Goudgeel

INFO SUPPORT B.V.

APPLICATIEBEHEER

Contents

[Gemaakte afwegingen/ontwerpen](#h.30j0zll)

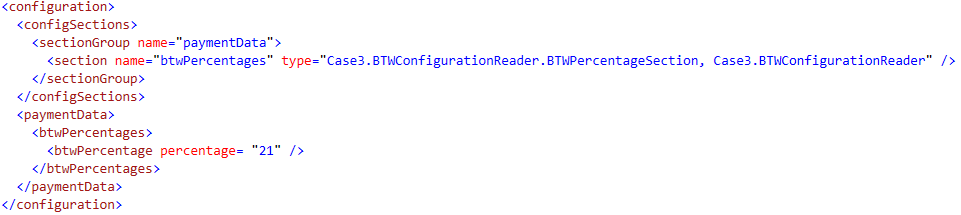
[BTWCalculator](#h.koo2wraz1qjv)

[Testhandleiding (alleen indien nodig)](#h.1fob9te)

# **Gemaakte afwegingen/ontwerpen**

Dit hoofdstuk beschrijft de gemaakte afwegingen/ontwerpen die team Gougeel gemaakt heeft.

## BTWCalculator

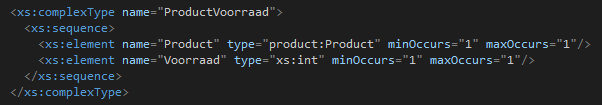


In de configuratie moet bovenstaande XML neergezet worden. De klasse BTWCalculator leest het BTW-percentage uit en gebruikt dit voor zijn berekeningen. We hebben hiervoor gekozen, zodat het BTW-percentage overal in de applicatie op een eenvoudige manier berekend kan worden. De keuze voor het plaatsen van het BTW percentage in een configuratie bestand is gemaakt op  basis van de wens van de product owner. Het BTW-percentage is veranderlijk en de product owner wilde dat dit eenvoudig aangepast kon worden, zonder dat dit veel geld zou kosten. Nu kan het configuratiebestand aangepast worden, zonder dat het systeem opnieuw gebuild en gedeployed hoeft te worden. Het BTW-percentage wordt gebruikt in de klasse BTWCalculator. Deze klasse is verantwoordelijk voor alle BTW-berekeningen. De BTWCalculator en het configuratiebestand worden in een NuGet package gezet waardoor de BTW-berekeningen eenvoudig op verschillende plekken herbruikt kunnen worden.

Een voorbeeld van een configuratie file is te vinden in ‘99-Shared’ in het bestand ‘*App.config*’

## Communicatie met BSCatalogusBeheer en BSVoorraadBeheer

Er is gekozen om de communicatie tussen FEWinkelen, BSCatalogusBeheer en BSVoorraadBeheer te laten verlopen via PcSWinkelen. Dit hebben we gedaan zodat we van de producten uit de catalogus de voorraad op kunnen halen en kunnen koppelen met elkaar. Hiervoor hebben we nieuwe XSD schema’s geïntroduceerd in het canomical data model waarin een product en een voorraad samen zitten:



Door in PcSWinkelen een mapping te maken van Product en Voorraad schema’s naar ProductVoorraad kunnen we dit rechtstreeks gebruiken in FEWinkelen en hoeft hier verder niet nagedacht te worden over deze mapping.

## Winkelmand (opslaan en toevoegen van artikelen)

**sprint 1**

Kees gaf aan tijdens een gesprek dat hij graag wilt dat de prijs die de klant moet betalen, de prijs van het artikel is die de klant ziet op het moment van het toevoegen aan zijn/haar winkelmand. Dit betekent dat wij de prijs ergens op moeten slaan en deze niet op moment van bestellen uit de database kunnen halen.

We moesten kiezen wat we in de cookies zouden stoppen, alle artikel gegevens of alleen het ID waarmee we de overige gegevens op kunnen halen uit de database.

We hebben ervoor gekozen om alle artikel gegevens in de cookies op te slaan. Op deze manier hoeven we op de winkelmand pagina geen requests te doen om de overige gegevens op te halen. Hiermee minderen we dus het aantal requests, wat de performance ten goede doet.

**sprint 2**

Echter kwamen wij erachter dat de prijs opslaan in een cookie geen veilige oplossing is. Cookies zijn namelijk aan te passen door gebruikers en hierdoor niet geschikt om een prijs in op te slaan.

Hierdoor zijn wij tijdens sprint 2 het winkelmandje op een andere manier gaan implementeren. Hierbij hadden we een aantal opties:

1. De cookie signen en encrypten
2. De winkelmand in de PcSWinkelen opslaan

De eerste optie zou het probleem niet oplossen aangezien er vanuit hidden input velden de waardes van een artikel in de cookie gezet worden. Dit is nog steeds onveilig omdat de HTML makkelijk aan te passen is.

Dus zijn wij voor optie 2 gegaan. Dit hebben wij geïmplementeerd door de cookie simpelweg een GUID te laten bevatten. Deze GUID wordt samen met artikelen opgeslagen in de database. Om ervoor te zorgen dat de prijs niet aan te passen is, worden er vanuit de frontend een id van het artikel samen met de GUID naar de PcSWinkelen gestuurd. Vervolgens haalt PcSWinkelen bij BSCatalogus het juiste artikel op, met prijs. Deze wordt samen met de GUID opgeslagen. Wanneer een gebruiker daarna zijn winkelmand bekijkt lezen we de cookie uit en halen we op basis van de GUID de juiste artikelen uit de database van de PcSWinkelen op. Dit is qua performance een minder fijne oplossing, maar het zorgt er wel voor dat het systeem voldoet aan de wensen van Kees en dat het op een veilige manier geïmplementeerd is.

**sprint 3**

Bij de review van sprint 2 gaf Kees (de opdrachtgever) aan dat het te lang duurde voordat er een melding kwam dat er een artikel toegevoegd wordt/is aan de winkelmand. Dit komt omdat bij het toevoegen van een artikel de view herladen wordt. Hierbij wordt de catalogus ook opnieuw geladen. Op dit moment is dat nog een langzame bewerking. Wij zijn ons ervan bewust dat dit op te lossen is door het  ophalen te parallelliseren of de catalogus in de cache te zetten. Echter kost dit meer tijd om te realiseren en weten wij niet of dit nog in de sprint past. Om er toch voor te zorgen dat voor de gebruiker duidelijk is dat het systeem bezig is met het toevoegen, kiezen wij ervoor om een simpele melding te tonen met javascript. Mochten wij nog mogelijkheid zien om de performance problemen op te lossen binnen deze sprint, zullen wij alsnog hiervoor gaan. Echter is het niet gelukt om dit te implementeren.

Klantgegevens invullen en opslaan

**sprint 3**

Binnen deze sprint is er afgesproken om alleen de klantgegevens op te slaan die nodig zijn voor het versturen van de bestelling. Echter staan er in het invoerformulier voor de klantgegevens al extra velden, die later bij een eventuele verdere uitbreiding van het systeem, nodig gaan worden. Op dit moment slaan wij deze velden nog niet op. Dit hebben wij met de opdrachtgever besproken en hij ging hiermee akkoord.

XSD’s aanpassing: veel items nillable die vaak verplicht zijn

**sprint 3**

In de bestelling XSD zijn de meeste elementen nillable gemaakt omdat deze niet altijd verplicht zijn. Meestal is dit wel het geval. Er is voor gekozen om de elementen in code te valideren op null waardes in plaats van dit door WCF te laten doen. Deze keuze is gemaakt omdat de XSD’s hierdoor generieker gebruikt kunnen worden en doordat de validatie van WCF niet duidelijk de foutmelding weergeeft.

# Testhandleiding

## Testen

De gerealiseerde applicatie voor Kantilever is ook uitvoerig getest. De testen kunnen uitgevoerd worden binnen Visual Studio en deze worden al automatisch uitgevoerd op de buildserver. We hebben ook integratietesten geschreven die afhankelijk zijn van de omgeving waarin ze gedraaid worden. Deze testen kunnen onderscheiden worden door het voorvoegsel: *“Integration\_”* of *“Acceptation\_”.* Deze testen zijn afhankelijk van de locaties van de services op de buildserver. Als de testen uitgevoerd worden is het belangrijk dat de testen gedraaid worden op Acceptatie ipv. Debug. Wel moeten de services dan beschikbaar zijn en dit is alleen binnen het domein van de buildserver. Deze kan namelijk niet bereikt worden vanuit buitenaf.