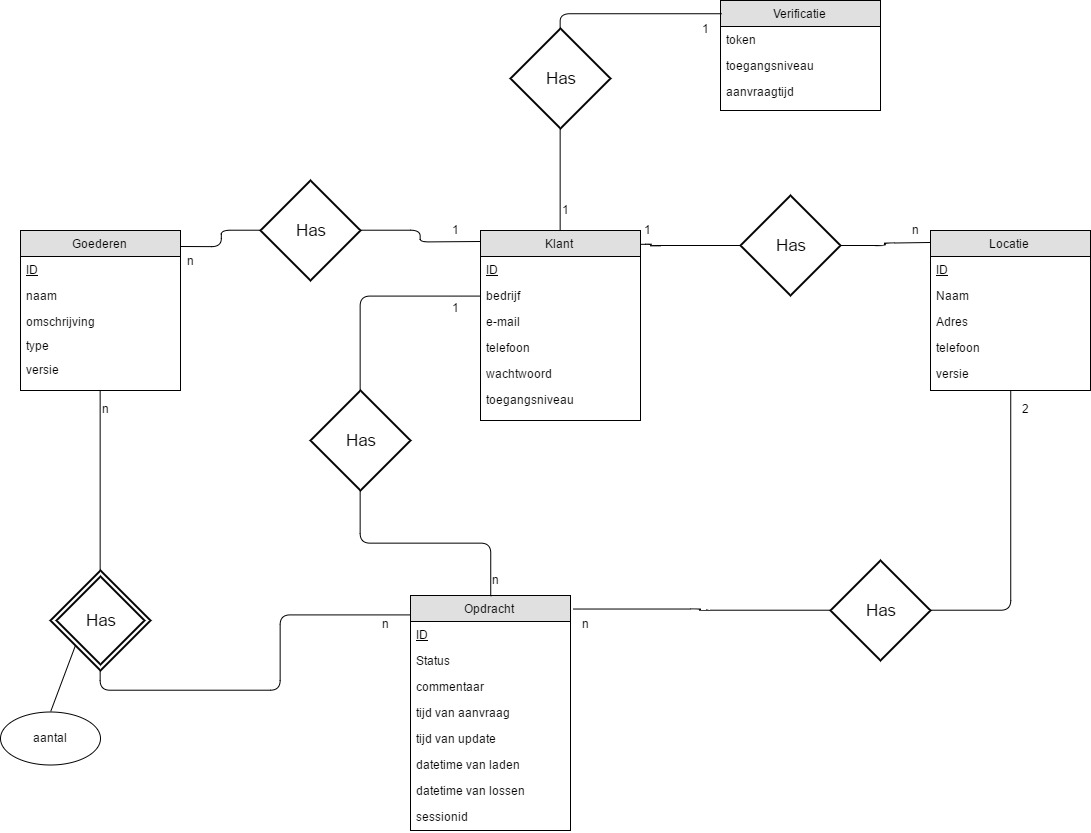
DATABANK

In dit onderdeel gaan we de structuur van de databank bespreken, zowel het ERD als de sql code. Ik ga niet overal in detail treden, maar slechts de minder voor de hand liggende relaties, of datatypes bespreken.

Om plaats te besparen ga ik verwijzen naar de create\_table.sql in de bijlagen.

Entity Relationship Diagram.



Hierboven ziet u her ER diagram van onze databank, met centraal de tabel klant. De naam is een beetje een misnomer, want in deze tabel worden de gegevens bijgehouden over alle gebruikersaccounts, dus niet enkel klanten van het transportbedrijf. De waarde van toegangsniveau is een getal van 0 tot en met 4. Deze waarden geven aan welke bevoegdheden een gebruiker heeft. 0 staat voor inactief/gebanned/verwijdert, 1 staat voor klant, 2 voor personeel, 3 voor admin, 4 voor superuser.

De rede waarom we verwijderde klanten niet regelrecht verwijderen uit de databank is omdat wanneer we dit doen, we problemen krijgen met de foreign keys uit andere tabellen die verwijzen naar een klant id. De transporteur zou dan ook niet meer in zijn geschiedenis de opdrachten van de verwijderde user kunnen opvragen. Om dit te vermijden zetten we de toegang van de gebruiker op 0. Een gebruiker met toegang 0 kan overigens niet meer inloggen.

Een admin heeft in deze versie enkel de rechten om gebruikers aan te maken, te beheren en te verwijderen.

Bij goederen wil ik de attributen type en versie toelichten.

Het type goederen wijst op de verpakking van de goederen. Op deze moment zijn er slechts 2 opties, doos en (euro)pallet. Dit lijkt vrij gelimiteerd, maar dit is afgestemd op de noden van onze opdrachtgever.

Versie wordt relevant wanneer de klant een product gaat aanpassen. Wanneer een klant een product aanpast gebeurt het volgende; er word een nieuw record aangemaakt met een nieuwe id, met de waarde die de klant heeft opgegeven en bij versie waarde 1, en bij de oude record word de versie op 0 gezet, om aan te duiden dat deze niet meer relevant is.

Wanneer de klant een product wil inladen worden enkel goederen weergegeven met versie waarde 1. Maar, en dit is waarom we de oude bijhouden, wanneer de klant of transportbedrijf een oude opdracht inkijken, wordt nog steeds het oude product weergegeven, aangezien de goederen via hun ID verbonden zijn met de opdracht.

Dit is analoog met het attribuut versie in de tabel locatie.

De goederen zijn met een many-to-many relatie verbonden met opdrachten, dit heeft tot gevolg dat er een extra table nodig is om deze succesvol te verbinden. Deze table heet opdracht\_has\_goederen en vindt u in de bijlage in onze create\_table.sql. Deze linkt de id’s van goederen samen met een aantal aan de id’s van opdrachten.

De relatie tussen opdrachten en locaties is ook wel opmerkelijk te noemen, hier gebruiken we een 2-to-many relatie, die toch wat ongebruikelijk is, maar in deze context werkt ze perfect. De 2-to-many wordt hier gebruikt omdat er in elke opdracht nood is aan 2 locaties: een voor het lossen en een voor het laden. Deze worden vertegenwoordigd in de sql file met los\_id en laad\_id.

Verder in de tabel opdracht ziet u ook session id staan. Dit was nodig omdat wanneer de opdracht aangemaakt wordt, er niet meteen goederen aan gekoppeld worden. Omdat we het id van de net gecreëerde record niet kennen, stoppen we er de session id bij als unieke key.

Wanneer het zo ver komt dat er producten bij de opdracht worden toegevoegd, word de opdracht ID geselecteerd waarvan de session id overeenkomt met dat van de gebruiker en die het recentst is aangemaakt.

Datetimepicker plugin

Voor alle instanties waar het nodig is een tijd en een datum in te geven maken we gebruik van een jquery plugin met de originele naam Bootstrap 3 datetimepicker v4. De volledige documentatie is te vinden op de volgende website: <http://eonasdan.github.io/bootstrap-datetimepicker/>

We gaan hier weer niet in detail, maar we gaan beknopt de requirements opnoemen en toelichten hoe we de plugin oproepen.

Om de plugin te kunnen gebruiken heb je jquery.js nodig, moment.js om parameters mee te kunnen geven en eventuele localisatie. Verder is het ook nodig om de bootsrap javascrip library te includen en de bootstrap css voor de opmaak. De plugin zelf is ook een js file en een css file die je moet includen.

Hieronder vind je de code die in de script tag moet staan van je html code, deze is afkomstig van createOrder.php.

1. $(**function** () {
2. $('#laaddatum').datetimepicker({
3. format: 'YYYY-MM-DD HH:mm',
4. stepping: 5,
5. minDate: moment()
7. });
8. $('#losdatum').datetimepicker({
9. format: 'YYYY-MM-DD HH:mm',
10. stepping: 5,
11. minDate: moment()
12. });
13. } );

Dankzij code weet de plugin op welke plaats hij de datetimepicker moet plaatsen, namelijk op de divisies met de id’s #laaddatum en #losdatum.

De parameter format bepaald welke gegevens je opvraagt in welke vorm, stepping geeft aan wat het kleinste verschil is tussen waarden. Dit is hier ingesteld op 5 minuten. Het is dus enkel mogelijk om minuten te selecteren die een veelvoud zijn van 5.

De minDate bepaald de kleinste datum die de gebruiker kan ingeven. Hier wordt deze ingesteld op de huidige datum/tijd met de functie moment().

Conclusie

Over het algemeen durven we zeggen dat we zijn geslaagd in het maken van een helder communicatiekanaal tussen klant en planningsafdeling. Hier en daar zijn we wel afgeweken van het vooraf opgestelde plan. Soms uit noodzaak, soms om een betere designkeuze te maken.

Een van de punten waar we sterk van zijn afgeweken was de mogelijkheid tot losse invoer in de pagina om een transportopdracht te maken. Het originele plan was om de klant de mogelijkheid te geven van locaties in te laden, maar hem ook de vrijheid te geven van de velden manueel in te voegen. Dit had als voordeel voor de klant dat wanneer hij slecht een keer ergens moest leveren, zoals bij een thuislevering, hij hier geen locatie template voor moest opslaan. Hierdoor hadden we in onze databank geen relatie tussen opdracht en locaties, en toen ook nog geen relatie tussen opdracht en goederen. Dit gaf als probleem dat we een grote hoeveelheid redundante informatie gingen opslaan in de tabel opdracht.

Toen we dit aanpasten zorgde dit wel voor een aanzienlijke vertraging, aangezien we onze databanken moesten herschrijven.

Dit is later nog een keer gebeurt, toen we ons realiseerden dat om meerdere producten te kunnen opslaan in de opdrachttabel, we een veel op veel relatie nodig hadden tussen opdrachten en goederen. Toen hebben we weer onze sql code grondig aangepast.

Waar we achteraf ook bedenkingen bij hebben is de admin user, nu kan die enkel gebruikers aanpassen, creëren en uitzetten. We denken dat het beter kan zijn om de admin ook dezelfde permissies te geven als de personeel accounts, aangezien de admin waarschijnlijk een interne werknemer is in het transportbedrijf en dus ook transportopdrachten zou moeten kunnen aanmaken.

Wat ik (wat hier volgt is de mening van Arno Willaert) ook heb ontdekt halverwege het project is dat wanneer ik hieraan opnieuw zou beginnen, zou ik zeker en vast gebruik maken van Angular.js om formulieren te verwerken, en responsieve elementen in de website te bouwen.

De jQuery plugins die we hebben gebruikt (datatables en datetimepicker) kunnen we alleen maar aanraden, ze waren relatief eenvoudig bij ingebruikname en configuratie.