EBOOST

Plan van aanpak
Eboost database

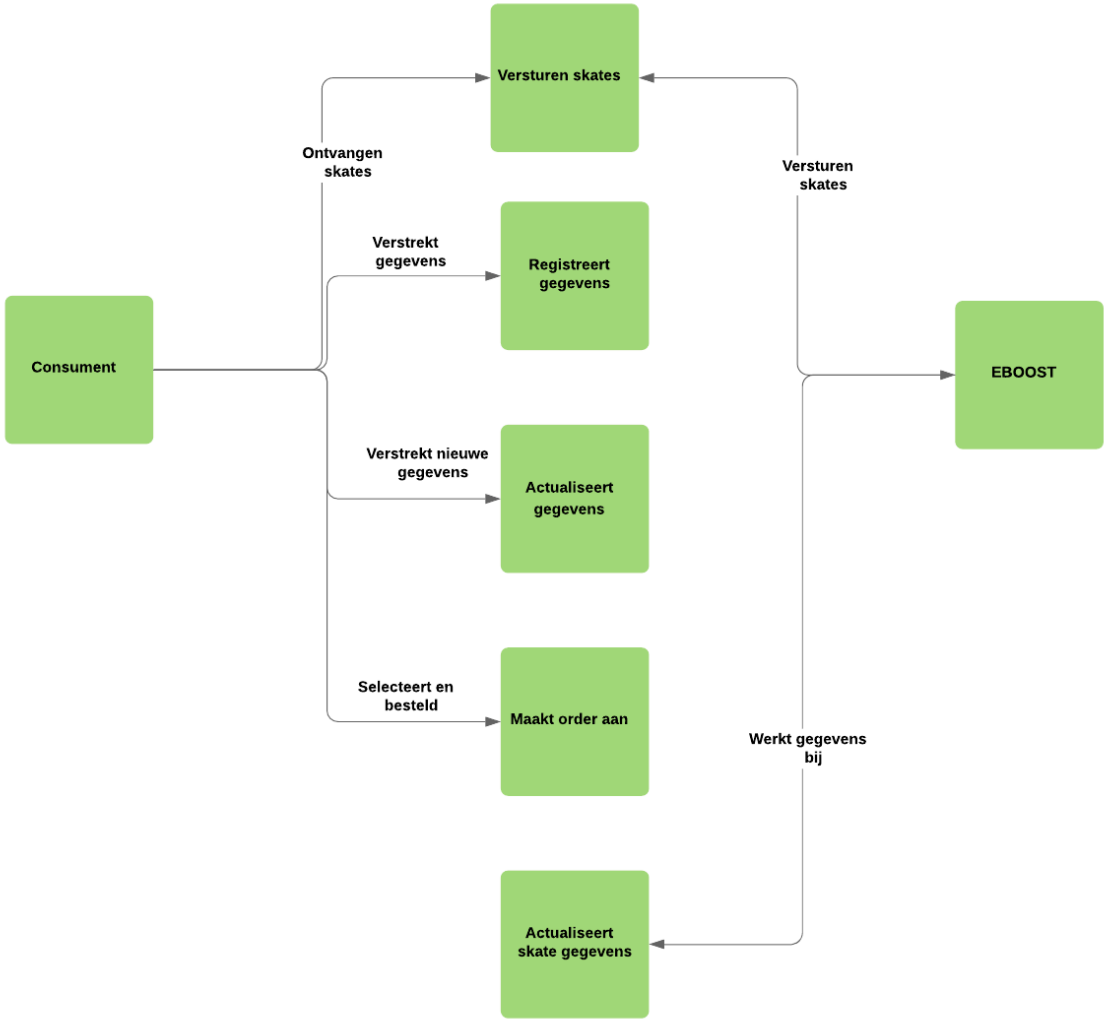
Inhoudsopgave

1 ALGEMENE BESCHRIJVING

De database applicatie voor Eboost heeft als hoofddoel het voor mensen mogelijk maken om skates van Eboost te kopen in drie stappen, te weten:

1. De klant bezoek de website;
2. De klant stelt zijn persoonlijke Eboost skate samen;
3. De klant vult zijn gegevens in en maakt de order definitief.

In het een 'use case diagram' hieronder is het visueel afgebeeld. De use case is vooral klant georiënteerd. Het gedeelte waarin Eboost de order ontvangt, verwerkt en afhandelt is vereenvoudigd tot het model hieronder.



Afbeelding 1 'Use case diagram'

Dit hoofddoel wordt uitgewerkt in het Klanten systeem.

De database applicatie kent nog drie andere systemen die ondersteunend zijn aan het hoofddoel, de system zijn:

1. Het klantensysteem waarin de klant zijn gegevens kan wijzigen;
2. Het beheersysteem waarin op verschillende niveaus 's onderhoud en beheer van de database kan worden uitgevoerd;
3. Het loginsysteem die de authenticatie en autorisatie verzorgt van de verschillende systemen.

2 AANPAK

2.1 PLANNING

Hieronder volgt de planning van het project alsmede de onderkende deeltaken in het geheel.

Start	Eind	Actie	Eigenaar
5-11-19	12-11-19	Start van de nieuwe fase, inlezen opdracht	
12-11-19		Besproken nieuwe fase en verdelen opdracht in GitHub	
19-11-19		Opstellen initiële documentatie	
12-11-19	19-11-19	Maken van eisen en wensen en ERD van Winkel systeem	
12-11-19	19-11-19	Maken eisen en wensen en ERD van Beheer systeem	
12-11-19	19-11-19	Maken van eisen en wensen en ERD van Klanten systeem	
12-11-19	19-11-19	Maken van eisen en wensen en ERD van Login systeem	
26-11-19	26-11-19	Samenvoegen in fysieke sessie van ERD en bespreken voortgang	
27-11-19	3-12-19	Visio beschikbaar	
27-11-19	27-11-19	PvA bijwerken en Klant systeem ERD en RD in Visio	
28-11-19	28-11-19	Beheer systeem bijvoegen in het ERD en RD	
29-11-19	29-11-19	Login systeem bijvoegen in het ERD en RD	
30-11-19	30-11-19	Winkel systeem ERD en RD e-mailen naar x	
1-2-19	2-12-19	Winkelsysteem ERD en RD samenvoegen in Visio	
3-12-12	3-12-12	PvA document bijwerken met ERD en RD voor evaluatie	
3-12-12	3-12-12	Evaluatie Willem	
4-12-19	28-12-19	Verdelen taken na evaluatie x, Taken verwerken. Tussendoor StaVaZa bespreken.	
29-12-19	29-12-19	Alles samenvoegen, documenteren en evalueren, optie tot inleveren bespreken	
30-12-19	30-12-19	Openstaande punten uit de evaluatie verwerken	
	31-12-19	Deadline inleveren	
	14-1-20	Assessment	

2.2 Afspraken

Afspraken die voortvloeien uit de besprekingen worden hieronder genoteerd.

Afspraak	Motivatie
Engels in code en modellen	Omdat de codetaal ook Engels is.
MS Visio voor modellen Sleutels in tabellen heten ID's	Beschikbaar voor alle proftaak leden Zodat het eenduidig is om aan te roepen
SQL-code in hoofdletters	Opgelegd door Avans
Cursieve tekst is invoer	Cursieve tekst is invoer, dus niet letterlijk nemen (Avans)
Tabelnaam meervoud	Eventueel tussen `backquotes` (`backticks`): Sporters, `Sporters` (Avans)
Kolomnaam enkelvoud	Met een hoofdletter, eventueel tussen `backquotes`: Club, `Club` (Avans)
Tekst	Datum, tijd: tussen enkele aanhalingstekens 'Joop', '2017-03-30', '13:30' (Avans)
Getallen	Getallen: geen aanhalingstekens nodig 42, `Bedrag` * 10 (Avans)
ERD Sheet Avans	De sheet voor Visio is door Avans verstrekt en staat op de onze share
RD Sheet	Visual Paradigm 16.0
Primaire sleutel	Een sterretje geeft het attribuut of de groep attributen aan die het entiteittype uniek identificeert.
Attributen	Attributen van de entiteit staan onder de scheidingslijn.
Attributen	Naamgeving van attributen in CamelCase.
GitHub	Voor versiebeheer werken we met GitHub op https://github.com/basvreeken/Eboost

2.3 EISEN EN WENSEN

E/W	*Specificatie*	*Toelichting*	*Implementatie*
		Algemeen	
E	Layout gebaseerd op Bootstrap	Zo wordt de beheerzijde consistent en helder weergegeven.	Dit volgt wanneer het front-end wordt gebouwd.
W	Layout gestyled volgens Eboost	Zodat de look & feel van Eboost ook in de beheerzijde terugkomt.	Dit volgt wanneer het front-end wordt gebouwd.
E	Formulier input wordt gecontroleerd voor uitvoer in PHP.	Om ongewenste effecten van kwaadwillende te voorkomen.	Dit volgt wanneer het front-end wordt gebouwd.
Winkel systeem			
E	Klant kan gewenste skates selecteren	De klant kan op de website zijn gewenste selecteren en bestellen: 1. Kleur 2. Maat 3. Wielkleur	Implementatie #4 In het webformulier verschijnen pull down menu's. Deze menu's worden gevuld met de inhoud uit de tabellen kleur, maat en wielkleur. De gekozen kleuren en maat worden ingevoerd. (Onderdeel van implementatie #5)
E	Klant kan orders plaatsen	Na het aanpassen van de skates kan de klant inloggen om dan een order te plaatsen. Bij het order plaatsen zijn de volgende punten nodig: 1. Adres 2. Postcode 3. Huisnummer 4. Stad 5. Land 6. Hoeveelheid 7. Gekozen kleur 8. Gekozen wielkleur 9. Gekozen maat 10. Orderbedrag	Implementatie #5 Klant zorgt voor de verzendgegevens en vult deze in. Met behulp van de INSERT_INTO wordt dit ingevoerd in de database. Met een trigger wordt het totale orderbedrag berekend. (Vervolg van implementatie #4)
E	Klant kan geplaatste order inzien	Doormiddel door in te loggen kan de klant zijn order inzien. Ordergegevens: 1. Verzendgegevens 2. Orderbedrag 3. Hoeveelheid 4. Datum/tijd Zo is de orderstatus te zien: 1. In behandeling 2. Geannuleerd 3. Voltooid	Implementatie #6 Met een behulp van een view kan de klant zijn ordergegevens oproepen.

Beheersysteem			
E	Admin moet rechten en toegang voor roles kunnen toewijzen	De admin role moet verschillende users aan kunnen maken, met daarin verschillende rechten opgenomen per user	Implementatie (DLC) Hoofdstuk 7 Admin maakt users aan, daarbij verleent de admin rechten aan de users
E	Log voor gewijzigde gegevens	De HRM-afdeling willen een log bijhouden wanneer er gegevens bewerkt worden bij de employees gegevens.	Doormiddel van een trigger worden wijzigingen aan de tabel employees geregistreerd Implementatie hoofdstuk 6.9 Triggers Implementatie #6.9.2 Implementatie #6.9.3
E	Gegevens in kunnen zien van het aantal employees per rol	Employees met de role manager willen up to date aantallen hebben van het aantal employees binnen de onderneming. Wanneer een afdeling onderbemand raakt kunnen zij gericht werven voor personeel.	Doormiddel van een view krijg je een overzicht van het aantal medewerkers met een specifieke role
Customer systeem			
E	De klant moet zijn persoonlijke gegevens kunnen wijzigen.	Eboost slaat tijdens de bestelling de volgende gegevens van de klant op: 1. Naam 2. Adres 3. Postcode 4. Plaats 5. E-mailadres 6. Telefoonnummer 7. Wachtwoord De gegevens moet de klant kunnen inzien en wijzigen. De gegevens mogen niet verwijderd worden. Het wachtwoord is niet verplicht, maar wel versleuteld.	Voor DML zie hoofdstuk 6.1. Versleuteling van het wachtwoord vindt plaats binnen PHP en is hier nog niet zichtbaar. Het wijzigen van het wachtwoord is nog niet geïmplementeerd. Voor uitleg zie Implementatie #1
E	De klant moet zijn order gegevens per order kunnen opvragen.	De klant ziet een lijst met geplaatste orders en kan door het klikken op een item de order bekijken en hier staat op: 1. Datum 2. Tijd 3. Verzendgegevens 4. Producten met specificaties als orderregels 5. Orderbedrag	Voor DML zie Hfdst 6. Stored Procedure voor de lijst van Ordernummer, Orderdatum en Orderstatus per klant. Implementatie #9. Voor uitleg zie Implementatie #2
W	De klant kan financiële informatie opvragen	De klant kan het totaalbedrag van zijn orders bekijken. De klant heeft hierbij aanvullende keuzes, te weten: 1. Filter periode van/tot	Voor DML zie Hfdst 6.

		2. Inclusief/exclusief BTW 3. Inclusief/exclusief verzendkosten	Stored Procedure voor het berekenen van een totaalbedrag per periode met een functie voor het berekenen van de BTW. Verzendkosten worden op dit moment niet aan de klant gerekend. Implementatie #3
Security			
E	De Admin rol moet de juiste rechten hebben.	De Admin role moet toegang hebben tot alles in de database en hier wijzigingen inbrengen: • Customers • Orders • Ordered_skates • Employees • Skates • SkateColors • WheelColors • Sizes	Zie hoofdstuk 7.1 "Rechten Admin"
E	De Manager rol moet de juiste rechten hebben.	De Manager role moet toegang hebben tot de volgende tabellen rechten hebben: • Employees • Ordered_skates	Zie hoofdstuk 7.2 "Rechten Manager"
E	De CustomerRelation rol moet de juiste rechten hebben.	De CustomerRelation role moet toegang hebben tot de volgende tabellen rechten hebben: • Orders • Customers	Zie hoofdstuk 7.3 "Rechten CustomerRelation"
E	De Production rol moet de juiste rechten hebben.	De Production role moet toegang hebben tot de volgende tabellen rechten hebben: • Skates • Ordered_skates • SkateColors • WheelColors • Sizes	Zie hoofdstuk 7.4 "Rechten Production"
E	De Customer rol moet de juiste rechten hebben.	De Customer role moet toegang hebben tot de volgende tabellen rechten hebben: • Orders • Ordered_skates • Customers	Zie hoofdstuk 7.5 "Rechten Customer"
E	De Guest rol moet de juiste rechten hebben.	De Customer role moet toegang hebben tot de volgende tabellen rechten hebben: • Skates • Ordered_skates • SkateColors • WheelColors • Sizes	Zie hoofdstuk 7.6 "Rechten Guest"

3 FUNCTIONELE BENODIGDHEDEN

Om te voldoen aan de eisen en wensen van de stake holders is het belangrijk om in kaart te brengen welke scenario's er zich zullen voor doen. Wat verwacht men wanneer de gebruiker bijvoorbeeld probeert in te loggen of zijn maat selecteert. Op basis van de eerdergenoemde eisen en wensen worden deze scenario's geschreven. Met behulp van 'acceptance criteria' worden de scenario's volgens de GWT-format geschreven, ook bekend als *'Given/ When/ Then'.* (Altexsoft, 2018).

3.1 ALGEMEEN

Voor het functioneren van de database is er gebruik gemaakt van Xampp. Deze applicatie bevat de MySQL server en de Webserver: Apache. Dit zijn de benodigdheden voor het gebruik maken van de Eboost Database. Hierin vinden wij onder de Webserver draaien we PHPMyadmin wat een beheerapplicatie is voor MySQL. Zowel PHPmyAdmin als de Console van MySQL kan gebruikt worden voor het beheren.

Scenario 1 Beheren van de de Database
User story: De Admin kan via PHPMyAdmin de database aanpassen.
Scenario: De database aanpassen.
Given: De Admin zal moeten navigeren nar PHPMyAdmin.
When*:* de Admin een tabel uit de Eboost database selecteert.
And*:* kunnen de regels hierin aangepast worden via de GUI, of met Queries.
Then*:* Deze worden vervolgens verwerkt door de Database: MySQL.

3.2 WINKELSYSTEEM

Omdat er volgens de Avans opdrachten geen PHP of front-end gemaakt worden. Zijn betrekkende informatie over front-end fictief geschreven.

Scenario 1 keuze maat en kleuren
User story: **de** gebruiker kan de kleuren kiezen voor de skates en wielen en zijn maat.
Scenario: kiezen tussen maat en kleuren van skate en wielen.
Given: de gebruiker zal op de website moeten navigeren naar de skate pagina.
When: de gebruiker heeft de via een drop down menu de keuze om zijn maat en gewenste kleuren te selecteren.
And: gebruiker heeft zijn keuzes geselecteerd via de drop down menu. Bestel knop zal groen gekleurd zijn, zodat men op de 'bestel knop' kan klikken.
Then: het system verwerkt volgens een INSERT INTO de geselecteerde gegevens in een order tabel.

Scenario 2 het bestellen van de skates
User story: na het selecteren van de gewenste keuzes en op bestel te hebben geklikt. Zal de gebruiken zijn ordergegevens willen ingeven.
Scenario: de gebruiker vult zijn order gegevens in.
Given: de gebruiker volgt op de bestelpagina de stappen om de volgende gegevens in te geven:

- Adres
- Postcode
- Huisnummer
- Stad
- Land

When: de gebruiker selecteert 'nu bestellen' op de website
Then: het system zal via een INSERT INTO ingevulde gegevens invullen in een order tabel.

3.3 BEHEERSYSTEEM

Scenario 1 Aanmaken users door admin
User story: De admin moet verschillende type accounts aanmaken die tot verschillende delen van de DB-toegang hebben.
Scenario: Om ongeautoriseerde toegang tot gegevens uit de DB te voorkomen is de admin belast met het creëren van verschillende type users
Given: Personen die toegang hebben tot de DB kunnen wijzigingen verrichten.
When*:* door onwetendheid en/of verkeerde handelingen kunnen er gegevens per ongeluk angest of verwijderd worden
And*:* Deze wijzigingen zijn niet makkelijk om te herstellen of om op te sporen
Then*:* Door de toegewezen rechten hebben de gebruikers alleen toegang voor de voor hun relevante gegevens.

Scenario 2 Log voor wijzigingen in employeebestand
User story: Logbestand voor mutaties aan het personeelbestand.
Scenario: De HRM-afdeling geeft aan de intakeprocedure oor medewerkers niet adequaat is er worden vachterafeel mutatis doorgevoerd daarbij de HRM-deling aangeeft dat het extra (onnodig) werk oplevert.
Given: Er worden mutaties verricht aan het personeelsbestand door de HRM.
When*:*Dit zijn extra handelingen achteraf, die er voor kunnen zorgen dat de DB vervuild kan raken.
And*:* Er wordt teen logbestand ontwikkeld die alle wijzigingen registreert aan het employee tabel
Then*:*Is het inzichtelijk hoeveel wijzigen er gedaan worden aan het Employee data.

Scenario 3 overzicht aantal werknemers per afdeling
User story: Management wilt in op tijd in kunne spleen op personeelstekort/overschot
Scenario: Het management wilt overzichten hebben van het aantal personeelsleden met een specifieke functie binnen de organisatie.
Given: Personeelsverloop kan in de huidige tijd snel gaan.
When*:* Wanneer er een personeelstekort ontstaat kan dit het primair proces bedreigen.
And*:* Hierdoor loopt de organisatie kans dat zij klanten verliezen
Then*:* Door het management te voorzien op actuele cijfers binnen de onderneming kan er snel en adequaat gehandeld worden bij een personeelstekort of overschot

3.4 CUSTOMERSYSTEEM

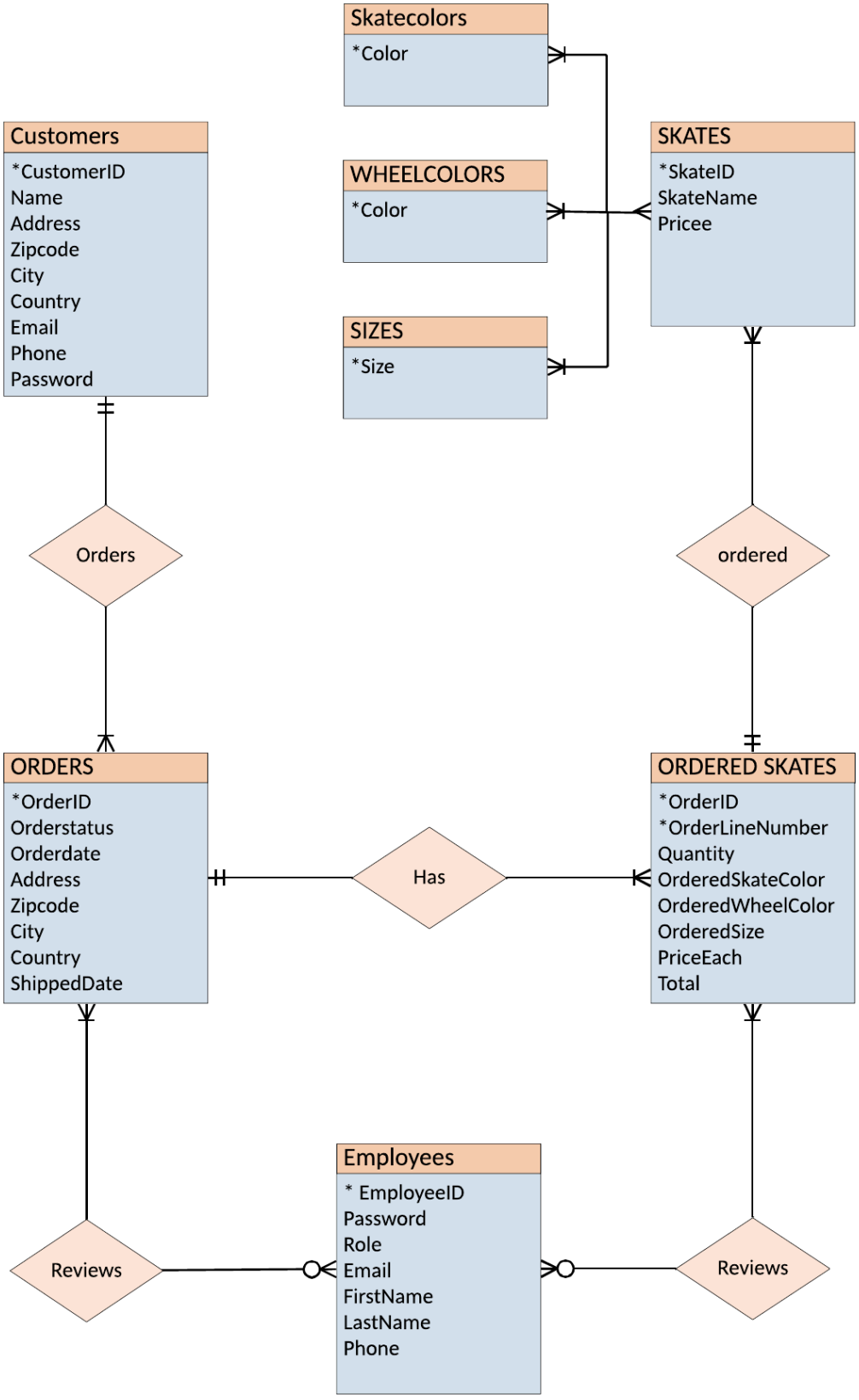
Scenario 1 De klant moet zijn persoonlijke gegevens kunnen wijzigen
User story: Er zijn klanten met en zonder accounts. Deze user story is van toepassing op klanten met accounts.
Scenario*:* De klant is ingelogd op de website in het klanten gedeelte en wil zijn persoonlijke gegeven wijzigen. Hij ziet een formulier waarop in de velden de huidige gegevens zijn ingevuld.
Given: Het formulier op de website is bijgewerkt door de klant.
When*:* De klant klikt op de knop `Bijwerken'.
Then*:* Het record van de klant wordt bijwerkt en het resultaat daarvan krijgt de klant in het scherm te zien in de vorm van de nieuwe gegevens.

Scenario 2 De klant moet zijn ordergegevens per order kunnen opvragen
User story: Er zijn klanten met en zonder accounts. Deze user story is van toepassing op klanten met accounts.
Scenario: De klant is ingelogd op de website in het klanten gedeelte en wil order gegevens inzien. Om te weten van welke order hij gegevens in kan zien moet hij weten welke orders hij heeft.
Given: De klant is ingelogd in het klanten gedeelte.
When*:* De klant klikt op 'Orders inzien'.

***Then**:* De klant krijgt een lijst te zien met Ordernummer, Order datum en order status.

Scenario 3 De klant kan financiële informatieve opvragen
User story: Er zijn klanten met en zonder accounts. Deze user story is van toepassing op klanten met accounts.
Scenario: De klant is ingelogd op de website in het klanten gedeelte en wil een totaalbedrag van zijn orders weten. Hij ziet een scherm waarbij hij de datum kan selecteren voor het begin van de periode en voor het einde van de periode. Daarnaast kan hij aangeven of hij de BTW wil zien.
Given: De klant heeft de begin en eindperiode aangegeven op het formulier.
***When**:* De klant klikt op 'Bereken bedrag'.
***Then**:* De klant krijgt het bedrag te zien met of zonder BTW.

4 Entity Relations Diagram (ERD)



Afbeelding 2 ERD Eboost

4.1 TOELICHTING ERD

4.1.1 SKATES

Aan de 'skates' entiteit zijn drie andere entiteiten gekoppeld, genaamd: SkateColors, WheelColors en sizes. Hiervoor is gekozen omdat deze entiteiten variabel zijn van de skates. De gebruiker kan hierdoor het model skate kiezen en daarbij zijn gewenste keuzes. In de toekomst kan er bij een uitbreiding makkelijker een nieuw model skate worden toegevoegd en deze te koppelen aan de variabele entiteiten.

4.1.2 ORDERS

Voor orders is er gekozen om twee lossen entiteiten te maken, zodat de klant meerdere skates per order kan bestellen.

4.1.3 CUSTOMERS

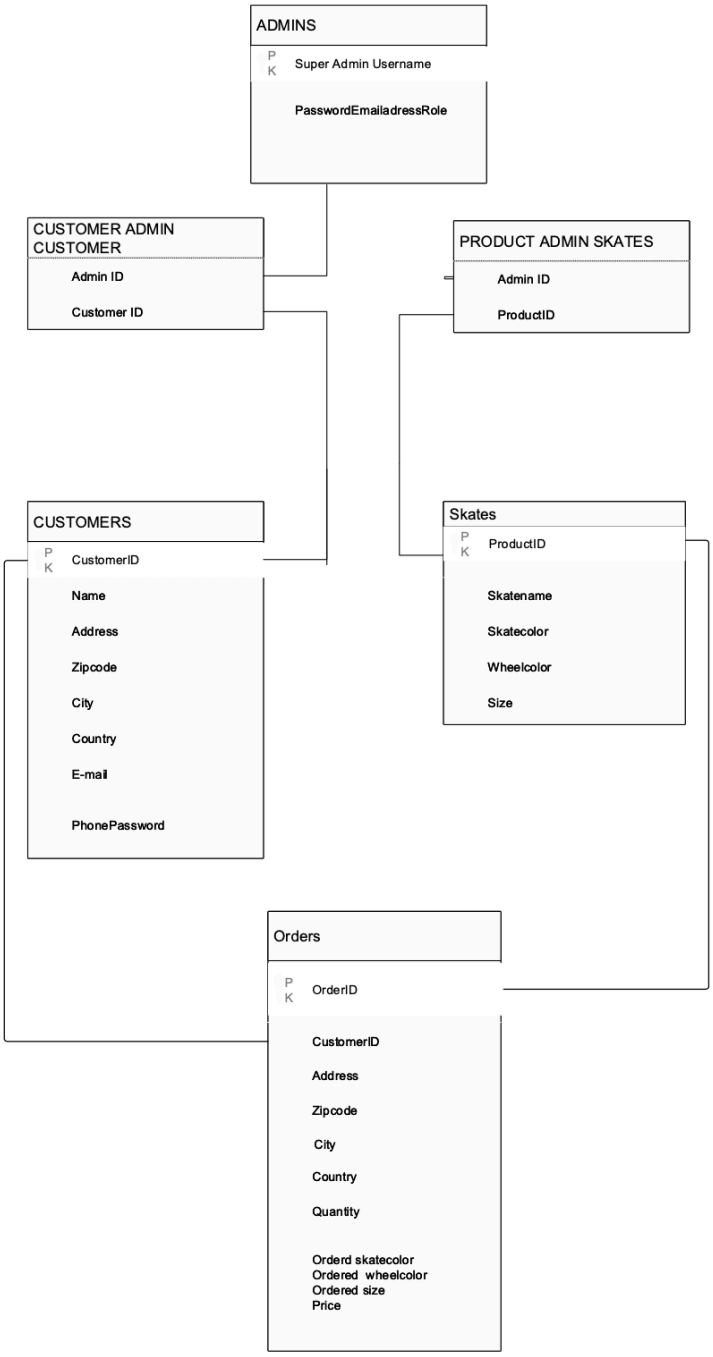
Klanten van Eboost zijn personen die een skate hebben besteld. De klanten worden weergeven door de entiteit 'CUSTOMERS'.

4.1.4 EMPLOYEES

Onder de entiteit 'Employees' vallen de gegevens van de verschillende personen binnen Eboost die toegang hebben tot de database. Elke Employee heeft een unieke ID. De role van de employee is bepalend tot welke gegevens een employee ter beschikking heeft.

5 Relationele Database (RD)

5.1 Relationele database 1e normalisatie vorm



Afbeelding 3 RD Eboost 1e normalisatievorm

5.1.1 Toelichting RD

SKATES

In de tabel skates zijn de skates en daarbij de keuzes van kleuren en maten in een tabel vastgelegd. Er wordt een productID vastgelegd en deze wordt als primary key vastgelegd.

ORDERS

In de tabel orders worden alle gegevens van de klant vastgelegd, daarbij ook de hoeveelheid bestelde skates en de geselecteerde kleuren en maat. In de orders wordt de prijs van de skates vastgelegd.

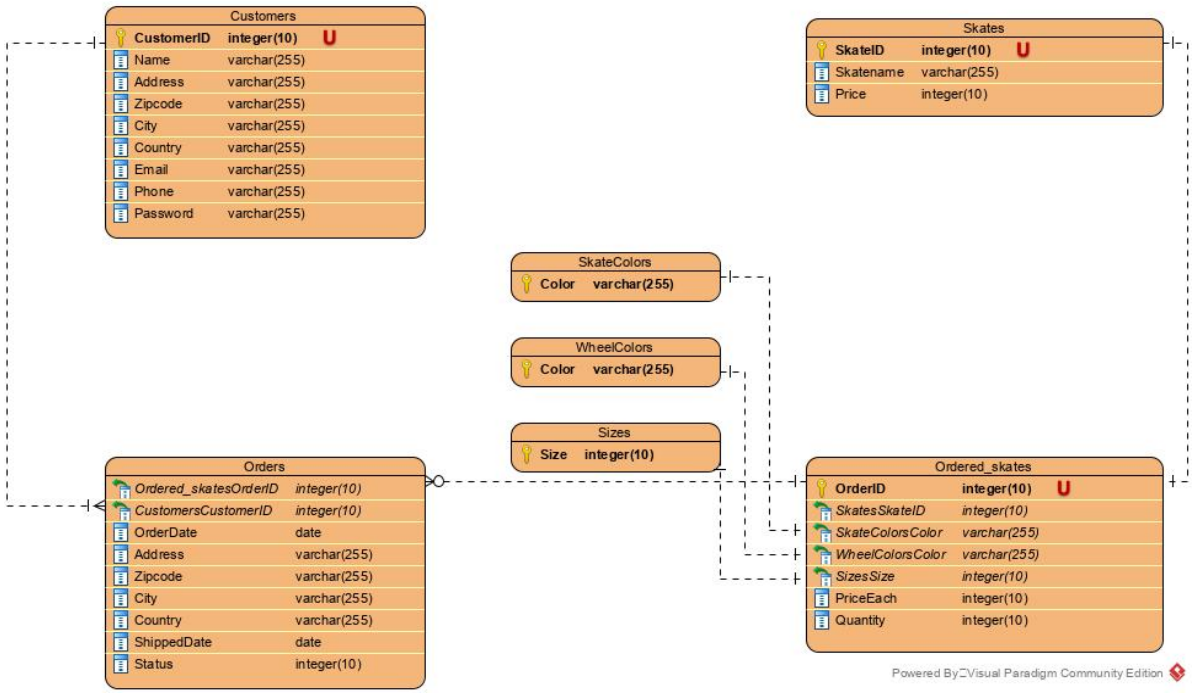
CUSTOMERS

De customers hebben hier geen enkele afhankelijkheid en dienen als actoren voor andere processen.

EMPLOYEES

De superAdmin beheert de accounts voor de productAdmin en de customerAdmin. De superAdmin kan de verschillende type accounts aanmaken en beheren. Zo kan de superAdmin rollen toebedelen. Zo is voor elke type admin alleen de benodigde gegevens beschikbaar en kunnen zij alleen wijzigingen toepassen op de voor hun van benodigde gegevens.

5.2 Relationele database 2e normalisatie vorm



Afbeelding 4 RD Eboost 2e normalisatie vorm

5.2.1 Toelichting RD

Skates
De keuzes van kleuren voor de skates en wielen en zo ook de maten zijn nu in een sub tabel geplaatst. In deze sub tabellen worden beschikbare kleuren en maten geplaatst. Binnen de tabel skates worden deze sub tabellen als foreign keys aangesteld. De prijs van de skates wordt nu als 'price' in de tabel skates geplaatst. Door deze wijzigingen is het nu eenvoudiger bij een uitbreiding van het assortiment een nieuw model skate te koppelen aan de sub tabellen. Of andersom bij het uitbreiden van kleuren en maten.

Orders
Er is een extra tabel gemaakt, genaamd: ordered_skates. In deze tabel wordt nu de gekozen kleuren en maat opgenomen. Dit zijn foreign keys van:

1. OrderedSkateColor verwijst naar de kolom Color van tabel SkateColors.
2. OrderedWheelColor verwijst naar de kolom Color van tabel WheelColors.
3. OrderedSizes verwijst naar de kolom Size van tabel Sizes.

SkateSkateID is een foreign key wat verwijs naar de tabel skateID van skates.

In de tabel orders zijn de ordered_skateOrderID en customerscustomerID als foreign keys vastgelegd.

1. CustomercustomerID verwijst naar de kolom customerID van de tabel customers.
2. Ordered_skateOrderID verwijst naar de kolom orderID van de tabel ordered_skates.

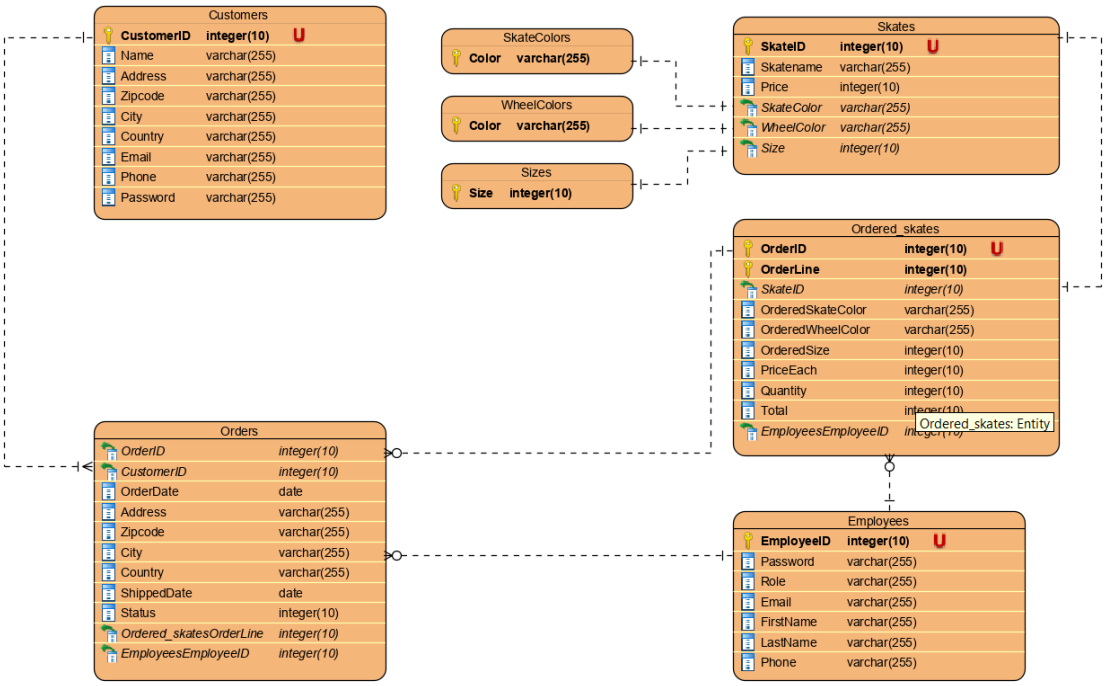
Daarbij zijn er nieuwe kolommen toegevoegd: orderdate, shippedDate en status.

Door deze wijziging is het nu mogelijk voor de klant om meerdere skates te bestellen in één order.

Customers
De tabel `Customers` heeft alle informatie over de klanten. Een klant is een klant wanneer de klant een aankoop doet. Voor iedere klant bestaat dus minimaal 1 order. Een goede klant heeft meerdere orders. Tussen de `Customers` en `Orders` bestaat een 1-op-veel relatie.

Employees
Na de derde normalisatie is ervoor gekozen om de splitsing in de admins anders in te delen. De onderneming heeft meerdere soorten werknemers die toegang nodig hebben tot verschillende delen van de DB. Er zijn vijf rollen aanwezig te weten: admin, manager, CR, HRM en productie. Elke rol heeft toegang tot de gegevens in de database, die benodigd zijn voor de taken die voortkomen uit de role. De admin hebben toegang tot de hele DB. Zij kunnen dan ook wijzigingen aanbrengen aan elke aanwezige tabel in de DB. De overige rollen hebben restricties en kunnen alleen bij de gegevens nadat een admin een role heeft toegekend.

5.3 Relationele database 3e normalisatie vorm



Afbeelding 5 RD 3e normalisatie vorm

6 DATABASE REALISATIE (DDL)

Voor het opzetten van de tabellen zijn de volgende eigenschappen gebruikt

Eigenschappen	Omschrijving
NOT NULL	Een NULL ingaven wordt niet geaccepteerd, wat zorgt voor een verplicht veld.
INT	Kan alleen worden ingevuld worden door cijfers.
VARCHAR	Kan ingevuld worden door alle soort karakters.
AUTO_INCREMENT	Genereerd bij het aanmaken van een nieuwe regel in de tabel automatisch een uniek nummer.
DEFAULT	Als er bij het invoeren van een nieuwe regel niks is ingegeven in de tabel wordt de waarde/tekst achter DEFAULT ingevuld.
DATE	Kan alleen worde ingegeven met een datum, bijvoorbeeld: 2019-12-25.

6.1 STRUCTUUR TABEL CUSTOMERS

```sql
/* Table structure for table Customers */
DROP TABLE IF EXISTS Customers;
CREATE TABLE `Customers` (
  `CustomerID` INT(10) AUTO_INCREMENT PRIMARY KEY,
  `Name` VARCHAR(255) NOT NULL,
  `Address` VARCHAR(255) NOT NULL,
  `City` VARCHAR(255) NOT NULL,
  `Zipcode` VARCHAR(255) NOT NULL,
  `Country` VARCHAR (255) NOT NULL,
  `Email` VARCHAR (255) NOT NULL,
  `Phone` VARCHAR(255) NOT NULL,
  `Password` VARCHAR(255) NULL
);
CREATE INDEX idx_customerEmail ON Customers(`Email`);
```

Allereerst wordt gekeken of de tabel Customers al bestaat en zonodig de tabel `Customers` verwijderd. De tabel wordt aangemaakt.

De eerste kolom is de `CustomerID` en tevens primaire sleutel.

Door de optie AUTO_INCREMENT zal MySQL steeds het CustomerID met 1 verhogen zodat de waarde uniek blijft.

De overige velden zijn standaard VARCHAR met voldoende lengte. Voor een goede order afhandeling is het noodzakelijk dat de velden zijn ingevuld en zodoende de constraint NOT NULL hebben.

Een account aanmaken om te bestellen is niet verplicht. Het wel of niet hebben van een account betekent in praktijk het verschil tussen NULL en NOT NULL voor het `Password` veld.

Om de mensen zonder account toch snel te bedienen hoeven ze niet verplicht te communiceren met het CustomerID. Er is een index gemaakt voor het e-mailadres zodat ze op basis van het e-mail adres contact kunnen hebben met de klantenservice.

6.2 Structuur tabel skateColors

```
/* Table structure for table color skates*/

DROP TABLE IF EXISTS skateColors;

CREATE TABLE skateColors (
Color VARCHAR(255) NOT NULL,

PRIMARY KEY (Color)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- Color is ingesteld met een VARCHAR-eigenschap met een limiet van 255 karakters en heeft een NOT NULL eigenschap.
- Color is verkozen als primary key, omdat er maar 1 kleur is heeft het een unieke waarde.

6.3 Structuur tabel wheelColors

```
/* Table structure for table color wheels*/

DROP TABLE IF EXISTS wheelColors;

CREATE TABLE wheelColors (
Color VARCHAR(255) NOT NULL,

PRIMARY KEY (Color)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- Color is ingesteld met een VARCHAR-eigenschap met een limiet van 255 karakters en heeft een NOT NULL eigenschap.
- Color is verkozen als primary key, omdat er maar 1 kleur is heeft het een unieke waarde.

6.4 Structuur tabel sizes

```
/* Table structure for table size skates*/

DROP TABLE IF EXISTS sizes;

CREATE TABLE sizes (
Size INT(10) NOT NULL,

PRIMARY KEY (Size)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- Size is ingesteld met een INT-eigenschap met een limiet van 10 karakters en heeft een NOT NULL eigenschap.
- Size is verkozen als primary key, omdat er maar 1 soort maat is heeft het een unieke waarde.

6.5 Structuur tabel skates

```
/* Table structure for table skates*/

DROP TABLE IF EXISTS skates;

CREATE TABLE skates (
SkateID INT NOT NULL AUTO_INCREMENT,
Skatename VARCHAR(50) NOT NULL,
Price INT(10) NOT NULL,

PRIMARY KEY (SkateID)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- SkateID wordt automatisch gegenereerd door de auto_increment en zorgt voor een unieke waarde en daardoor is SkateID gelijk de primary key.
- Skatename heeft een NOT NULL eigenschap waarbij de kolom ingevuld kan worden met VARCHAR-karakters met een limiet van 50 karakters.
- Price heeft een NOT NULL eigenschap en kan ingevuld worden met de INT-eigenschappen.

6.6 Structuur tabel ordered_skates

```
/* Table structure for table ordered skates*/

DROP TABLE IF EXISTS ordered_skates;

CREATE TABLE ordered_skates (
    OrderID INT(10) NOT NULL,
    SkateID INT(10) NOT NULL,
    OrderedSkateColor VARCHAR(255) NOT NULL,
    OrderedWheelColor VARCHAR (255) NOT NULL,
    OrderedSize INT(10) NOT NULL,
    Quantity INT(10) NOT NULL,
    PriceEach INT(10) NOT NULL DEFAULT '300',
    Total INT(10) NULL,
    OrderLineNumber INT(6) NOT NULL,

PRIMARY KEY (OrderID, OrderLineNumber),
CONSTRAINT orderSkates FOREIGN KEY (SkateID) REFERENCES skates (SkateID),
CONSTRAINT skateColors FOREIGN KEY (OrderedSkateColor) REFERENCES skateColors (Color),
CONSTRAINT wheelColors FOREIGN KEY (OrderedWheelColor) REFERENCES wheelColors (Color),
CONSTRAINT sizes FOREIGN KEY (OrderedSize) REFERENCES sizes (size)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- OrderID heeft een INT-eigenschap en is een verplichte kolom. De OrderID is de primary key samen met OrderLineNumber.
- OrderLineNumber heeft een INT-eigenschap en is een verplichte kolom. Samen met de OrderID is het de primary key. Hierdoor kan de klant meerdere skates bestellen in één order, onder één OrderID.
- SkateID heeft een INT-eigenschap en is een verplichte kolom. Het is een foreign key wat gekoppeld is aan de tabel skates en kolom SkateID. De gebruiker kan hierdoor alleen kiezen tussen de skates die zijn ingegeven in de tabel skates en kolom SkateID.

- OrderedSkateColor heeft een VARCHAR-eigenschap en is een verplichte kolom met een limiet van 255 karakters. Het is een foreign key wat gekoppeld is aan de tabel skateColors en kolom Color. De gebruiker kan hierdoor alleen kiezen tussen de kleuren die zijn ingegeven in de tabel skateColors en kolom Color.
- OrderedwheelColor heeft een VARCHAR-eigenschap en is een verplichte kolom met een limiet van 255 karakters. Het is een foreign key wat gekoppeld is aan de tabel wheelColors en kolom Color. De gebruiker kan hierdoor alleen kiezen tussen de kleuren die zijn ingegeven in de tabel wheelColors en kolom Color.
- Orderedsizes heeft een INT-eigenschap en is een verplichte kolom met een limiet van 255 karakters. Het is een foreign key wat gekoppeld is aan de tabel sizes en kolom Size. De gebruiker kan hierdoor alleen kiezen tussen de maten die zijn ingegeven in de tabel sizes en kolom Size.
- Quantity heeft een INT-eigenschap en is verplichte kolom met een limiet van 10 getalen.
- PriceEach heeft een INT-eigenschap en is een verplicht veld wat met default als 300,- euro is ingesteld.
- Total heeft een INT-eigenschap met een limiet van 10 getallen en is ingesteld als niet verplicht. Echter wordt met een trigger (hoofdstuk 9.2.1) een som gemaakt van de kolommen quanitity * PriceEach = Total. Het totaalbedrag wordt hierdoor automatisch ingevuld.

6.7 Structuur tabel orders

```sql
/* Table structure for table orders*/

DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
    OrderID INT(10) NOT NULL,
    CustomerID  INT(10) NOT NULL,
    OrderDate DATE NOT NULL,
    Address VARCHAR(50) NOT NULL,
    Zipcode VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    Country VARCHAR (50) NOT NULL,
    ShippedDate DATE NOT NULL,
    orderStatus VARCHAR(255) NOT NULL,

PRIMARY KEY (OrderID),
CONSTRAINT customerOrders FOREIGN KEY (CustomerID) REFERENCES customers (CustomerID)
);
```

De structuur van de tabel heeft de volgende eigenschappen:

- OrderID heeft een INT-eigenschap met een limiet van 10 getallen. Het is een primay key.
- CustomerID heeft een INT-eigenschap en is een verplichte kolom. Het is een foreign key wat gekoppeld is aan de tabel customers en kolom CustomerID.
- OrderDate heeft een DATE-eigenschap en is een verplichte kolom.
- Address, zipcode, city en country hebben alle vier de VARCHAR-eigenschap en zijn ook alle vier verplichten kolommen.
- ShippedDate heeft een DATE-eigenschap en is een verplichte kolom.
- OrderStatus heet een VARCHAR-eigenschap met een limiet van 255 karakters. Er zijn drie verschillende status: shipped, paid en ordered.

6.8 Structuur tabel employees

```
/* Table structure for table employees*/

DROP TABLE IF EXISTS `employees`;

CREATE TABLE `employees` (
    `EmployeeID` INT(255) NOT NULL AUTO_INCREMENT,
    `Password` VARCHAR(255) NOT NULL ,
    `Role` VARCHAR(255) NOT NULL ,
    `Email` VARCHAR(255) NOT NULL ,
    `FirstName` VARCHAR(255) NOT NULL ,
    `LastName` VARCHAR(255) NOT NULL ,
    `Phone` VARCHAR(255) NULL ,
PRIMARY KEY (`EmployeeID`)
```

De opbouw van de employee tabel heeft de volgende eigenschappen:
De EmployeeID is een integer van maximaal 255 karakters en is tevens de primary key. Door de auto increment functie wordt deze automatisch toebedeeld bij het aanmaken van een record.

Password bestaat uit maximaal 255 verschillende karakters.

Role wordt toegewezen door de admin, deze is bepalend voor de rechten van de user. Email, FirstName en LastName bestaan uit maximaal 255 karakters en zijn de persoonlijke gegevens van de user. Deze moeten ingevuld worden.

Phone is het telefoonnummer die al dan niet bekend is.

6.9 Triggers

6.9.1 Trigger totalOrdered

```
/* Trigger to sum up the total ordered amount */

CREATE TRIGGER totalOrdered BEFORE INSERT
ON ordered_skates
FOR EACH ROW
SET NEW.Total = NEW.Quantity * NEW.PriceEach;
```

Het moet voor de klant mogelijk zijn om meerdere skates te bestellen om te zorgen voor een goed overzicht is het goed om te weten wat het totaalbedrag is van een order. Door de trigger 'totalOrdered' worden de kolommen Quantity en PriceEach met elkaar vermenigvuldigd. Dit wordt voordat het wordt doorgevoerd uitgevoerd.

6.9.1 Trigger employee changes

```
/* Trigger staff update changes HRM */

DELIMITER $$

CREATE TRIGGER register_employee_changes
    BEFORE UPDATE
    ON employees FOR EACH ROW
 BEGIN
```

```
INSERT INTO employeesChanges
 SET action = 'update',
     EmployeeID = OLD.EmployeeID,
     FirstName = OLD.FirstName,
     LastName = OLD.LastName,
     Role = OLD.Role,
     Email = OLD.Email,
     Phone = OLD.Phone,
     changedat = NOW();
END$$

DELIMITER ;
```

Bij het aanbrengen van wijzigingen worden de oude gegevens van geregistreerd door een trigger.

6.9.2 TRIGGER EMPLOYEE DELETE

```
DELIMITER $$

CREATE TRIGGER register_employee_delete
    BEFORE DELETE
    ON employees FOR EACH ROW
 BEGIN
 INSERT INTO employeesChanges
 SET action = 'delete',
     EmployeeID = OLD.EmployeeID,
     FirstName = OLD.FirstName,
     LastName = OLD.LastName,
     Role = OLD.Role,
     Email = OLD.Email,
     Phone = OLD.Phone,
      changedat = NOW();
END$$

DELIMITER ;
```

Wanneer er een record verwijderd wordt uit het employee bestand, wordt deze handeling geregisteerd door een trigger.

6.10 VIEWS

6.10.1 VIEW ORDEREDSIZES

```
/* Create view for counting ordered sizes*/

CREATE VIEW orderedSizes AS
SELECT OrderedSize, COUNT(*)
FROM ordered_skates
GROUP BY OrderedSize;
```

Voor Eboost is het belangrijk om te weten welke maat goed wordt verkocht en welke niet. Door de view 'orderedSizes' is het mogelijk om een lijst te krijgen met de orderedSize om de bestelde maten met elkaar op te tellen. Dit ziet er als volgt uit:

OrderedSize	COUNT(*)
39	1
40	1
41	1
42	2
43	2
44	2

Onder kolom 'orderedSize de maat en onder kolom COUNT(*) opgeteld de bestelde hoeveelheid maten.

6.10.2 VIEW VIEWORDERS

```
/* Create view for summary order*/

  CREATE VIEW viewOrders AS
  SELECT orders.OrderID, CustomerID, OrderDate, Address, Zipcode, City, Country, ShippedDate,
orderStatus, SkateID, OrderedSkateColor, OrderedWheelColor, OrderedSize, Quantity, PriceEach,
OrderlineNumber FROM orders, ordered_skates
  WHERE ordered_skates.OrderID = orders.OrderID;
```

Het is een eis dat het voor de klant mogelijk is om na de bestelling zijn order in te zien. Door de stored procedure 'viewOrder' op te roepen zal er een overzicht worden gemaakt waarbij de tabellen orders en ordered_skates worden samengevoegd. Er wordt gekeken of de waarde van tabel orders en kolom ordered_skates.OrderID en tabel order.OrderID gelijk zijn aan elkaar.

6.10.3 VIEW STAFFUPDATE

```
/* view for staff update manager */

CREATE VIEW StaffUpdate AS
    SELECT Role, COUNT(Role)
    FROM `employees`
    GROUP BY Role
```

Met deze view is het inzichtelijk hoeveel roles er onder de employees aanwezig zijn

De view kan wordt opgeroepen door:

```
SELECT * FROM `staffupdate`
```

6.10.4 VIEW CUSTOMERSLOCATION

```
/* view for customer location marketing */

CREATE VIEW CustomersLocation AS
    SELECT City, COUNT(City)
    FROM `customers`
    GROUP BY City;
```

Om inzicht te krijgen uit welke omgeving de customers komen is er een view gemaakt die de customers indeelt op basis uit welke stad zij komen. Hierbij worden het totaalaantal consumenten per stad aangegeven.

De view kan opgeroepen worden door

```
SELECT * FROM `customerslocation`
```

6.10 Stored Procedure

```
/* Stored Procedure for getting an orderlist */
DELIMITER //
CREATE PROCEDURE GetOrderlistByCustomer(
    IN customer INTEGER(10)
)
BEGIN
    SELECT OrderID, OrderDate, orderStatus
    FROM orders
    WHERE CustomerID = customer;
END //

DELIMITER ;
```

Wanneer de klant is ingelogd kan in een sessie variabele zijn klantnummer worden opgeslagen. Dit klantnummer is invoer voor de procedure en retourneert de lijst.

```
/* Stored Procedure for getting an Total amount with TAX */
DELIMITER //

CREATE PROCEDURE TotalAmountByCustomer(
    IN customer INTEGER(10),
       startd DATE,
       endd DATE
)
BEGIN
  SELECT SUM(Total) AS Totaal, SUM(calculateTax(Total)) AS BTW
  FROM orders
  INNER JOIN ordered_skates USING(OrderID)
  WHERE CustomerID=customer AND OrderDate BETWEEN startd AND endd;
END //

DELIMITER ;
```

Wanneer de klant is ingelogd kan in een sessie variabele zijn klantnummer worden opgeslagen. Dit klantnummer is invoer voor de procedure. De klant geeft zelf een start en einddatum op en daarna komt de lijst terug.

6.11 Function

```
CREATE FUNCTION calculateTax(
  amount INT
)
RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE tax INT(10);
  SET tax = amount/100*21;
```

```
  -- return the VAT 21%
  RETURN (tax);
END//

DELIMITER ;
```
Deze functie krijgt een bedrag en rekent 21% BTW uit. De BTW krijg je terug als bedrag.

7 Security (DCL)

7.1 Rechten Admin

```
CREATE USER 'Admin '@' % ' IDENTIFIED BY 'AdminPassword';
GRANT ALL PRIVILEGES ON EBOOST.* TO ‘Admin’@'%' WITH GRANT OPTION;
```
De Admin rol heeft overal de rechten toe.

7.2 Rechten Manager

```
CREATE USER ' Manager'@' % ' IDENTIFIED BY 'ManagerPassword';
GRANT ALL PRIVILEGES ON EBOOST.Employees TO 'Manager'@'%' WITH GRANT OPTION;
```
De manager rol heeft de volledige rechten tot de medewerkerstabel.

```
GRANT ALL PRIVILEGES ON EBOOST. Ordered_skates TO 'Manager'@'%' WITH GRANT OPTION;
```
De manager rol heeft de volledige rechten op de ordered_Skatestabel.

7.3 Rechten CustomerRelation

```
CREATE USER 'CustomerRelation'@' % ' IDENTIFIED BY 'CustomerRelationPassword';
GRANT SELECT, INSERT, UPDATE, DELETE ON EBOOST.Customers TO 'CustomerRelation'@'%' WITH
GRANT OPTION;
```
De CustomerRelation rol heeft alleen de Select, Insert, Update en Delete op de Customers tabel.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON EBOOST.Orders TO ‘'CustomerRelation'@'%' WITH GRANT
OPTION;
```
De CustomerRelation rol heeft alleen Select, Insert, Update en Delete rechten op de tabel Orders.

7.4 Rechten Productie

```
USER 'Production'@' % ' IDENTIFIED BY 'ProductionPassword';
GRANT ALL PRIVILEGES ON EBOOST.Skates TO 'Production'@'%' WITH GRANT OPTION;
GRANT INSERT, UPDATE, DELETE ON EBOOST.Ordered_skates TO 'Production'@'%' WITH GRANT
OPTION;
GRANT INSERT, UPDATE ON EBOOST.SkateColors TO 'Production'@'%' WITH GRANT OPTION;
GRANT INSERT, UPDATE ON EBOOST.WheelColors TO 'Production'@'%' WITH GRANT OPTION;
GRANT INSERT, UPDATE ON EBOOST.Sizes TO 'Production'@'%' WITH GRANT OPTION;
```
De Productie rol heeft de volgende rechten:

- Volledige rechten op de tabel Skates.
- Insert, Update, Delete rechten op de Ordered_skates Tabel.
- Insert, Update rechten op de tabellen SkateColors, WheelColors en Sizes.

7.5 Rechten Customer

```
CREATE USER 'Customer'@' % ' IDENTIFIED BY ‘CustomerPassword';
GRANT SELECT ON EBOOST.Orders TO ‘Customer’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Customer rol heeft alleen de Select rechten op de tabel Orders. Oftewel alleen lezen.

```
GRANT UPDATE, SELECT ON EBOOST.Customers TO ‘Customer’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Customer rol heeft alleen de Select en update rechten op de tabel Customers.

```
GRANT SELECT, INSERT ON EBOOST.Ordered_Skates TO ‘Customer’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Een Customer kan altijd een bestelling plaatsen

7.6 Rechten Guest

```
CREATE USER 'Guest'@' % ' IDENTIFIED BY ‘GuestPassword';
GRANT SELECT ON EBOOST.Skates TO ‘Guest’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Gasten rol heeft alleen lees rechten.

```
GRANT SELECT ON EBOOST.SkateColors TO ‘Guest’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Gasten rol heeft alleen lees rechten.

```
GRANT SELECT ON EBOOST.WheelColors TO ‘Guest’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Gasten rol heeft alleen lees rechten.

```
GRANT SELECT ON EBOOST.SizesTO ‘Guest’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Gasten rol heeft alleen lees rechten.

```
GRANT SELECT, INSERT ON EBOOST.Ordered_Skates TO ‘Guest’@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

De Gasten rol heeft lees en Insert rechten.

8 ONDERHOUD

De database wordt onderhouden via phpMyAdmin. Het onderhoud verdelen we in twee aparte delen. Het eerste deel is het veiligstellen van de structuur en data van de database. Het tweede deel is het herstellen van fragmentatie. Door het gebruik worden gegevens fysiek niet bij elkaar gehouden. Om de gegevens fysiek weer op orde te brengen optimaliseren wij de database.

Instructie voor Backup

1. Start phpMyAdmin;
2. Selecteer de Eboost database;
3. Zet een vinkje bij "Selecteer alles";
4. Kies uit het dropdown menu "Met geselecteerde" de keuze "Exporteren";
5. Druk op "Starten";
6. Het .sql bestand staat in de map "Downloads".

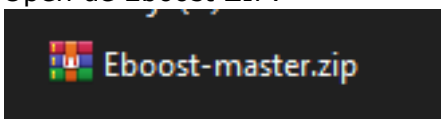
Instructie voor Optimaliseren

1. Start phpMyAdmin;
2. Selecteer de Eboost database;
3. Zet een vinkje bij "Selecteer alles";
4. Kies uit het dropdown menu "Met geselecteerde" de keuze "Optimaliseren";
5. Het proces wordt op de achtergrond zonder feedback uitgevoerd.

9 INSTALLATIE

9.1 STAPPENPLAN

Stap 1:
Open de Eboost ZIP:



Stap 2:
Ga naar de MYSQL Command lijn:

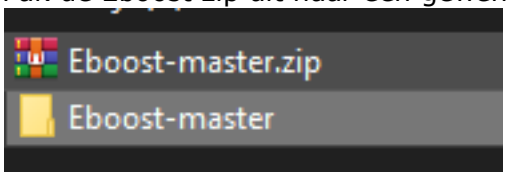
```
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.4.8-MariaDB mariadb.org binary distribution

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)]>
MariaDB [(none)]>
MariaDB [(none)]>
MariaDB [(none)]>
```

Stap 3:
Pak de Eboost zip uit naar een gewenste locatie.



Hierin vind je de 3 bestanden om de database op te zetten (DDL, DCL, DML).

Stap 4:
Typ in de command lijn van SQL het volgende:
Source "Pad waar de bestanden staan"/DDL.sql
Source "Pad waar de bestanden staan"/DCL.sql
Source "Pad waar de bestanden staan"/DML.sql

9.2 INSERT DATA

Om te werken met een database zijn onderstaande fictieve gegevens ingevoerd.

Gegevens van de klanten

```
/* Insert data for customers*/

INSERT INTO customers (Name, Address, City, Zipcode, Country, Email, Phone, Password) VALUES
('Bert Jansen', 'Kerkstraat 1', 'Den Bosch', '5213HG', 'NL', 'bert@avans.nl','0612345678', 'te
st1'),
('Niels Peters', 'Snellestraat 3', 'Den Bosch', '5211KG', 'NL', 'niels@avans.nl','0612345400',
 'test2'),
```

```
('Pieter Jansen', 'Voltastraat 12', 'Den Bosch', '5215KJ', 'NL', 'pieter@avans.nl','0612345211
',  'test3'),
('Piet Pieters', 'Langestraat 43', 'Den Bosch', '5212ER', 'NL', 'piet@avans.nl','0612345700',
 'test4'),
('Sjefke van de Sterre', 'Dorpstraat 20', 'Den Bosch', '5221TK', 'NL', 'sjefke@avans.nl','0612
345711', 'test5'),
('Jan Jansen', 'Stationstraat 56', 'Den Bosch', '5214WE', 'NL', 'jan@avans.nl', '0612345822','
test6'),
('Bas Peter', 'Kerkstraat 100', 'Den Bosch', '5234FR', 'NL', 'bas@avans.nl', '0612345933', 'te
st7'),
('Geintreseerde Skater', 'Kerklaan 1', 'Breda', '4800HG', 'NL', 'snelleskater@avans.nl','06105
45400', 'test8'),
('Ikke Ook',  'Snelleweg 3', 'Breda', '4834KG', 'NL', 'Ikkeook@avans.nl','0633345211', 'test9'
),
('Loop Nietmeer',  'Voltadrift 12', 'Breda', '4827KJ', 'NL', 'nietmeer@avans.nl','066645700',
'test10'),
('Bassie de Clown',  'Langepad 43', 'Utrecht', '3455ER', 'NL', 'Bassie@avans.nl','0689765411',
 'test11'),
('Sjefke van weghorst',  'Dorpstraat 20', 'Utrecht', '3455TK', 'NL', 'sjefke@avans.nl','061204
5822', 'test12'),
('Jan Jans',  'Weg achter Stationstraat 9', 'Tilburg', '5000WE', 'NL', 'jan@avans.nl','0612345
933',  'test13'),
('Bassie Adriaan', 'lijsterstraat 100', 'Tilburg', '5000FR', 'NL', 'bassieandriaan@avans.nl','
0623456789', 'test14'),
('Bert-
Willen van der Jansen',  'stavorenstraat 13', 'Breda', '4834HG', 'NL', 'bertwillem@avans.nl','
0613452119', 'test15'),
('Niek Peeters',  'staorenstraat 3', 'Breda', '4826KG', 'NL', 'niek@avans.nl','0612893000', 't
est16'),
('Piet Janssen',  'Voltastraat 1', 'Utrecht', '3512KJ', 'NL', 'pietp@avans.nl','0611145711', '
test17'),
('Karel Pieters',  'Langeweg 4', 'Tilburg', '5000ER', 'NL', 'karelpiet@avans.nl', '061765822',
 'test18'),
('Jeffrey Sterre',  'Dorpweg 20', 'Utrecht', '35661TK', 'NL', 'jeff@avans.nl','0634544070', 't
est19'),
('Japie meerkoet',  'ijlsterstraat 5', 'Breda', '4826WE', 'NL', 'japie@avans.nl','0612333309',
 'test20'),
('Bastiaan Peterson',  'Kerkveste 90', 'Den Bosch', '5234FR', 'NL', 'bastiaan@avans.nl','06123
45678', 'test21'),
('Bert Jacobsen',  'Kerkerstraat 1', 'Utrecht', '3506HG', 'NL', 'bert@avans.nl','0612345999',
'test22'),
('Nielson von Peterson','Snellwegstraat 3','Breda','5211KG','NL','nielson@avans.nl','068788864
5', 'test23'),
('Pieteria van der Jansen',  'poltastraat 12', 'Breda', '5215KJ', 'NL', 'pieteria@avans.nl','0
699345211', 'test28'),
('Peter Pietsinho','Langerestraat 43', 'Breda', '5212ER', 'NL', 'peter@avans.nl','0612543210',
 'test24'),
```

```
('jefke van der Sterrenhemel',  'Dorpssestraat 20', 'Utrecht', '3500TK', 'NL', 'jefke@avans.nl
','0607645711', 'test25'),
('Jan-
Willem Johansen','Stationparkstraat 56', 'Tilburg', '5214WE', 'NL', 'janwillem@avans.nl','0612
876822', 'test26'),
('Baltasar Peteresco', 'Kerktorenstraat 100', 'Tilburg', '5234FR', 'NL', 'baltasars@avans.nl',
'0667445933', 'test27');
```

Gegevens van de orders/verzendgegevens

```
/* Data for the table orders*/
INSERT INTO orders (OrderID, CustomerID, Orderdate, Address, Zipcode, City, Country,
ShippedDate, orderStatus) VALUES

('1', '1', '2019-11-01', 'Kerkstraat 1', '5211KG', 'Den Bosch', 'NL', '2019-11-15',
'Shipped'),

('2', '2', '2019-11-04', 'Kerkstraat 20', '5211KG', 'Den Bosch', 'NL', '2019-11-17',
'Shipped'),

('3', '3', '2019-11-09', 'Vughterstraat 34', '5216KJ', 'Den Bosch', 'NL', '2019-11-19',
'Shipped'),

('4', '4', '2019-11-14', 'Snellestraat 78', '5217JK', 'Den Bosch', 'NL', '2019-11-22',
'Shipped'),

('5', '5', '2019-11-23', 'Verwerstraat 98', '5221LK', 'Den Bosch', 'NL', '2019-11-30',
'Shipped'),

('6', '6', '2019-11-27', 'Kortestraat 12', '5219ER', 'Den Bosch', 'NL', '2019-12-02',
'Shipped'),

('7', '7', '2019-11-30', 'Stationstraat 43', '5211KG', 'Den Bosch', 'NL', '2019-12-05',
'Shipped'),

('8', '1', '2019-12-04', 'Voltastraat 13', '5241HG', 'Den Bosch', 'NL', 'NULL', 'Paid'),

('9', '2', '2019-12-07', 'Hinthamerstraat 20', '5234EK', 'Den Bosch', 'NL', 'NULL', 'Paid');
```

Gegevens van de bestelde producten

```
/* Insert data for ordered skates*/

INSERT INTO ordered_skates (OrderID, SkateID, OrderedSkateColor, OrderedWheelColor,
OrderedSize, Quantity, PriceEach, Total, OrderLineNumber) VALUES

('1', '1', 'Black', 'Black', '42', '2', '300', '0', '1'),
('1', '1', 'Black', 'White', '43', '1', '300', '0', '2'),
('3', '1', 'White', 'Black', '44', '1', '300', '0', '1'),
('4', '1', 'White', 'White', '39', '1', '300', '0', '1'),
('5', '1', 'White', 'Black', '40', '1', '300', '0', '1'),
('6', '1', 'Black', 'White', '41', '1', '300', '0', '1'),
('7', '1', 'Black', 'Black', '42', '1', '300', '0', '1'),
```

```
('8', '1', 'White', 'White', '44', '1', '300', '0', '1'),
('9', '1', 'Black', 'White', '43', '1', '300', '0', '1');
```

Gegevens van de Employees

```
INSERT INTO `employees` (`Password`, `Role`, `Email`, `FirstName`, `LastName`, `Phone`) VALUES

('ChangeWhenPossible!1', 'Admin', 'eboostadmin@eboost.nl', 'x', 'x x', '061234567
8'),
('ChangeWhenPossible!2', 'Admin', 'eboostcustomer@eboost.nl', 'Klant', 'vriendelijk', '+316345
67890'),
('ChangeWhenPossible!3', 'Admin', 'eboostProduct@eboost.nl', 'Productie', 'Werker', '
3'),
('ChangeWhenPossible!4', 'CustomerRelation', 'eboostmarketing@eboost.nl', 'x', 'x
', ''),
('ChangeWhenPossible!5', 'CustomerRelation', 'eboostmarketing2@eboost.nl', 'Sales', 'Man', '06
-'),
('ChangeWhenPossible!6', 'CustomerRelation', 'eboostmanager@eboost.nl', '', '', '+316-'),
('ChangeWhenPossible!7', 'Manager', 'eboostmanager@eboost.nl', 'Big', 'Boss', '+'),
('ChangeWhenPossible!8', 'HRM', '', 'Peoples', 'Person', ''),
('ChangeWhenPossible!9', 'HRM', 'eboostHRM@eboost.nl', 'Human', 'Resources', '+'),
('ChangeWhenPossible!10', 'HRM', 'Username@eboost.nl', '', '', '');
```

10 Uitwerking (DML)

10.1 Implementatie #1

De klant moet zijn persoonlijke gegevens kunnen wijzigen. In dit voorbeeld heeft de is de ingelogde klant de klant met het CustomerID 4.

De klant krijgt na het correct inloggen een scherm te zien met verschillende keuzes, wanneer hij kiest om zijn persoonlijke gegevens aan te passen ziet hij een webformulier die gevuld is met informatie uit de query:

```
SELECT Name, Address, City, Zipcode, Country, Email, Phone FROM customers WHER
E CustomerID = 4;
```

De klant ziet zijn gegevens in en corrigeert deze zo nodig. Na het drukken op de knop `Bijwerken` wordt de volgende query uitgevoerd:

```
UPDATE customers SET Name = 'Piet Pieters 2', Address = 'Langestraat 432', Cit
y = 'Den Bosch 2', Zipcode = '512ER2', Country = 'NL2', Email = 'piet@avans.nl
2', Phone = '0622222222' WHERE CustomerID = 4;
```

De gegevens van de klant zijn nu bijwerkt.

10.2 Implementatie #2

De klant moet zijn gegevens per order kunnen opvragen. In dit voorbeeld heeft de is de ingelogde klant de klant met het CustomerID 4.

De klant krijgt na het correct inloggen een scherm te zien met verschillende keuzes, wanneer hij kiest om zijn orders te bekijken ziet hij een lijst die is verkregen met de volgende Stored Procedure:

```
DELIMITER //

CREATE PROCEDURE GetOrderlistByCustomer(
    IN customer INTEGER(10)
)
BEGIN
    SELECT OrderID, OrderDate, orderStatus
    FROM orders
    WHERE CustomerID = customer;
END //

DELIMITER ;
```

De procedure wordt aangeroepen door:

```
CALL GetOrderlistByCustomer(4);
```

Door het klikken op een order in de lijst wordt Implementatie #9 uitgevoerd.

10.3 Implementatie #3

De klant moet een financieel overzicht kunnen opvragen. Om de BTW te berekenen is een aparte functie toegevoegd zoals hieronder:

```
/* Function for calculating VAT */
DELIMITER //

CREATE FUNCTION calculateTax(
  amount INT
)
RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE tax INT(10);
  SET tax = amount/100*21;

  -- return the VAT 21%
  RETURN (tax);
END//

DELIMITER ;
```

Er is uitgegaan van 21% BTW en die wordt teruggegeven aan MySQL.
Daarnaast is er een view gemaakt zodat het datumfilter kan werken. De klant selecteer de begin en start datum in het scherm en het CustomerID bestaat in de sessie variabele.

```
/* Stored Procedure for getting an Total amount with TAX */
DELIMITER //

CREATE PROCEDURE TotalAmountByCustomer(
    IN customer INTEGER(10),
       startd DATE,
       endd DATE
)
BEGIN
  SELECT SUM(Total) AS Totaal, SUM(calculateTax(Total)) AS BTW
  FROM orders
  INNER JOIN ordered_skates USING(OrderID)
  WHERE CustomerID=customer AND OrderDate BETWEEN startd AND endd;
END //

DELIMITER ;
```

Gebruik:

```
CALL TotalAmountByCustomer(1,'2019-11-01','2019-12-31');
```

We krijgen nu een Totaal terug en een BTW. Deze kunnen we zelf naar wens representeren in de front end.

10.4 Implementatie #4

```
/* Insert order in table orders and ordered_skates*/

START TRANSACTION;

SELECT
    @orderID:=MAX(orderID)+1
FROM
    orders;

/* Implementatie #7 */

INSERT INTO ordered_skates(OrderID, SkateID, OrderedSkateColor, OrderedWheelColor,
OrderedSize, Quantity, PriceEach, Total, OrderLineNumber) VALUES
(@orderID, '1', 'Black', 'Black', '42', '2', '300', '0', '1'),
(@orderID, '1', 'Black', 'Black', '42', '5', '300', '0', '2');


/* Implementatie #8 */

INSERT INTO orders (OrderID, CustomerID, Orderdate, Address, Zipcode, City, Country) VALUES
(@orderID, '2', '2019-12-30', 'raadstraat 2', '5241BL', 'Rosmalen', 'NL');


COMMIT;
```

In één transaction worden de gegevens voor de tabellen ordered_skates en orders ingevoerd door de klant. Het eerste gedeelte is voor de tabel ordered_skates. De klant kiest de kleuren van de skates en wielen en de juiste maat. Om te zorgen dat 1 orderID meerdere OrderLineNumbers kan hebben. Wordt er verwezen naar de orderID van de tabel orders. Hierdoor worden elke OrderLineNumber met dezelfde orderID ingevuld, welke wordt bepaald door "@order". De klant geeft aan hoeveel skates hij wil bestellen de prijs staat vast op 300,- euro.

Door de trigger totalOrdered wordt de het totaalbedrag gerekend voordat het wordt doorgevoerd.

10.5 Implementatie #5

Doormiddel door het eerdere genoemde @orderID wordt er eerst gekeken naar het hoogste orderID nummer en wordt het nieuwe orderID met 1 verhoogd. Dit orderID wordt dan ook gebruikt voor de OrderLineNumbers.

De klant geeft zijn adresgegevens in, wat behulp van de INSERT INTO wordt doorgevoerd.

10.6 Implementatie #6

Wanneer de klant zijn order wil inzien kan men door het invullen van zijn orderID (als voorbeeld orderID '1') op de website zijn gegevens inzien. Daarbij wordt de view 'viewOrders' oproepen. De view haalt alle kolommen van orders en ordered_skates op waarbij de orderID gelijk is in beide tabellen.

```
SELECT * FROM `viewOrders` WHERE OrderID = 1;
```

De uitkomst is te zien in bijlage 1.

10.7 Insert data

```
/* Insert colors for skatecolors*/

INSERT INTO skateColors (Color) VALUES
('Black'),
('White');
```

Eboost heeft ervoor gekozen om met twee kleuren skates uit te brengen. De gebruiker kan kiezen tussen zwart en wit.

```
/* Insert colors for the wheelcolors*/

INSERT INTO wheelColors (Color) VALUES
('Black'),
('White');
```

Eboost heeft ervoor gekozen om met twee kleuren wielen uit te brengen. De gebruiker kan kiezen tussen zwart en wit.

```
/* Insert sizes for skates*/

INSERT INTO sizes (Size) VALUES
('38'),
('39'),
('40'),
('41'),
('42'),
```

```
('43'),
('44'),
('45');
```

Eboost heeft ervoor gekozen om 8 verschillende maten beschikbaar te maken voor de klant. De maten zijn er van maat 38 tot en met maat 45.

```
/* Insert data for the skates*/

INSERT INTO skates (SkateID, Skatename, Price) VALUES

('1', 'Eboost ONE', '300');
```

Bij het begin van Eboost is er één model ontworpen genaamd de Eboost ONE. De Eboost ONE worden voor 300,- EUR aangeboden.

BIBLIOGRAFIE

Altexsoft. (2018, Juli 28). *Acceptance Criteria: Purposes, Formats, and Best Practices*. Opgehaald van Altexsoft: https://www.altexsoft.com/blog/business/acceptance-criteria-purposes-formats-and-best-practices/

Bijlagen

OrderID	CustomerID	OrderDate	Address	Zipcode	City	Country	ShippedDate	orderStatus	SkateID	OrderedSkateColor	OrderedWheelColor	OrderedSize	Quantity	PriceEach	OrderlineNumber
1	1	2019-11-01	Kerkstraat 1	5211KG	Den Bosch	NL	2019-11-15	Shipped	1	Black	Black	42	2	300	1
1	1	2019-11-01	Kerkstraat 1	5211KG	Den Bosch	NL	2019-11-15	Shipped	1	Black	White	43	1	300	2

Bijlage