van voorraad etc. Deze wijzigingen zullen opgeslagen worden in de entiteit `ProductLogs`.

4.1.5 Users

Klanten van StilteAUBv hebben de mogelijkheid om een account aan te maken op de website. Met dit account hebben zij toegang tot de bestellingen en statussen. Deze gegevens worden opgeslagen in de entiteit `Users`.

5. Relational diagram (RD)



5.1 Toelichting RD

5.1.1 Employees

De medewerkers van StilteAUBv met een account staan als record in de tabel `Employees`. Elke record heeft een PRIMARY KEY `EmployeeID` en een FOREIGN KEY relatie zijn eigen table `Employees`. De FOREIGN KEY is `Created_By`.

5.1.2 Roles

De rollen die gebruikt worden voor de autorisatie op de website worden opgeslagen in de tabel `Roles`. Elke record heeft een PRIMARY KEY `RoleID`. Aangezien de rollen aangemaakt worden door employees heeft de tabel `Roles` een FOREIGN KEY relatie met de tabel `Employees`. Deze FOREIGN KEY is `Created_By` en bevat de `EmployeeID` van de employee die de rol heeft aangemaakt.

5.1.3 Employee-Roles

Een employee heeft tenminste 1 of meerde rollen. Om deze verbinding te kunnen maken word er gebruik gemaakt van een tussen tabel genaamd `Employees-Roles`. Elke record heeft een PRIMARY KEY genaamd `TableID` en 2 FOREIGN KEY relaties. De eerste relatie is met de table `Employees` en heet `EmployeeID`. De 2^e relatie is met de tabel `Roles` en heet `RoleID`.

5.1.4 Products

De operations medewerkers van StilleAUBv kunnen producten aanmaken doormiddel van het medewerkersportaal. Deze producten worden opgeslagen in de tabel `Products`. Elke record heeft een PRIMARY KEY genaamd `ProductID`. Omdat de producten aangemaakt worden door employees is er een FOREIGN KEY relatie met de tabel `Employees` en de kolom heet `Created_By`.

5.1.5 ProductLogs

De operations medewerkers kunnen de producten wijzigingen in de medewerkersportaal, denk hierbij aan het wijzigen van de prijs, voorraad of omschrijving. Deze wijzigingen/logging worden opgeslagen in de tabel `ProductLogs`. Elke record heeft een PRIMARY KEY genaamd `LogID`. En twee FOREIGN KEY relatie. De eerste relatie is met de tabel `Products`, de kolom heet `ProductID`. De tweede relatie is met de tabel `Employees`, de kolom heet `Modified_By`.

5.1.6 Users

Klanten die een account hebben aangemaakt krijgen een record in de tabel `Users`. Elke record heeft een PRIMARY KEY genaamd `UserID`.

5.1.7 OrderHeaders

Klanten hebben de mogelijkheid om bestellingen te plaatsen. Deze bestellingen worden weggeschreven in twee tabellen. De samenvatting van de bestelling inclusief verzend gegevens worden opgeslagen in de tabel `OrderHeaders`. Elke record heeft een PRIMARY KEY genaamd `HeaderID` en heeft een FOREIGN KEY relatie met de tabel `Users`. Deze kolom heet `Order_By`.

5.1.8 OrderLines

Zoals hierboven beschreven staat word een bestelling weggeschreven in twee tabellen. Naast de tabel `OrderHeaders` die een samenvatting is van de bestelling worden de details weggeschreven in de tabel `OrderLines`. Elke record heeft een PRIMARY KEY genaamd `LineID` en twee FOREIGN KEY relaties. De eerste relatie is met de tabel `OrderHeaders`, de kolom heet `OrderID`. De tweede relatie is met de tabel `Products`, de kolom heet `ProductID`.

6. Database realisatie (DDL)

6.1 Tables

6.1.1 Structuur table Employees

```
1 /*Creation query for Employees table*/
2 DROP TABLE IF EXISTS Employees;
3
4 CREATE TABLE `Employees` (
5     `EmployeeID` INT NOT NULL AUTO_INCREMENT,
6     `First_Name` VARCHAR(25) NOT NULL,
7     `Middle_Name` VARCHAR(25) NULL,
8     `Last_Name` VARCHAR(25) NOT NULL,
9     `Email` VARCHAR(50) NOT NULL,
10    `Password` VARCHAR(25) NOT NULL,
11    `Creation_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
12    `Created_By` INT NULL,
13    `Last_Active` DATETIME NULL,
14    `ACTIVE` BOOLEAN NOT NULL default 1,
15
16    PRIMARY KEY(`EmployeeID`),
17    CONSTRAINT `EmpCreated_By` FOREIGN KEY (`Created_By`) REFERENCES `Employees` (`EmployeeID`) ON DELETE SET NULL
18 );
19
```

In de CREATE-statement voor het maken van de tabel `Employees` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE-statement opgebouwd. Lijn 5 creëert de kolom `EmployeeID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. Ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 11 creëert de kolom `Creation_Date`. Deze heeft als standaardwaarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

Als laatste is de kolom `Created_By` die aangemaakt wordt op lijn 12 een FOREIGN KEY. Deze kolom refereert naar de kolom `EmployeeID` en dezelfde tabel. Deze relatie wordt gemaakt op rij 17. Mochten er een record verwijderd worden dan zal de waarde in `Created_By` geüpdatet worden naar NULL.

De overige velden zijn standaard VARCHAR, INT of DATETIME en hebben verder geen uitleg nodig.



Employees

6.1.2 Structuur table Roles

```
1  /*Creation query for Roles table*/
2  DROP TABLE IF EXISTS Roles;
3
4  CREATE TABLE `Roles` (
5      `RoleID` INT NOT NULL AUTO_INCREMENT,
6      `Name` VARCHAR(25) NOT NULL,
7      `Description` VARCHAR(255) NULL,
8      `Creation_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
9      `Created_By` INT NULL,
10
11     PRIMARY KEY(`RoleID`),
12     FOREIGN KEY(`Created_By`) REFERENCES `employees`(`EmployeeID`) ON DELETE SET NULL
13 );
14
```

In de CREATE-statement voor het maken van de tabel `Roles` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE-statement opgebouwd. Lijn 5 creëert de kolom `RoleID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. Ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 8 creëert de kolom `Creation_Date`. Deze heeft als standaardwaarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

Als laatste is de kolom `Created_By` die aangemaakt wordt op lijn 9 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID`. Deze relatie wordt gemaakt op rij 12. Mochten er een record verwijderd worden dan zal de waarde in `Created_By` geüpdatet worden naar NULL.

De overige velden zijn standaard VARCHAR-kolommen en hebben geen verdere uitleg nodig.



Roles

6.1.3 Structuur table Employees-Roles

```
1  /*Creation query for Employees-Roles table*/
2  DROP TABLE IF EXISTS `Employees-Roles`;
3
4  CREATE TABLE `Employees-Roles` (
5      `TableID` INT NOT NULL AUTO_INCREMENT,
6      `EmployeeID` INT NOT NULL,
7      `RoleID` INT NOT NULL,
8
9      PRIMARY KEY(`TableID`),
10     FOREIGN KEY(`EmployeeID`) REFERENCES `Employees`(`EmployeeID`) ON DELETE CASCADE,
11     FOREIGN KEY(`RoleID`) REFERENCES `Roles`(`RoleID`) ON DELETE CASCADE
12 );
13
```

In de CREATE-statement voor het maken van de tabel `Employee-Roles` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE-statement opgebouwd. Lijn 5 creëert de kolom `TableID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. Ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Omdat deze tabel dient als een verbinding tussen de tabellen `Employees` en `Roles` worden er 2 FOREIGN KEY relaties gelegd. Op rij 6 wordt de kolom `EmployeeID` aangemaakt die een FOREIGN KEY relatie heeft naar de tabel `Employees` en kolom `EmployeeID`. Op rij 7 wordt de kolom `RoleID` aangemaakt die een FOREIGN KEY relatie heeft naar de tabel `Roles` en kolom `RoleID`.

Voor beide FOREIGN KEY relaties geldt de regel ON DELETE CASCADE. Mochten er dus records verwijderd worden in de tabellen waar een relatie mee is dan zal de bijbehorende record ook verwijderd worden in de tabel `Employees-Roles`.



Employees-Roles

6.1.4 Structuur table Products

```
1  /*Creation query for Products table*/
2  DROP TABLE IF EXISTS `Products`;
3
4  CREATE TABLE `Products` (
5      `ProductID` INT NOT NULL AUTO_INCREMENT,
6      `Name` VARCHAR(50) NOT NULL,
7      `Description` VARCHAR(255) NULL,
8      `Stock` INT NOT NULL,
9      `Price` DECIMAL(65,2) NOT NULL,
10     `Creation_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
11     `Created_By` INT NULL,
12
13     PRIMARY KEY(`ProductID`),
14     FOREIGN KEY(`Created_By`) REFERENCES `Employees`(`EmployeeID`) ON DELETE SET NULL
15 );
16
```

In de CREATE-statement voor het maken van de tabel `Products` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE-statement opgebouwd. Lijn 5 creëert de kolom `ProductID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. Ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 9 creëert de kolom `Price`. Deze kolom heeft als datatype DECIMAL (65,2). 65 geeft aan hoeveel cijfers er voor de komma mogen staan en de 2 geeft de hoeveelheid cijfers achter de komma aan.

Lijn 10 creëert de kolom `Creation_Date`. Deze heeft als standaardwaarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

Als laatste is de kolom `Created_By` die aangemaakt wordt op lijn 11 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID`. Deze relatie wordt gemaakt op rij 14. Mochten er een record verwijderd worden dan zal de waarde in `Created_By` geüpdatet worden naar NULL.

De overige kolommen hebben als datatype VARCHAR of INT en hebben geen verdere uitleg nodig.



Products

6.1.5 Structuur table ProductLogs

```
1  /*Creation query for ProductLogs table*/
2  DROP TABLE IF EXISTS `ProductLogs`;
3
4  CREATE TABLE `ProductLogs` (
5      `LogID` INT NOT NULL AUTO_INCREMENT,
6      `ProductID` INT NOT NULL,
7      `Modified_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
8      `Modified_By` INT NULL,
9      `Name` VARCHAR(50) NULL,
10     `Description` VARCHAR(255) NULL,
11     `Stock` INT NULL,
12     `Price` DECIMAL(65, 2) NULL,
13
14     PRIMARY KEY(`LogID`),
15     FOREIGN KEY(`ProductID`) REFERENCES `Products`(`ProductID`) ON DELETE CASCADE,
16     FOREIGN KEY(`Modified_By`) REFERENCES `Employees`(`EmployeeID`) ON DELETE SET NULL
17 );
18
```

In de CREATE statement voor het maken van de tabel `ProductLogs` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE statement opgebouwd. Lijn 5 creëert de kolom `LogID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 6 creëert de kolom `ProductID`. Dit is een FOREIGN KEY die refereert naar de tabel `Products` en kolom `ProductID`. Deze relatie wordt gemaakt op rij 15. Voor deze relatie geldt de regel ON DELETE CASCADE. Mochten er dus records verwijderd worden in de tabel `Products` waar een relatie mee is dan zal de bijbehorende record ook verwijderd worden in de tabel `ProductLogs`.

Lijn 7 creëert de kolom `Modified_Date`. Deze heeft als standaard waarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

Als laatste is de kolom `Modified_By` die aangemaakt wordt op lijn 9 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID`. Deze relatie wordt gemaakt op rij 16. Mochten er een record verwijderd worden dan zal de waarde in `Created_By` geüpdatet worden naar NULL.



ProductLogs

6.1.6 Structuur table Users

```
1  /*Creation query for Users table*/
2  DROP TABLE IF EXISTS `Users`;
3
4  CREATE TABLE `Users` (
5      `UserID` INT NOT NULL AUTO_INCREMENT,
6      `First_Name` VARCHAR(25) NOT NULL,
7      `Middle_Name` VARCHAR(25) NULL,
8      `Last_Name` VARCHAR(25) NOT NULL,
9      `Email` VARCHAR(50) NULL,
10     `Phone_Number` VARCHAR(20) NULL,
11     `Password` VARCHAR(25) NOT NULL,
12     `Street` VARCHAR(100) NULL,
13     `House_Number` INT NULL,
14     `House_Number_Addition` VARCHAR(25) NULL,
15     `Zipcode` VARCHAR(10) NULL,
16     `City` VARCHAR(50) NULL,
17     `Creation_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
18
19     PRIMARY KEY(`UserID`)
20 );
21
```

In de CREATE statement voor het maken van de tabel `Users` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE statement opgebouwd. Lijn 5 creëert de kolom `UserID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 9 creëert de kolom `Creation_Date`. Deze heeft als standaard waarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

De overige kolommen hebben de datatypes VARCHAR of INT en hebben geen verdere uitleg nodig.



Users

6.1.7 Structuur table OrderHeaders

```
1  /*Creation query for OrderHeaders table*/
2  DROP TABLE IF EXISTS `OrderHeaders`;
3
4  CREATE TABLE `OrderHeaders` (
5      `HeaderID` INT NOT NULL AUTO_INCREMENT,
6      `Order_By` INT NULL,
7      `Total_Price` DECIMAL(65, 2) NOT NULL,
8      `Deliver_Adres` VARCHAR(255) NOT NULL,
9      `Deliver_Zipcode` VARCHAR(10) NOT NULL,
10     `Deliver_City` VARCHAR(50) NOT NULL,
11     `Creation_Date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
12     `Finished_Date` DATETIME NULL,
13     `Status` VARCHAR(25) NOT NULL,
14
15     PRIMARY KEY(`HeaderID`),
16     FOREIGN KEY(`Order_By`) REFERENCES `Users`(`UserID`) ON DELETE SET NULL
17 );
18
```

In de CREATE statement voor het maken van de tabel `OrderHeaders` wordt er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat wordt de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 wordt de CREATE statement opgebouwd. Lijn 5 creëert de kolom `HeaderID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 5 creëert de kolom `Order_By` en is een FOREIGN KEY die een relatie heeft met de tabel `Users` en kolom `UserID`. Deze relatie wordt gemaakt op lijn 16 en heeft de regel ON DELETE SET NULL. Mocht er een record verwijderd worden uit de tabel `Users` dan zal de waarde in `Order_By` geüpdatet worden naar NULL.

Lijn 11 creëert de kolom `Creation_Date`. Deze heeft als standaard waarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

Lijn 12 creëert de kolom `Finished_Date`. Deze heeft als standaard waarde CURRENT_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld wordt.

De overige kolommen hebben de standaard types VARCHAR, INT, DATETIME en DECIMAL en hebben geen verdere uitleg nodig.



OrderHeaders

6.1.8 Structuur table OrderLines

```
1 /*Creation query for OrderLines table*/
2 DROP TABLE IF EXISTS `OrderLines`;
3
4 CREATE TABLE `OrderLines`(
5     `LineID` INT NOT NULL AUTO_INCREMENT,
6     `HeaderID` INT NOT NULL,
7     `ProductID` INT NULL,
8     `Amount` INT NOT NULL,
9     `Line_Price` DECIMAL(65, 2) NOT NULL,
10
11     PRIMARY KEY(`LineID`),
12     FOREIGN KEY(`HeaderID`) REFERENCES `OrderHeaders`(`HeaderID`) ON DELETE CASCADE,
13     FOREIGN KEY(`ProductID`) REFERENCES `Products`(`ProductID`) ON DELETE SET NULL
14 );
```

In de CREATE statement voor het maken van de tabel `OrderLines` word er allereerst op lijn 2 gekeken of de tabel al bestaat. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `LineID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO_INCREMENT.

Lijn 6 creëert de kolom `HeaderID` en is een FOREIGN KEY die een relatie heeft met de tabel `Headers` en kolom `HeaderID`. Deze relatie word gemaakt op lijn 15 en heeft de regel ON DELETE CASCADE. Mocht er een record verwijderd worden uit de tabel `Headers` dan zullen alle records met de daarbij horende HeaderID ook verwijderd worden in de tabel `OrderLines`.

Lijn 7 creëert de kolom `ProductID` en is een FOREIGN KEY die een relatie heeft met de tabel `Products` en kolom `ProductID`. Deze relatie word gemaakt op lijn 16 en heeft als regel ON DELETE SET NULL. Mochten er records verwijderd worden uit de tabel `Products` dan zullen alle records met de daarbij horende ProductID geupdate worden met waarde null in de tabel `OrderLines`.

De overige kolommen hebben de datatypes VARCHAR, INT en DECIMAL en vergen geen verdere uitleg.



OrderLines

6.2 Triggers

6.2.1 Trigger LineTotalPrice

Beschreven in scenario 2 van hoofdstuk 3.2

```
1 /* Trigger for OrderLines Line_Price */
2
3 CREATE TRIGGER LineTotalPrice BEFORE INSERT
4 ON orderlines
5 FOR EACH ROW
6 SET NEW.Line_Price = NEW.Amount * (SELECT `prd`.`Price` FROM `Products` `prd` WHERE new.`ProductID` = `prd`.`ProductID`);
```

De trigger `LineTotalPrice` zorgt ervoor dat voor elke INSERT de kolom `Line_Price` in de tabel `OrderLines` de uitkomst krijgt van de kolommen `Amount` keer `Price` van de tabel `Products`. Dit

zorgt ervoor dat de berekening gedaan word vanuit de database en dat deze waarde niet vanuit de website gevuld hoeft te worden. Elke regel krijgt hierdoor een totaal bedrag die weer noodzakelijk is voor het totaal bedrag van de kolom `Total_Price` in de tabel `OrderHeaders`.



LineTotalPrice

6.2.2 Trigger UpdateTotalPriceHeader

Beschreven in scenario 2 van hoofdstuk 3.2

```
1 /* Trigger for updating the total price in OrderHeaders */
2
3 CREATE TRIGGER `UpdateTotalPriceHeader` AFTER INSERT ON `OrderLines` FOR EACH ROW
4 UPDATE `OrderHeaders` SET
5     `Total_Price` = (SELECT SUM(`Line_Price`) FROM `OrderLines` WHERE `HeaderID` = NEW.`HeaderID`)
6     WHERE `HeaderID` = NEW.`HeaderID`
7
```

De trigger `UpdateTotalPriceHeader` zorgt ervoor dat na elke INSERT in de tabel `OrderLines` er een UPDATE plaats vindt in de tabel `OrderHeaders`. De trigger telt alle waardes van de kolom `Line_Price` met dezelfde HeaderID van de tabel `OrderLines` bij elkaar op en update vervolgens de kolom `Total_Price` in de tabel `OrderHeaders` met de daarbij horden HeaderID.

Dit zorgt ervoor dat de berekening automatisch word gedaan vanuit de database en dat de waarde niet vanuit de website gevuld hoeft te worden.



UpdateTotalPriceHeader

6.3 Views

6.3.1 View Top10ProductsAmount

Beschreven in scenario 6 van hoofdstuk 3.4

```
1 /* View for top 10 ordered products based on amount*/
2
3 CREATE VIEW Top10ProductsAmount AS
4     SELECT
5         `prd`.`Name` as `ProductName`,
6         `prd`.`Description` as `ProductDescription`,
7         SUM(`ol`.`Amount`) as `TotalAmountSold`
8     FROM `OrderLines` `ol`
9     LEFT JOIN `Products` `prd` on `ol`.`ProductID` = `prd`.`ProductID`
10    GROUP BY
11        `ol`.`ProductID`
12    ORDER BY
13        SUM(`ol`.`Amount`) DESC LIMIT 10;
```

De view `Top10ProductsAmount` toont een overzicht van de bestelde producten op basis van de hoeveelheid verkochte exemplaren. In de CREATE statement word er een SUM gedaan op Amount om per ProductID de totale hoeveelheid te kunnen tonen.

Doormiddel van een LEFT JOIN word de tabel `Products` gekoppeld op basis van de `ProductID` uit `OrderLines` en de ProductID uit de tabel `Products`. Vervolgens kunnen de waardes `Name` en `Description` nu getoond worden die bij de `TotalAmountSold` horen.

Als laatste word de uitkomst gegroepeerd op `ProductID` en aflopend geordend om ervoor te zorgen dat het hoogste getal bovenaan komt te staan. Ook toont de uitkomst niet meer dan 10 records aangezien het hier gaat om de top 10.



Top10ProductsAmount

6.3.2 View Top10ProductsPrice

Beschreven in scenario 7 van hoofdstuk 3.4

```
1 /* View for top 10 ordered products based on Total Price*/
2
3 CREATE VIEW Top10ProductsPrice AS
4     SELECT
5         `prd`.`Name` as `ProductName`,
6         `prd`.`Description` as `ProductDescription`,
7         SUM(`ol`.`Line_Price`) as `TotalPriceSold`
8     FROM `OrderLines` `ol`
9     LEFT JOIN `Products` `prd` on `ol`.`ProductID` = `prd`.`ProductID`
10    GROUP BY
11        `ol`.`ProductID`
12    ORDER BY
13        SUM(`ol`.`Line_Price`) DESC LIMIT 10;
14
```

De View `Top10ProductsPrice` word op dezelfde manier opgebouwd als de view `Top10ProductsAmount`. Het verschil is echter dat deze een SUM doet op `Line_Price` en ook aflopend geordend word op `Line_Price`.



Top10ProductsPrice

6.3.3 View RevenuePerMonth

Beschreven in scenario 8 van hoofdstuk 3.4

```
1  /* Total Revenue Per Month */
2
3  CREATE VIEW `RevenuePerMonth` AS
4      SELECT
5          YEAR(`Finished_Date`) AS `Year`,
6          CONCAT(MONTH(`Finished_Date`), ' ', MONTHNAME(`Finished_Date`)) AS `Month`,
7          SUM(`Total_Price`) AS `Total`
8      FROM `OrderHeaders`
9      WHERE `Status` = 'Geleverd'
10     GROUP BY
11         CONCAT(MONTH(`Finished_Date`), ' ', MONTHNAME(`Finished_Date`)),
12         YEAR(`Finished_Date`)
13     ORDER BY
14         YEAR(`Finished_Date`), CONCAT(MONTH(`Finished_Date`), ' ', MONTHNAME(`Finished_Date`))
```

De view `RevenuePerMonth` toont de omzet vanuit de tabel `OrderHeaders` per jaar en maand.

Rij 5 gebruikt de functie YEAR om het jaar uit de kolom `Finished_date` te halen.

Rij 6 gebruikt de functie CONCAT om de waarden aan elkaar toe te voegen en vervolgens in een kolom te tonen. Ook word hier de functie MONTH gebruikt om de maand als nummer uit de kolom `Finished_date` te halen. Vervolgens word de functie MONTHNAME gebruikt om de naam van de maand uit de kolom `Finished_date` te halen. Dit word aan elkaar geplakt doormiddel van CONCAT met een spatie tussen de twee waardes.

De reden dat de nummer van de maand voor de naam van de maand word geplakt is om deze in de juiste volgorde te tonen.

Verder word de uitkomst gegroepeerd over de uitkomst van CONCAT en de uitkomst van YEAR. Als laatste word de uitkomst op volgorde gezet doormiddel van de uitkomst van YEAR en de uitkomst van CONCAT.



RevenuePerMonth

6.3.4 View Invoices

Beschreven in scenario 10 hoofdstuk 3.4

```
1  CREATE OR REPLACE VIEW `invoices` AS (
2      SELECT `users`.`First_Name`, `users`.`Last_Name`, `users`.`Email`, `orderheaders`.`Total_Price`, `orderheaders`.`Status`
3      FROM `users`
4      INNER JOIN `orderheaders` ON `users`.`UserID` = `orderheaders`.`Order_By`
5  )
```

Bovenstaande view toont gebruikersdata met een bestelling en de daarbij horende prijs.



invoices

6.4 Stored Procedure

6.4.1 Profielgegevens tonen

Beschreven in scenario 1 van hoofdstuk 3.3, worden eerst de gegevens getoond voordat deze kunnen worden aangepast.

```
1 DELIMITER //
```

```
2
```

```
3 CREATE PROCEDURE `ShowUserProfile`(  
4     IN  
5     `SP_ID` INT  
6 )  
7 BEGIN  
8     SELECT  
9         `First_Name`,  
10        `Middle_Name`,  
11        `Last_Name`,  
12        `Email`,  
13        `Phone_Number`,  
14        `Street`,  
15        `House_Number`,  
16        `House_Number_Addition`,  
17        `Zipcode`,  
18        `City`  
19     FROM `users`  
20    WHERE `UserID` = `SP_ID`;  
21 END //
```

Op basis van de `SP_ID` wat de UserID is kan met deze stored procedure de gebruikers data opgehaald worden vanuit de tabel `Users`.



ShowUserProfile

6.4.2 Status bestelling bekijken

Beschreven in scenario 2 hoofdstuk 3.3

```
1 -- Stored procedure, retrieve orders from user based on userID  
2 DELIMITER //
```

```
3 CREATE PROCEDURE `getOrdersUser`  
4     (IN `userID` INT)  
5 BEGIN  
6     SELECT `Creation_Date`, `HeaderID`, `Total_Price`, `Status`  
7     FROM `OrderHeaders`  
8     WHERE `Order_by` = `userID`;  
9 END //
```

Op basis van de userID kan met deze stored procedure de lijst met bestelling worden opgevraagd vanuit `OrderHeaders` en worden de gegevens: `Creation_Date`, `HeaderID`, `Total_Price`, `Status` per bestelling opgehaald en getoond.



getOrdersUser

6.4.3 Bestelling details bekijken

Beschreven in scenario 2 hoofdstuk 3.3

```

1 DELIMITER //
2 CREATE PROCEDURE `OrderDetails`
3     (IN `SP_HeaderID` INT)
4 BEGIN
5     SELECT
6         `prd`.`Name`,
7         `prd`.`Description`,
8         `ol`.`Amount`,
9         `ol`.`Line_Price`
10    FROM `OrderLines` `ol`
11   LEFT JOIN `Products` `prd` ON `prd`.`ProductID` = `ol`.`ProductID`
12   WHERE `ol`.`HeaderID` = `SP_HeaderID`;
13 END //
```

Op basis van de `SP_HeaderID` kan met deze stored procedure de details van het bestelling op regel niveau bekeken worden vanuit de tabel `OrderLines` in combinatie met `Products`.



OrderDetails

6.5 Stored Function

6.5.1 users_updateName

Deze functie is een voorbeeld voor het aanpassen van gegevens van een gebruiker. Dit kan worden uitgebreid met meerdere parameters indien nodig. Dit is beschreven in scenario 1 van hoofdstuk 3.3

```

1 DELIMITER //
2
3 CREATE OR REPLACE FUNCTION `users_updateName`(`id` INT(11), `FN` VARCHAR(25), `LN` VARCHAR(25))
4 RETURNS INT
5 DETERMINISTIC
6 BEGIN
7     UPDATE `users` SET `First_Name` = `FN`, `Last_Name` = `LN` WHERE `users`.`UserID` = `id`;
8     RETURN 1;
9 END //
```



users_updateName

De function 'users_updateName' is een functie die de voornaam en achternaam van een 'user' verandert op basis van een meegegeven userId.

Functie is bedoeld als voorbeeld om een update te schetsen, kan zo breed gemaakt worden als nodig op basis van de parameters die meegegeven kunnen worden aan de function.

Als je de functie wilt gebruiken, dan kan dat met onderstaand statement als voorbeeld:

```

1 SET @voorbeeld = `users_updateName`(1, 'voornaam', 'achternaam');
```

6.5.2 Toon jaren in dienst medewerker

Zoals beschreven in scenario 9 van hoofdstuk 3.4.

```
1 DELIMITER //
```

```
2 CREATE OR REPLACE FUNCTION `yearsEmployeeInCompany` (`empID` INT)
```

```
3     RETURNS INT
```

```
4 BEGIN
```

```
5     -- Variable voor het startJaar
```

```
6     DECLARE `startDate` INT;
```

```
7
```

```
8     SELECT YEAR(`Creation_Date`)
```

```
9     INTO `startDate`
```

```
10    FROM `employees`
```

```
11    WHERE `EmployeeID` = `empID`;
```

```
12
```

```
13    -- Bepaal huidig jaar en trek daar het startJaar van af
```

```
14    RETURN YEAR(CURRENT_TIMESTAMP()) - `startDate`;
```

```
15
```

```
16 END //
```

```
17
```

```
18 DELIMITER ;
```



yearsEmployeeInCo
mpany

Op basis van het `EmployeeID` kan per medewerker het aantal jaren in dienst worden opgeroepen.

Dit kan in de aanroep worden aangevuld met de gewenste gegevens zoals het `EmployeeID`.

Deze functie kan op de volgende manier worden gebruikt:

```
21 -- Voorbeeld aanroep
```

```
22 SELECT `EmployeeID`, yearsEmployeeInCompany (EmployeeID)
```

```
23 FROM `employees`;
```

7. Security (DCL)

7.1 Admin

Voor de toegang van de database zelf zal er een admin account aangemaakt worden die volledige rechten heeft om alle tabellen.

```
1 CREATE USER 'admin'@'%' IDENTIFIED BY 'P@ssw0rd';  
2 GRANT ALL PRIVILEGES ON stilteaubv.* TO 'admin'@'%';
```



Admin

De user `admin` word aangemaakt inclusief wachtwoord op lijn 1. Achter `admin`@ staat een % teken, dit betekent dat de user een connectie kan maken vanaf elke host. Als voorbeeld had hier ook localhost kunnen staan wat in dat geval zou betekenen dat de user alleen had kunnen inloggen via de server zelf.

Op lijn 2 word aangegeven dat de user `admin` volledige rechten heeft op de gehele database `stilteaubv`.

7.2 system

Vanuit de site zullen de SELECT, INSERT, UPDATE, en DELETE statements plaatsvinden. Voor de site zal een aparte user aangemaakt worden die deze functies mogen gebruiken op alleen de benodigde tabellen.

```
1 CREATE USER 'system'@'%' IDENTIFIED BY 'P@ssw0rd';  
2 GRANT SELECT, INSERT, UPDATE,DELETE ON stilteaubv.* TO 'system'@'%';
```



system

De user `system` word aangemaakt inclusief wachtwoord op lijn 1. Ook hier staat een % teken wat ervoor zorgt dat de user een connectie kan maken vanaf elke host.

Op lijn 2 worden de rechten SELECT, INSERT, UPDATE, en DELETE gegeven op alle tabellen in stilteaubv.

7.3 wachtwoorden

Om de veiligheid van wachtwoorden te waarborgen zijn er een aantal stappen die gezet (zullen) worden, wachtwoorden voor werknemers zullen minstens 15 karakters lang zijn met gebruik van tenminste hoofdletters en kleine letters. Wachtwoorden zullen gehasht worden met een custom salt op applicatieniveau, zodat er geen wachtwoorden in normale tekst bij de database terecht komen. Wachtwoorden zijn ook nooit onderdeel van een authenticatie, maar worden opgevraagd op basis van de username en vervolgens teruggegeven aan de applicatie en op applicatieniveau vergeleken met het ingevoerde wachtwoord.

8. Maintenance

De database wordt onderhouden via phpMyAdmin. Er zijn een aantal functies binnen phpMyAdmin die gebruikt kunnen worden voor het onderhouden van de database. Zie hieronder de instructies voor deze functies:

8.1 Instructie backup

1. Start **phpMyAdmin**.
2. Selecteer de database **`stilteaubv`**.
3. Druk op de knop **Export**.
4. Druk op de knop **Go**.
5. Het SQL bestand word gedownload en staat in je download folder. Verplaats dit bestand naar de backup server.

8.2 Instructie analyze table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database **`stilteaubv`**.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **analyze table**.
7. De analyse loopt nu en geeft de status per tabel aan.

8.3 Instructie Check table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database **`stilteaubv`**.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **check table**.
7. De analyse loopt nu en geeft de status per tabel aan.

8.3 Instructie optimize table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database **`stilteaubv`**.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **optimize table**.
7. De analyse loopt nu en geeft de status per tabel aan.

9. Installatie

9.1 Installatie handleiding

Voor de installatie van de database kan gebruik worden gemaakt van MySQL of MariaDB. Hoe deze software geïnstalleerd moet worden staat op de websites van de leverancier per besturingssysteem uitgelegd. Tevens kan er een webapplicatie worden geïnstalleerd voor de database administratie via een webbrowser, als voorbeeld wordt phpMyAdmin gebruikt. Hoe phpMyAdmin geïnstalleerd moet worden staat beschreven op de website van de leverancier.

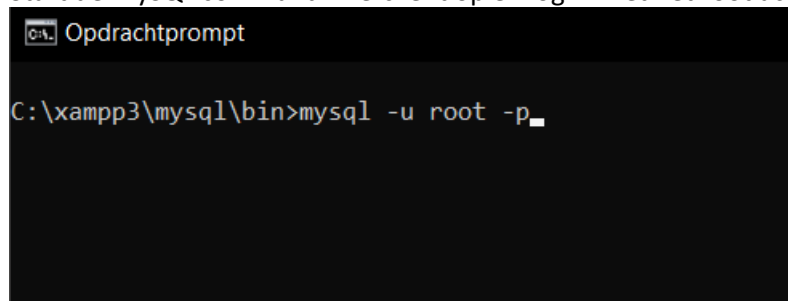
9.1.1 Installatiebestand uitpakken

Het database installatiebestand wordt aangeleverd in het bestand 'StilteAubBV.zip' en deze moet worden uitgepakt op een locatie naar keuze. Na het uitpakken is het volgende bestand aanwezig:

- StilteAuBV.sql

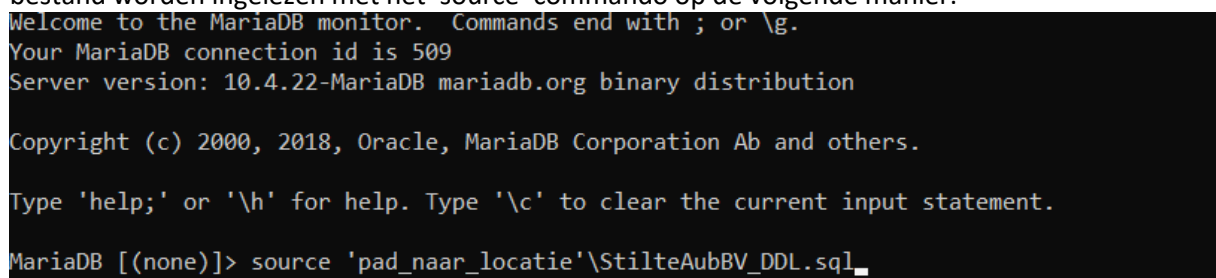
9.1.2 Installatie via de command line

Start de MySQL command-line-client op en log in met het root account via het volgende commando:



```
C:\xampp3\mysql\bin>mysql -u root -p
```

Indien een wachtwoord is ingesteld voor het root account, voert u deze in. Na het inloggen kan het bestand worden ingelezen met het 'source' commando op de volgende manier:



```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 509
Server version: 10.4.22-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> source 'pad_naar_locatie'\StilteAubBV_DDL.sql
```

Dit geeft het volgende commando:

- Source 'pad_naar_locatie'\StilteAuBV.sql

9.1.3 Importeren via phpMyAdmin

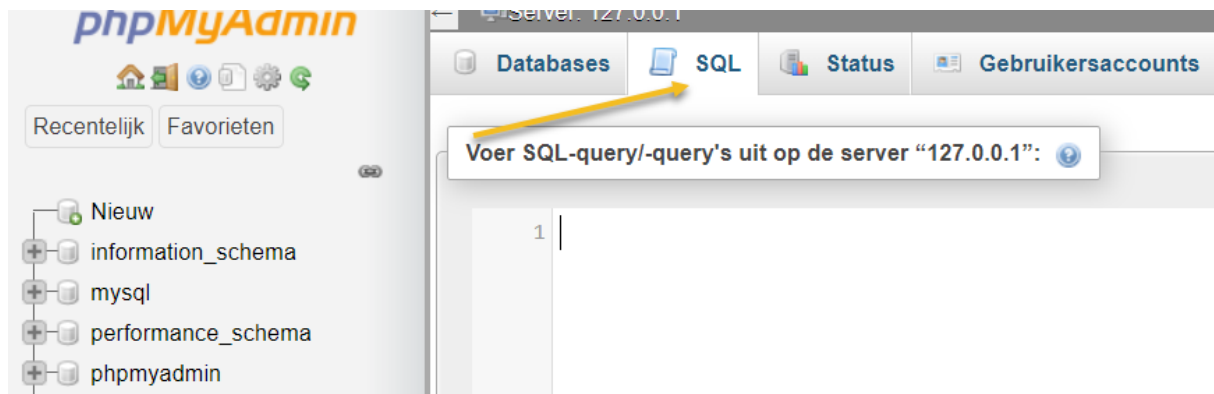
In phpMyAdmin kan via de optie 'Importeren' een bestand worden ingelezen door via 'Bestand Kiezen' het bestand te selecteren:



Druk daarna op 'Starten' op de query uit te voeren.

9.1.4 Query handmatig kopiëren en plakken via phpMyAdmin

Na het openen van phpMyAdmin kan via de optie SQL, een query worden ingelezen:



Na het selecteren van deze optie, kan de inhoud van een bestand worden ingelezen. Dit kan worden gedaan door de bestanden te openen in een tekstverwerker zoals Notepad++, de inhoud selecteren en vervolgens plakken in phpMyAdmin. Met de knop *Starten*, wordt de query doorgevoerd.

9.2 Insert data

9.2.1 Insert Employees

```

1 /*Data for Employees table*/
2
3 INSERT INTO `Employees` (`First_Name`, `Middle_Name`, `Last_Name`, `Email`, `Password`, `Created_By`, `ACTIVE`)
4 VALUES
5 ('Wesley', NULL, 'Geboers', 'w.geboers@student.avans.nl', 'P@ssw0rd@2022!', NULL, 1),
6 ('Marcel', NULL, 'Forman', 'm.forman@student.avans.nl', 'P@ssw0rd@2022!', 1, 1),
7 ('Bart', NULL, 'Frijters', 'bjal.frijters@student.avans.nl', 'P@ssw0rd@2022!', 1, 1),
8 ('Thomas', NULL, 'Daane', 'trbl.daane@student.avans.nl', 'P@ssw0rd@2022!', 1, 1),
9 ('Sanel', 'van den', 'Bogert', 'avd.bogert@student.avans.nl', 'P@ssw0rd@2022!', 1, 1),
10 ('Lysette', NULL, 'Schipper', 'l.schippers@student.avans.nl', 'P@ssw0rd@2022!', 1, 1)

```



InsertEmployees

9.2.2 Insert Users

```
1 /*Data for Users table*/
2
3 INSERT INTO `Users` (`First_Name`, `Middle_Name`, `Last_Name`, `Email`, `Password`, `Phone_Number`, `Street`, `House_Number`, `House_Number_Addition`, `Zipcode`, `City`)
4 VALUES
5 ('Hans', NULL, 'Poppelaars', 'HansPoppelaars@teleworm.us', 'gohm6Ahquae', '06-19066719', 'Midscheeps', 65, NULL, '8899 BT', 'Vlieland'),
6 ('Gerbert', 'van', 'Muijen', 'GerbertvanMuijen@rhyta.com', 'Oosh4eif', '06-37008585', 'Irenestraat', 32, NULL, '9744 CV', 'Groningen'),
7 ('Marjolijn', NULL, 'Steverink', 'MarjolijnSteverink@rhyta.com', 'tohKeIIae', '06-13191051', 'Blauwe Pan', 56, '-A', '1317 AP', 'Almere'),
8 ('Faik', NULL, 'Penterman', 'FaikPenterman@dayrep.com', 'OoD1ahch4ee', '06-75750105', 'Ingenieur Lelystraat', 128, NULL, '5142 AM', 'Waalwijk'),
9 ('Marike', NULL, 'Vlieland', 'MarikeVlieland@jourrapide.com', 'ciez8egeila', '06-48991805', 'Jacobshoeve-Erf', 153, NULL, '3931 RZ', 'Woudenberg')
```



InsertUsers

9.2.3 Insert Roles

```
1 /*Data for Roles table*/
2
3 INSERT INTO `Roles` (`Name`, `Description`, `Created_By`)
4 VALUES
5 ('Directie', 'Toegang tot medewerkersportaal (lees rechten)', 1),
6 ('IT', 'Toegang tot medewerkersportaal en de database (lees en schrijf rechten)', 1),
7 ('Operations', 'Toegang tot medewerkersportaal (lees en schrijf rechten)', 1),
8 ('Marketing', 'Toegang tot medewerkersportaal (lees en schrijf rechten)', 1),
9 ('Administratie', 'Toegang tot medewerkersportaal (lees rechten)', 1)
```



InsertRoles

9.2.4 Insert Employees-Roles

```
1 /*Data for Employees-Roles table*/
2
3 INSERT INTO `Employees-Roles` (`EmployeeID`, `RoleID`)
4 VALUES
5 (1, 2),
6 (2, 1),
7 (3, 3),
8 (4, 4),
9 (5, 5),
10 (5, 5)
```



InsertEmployees-Roles

9.2.5 Insert Products

```
1 /*Data for Products table*/
2
3 INSERT INTO `Products` (`Name`, `Description`, `Price`, `Stock`, `Created_By`)
4 VALUES
5 ("60's Package", 'Jaren 60 thema met 10 headsets en 1 zender', 64.99, 12, 1),
6 ("70's Package", 'Jaren 70 thema met 10 headsets en 1 zender', 64.99, 15, 1),
7 ("80's Package", 'Jaren 80 thema met 10 headsets en 1 zender', 69.99, 6, 1),
8 ("90's Package", 'Jaren 90 thema met 10 headsets en 1 zender', 69.99, 18, 1),
9 ("00's Package", 'Jaren 00 thema met 10 headsets en 1 zender', 74.99, 3, 1)
```



InsertProducts

9.2.6 Insert ProductLogs

```
1 /*Data for Productlogs table*/
2
3 INSERT INTO `Productlogs` (`Name`, `Description`, `Price`, `Stock`, `Modified_By`, `ProductID`)
4 VALUES
5 (NULL, NULL, NULL, NULL, 3, 2),
6 (NULL, NULL, 60.99, NULL, 3, 1),
7 (NULL, NULL, NULL, 10, 3, 3),
8 (NULL, NULL, NULL, 20, 3, 2),
9 (NULL, NULL, NULL, 15, 3, 5)
```



InsertProductLogs

9.2.7 Insert OrderHeaders

```
1 /*Data for OrderHeaders table*/
2
3 INSERT INTO `OrderHeaders` (`Order_By`, `Deliver_Adres`, `Deliver_Zipcode`, `Deliver_City`, `Status`, `Finished_Date`)
4 VALUES
5 (1, 'Midscheeps 65', '8899 BT', 'Vlieland', 'In behandeling', NULL),
6 (3, 'Blauwe Pan 56-A', '1317 AP', 'Almere', 'Geleverd', NOW())
7
```



InsertOrderHeaders

9.2.8 Insert OrderLines

```
1 /*Data for OrderLines table*/
2
3 INSERT INTO `OrderLines` (`HeaderID`, `ProductID`, `Amount`)
4 VALUES
5 (1,2,1),
6 (1,3,3),
7 (1,5,1),
8 (2,4,1)
```



InsertOrderLines

10 Uitwerking (DML)

De site zal verschillende scenario's moeten kunnen uitvoeren op de database. Deze scenario's zijn hieronder uitgeschreven.

10.1 Edit user

De klant moet de mogelijkheid hebben om zijn of haar gegevens in te zien en deze te wijzigen. In dit voorbeeld word er gebruik gemaakt van de user Marike Vlieland met UserID 5 en wijzigen we het email adres van Marike.

In de onderstaande query word de SELECT statement gebruikt om de data van Marike op te halen. Om te bepalen dat het om Marike gaat word er gebruik gemaakt van WHERE `UserID` = 5.

```
1 SELECT
2     `First_Name`,
3     `Middle_Name`,
4     `Last_Name`,
5     `Email`,
6     `Phone_Number`,
7     `Street`,
8     `House_Number`,
9     `House_Number_Addition`,
10    `Zipcode`,
11    `City`
12 FROM `users`
13 WHERE `UserID` = 5;
```



SelectUser

In de onderstaande query heeft Marike bepaalt dat het email adres gewijzigd moet worden. Dit zal voor een UPDATE statement zorgen waarin het email adres gewijzigd word en de user bepaald word doormiddel van WHERE `UserID` = 5.

```
1 UPDATE `Users`
2 SET `Email` = 'Marike123@gmail.com'
3 WHERE `UserID` = 5
```



UpdateUser

10.2 Show orders

De klant heeft in het klantenportaal ook de mogelijkheid om zijn of haar bestellingen in te zien. Voor dit voorbeeld gebruiken we de user Hans met UserID 1. Voor het ophalen van de bestellingen is er een stored procedure geschreven genaamd `getOrdersUser`. In de onderstaande query is te zien hoe deze aangeroepen kan worden:

```
1 SET @user =1; CALL `getOrdersUser`(@user);
```

10.3 Show order details

In hoofdstuk 10.2 Show orders worden de bestellingen van Hans getoond op de tabel `OrderHeaders`. Dit levert als resultaat het bestelnummer, aflevergegevens en de status op. De gebruiker moet ook de mogelijkheid hebben om de details op regel niveau te bekijken. Voor het ophalen van deze details is een stored procedure geschreven genaamd `OrderDetails`. In de onderstaande query is te zien hoe deze aangeroepen kan worden:

```
1 SET @header= 1; CALL `OrderDetails`(@header);
```

10.4 Edit products

Een operations medewerkers moet de mogelijkheid hebben om producten te wijzigen. In dit voorbeeld wijzigen we de voorraad en prijs van het product 90's Package met ProductID 4. De wijzigingen zullen doorgevoerd worden door de user Bart met EmployeeID 3.

In onderstaande query word er gebruik gemaakt van de UPDATE statement waarin de voorraad en prijs gewijzigd word. Ook zorgt het wijzigen van een product voor een INSERT op de tabel `ProductLogs`.

```
1 START TRANSACTION;
2
3 UPDATE `products`
4 SET `Price` = 59.99, `Stock` = 20
5 WHERE `ProductID` = 4;
6
7 INSERT INTO `Productlogs` (`Name`, `Description`, `Price`, `Stock`, `Modified_By`, `ProductID`)
8 VALUES (NULL, NULL, 59.99, 20, 3, 4);
9
10 COMMIT;
```



UpdateProduct

10.5 Delete user role

Een IT medewerker moet de mogelijkheid hebben om rechten van een andere gebruiker af te nemen. In dit voorbeeld nemen wij de rol administratie met `RoleID` 5 weg bij Sanel met `EmployeeID` 5.

In onderstaande query word er gebruik gemaakt van de DELETE statement om de rol weg te halen.

```
1 DELETE FROM `Employees-Roles` WHERE `EmployeeID` = 5 and `RoleID` = 5
```