Webwinkelproject

Plezier en nut met actuele webtechnologie

https://www.erichoekstra.com/webwinkel

# Inleiding en doel

Het doel van het project is dat ik kennismaak met webtechnologieën in een concrete toepassing. Het gaat om een webwinkel in artikelen die in de supermarkt verkrijgbaar zijn. Een webwinkel is een op het web relevante toepassing, die klanten en de bedrijfsprocessen op een hedendaagse wijze verbindt. De webwinkel biedt ook de mogelijkheid tot het kennismaken met technologieën die aan de client (browser) en de serverkant voorkomen. In dit project zijn dit:

|  |  |
| --- | --- |
| Primair | Secundair |
| 1. JavaScript, 2. AngularJS, 3. Node.js, 4. REST-services, 5. Api’s zoals de beroemde van Google. | 1. HTML, CSS, secundair omdat dit het terrein van designers is, 2. SQLite, omdat ik al voldoende bekend ben met databases/ SQL,   **Tertiar**   1. Php, omdat de taal conceptueel te zwak is om hier helemaal ‘los’ in te gaan. |

De aanpak kan pragmatisch blijven: ontwerp schrijven, prototype maken en module-voor-module uitvoeren.

# Webwinkel

## Eisen

Het antwoord op de vraag wat een webwinkel is, wordt gegeven door de groten, zoals Bol.com en Amazon.com. Het concept van deze winkels wordt gevolgd, maar natuurlijk met een eigen variant. De webwinkel gaat supermarktartikelen verkopen. De belangrijkste functionele eisen waarmee rekening wordt gehouden:

| Stakeholder | # | Toelichting |
| --- | --- | --- |
| Winkelier, klant |  | Bestellen kan met én zonder account. |
| Winkelier |  | De openingspagina is eenvoudig aanpasbaar zodat de winkelier hier real-time, zijn aanbiedingen en andere mededelingen kan doen. |
| Winkelier, klant |  | Ondersteuning van meerdere talen. |
| Winkelier, klant |  | Afrekenen kan op verschillende wijzen, zoals: acceptgiro, iDeal, PayPal, creditcard, enz. |
| Winkelier |  | De winkel ondersteunt productgroepen. Eén product kan tot meerdere groepen behoren. |
| Winkelier |  | De voorraadstand van een product is zichtbaar in de winkel. |
| Winkelier |  | De webwinkel biedt een voer-één-woord-in-en-vindt-alles zoekmogelijkheid. |
| Winkelier |  | Een lijstje met laatst bekeken producten ‘volgt’ de klant. |
| Winkelier |  | Klant kunnen onafhankelijk van het (mobiele) apparaat de winkel bezoeken. |
| Bedrijfsleider |  | De omzet, bestellingen en voorraden kunnen eenvoudig opgevraagd worden. |
| Beheerder |  | De winkel kan (tijdelijk) gesloten worden, zodat alleen een onderhoudspagina zichtbaar is. |
| Ontwikkelaar |  | De opmaak van de webpagina’s is gescheiden van de inhoud. |

Deze tabel is vanzelfsprekend fictief, omdat geen werkelijke stakeholderanalyse is uitgevoerd. De rollen spreken voor zich, behalve dat een winkelier ook een bedrijfsleider kan zijn, echter de laatste kijkt vooral bedrijfseconomisch naar de winkel.

## Winkelpui en -inrichting

Het opvallende aan webwinkels is dat deze vooral de productboom volgen. De wortel van de boom is niet zichtbaar en daarna volgen een aantal niveaus van productgroepen die steeds verfijnder worden. Aan het einde van de productgroeptakken hangen de bladeren, de producten die besteld kunnen worden.

Meerdere taxonomieën kunnen dezelfde producten indelen, zodat naast de standaardindeling ook een alternatieve indeling kan worden toegepast, zoals op aanbieding, jaargetijde, doelgroep, enz.

Hoewel productgroepen niet gekocht kunnen worden, hebben deze wel een eigen presentatiepagina.

Bepaalde takken zijn zwaar beladen met bladeren (producten) en daarom is een nader filter gebruikelijk, zoals op prijs of op specifieke kenmerken van de producten in kwestie.

Hoe weet de klant nu welk artikel hij moet kopen? Bijvoorbeeld door reviews van andere klanten. Een andere mogelijkheid is een chatsysteem.

Van eenmaal bestelde spullen moet een administratie worden bijgehouden: bestelhistorie, bestelstatus, retourneren, inruilen, enz. Dit leidt tot financiële boekingen die in een extern pakket verwerkt moeten kunnen worden.

## Algemene Verordening Gegevensbescherming

De AVG geeft betrokkenen controle over de persoonsgegevens die over hun zijn geregistreerd. Wat betekent dat voor dit informatiesysteem? Volgens de AVG zijn persoonsgegevens: “alle gegevens over een geïdentificeerde of identificeerbare natuurlijke persoon”. Deze webwinkel houdt in die zin persoonsgegevens bij:

1. naam, e-mailadres, foto, geslacht, taal;
2. of de klant geblokkeerd is, het tijdstip van de laatste inlog;
3. het verzend- en/of factuuradres;
4. alle aankopen van de klant;
5. zijn gedrag op de website middels de serverlog waarin een ip-adres is geregistreerd.

Het streven in dit ontwerp is om een stap verder te komen dan een privacyverklaring, maar om ook daadwerkelijk betrokkenen (klanten) controle over hun persoonsgegevens te geven. De AVG als een bron van eisen.

Hoe groot risico levert het bijhouden van deze gegevens voor de rechten en vrijheden van betrokkenen op? Het openbaar maken van de gegevens zonder tussenkomst van de betrokkene levert een bedreiging op voor zijn privacy. Het kan hem in ongemakkelijke situaties brengen waarvoor deze webwinkel niet verantwoordelijk wil zijn. Dit leidt tot technische en organisatorische maatregelen.

Deze webwinkel heeft persoonsgegevens nodig voor de verkoop van artikelen, het afhandelen van bestelde artikelen en het verlenen van klantenservice.

Alle bewerkingen van gegevens in een willekeurig informatiesysteem kunnen worden ingedeeld in de bekende categorieën: create, read, update en delete (crud). Hieronder is dat uitgewerkt samen met de consequenties voor de webwinkel.

|  |  |  |
| --- | --- | --- |
| Bewerking | AVG | Eisen aan de webwinkel |
| Create | Rechtmatigheid | De klant geeft toestemming voor de opname van zijn persoonsgegevens in de database. Die opname is de basis voor alle verdere verwerking in de zin van AVG.   * toestemming vragen   Door het plaatsen van een bestelling gaat de klant een overeenkomst aan met deze webwinkel. Op grond daarvan worden gegevens over die bestelling verwerkt.   * geen actie nodig; de verwerking is zondermeer rechtmatig |
| Doelbinding |  |
| Minimale gegevensverwerking |  |
| Opslagbeperking | Dit is een probleem voor statistieken! Volg je de wet zuiver, dan kun je alleen gegevens over bestelde producten bijhouden, dus niet voor welke klant en naar welk adres. |
| Informeren | Het registratiescherm informeert de klant alvorens hij toestemming geeft tot de verwerking. Dit geldt met name op het moment dat deze via Google gegevens overdraagt naar de webwinkel. |
| Read | Integriteit en vertrouwelijkheid |  |
| Inzage | Dit leidt tot twee schermen: (1) een overzicht van zijn klantgegevens, zoals naam, adres, enz. en (2) een overzicht over zijn bestellingen. |
| Overdraagbaarheid | Een export van zijn gegevens mogelijk maken in een nader te bepalen opmaak. |
| Update | Rectificatie |  |
| Juistheid |  |
| Delete | Vergetelheid |  |
| Beperking van de verwerking |  |

Deze webwinkel doet niet aan geautomatiseerde besluitvorming of profilering. Naast de verwerkingsverantwoordelijke is er één verwerker: een back-uplocatie in San Diego verzorgt door RSync.net.

# Gegevensmodel

Het gegevensmodel wordt als cruciaal beschouwd in de gehele oplossing. Het is een antwoord op de bovengenoemde eisen, voorzover de verwerking van gegevens een rol speelt, en het werkt door in heel de technische oplossing.

Het gegevensmodel opgenomen in dit document wordt bijgewerkt wanneer het model in de database ook bijgewerkt wordt. Echter, het model in de database mag meer entiteiten bevatten dan het hier getekende.

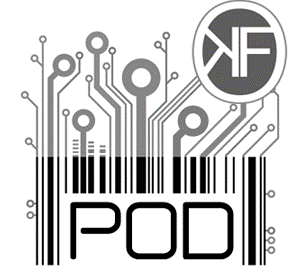
Van de volgende objecten wordt de levenscyclus (deels) vastgelegd in dit gegevensmodel.

|  |  |  |
| --- | --- | --- |
| product | klant | bestelling en -regels |
| retouren | verzending en -regels | adres |
| betaling | kortingsbon |  |

Omdat het gegevensmodel zo bepalend is voor het gehele systeem volgt hier een verantwoording van zaken die (niet) worden opgenomen in het model:

* Het is een webwinkel en geen erp-systeem, daarom zijn andere processen dan die direct nodig voor de verkoop uitgesloten, zoals: inkoop, voorraadbeheer, boekhouding, enz.
* Het verkoopproces wordt in zijn geheel gedekt, inclusief de debiteurenadministratie.
* Verkoopcontracten worden niet ondersteund, dus klanten moeten steeds opnieuw een bestelling plaatsen.
* Klanten kunnen géén artikelen bestellen die niet op voorraad zijn. Dus het systeem houdt geen backorders bij.
* Magazijnadministratiestukken, zoals pakbonnen, routelijsten, enz. worden niet door dit systeem gegenereerd.
* Het systeem slaat geen creditcardgegevens of soortgelijke gegevens op.

## Producten

De Open Knowledge Foundation werkt aan Open Product Data. De website biedt de mogelijkheid tot het downloaden van de door hun verzamelde gegevens: http://product-open-data.com/.

Het project ligt stil sinds 2015 ([bron](https://discuss.okfn.org/t/can-we-resurrect-product-open-data-com/173)), maar hoe dan ook: het is een behoorlijke verzameling gegevens die men inmiddels heeft aangelegd en ook een datamodel wordt aangeboden. De gegevens zijn gestructureerd in een stermodel. Uit de data is een selectie gemaakt en geïmporteerd als testgegevens in dit project.

Ieder product heeft een gtin-code en is ingedeeld in een gcp segment, familie, klasse of bouwsteen. De hiërarchie die in de gcp classificering zit, is overgenomen in de productgroep en productgroeptype tabellen.

## Voorraad

In deze oplossing is gekozen voor een vereenvoudigde voorraadadministratie. In een complexe situatie moet gewerkt worden met voorraadmutaties, maar deze oplossing beperkt zich tot het vastleggen van individuele productexemplaren. Het gevolg is dat producten alleen besteld kunnen worden wanneer deze als exemplaar geregistreerd zijn, wat suggereert dat deze werkelijk in de schappen liggen. Dat is in een complexe situatie ongewenst. Neem bijvoorbeeld de situatie dat een bedrijf een levering in de middag verwacht, dan kunnen deze pas dan ook verkocht worden, en ligt in de ochtend de verkoop stil. Het voordeel voor dit project is dat de oplossing een wat meer modelmatig en minder administratief karakter krijgt.

Producten die niet op voorraad zijn kunnen in dit project niet besteld worden, dus geen backorders.

## Adressen

Doorgaans werkt een webwinkel met twee adressen: één waarop de factuur gezet moet worden, of deze nu per post verstuurd wordt of niet, en een tweede waarnaar de bestelling verzonden moet worden. Omwille van de eenvoud wordt het definitieve verzendadres pas bepaald op het moment van verzending. Alle adreswijzigingen tot die tijd hebben dus effect op de distributie.

## Verzendkosten

### Berekening, constraint 4

|  |  |  |
| --- | --- | --- |
| Verzendkosten | | |
| prijs | afstand | kosten |
| € 0,00 | 0 meter | € 1,00 |
| € 0,00 | 10.000 | € 1,10 |
| € 0,00 | 40.000 | € 1,40 |
| € 0,00 | 70.000 | € 1,70 |
| € 15,00 | 0 | € 0,00 |
| De combinatie van afstand en prijs moet uniek zijn. | | |

Deze webwinkel laat de verzendkosten afhangen van de afstand tussen het verzendadres en het distributiecentrum aan de Atoomweg 60 in Utrecht. Hoe verder weg de klant woont hoe hoger de bestelkosten, maar wanneer voor meer dan 15 euro besteld wordt, dan zijn de verzendkosten voor rekening van de winkel. Dit wordt vastgelegd in een staffel genaamd ‘Verzendkosten’. Rechts een voorbeeld daarvan.

Voor het bepalen van de verzendkosten wordt eerst begonnen met de prijs van de bestelling, in het voorbeeld kan deze in één van de twee categorieën, 0 tot 15 euro of 15 euro of meer, vallen. Daarna wordt binnen de categorie naar de afstand tot het verzendadres gekeken. Bij een prijs van 0 tot 15 euro is dat bijvoorbeeld 1,10 euro wanneer de afstand in de (sub-)categorie vanaf 10 tot 40 km valt.

Logisch gezien valt dit uiteen in twee onderdelen.

Dit is ten eerste de lineaire ordening in de productverzameling prijs afstand: . Een tupel gaat vooraf aan wanneer . In de verzameling uit het voorbeeld gaat vooraf aan en alle andere elementen, vooraf aan , enz.

Ten tweede een equivalentierelatie in prijs afstand gedefinieerd door de elementen van . Ieder element uit definieert een equivalentieklasse. Een zekere bestelling valt in dezelfde klasse als wanneer en er geen andere gevonden kan worden waarvoor geldt .

De verzendkosten zijn de kosten vastgelegd bij de overeenkomstige equivalentieklasse.

Is ordening een lineaire ordening? Een ordening is lineair wanneer elk tweetal elementen vergelijkbaar is. Gegeven de tupels en , dan moeten deze altijd vergelijkbaar zijn, dus gaat vooraf aan of gaat vooraf aan . Wanneer niet geldt dat , wat de relatie gelijk zou leggen, dan moet gelden, waarbij de relatie alsnog gelegd wordt, of anders geldt . Wanneer geldt dat moet waar zijn, en als dat niet waar is, dan geldt altijd , want levert per definitie een lineaire ordening op, maar dan moet ook gelden, wat per definitie waar is, omdat de volgorde van de termen rond het -teken niet uitmaakt.

De prijzen en kosten zijn gebroken positieve getallen, dus een deelverzameling van . De afstanden zijn deelverzameling van de natuurlijke getallen, dus van .

### Technische oplossing

De verzendkosten zijn afleidbaar uit andere gegevens in de database en omdat je een view kunt maken, en die is ook gemaakt, met voor iedere bestelling de verzendkosten, zou bij iedere bevraging van de bestelling een join met die view voldoen. Dat is echter wel een heel dynamische relatie tussen de verzendkosten en een bestelling, en daarom wordt de view alleen gebruikt voor het bepalen van de primaire sleutel van de verzendkosten, die vervolgens wordt overgenomen in de bestelling.

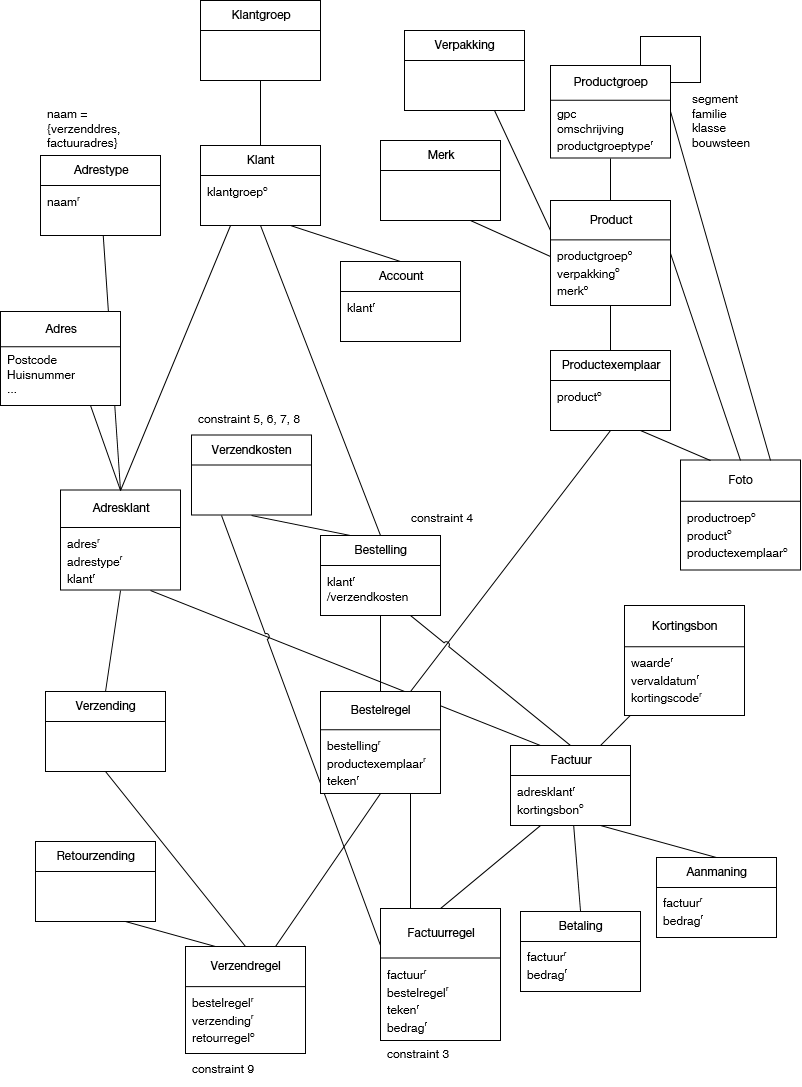
Een probleem is dat staffel met verzendkosten zal wijzigen in de toekomst. Dat betekent dat wanneer een relatie is gelegd tussen een verzendkostenrecord en een bestelling, het verzendkostenrecord niet meer gewijzigd mag worden. Om wijzigen van verzendkosten toch mogelijk te maken moet iets als een ‘verzendkostenplan’ ingevoerd worden. Eén bepaald plan is dan actief, waardoor nieuw versies van de bovengenoemde staffel ingevoerd kunnen worden. Die oplossing is verder niet uitgewerkt in deze webwinkel.

Wanneer een factuur wordt aangemaakt, dan wordt voor de verzendkosten een factuurregel aangemaakt. Dat betekent dat een factuurregel verbonden is aan een bestelregel of een ‘verzendkostenregel’ en niet gelijktijdig aan beiden.

Technisch blijft nog het probleem van de onzekerheid in de bepaling van de afstand, omdat hiervoor de Google Maps API Distance Matrix wordt gebruikt. Iedere keer wanneer een klant een adres invoert, dan wordt die api geraadpleegd. Maar wat als die geen antwoord geeft? Een afstand van 0 km wordt dan maar aangenomen. In deze webwinkel, die als studieobject dient, is dat geen probleem, maar wanneer de eisen hoger liggen, dan kan dat niet. Dan zouden bijvoorbeeld alle adressen in de tabel ‘adres’ bij voorbaat verrijkt moeten worden met de afstand. Of berekent de winkel de hemelsbrede afstand met de x-y-coördinaten die bij sommige postcodetabellen geleverd worden.

## Diagram

Het gegevensmodel. Het origineel wordt [hier](https://1drv.ms/u/s!Ahx0RA94MlWOoEh2DjSCdSmqdxXU) gevonden en kan worden bewerkt met [draw.io](https://www.draw.io/). Zie bijlage voor een toelichting op de gebruikte techniek.



### Constraints

Constraints op het gegevensmodel geformuleerd in RuleSpeak. Een eenmaal toegekend nummer wordt niet meer gewijzigd, wegens vermelding in de broncode (traceability).

| # | Constraint | Realisatie |
| --- | --- | --- |
| 3 | De factuurregel moet bij dezelfde bestelling horen als de bestelregel waarvoor deze aangemaakt is.  context: factuurregel invariant: self.bestelregel.bestelling.nummer = self.factuur.bestelling.nummer | In de database: bij het vastleggen van een factuurregel, bij het wijzigen van een bestelregel en bij het wijzigen van een factuur. Alle drie keer in een trigger. De triggers maken gebruik van een view genaamd Factuurregel\_constraint3. |
| 4 | De verzendkosten moeten worden afgeleid uit de afstand tussen het distributiecentrum en het verzendadres, en uit de prijs van de bestelling.  context: bestelling derive: self.verzendkosten = “Verzendkosten(Kosten)”  Zie pagina . | In de database: bij het toevoegen van een adresklant, indien een afstand gegeven is, bij het wijzigen van de afstand in adresklant, bij het toevoegen of verwijderen van een bestelregel.  In het systeem komt het wijzigen van een bestelregel niet voor; alleen toevoegen en verwijderen.  Het verwijderen van een adresklant heeft geen betekenis, tenzij het het laatste adres is, maar ook het verwijderen van een adresklant komt in dit systeem niet voor.  De triggers maken gebruik van de view Bestelling\_constraint4. |
| 5 | De combinatie van afstand en prijs in de verzendkosten-tabel moet uniek zijn. | Een unieke index aangemaakt genaamd Verzendkosten\_constraint5. |
| 6 | De prijs en de kosten in de verzendkostentabel zijn een gebroken positieve getallen inclusief 0. | Aan de tabel Verzendkosten zijn de constraints: Prijs\_constraint6 en Kosten\_constraint6 toegevoegd. Deze testen op .  Idem. aan de tabel Bestelling\_Prijs\_Afstand\_Invoer. |
| 7 | De afstand in de verzendkostentabel is een natuurlijk getal groter dan of gelijk aan 0. | Aan de tabel Verzendkosten is de constraint Afstand\_constraint7 toegevoegd. |
| 8 | De verzendkostentabel bevat een rij met prijs 0,00 en afstand 0 en een zekere waarde voor de verzendkosten. | Drie triggers op de tabel Verzendkosten: na het toevoegen, na het wijzigen en na het verwijderen.  De triggers maken gebruik van de view Verzendkosten\_constraint8. |
| 10 | Een factuurregel is afgeleid van een bestelregel of van een verzendkostenregel, maar nooit van beiden. | Niet vastgelegd in het model. |
| 11 | In de factuurregels die bij een zekere factuur horen, komt maximaal één regel voor die afgeleid is uit de verzendkosten. | Niet vastgelegd in het model. |
| 1 | De verzending is naar het verzendadres dat bij de klant van de bestelling is vastgelegd. | Niet vastgelegd in het model. |
| 2 | Een bestelling mag alleen aangemaakt worden indien een verzendadres bekend is. | Niet vastgelegd in het model. |
| 9 | Het adres van de klant vastgelegd bij de verzending moet tot dezelfde klant behoren als de bestelling. | Niet vastgelegd in het model. |

# Technische oplossing

## Overzicht

De volgende thema’s spelen een rol in het bepalen van de technische oplossing.

|  |  |  |
| --- | --- | --- |
| AngularJS, of Angular | Databaseontwerp | Winkelwagentje |
| JavaScript of TypeScript | Indelen van de applicatie in modulen | Betalen |
| Node.js | Status beheersen in de applicatie | REST API[[1]](#footnote-2) |
| SQLite database, DBMS | Loginprocedure | Compatibiliteit browsers |
| Bootstrap |  |  |

De algemene architectuur van de applicatie:

* Een webapplicatie.
* De database is SQLite3. Dit is voor het doel van deze toepassing ruim voldoende en kan, indien nodig nodig, worden gewijzigd naar bijvoorbeeld MySQL.[[2]](#footnote-3)
* De gebruikersinterface wordt geheel door de webbrowser verzorgt. In de hele interface wordt geschreven in AngularJS of Angular, en dus in JavaScript en HTML5.
* De serverlogica wordt geprogrammeerd in Node.js. PHP is op deze manier niet nodig.[[3]](#footnote-4)
* Angular dwingt werken volgens het MVC-patroon, dus dat is geen keuze. Node.js lijkt ook het beste te werken wanneer de applicatie volgens MVC wordt opgezet, en dat wordt dus gevolgd.
* Node.js, één router, maar meerdere controllers, bijvoorbeeld per deelgebied één, zoals een controller voor productfiltering, één voor het inloggen, één voor de winkelwagen, enz.
* Node.js, één module voor een resource.
* Bootstrap is gebouwd op flexbox, dus zal de gebruikersinterface op flexbox gebasseerd zijn, wat goed aansluit bij de wens dat de winkel ook op mobiele devices werkt.

Het lijkt wel dat webwinkels gemaakt zijn door mensen met kennis van webdesign óf door mensen die graag programmeren, maar nooit door beiden. Ook in dit project ligt het accent op één van beide: op de functionaliteit en minder op de flitsende vormgeving van de webpagina’s.

## Inlogprocedure en bestellen

Nee, niet de zoveelste eigen inlogprocedure, maar aansluiten bij OpenID Connect, https://openid.net/, https://developers.google.com/identity/protocols/OpenIDConnect. Hierdoor worden ook geen inlog­gegevens op een toch te slecht beveiligde server bewaart.

Bestellen als een gast moet mogelijk zijn. Wanneer iemand zijn e-mailadres voorkomt in de database, dan wordt hem een inlog aangeboden. Maakt iemand daar geen gebruik van, dan wordt bestelling genoteerd alsof het nieuwe klant is, omdat inloggen enige manier waarop een klant geauthentiseerd kan worden. De beslissing wordt genomen op het moment dat de bestelling wordt geplaatst, dus op het moment dat de klant het complete overzicht van de bestelling, inclusief verzendkosten, voor zich ziet en akkoord gaat. Dat is nog voor het afrekenen.

Overigens komt inloggen voor klanten pas aan de orde tijdens het bestelproces. Wanneer iemand al iets besteld heeft als anonieme klant, en hij logt in, dan wordt zijn bestelling overgeheveld naar zijn account. Hetzelfde geldt voor adressen die reeds ingevoerd zijn. Ook die worden overgenomen.

## Identificatie van de client en bestellen

De database kent een hiërarchische opbouw, zoals de klant die boven een bestelling staat en de bestelling die boven de bestelregel staat. Wanneer een klant een bestelling plaatst moet ten eerste de klant bepaald worden, daarna de bestelling en vervolgens kan een bestelregel toegevoegd worden. Dat is echter niet waarop de client werkt: die stuurt een product en een aantal naar de server en verwacht dat deze het verder verwerkt tot een bestelling.

De server reageert steeds met een poging waarin een bestellingnummer wordt vastgesteld. De client kan bijvoorbeeld eerder een bestelregel (product en aantal) hebben gezonden. Dan kan het nummer uit de sessie (cookie) worden gehaald. Wanneer dat niet zo is, en het nummer ook niet op een andere manier bepaald kan worden, dan moet een nieuwe bestelling aangemaakt worden. Deze procedure is uitgewerkt in de tabel op pagina .

Een probleem ontstaat echter wanneer de client zeer kort achter elkaar twee bestelregels naar de server stuurt. De eerste bestelregel veroorzaakt een nieuwe bestelling. Maar het nummer van die bestelling is bij aankomst van de tweede bestelregel nog niet met de client gecommuniceerd.[[4]](#footnote-5) Het gevolg is dat voor de tweede bestelregel ook een nieuwe bestelling wordt aangemaakt. Wanneer de client vervolgens de lopende bestelling opvraagt ziet deze alleen de tweede bestelling en is de eerste bestelregel verloren gegaan! Dit is één van de ergste dingen die in een webwinkel kan gebeuren.[[5]](#footnote-6)

Het probleem is terug te brengen naar de anonimiteit van de client en de mogelijkheid om parallel bewerkingen te laten uitvoeren op de server. Een aantal oplossingsrichtingen:

* Zou een bestelling eerst samengesteld worden op de client en daarna in één keer verzonden, dan bestaat dit probleem niet. Wel ontstaan dan weer nieuwe problemen van een grotere orde, zoals klanten die gelijktijdig de laatste productexemplaren bestellen.
* Ingelogde gebruikers kennen dit probleem niet, doordat de client gedwongen wordt zich eerst te identificeren bij de server alvorens een eerste bestelregel verzonden kan worden.
* Met het eerste elektronisch contact met een niet-ingelogde gebruiker kan ook een identificatie worden meegestuurd. Dan wordt verondersteld dat die identificatie ruim voor de eerste bestelling wordt opgeslagen in de sessie, zodat bij de eerste bestelregel deze al meegestuurd kan worden. Dat is manier waarop php werkt.
* Als de client de bestelregels serieel naar de server overbrengt, of tenminste de eerste twee serialiseert, dan kan bij de tweede bestelregel in de sessie het bestellingnummer van de eerste worden meegestuurd.

Gekozen is voor de laatste oplossing. Dit stelt een extra eis aan de client die de API minder generiek maakt, maar in dit project is dit de eenvoudigste en een betrouwbare oplossing.

## Foto’s

SQLite3 ondersteunt het opslaan van binaire gegevens met de type affinity blob. Foto’s zullen in dat type worden opgeslagen maar wel in een aparte tabel. Op deze manier kan deze gemakkelijker afgesplitst worden indien de verzameling te omvangrijk wordt. Zie gegevensmodel.

De foto’s zijn een aparte resource in de API. Door een get-request op de foto-resource samen met de primaire sleutel van de foto kan deze worden opgevraagd. Overwogen is een oplossing met zg. data-url’s, maar deze loopt tegen ongespecificeerde beperkingen van de grootte van de afbeelding.

## Analytics

Aan het gedrag van de bezoeker in deze webwinkel wordt geen speciale waarde toegekend. Met andere woorden: het wordt niet geanalyseerd.

Wel wordt een log bijgehouden van bezoekers in de vorm van een ip-adres en tijdstip. Natuurlijk heeft de webserver Node.js ook een weblog met alle requests, die bijvoorbeeld met GoAccess (populaire Linux tool) tot een rapport gemaakt kunnen worden.

## Deployment

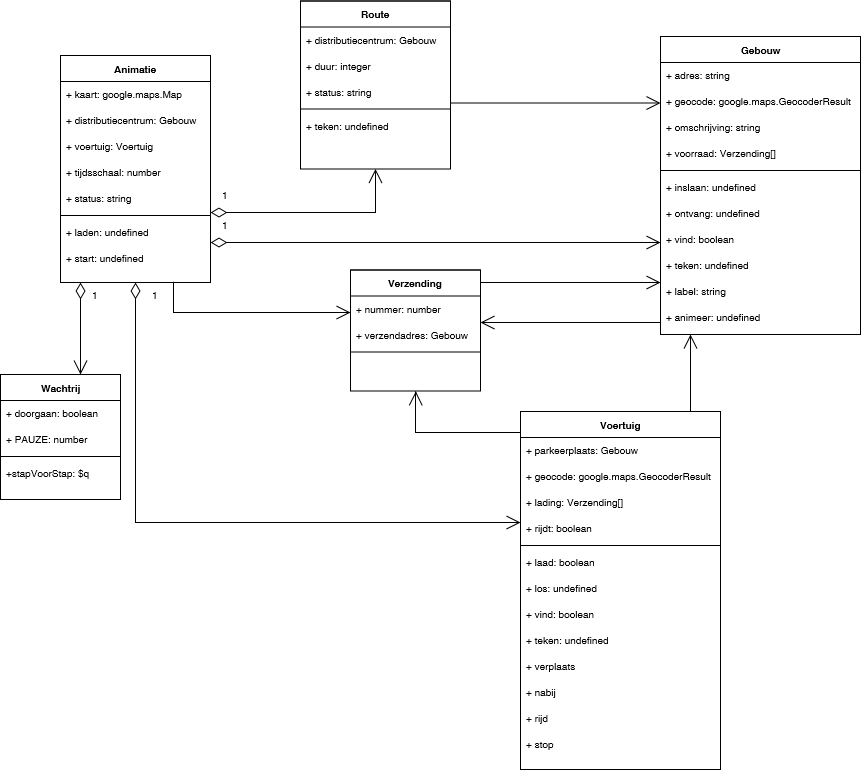
Deployment van de ontwikkelomgeving naar de productie zoveel mogelijk met npm. Op de client is geen sprake van een deployment in het bereik van dit project, maar wel worden eisen gesteld aan de browsers op de client. Deze moeten HTML5 ondersteunen. Dit zal worden ‘afgedwongen’ door een test op de HTML5-features van de browser die verbinding maakt, om vervolgens deze uit te sluiten wanneer de test faalt.

## Simulatie van de distributie

Wanneer in het datamodel voor een bestelregel een verzendregel wordt aangemaakt, dan start de distributie. Met de Google Maps JavaScript API kan een simulatie worden gemaakt van een vrachtauto die op een kaart een rondje rijdt langs de klanten. De vrachtauto laadt verzendingen die overeenkomen met de verzendregels. Google Maps berekent de optimale route en vervolgens laat de software de vrachtauto over de route rijden.

Wanneer de auto bij een verzendadres aankomt, dan moet een verzending worden afgeleverd. Dat wordt gerealiseerd met een aanroep naar de API die in het datamodel Verzending(afgeleverd) bijwerkt.

Een simulatie leent zich uitstekend voor een object-georiënteerde aanpak. Hieronder is het klassenmodel afgebeeld.



Het model wordt gemaakt in de methode $scope.distribueer(). Die methode wordt getriggert door de Distribueer-knop in het scherm. De methode converteert verzendregels naar bovenstaande objecten, uitgezonderd het adres van het distributiecentrum, dat uit de configuratie van de client afkomstig is.

Het gebouw heeft een dubbelrol in het model. Ten eerste dient het als de locatie waar iets is, of moet zijn. Een verzending heeft een referentie naar het distributiecentrum, waar het is, en één naar het verzendadres, waar het moet zijn. Het voertuig wordt aangemaakt bij een gebouw, de locatie waar het geparkeerd staat. Een route begint altijd bij het distributiecentrum. De andere rol van het gebouw is een plaats waar voorraad gestald kan worden. Een voertuig laadt verzendingen bij een gebouw, het distributiecentrum, en lost deze bij een gebouw, het verzendadres.

Hoewel intensief gebruik wordt gemaakt van klassen uit de Google Maps JavaScript API, hoeven deze bij aanvang niet geïnstantieerd te zijn, de objecten maken deze zelf aan, behalve één object van de klasse google.maps.Map. Dat object representeert de kaart. Dat Map-object wordt aangemaakt in een directive waarin het aan een HTML-element verbonden wordt.

De verzendregels worden door de wijze waarop de $resource-service van AngularJS werkt, steeds als een snapshot uit de API gehaald. Het gevolg is dat bovenstaand model na iedere update opnieuw moet worden opgebouwd. In een toekomstige versie zouden de verzendregels ook ‘opgevangen’ kunnen worden in een objectmodel, waardoor bijvoorbeeld ook meerdere voertuigen verschillende verzendingen gelijktijdig kunnen distribueren.

# Plan van Aanpak

De aanpak zal worden gevolgd. Het ligt voor de hand dat database steeds zo goed mogelijk uitgewerkt is en voldoende testgegevens bevat. Andere zaken kunnen meer iteratief uitgevoerd worden, zoals bijvoorbeeld eerst de winkelpui schrijven (producten) en daarna de login.

1. Database, DDL schrijven. In zijn geheel.
2. Database bijwerken met basisgegevens (de eerder genoemde productdatabase)
3. De productpresentatie realiseren
   1. REST-services in Node.js schrijven.
   2. Client schrijven.
4. Het bestelproces realiseren, idem. REST-services en client.
5. Managementinformatie realiseren: voorraadstand, verkoopcijfers, bezoekcijfers, enz. in het portaal voor de bedrijfsleider.

# Test

De belangrijkste vraag over een test is de dekking. In welke mate toont de test de gespecificeerde werking aan. Veel zaken zijn niet gespecificeerd maar vallen onder een goede werking. Een wiskundig bewijs dat de programmatuur werkt zoals gespecificeerd, of verwacht, is meestal niet haalbaar. Men kan tenminste equivalentieklassen onderscheiden en daarvoor testgevallen opstellen. Daarvoor zouden beslissingstabellen gebruikt kunnen worden. Als ook dat niet lukt of zinvol is, dan kunnen tenminste enkele uitspraken gecontroleerd worden. Die uitspraken moeten zo dicht mogelijk bij de specificatie of verwachting liggen. Een formulering in RuleSpeak ligt dan voor de hand (zie bijlage).

* ~~correctheid bewijzen~~: logica;
* equivalentieklassen en testgevallen: beslissingstabellen;
* uitspraken controleren: RuleSpeak.

De test op de server wordt uitgevoerd met Mocha en van de client met Jasmine. In de tabel hieronder een verantwoording van de tests die geschreven zijn. Niet alle onderdelen worden in dit project getest, omdat het doel van het project niet een perfect werkend systeem is.

| Subsysteem | Component | Onderdeel | Dekking | Uitvoering |
| --- | --- | --- | --- | --- |
| database | DML |  |  |  |
| DDL |  |  |  |
| server | model | product |  | Mocha |
| service |  |  |
| controller |  |  |
| router |  |  |
| client | service |  |  | Jasmine |
| controller |  |  |
| view |  |  |

# Wensen

De volgende wensen staan nog open.

|  |  |  |
| --- | --- | --- |
| # | Deelsysteem |  |
| 1 | Klant-client | In het scherm met productgroepen komen lege productgroepen voor. Dat is een correcte weergave van de database, maar ongewenst in de gebruikersinterface. |
| 2 | Klant-client | Je moet kunnen klikken op een plaatje. |
| 3 | Database | Is de historie in de tabel adresklant een probleem? |
| 4 | Klant-client | Houdt de laatst bekeken producten bij van een klant, bijv. met local-storage, en toon deze op het scherm. |
| 5 | Klant-client | Introduceer een ‘mogelijk ook interessant voor u’ nadat iemand iets in de winkelwagen legde. |
| 6 | API | Maak de REST-API discoverable. |
| 7 | Klant-client | Geef een anonieme klant een hyperlink waarmee hij zijn bestelling kan bekijken/ volgen. |
| 8 | Klant-client | Toon een foto bovenin de e-mail die de bestelling bevestigd, i.p.v. de huidige kop. |
| 9 | Klant-client | Het scherm geeft verkeerde feedback wanneer een item is uitverkocht en deze toch gekozen wordt. |

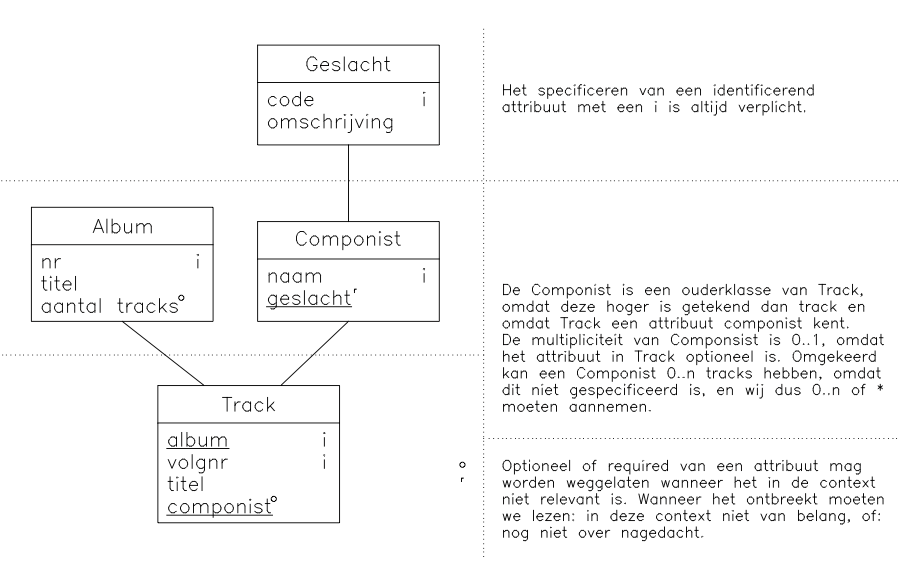
# Bijlage I – Informatiediagram

Uit MDD-cursus gevolgd aan de OU

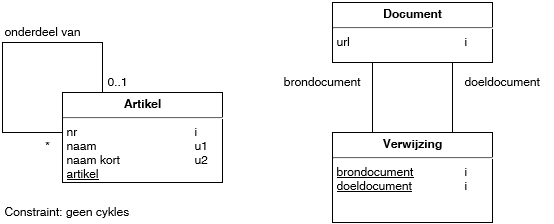
De cursus gebruikt een UML-profile van het klassendiagram. Een klassendiagram in een PIM kan vertaald worden naar een relationeel PSM. Dat relationeel PSM wordt getekend als een strokendiagram. Ook dat diagram wordt hier beschreven.

## Tekenconventies

Een aantal afspraken over het UML-profile van het klassendiagram dat in deze cursus wordt gebruikt. Zie pagina 89 van cursusboek 1.



In deze notatiewijze is één van beide klassen de ouder, en daarmee de ander het kind. In opgave 3.8 van het lesboek is een situatie dat werknemers voor een filiaal werken, maar dat een filiaal geleidt wordt door een werknemer (nl. de bedrijfsleider). In zo'n situatie kies je één van beide klassen als ouder, bijvoorbeeld het filiaal. De situatie is dan correct voor de associatie tussen het filiaal en de werknemer. Om nu aan te geven dat een werknemer één of meer filialen kan leiden als bedrijfsleider worden de multipliciteit op de verkeerd-om-associatie expliciet vermeld. Zie uitwerking op pagina 110 van het boek.

De linkerfiguur toont twee uniciteitsregels die apart gespecificeerd zijn als u1 en u2. De inhoud van beide attributen is onafhankelijk van elkaar uniek, zou alleen u genoteerd zijn, dan moet de combinatie uniek zijn. Verder toont de figuur de constraint die voorkomt dat een artikel zichzelf (via een omweg) als ouder krijgt. In de rechterfiguur is de consistente notatie van de rolnaam met de attribuutnaam afgebeeld. In deze situatie (netwerkpatroon) kan niet volstaan worden met document als attribuut in Verwijzing, daarom moet gebruik worden gemaakt van de rolnamen.

Een veel-op-veel associatie uitmodelleren met een nieuwe klasse, of niet? Wanneer die nieuwe klasse geen attributen draagt anders dan de verwijzingen, is het niet nodig voor het presenteren van een ontwerp. Wanneer het ontwerp vertaald wordt naar Cathedron, dan natuurlijk wel.

De plaatsing van de klassen in het vlak van het diagram heeft een betekenis, zoals hierboven beschreven, maar die betekenis verdwijnt in Cathedron. Daar tellen alleen nog de sleutelverwijzingen naar andere klassen.

## Afleidingsregels

Attributen waarvan de waarde wordt berekend worden genoteerd als '/bedrag' aanvullend wordt er een afleidingsregel opgenomen in het model. Wanneer 'i/bedrag' wordt genoteerd, dan is het bedrag initieel afleidbaar wat betekent dat het later kan worden overschreven met een andere waarde die niet voldoet aan de afleidingsregel.

Ook een klasse als geheel kan onderworpen worden aan een afleidingsregel. Ook dat wordt genoteerd met een /, maar dan voor de klassenaam, bijvoorbeeld: '/Resultaat'. Dit betekent dat de instanties van die klasse geheel afleidbaar zijn uit andere instanties in het model.

# 

# Bijlage II – Beslissingstabel bestellingnummerservice

Wanneer een client contact maakt met de server voor het bewerken of plaatsen van een bestelling moet het lopende bestellingnummer bepaald worden. Dat nummer is de primaire sleutel van de tabel Bestelling. Het nummer kan soms gegeven worden door de client, bijvoorbeeld wanneer de bestelhistorie opgevraagd wordt, maar het nummer kan ook onbekend zijn voor de client, en moet dan door de server bepaald worden. De onderstaande tabel beschrijft de wijze waarop het nummer bepaald kan worden.

| **C** | **Conditie** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **D10** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Is een bestellingnummer gegeven in de url van de request? | J | J | J | N | N | N | N | N | N | N |
| 2 | Is een bestellingnummer gegeven in de body van de request? | - | - | - | J | J | J | N | N | N | N |
| 3 | Is een bestellingnummer bekend in de sessie? | - | - | - | - | - | - | J | N | N | N |
| 4 | Is de client ingelogd? | J | J | N | J | J | N | - | J | J | N |
| 5 | Hoort een zeker bestellingnummer bij de ingelogde gebruiker? | J | N | - | J | N | - | - | - | - | - |
| 6 | Is van de ingelogde gebruiker een ongefactureerde bestelling bekend? | - | - | - | - | - | - | - | J | N | - |
| **A** | **Actie** |  |  |  |  |  |  |  |  |  |  |
| 1 | Gooi een fout op. | - | × | × | - | × | × | - | - | - | - |
| 2 | Neem het opgegeven bestellingnummer uit de url van de request over. | × | - | - | - | - | - | - | - | - | - |
| 3 | Neem het opgegeven bestellingnummer uit de body van de request over. | - | - | - | × | - | - | - | - | - | - |
| 4 | Neem het bestellingnummer uit de sessie over. | - | - | - | - | - | - | × | - | - | - |
| 5 | Neem het bestellingnummer van de meest recente ongefactureerde bestelling over. | - | - | - | - | - | - | - | × | - | - |
| 6 | Maak een klant aan. | - | - | - | - | - | - | - | - | - | × |
| 7 | Maak een bestelling aan. | - | - | - | - | - | - | - | - | × | × |
|  | **Controle** |  |  |  |  |  |  |  |  |  |  |
|  | Controlegetal = 26 = 64 = de som rechts | 8 | 8 | 16 | 4 | 4 | 8 | 8 | 2 | 2 | 4 |

# Bijlage III – RuleSpeak

Uit MDD-cursus gevolgd aan de OU

RuleSpeak is een verzameling richtlijnen voor het formuleren van bedrijfsregels in natuurlijke taal. Het perkt de vrijheid van natuurlijke taal in.

Modaliteiten of modale begrippen: begrippen zoals uitgedrukt door de woorden: toegestaan, verboden, mogelijk, noodzakelijk, moeten en hun ontkenningen. Op talloze manieren kan met deze woorden dezelfde modaliteit worden uitgedrukt, bijvoorbeeld dat meer dan 5 commissariaten voor een manager niet is toegestaan. Een modaliteit die altijd en overal geldt kan op verschillende manieren uitgedrukt worden, en daardoor onduidelijkheid geven, maar helemaal problematisch wordt het gebruik van condities. Wat als de conditie niet geldt? Wordt dan de modaliteit bepaald door het omgekeerde van wanneer de conditie wel geldt?

In 2005 heeft de Object Management Group (OMG) de Semantics of Business Vocabulary and business Rules (SBVR) goedgekeurd. SBVR is daarmee de standaard voor het formuleren van bedrijfsregels geworden, maar het is geen syntaxis voor bedrijfsregels. RuleSpeak is dat wel en voldoet aan SBVR. RuleSpeak is opgesteld door Ronald G. Ross.

RuleSpeak gaat over bedrijfsregels. Een bedrijfsregel is altijd op de volgende manier geformuleerd. Dit zijn de basisbeginselen.

* De formulering bevat één van de sleutelwoorden: moet, mag niet of mag alleen. Alleen mag is slechts een advies en geen bedrijfsregel.
* Het onderwerp staat aan het begin van de zin. Je moet dus gelijk duidelijk maken waarop de regel van toepassing is.
* Ruimte: bepalingen over situaties waarop zij van toepassing zijn (meestal) aangeven met indien.
* Tijd: een regel is op een bepaald moment of in een bepaalde periode van toepassing: wanneer.

Samengevat: moet, mag niet, mag alleen, onderwerp-begin, indien en wanneer.

1. Een REST API is meer dan bedienen van de HTTP methoden. [↑](#footnote-ref-2)
2. SQLite3 bleek gaandeweg een gelukkige keuze. Het DBMS is buitengewoon eenvoudig opgebouwd, waardoor ook een eenvoudige oplossing afgedwongen wordt. SQLite3 ondersteunt bijvoorbeeld geen stored procedures, waardoor ik gedwongen werd het MVC-ontwerp beter te volgen. [↑](#footnote-ref-3)
3. Niets mis met PHP, maar JavaScript (Node.js) is conceptueel een stuk sterkere taal. ‘Downgraden’ naar PHP kan altijd nog. [↑](#footnote-ref-4)
4. Het wordt in de response teruggestuurd en als cookie in de client opgeslagen. Bij de volgende request stuurt de client dat cookie mee. [↑](#footnote-ref-5)
5. Hetzelfde probleem doet zich ook voor wanneer een klant begint met het registreren van zijn adressen, maar daar heeft niemand last van. [↑](#footnote-ref-6)