

**Syntra West Brugge**  
**Opleiding C# programmeur**

Avondonderwijs

# **Controleformulieren Footstep**

Eindwerk voor het behalen van het getuigschrift van  
C# programmeur

door **Yordi Van Nieuwenhuyse**

o.l.v. Helena Coppieters

Academiejaar 2019 – 2020

## Woord vooraf

Om mijn opleiding tot C# programmeur te volbrengen, kregen wij als opdracht om een eindwerk te maken. Hierbij werd van ons verwacht om een webbrowser te maken die werkt en waarin we de aangeleerde technieken tonen. Voor mij was de keuze van een project heel makkelijk aangezien ik op mijn werk al enkele mogelijkheden zag om zaken te verbeteren. In overleg met de verantwoordelijken van het bedrijf werd beslist om het systeem van controles uitvoeren en registreren gebruiksvriendelijker te maken. Op die manier is er een database die door iedereen via de site kan geraadpleegd worden. Dit bespaart niet alleen tijd, maar ook veel papier.

Ik werkte vooral thuis aan dit project. Bij deze wil ik dan ook graag mijn vriendin, Emelie Deklerck, bedanken voor de steun die ik tijdens mijn studie en het maken van dit eindwerk kreeg.

Anderzijds wil ik ook nog de volgende personen bedanken die mij deze kennis hebben bijgeleerd of mij ondersteunden bij het verwerken ervan. Op deze manier zorgden zij ervoor dat ik dit eindwerk kon realiseren.

- Mevr. H. Coppeters (docent C# programmeur)
- Mr. D. Henderyck (verantwoordelijke metaal Footstep)

Ook wil ik graag nog mijn ouders, vrienden en klasgenoten bedanken voor hun steun die ze mij gegeven hebben tijdens deze opleiding.

*Ik hoop dat u, als lezer, mijn visie en uitwerking van dit project interessant vindt en veel bijkeert!*

## Inhoudsopgave

Woord vooraf .....	2
Inhoudsopgave.....	3
Lijst met illustraties.....	4
Gebruikte afkortingen en symbolen.....	5
Inleiding.....	6
Over het bedrijf.....	6
Controleformulieren.....	7
Probleemstelling.....	9
Gebruiksvriendelijkheid.....	9
Laksheid.....	10
Milieuvriendelijkheid.....	10
Raadpleging.....	10
Oplossingen.....	10
Gebruiksvriendelijkheid.....	10
Laksheid.....	13
Milieuvriendelijkheid.....	13
Raadpleging.....	14
Codevoorbeelden.....	15
De API.....	15
A. Database .....	15
B. Datalayer.....	19
C. Business layer.....	22
D. Controllers.....	26
Blazor Server.....	27
A. Data acces.....	27
B. Client.....	30
Besluit.....	37
Bijlagen .....	38
Bijlage 1: Kostenprijsberekening.....	38
Bijlage 2: Gepresteerde uren.....	42
Bijlage 3: Database Model.....	48
Geraadpleegde links.....	50

## Lijst met illustraties

Figuur 1: Onthaal VZW Footstep afdeling metaal.....	6
Figuur 2: Afbeelding homepage.....	8
Figuur 3: Origineel controleformulier om handmatig in te vullen.....	9
Figuur 4: Voorbeeld overvol archief .....	10
Figuur 5: Overzicht vragen met type antwoorden.....	11
Figuur 6: Enkele mogelijkheden type antwoorden .....	11
Figuur 7: Vragen naar parameters .....	12
Figuur 8: Foutmelding bij het vergeten van essentiële informatie .....	13
Figuur 9: Mogelijkheid om te filteren op verschillende onderdelen.....	14
Figuur 10: Invoeren van de parameters .....	14
Figuur 11: Voorbeeld van create stored procedure .....	15
Figuur 12: Voorbeeld van delete stored procedure.....	15
Figuur 13: Voorbeeld van update stored procedure .....	16
Figuur 14: Class DO_Question .....	16
Figuur 15: View QuestionViewAll .....	17
Figuur 16: Een klein deel van de Dummy data voor de database.....	18
Figuur 17: Class DO_RelationBase .....	19
Figuur 18: Class SqlDatabase .....	20
Figuur 19: Interface IBaseRepository<T>.....	20
Figuur 20: Class QuestionRepository .....	21
Figuur 21: Drielagige architectuur.....	22
Figuur 22: Class BO_Question .....	22
Figuur 23: Class BusinessObjectBase .....	23
Figuur 24: Deel van Class UC_Question.....	23
Figuur 25: Deel van Class StandardUseCase .....	24
Figuur 26: Class UC_ExceptionHandling .....	25
Figuur 27: Class QuestionController.....	26
Figuur 28: Class RelationBase.....	27
Figuur 29: Class RelationBaseInput .....	27
Figuur 30: Class Serializer.....	28
Figuur 31: Class QuestionData .....	29
Figuur 32: Parameters DataGrid Component .....	30
Figuur 33: DataGrid HTML Component .....	31
Figuur 34: QuestionColumns Component .....	32
Figuur 35: QuestionFormLayout Component .....	33
Figuur 36: Questions Page.....	34
Figuur 37: declaratie variabelen my-styles .....	35
Figuur 38: Deel van my-styles.css.....	36
Figuur 39: Eigen Database diagram .....	48

## Gebruikte afkortingen en symbolen

API = Application Programming Interface

JSON formaat = JavaScript Object Notation formaat

ERP systeem = Enterprise Resource Planning systeem

EF Core = Entity Framework Core

TPT = Table Per Type

DB = DataBase

BC = Business Central

CF = ControleFormulieren

DRY = Don't Repeat Yourself

CRUD = Create Read Update Delete

TSQL = Transact Structured Query Language

SQL = Structured Query Language

## Inleiding

### Over het bedrijf

VZW Footstep is een maatwerkbedrijf in de Brugse regio. Footstep is ontstaan uit de fusie van de beschutte werkplaats Arcotec en de sociale werkplaats Loca Labora. Met zijn 500 medewerkers is het hoofddoel van Footstep om aangepaste tewerkstelling te creëren voor mensen die niet meer aan de slag kunnen in een traditionele werkomgeving. De medewerkers worden begeleid door jobcoaches die hen op deze manier een zo veilig mogelijke werkomgeving aanbieden.

Met een oppervlakte van ruim 2 hectare biedt Footstep heel wat verschillende activiteiten en diensten aan waarin de medewerkers kunnen werken. Er zijn enkele verpakkingsdiensten in het bedrijf en ze doen ook aan productie van metaaldraad en –artikelen. Ook een groendienst, poetsdienst en schilderwerken behoren tot de aangeboden diensten. Daarnaast baat Footstep met zijn medewerkers ook de cafetaria's van kinderboerderijen in Brugge en Torhout uit, alsook Hotel-Restaurant De Grote Wateringe te Damme. Footstep biedt foodboxen aan die ter plaatse worden samengesteld, maar ook hand gepelde garnalen en kruiden worden door de medewerkers verzorgd.

Door het aanbieden van deze producten en diensten, probeert Footstep de medewerkers een zinvolle tewerkstelling te bieden. De producten en diensten worden daarna ook te koop aangeboden aan externe bedrijven.



Figuur 1: Onthaal VZW Footstep afdeling metaal

## Controleformulieren

Ikzelf werk op de afdeling metaal in Footstep. Hier sta ik in voor het inplannen van de productie van metaalartikelen alsook het plannen van de aankoop van materialen. De medewerkers van het bedrijf die instaan voor het vervaardigen van deze artikelen, voeren regelmatig kwaliteitscontroles uit.

Op dit moment gebeuren deze controles met standaard controleformulieren. Hierop worden eventuele bevindingen en/of tekorten genoteerd per artikel. Deze standaardformulieren werden opgesteld in Excel. Ze worden oningegevuld afgeprint om dan met de hand in te vullen.

Er worden ongeveer 150 controles uitgevoerd per dag. Deze controles en de bijhorende bevindingen worden manueel bijgehouden op dit geprinte formulier. De controles worden verdeeld in ochtend-, middag- en avondcontroles. Hierbij komt het erop neer dat er per dag 50 formulieren, met telkens 3 controles, worden ingevuld.

Op die manier is er een groot archief ontstaan van veel papierwerk dat enkele jaren moeten bijgehouden worden. Wanneer er zich dan een probleem voordoet, is het natuurlijk heel moeilijk om de juiste documenten snel te raadplegen. Dit vergt heel wat zoekwerk in het papieren archief aangezien er niets op de computer wordt bijgehouden.

Daarnaast zijn er nog andere struikelblokken waartegen we aanbotsten bij het gebruik van dit formulier. Mede door deze struikelblokken leek het me productief om voor de Excel formulieren een andere en efficiëntere oplossing te vinden. Daarom heb ik ervoor gekozen om mijn eindwerk op te delen in drie delen:

1. De API. Deze verzorgt alle communicatie naar de database en vertaalt deze naar een JSON formaat.
2. De Blazor server applicatie. Deze verzorgt de webbrowser visuals voor het aanmaken van de controleformulieren. Deze spreekt met de API en vertaalt de teksten in JSON formaat naar zichtbare gegevens in de site.
3. De smartphone/tablet applicatie (deze behoort niet tot het eindproduct van dit eindwerk). Deze zal eveneens de API aanspreken en zal worden gebruikt om ten velde controles uit te voeren.

De API haalt uit twee verschillende databasen zijn gegevens:

1. De reeds bestaande database van ons ERP systeem (Navision, Business Central). Deze database heeft enkel leesrechten.
2. De eigen gemaakte database van mijn project met de gegevens van de controleformulieren.

Mijn eindwerk bestaat dus uit mijn eigen gemaakte database en de API die zal zorgen voor de communicatie naar de database en de webbrowser. Later zal deze ook gebruikt worden door de applicatie. Ik maakte ook een webbrowser waarop de controleformulieren aangemaakt kunnen worden. Deze formulieren zullen later aangeroepen en geregistreerd worden in de applicatie.

Het is de bedoeling om later dit eindwerk aan te wenden bij het maken van een applicatie voor het bedrijf. In deze applicatie zullen alle controleformulieren digitaal ingevuld en geregistreerd kunnen worden.



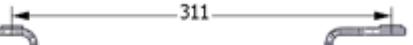
Figuur 2: Afbeelding homepage

## Probleemstelling

### Gebruiksvriendelijkheid

We merkten op dat het reeds bestaande controleformulier niet zo gebruiksvriendelijk was. De hokjes waarin er moet geschreven worden, zijn veel te klein waardoor er al snel een 'OK' ingevuld werd terwijl dit misschien niet aan de orde was. Ook werden er soms fouten gemaakt bij datum, naam van het product,... waardoor het controleformulier niet klopte. Er werd daarna ook geen feedback gegeven aan de medewerker die het formulier invulde. Op die manier bleven dezelfde fouten zich voordoen.

Aangezien de medewerkers zelf opmerkingen kunnen noteren, zagen we dat dit niet altijd volledig of correct gebeurde. Het bestaande controleformulier ging zo zijn doel voorbij. Voor sommigen was het formulier eerder een last dan een nuttig instrument.

		8 Meting, analyse en verbetering 8.2. Bewaking en meting 8.2.4. Bewaking en meting van producten <b>CONTROLEFORMULIER ESC-SPC</b>	CF.DAI800.35 28/03/2017 Ann
<b>LASSEN</b>			
Artikel : DAI800.35	Lotnummer :		
EP - DAI800.35	Productie-order :		
Kaliber :	Aantal :		
Datum :			
ESC of SPC			
Tijdstip controle			
Plaatjes is <u>niet repositioneerd &gt;&gt;</u> <u>centerafstand - 311+0,5</u> >> extra controle met bout + flens : MOET er vlot tussen passen ! Zie ook Q-sandachtpunt			
Plaatjes en draden zijn niet vervormd			
Aspect van de puntjes >> geen ruwe kraagvorming			
Monitor of tiknummer			
Opmerkingen :			
<small>* ESC = Eerste Stuk Controle gebeurt bij machine-instelling, nieuw lot, andere persoon, kaliberwissel en bobijnwissel.  <small>SPC = SteekProofControle gebeurt <u>na elke pauze</u>.</small></small>			

Figuur 3: Origineel controleformulier om handmatig in te vullen

## Laksheid

Zoals eerder aangehaald, moesten de medewerkers de formulieren volledig met de hand invullen. Er zijn geen voorgestelde invullingen, waardoor iedere medewerker dit op zijn eigen manier deed. De één deed dit punctueler en correcter dan de ander. Snel overal 'OK' invullen zodat het verplichte papier in orde was, gebeurde ook soms.

## Milieuviriendelijkheid

Een bijkomend nadeel van deze manier van werken, is dat er heel veel papieren worden geprint. Elke dag 50 controleformulieren, dat betekent 11 000 bladzijden per jaar. Als je dan berekent wat dit het bedrijf elk jaar kost aan papier en inkt, merk je ook op dat dit eigenlijk een verloren kost is. Door alles te digitaliseren, wordt deze kost al uitgespaard.



Figuur 4: Voorbeeld overvol archief

## Raadpleging

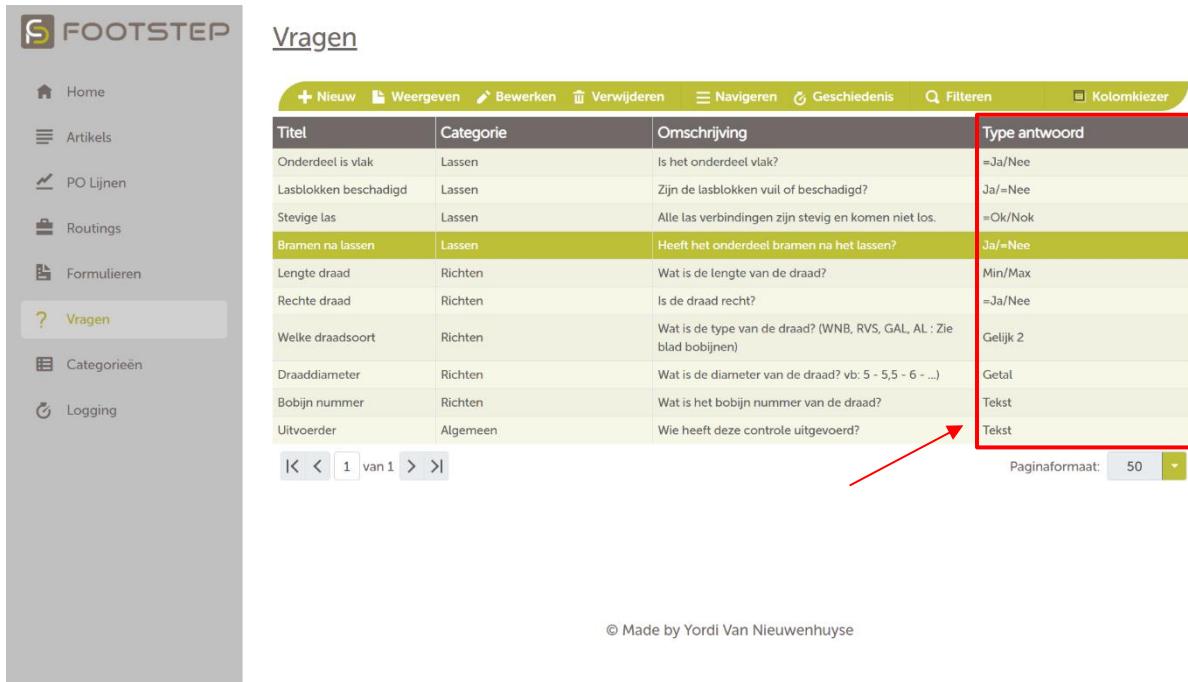
Tot nu toe werden alle controleformulieren leeg afgedrukt om daarna ingevuld te worden met de hand. Deze werden dan 5 jaar lang bijgehouden in mappen. De formulieren werden gewoon op elkaar gelegd en zo bewaard, zonder enige vorm van structuur. Wanneer er zich echter een probleem voordeed bij een product op vlak van kwaliteit, was het een hele zoektocht om het juiste controleformulier terug te vinden. Hierbij moest er gezocht worden in mappen zonder structuur of aanduiding waar je het formulier kan terugvinden. Aangezien er geen digitaal controleformulier werd bijgehouden, kon er ook niet digitaal gezocht worden.

## Oplossingen

Voor alle bovenstaande problemen probeerde ik via het uitwerken van mijn project de gepaste oplossingen te vinden. Het doel was om hierdoor correcter en efficiënter aan de slag te kunnen. Als eindwerk maakte ik een programma waarin de controleformulieren digitaal kunnen aangemaakt worden. Deze worden opgeslagen in een database en zullen, in een later stadium na dit eindwerk, kunnen aangeroepen en geregistreerd worden in een applicatie.

## Gebruiksvriendelijkheid

De controleformulieren zijn voorzien van vooraf opgestelde antwoorden die de medewerkers kunnen aanklikken. Bij bepaalde controlevragen worden er ja-neevragen gesteld die de medewerker juist moet aanklikken. Soms is hier 'ja' het juiste antwoord, maar soms ook 'nee'. Op die manier wordt er vermeden dat de vragenlijst snel en ondoordacht wordt ingevuld.

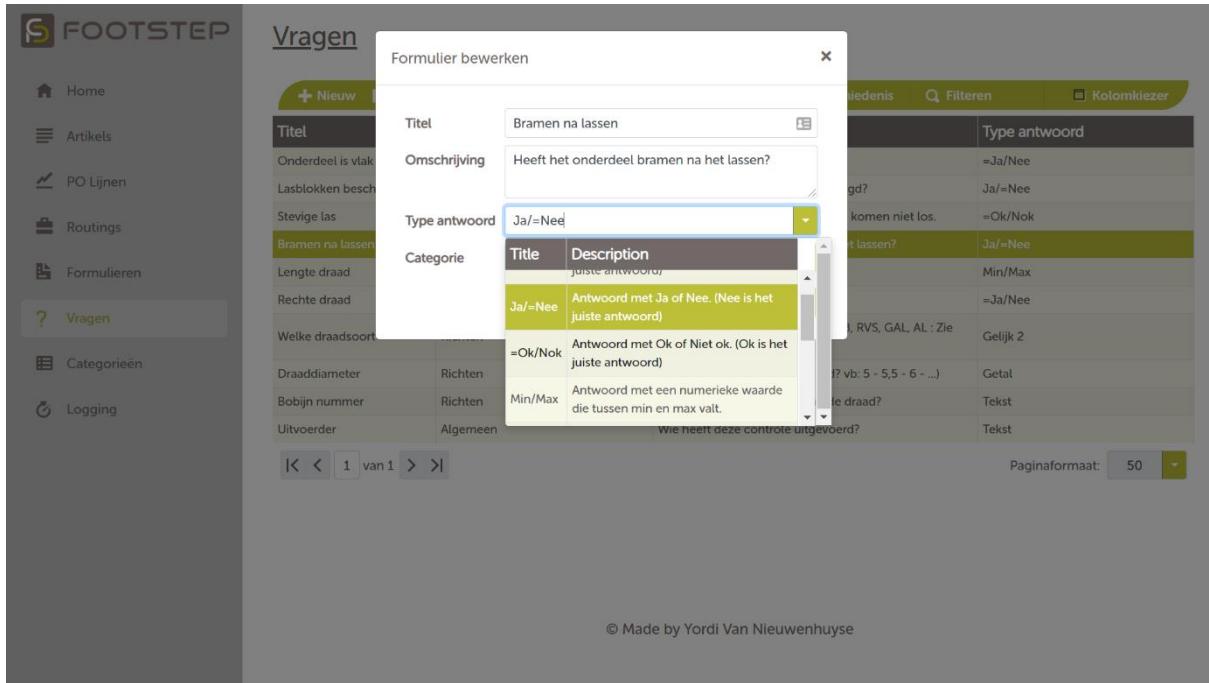


**Vragen**

+ Nieuw Weergeven Bewerken Verwijderen Navigeren Geschiedenis Filteren Kolomkiezer			
Titel	Categorie	Omschrijving	Type antwoord
Onderdeel is vlak	Lassen	Is het onderdeel vlak?	=Ja/Nee
Lasblokken beschadigd	Lassen	Zijn de lasblokken vuil of beschadigd?	=Ja/Nee
Stevige las	Lassen	Alle las verbindingen zijn stevig en komen niet los.	=Ok/Nok
<b>Bramen na lassen</b>	<b>Lassen</b>	<b>Heeft het onderdeel bramen na het lassen?</b>	<b>Ja/Nee</b>
Lengte draad	Richten	Wat is de lengte van de draad?	Min/Max
Rechte draad	Richten	Is de draad recht?	=Ja/Nee
Welke draadsoort	Richten	Wat is de type van de draad? (WNB, RVS, GAL, AL : Zie blad bobijnen)	Gelijk 2
Draaddiameter	Richten	Wat is de diameter van de draad? vb: 5 - 5,5 - 6 - ...)	Getal
Bobijn nummer	Richten	Wat is het bobijn nummer van de draad?	Tekst
Uitvoerder	Algemeen	Wie heeft deze controle uitgevoerd?	Tekst

Paginaformaat: 50

Figuur 5: Overzicht vragen met type antwoorden



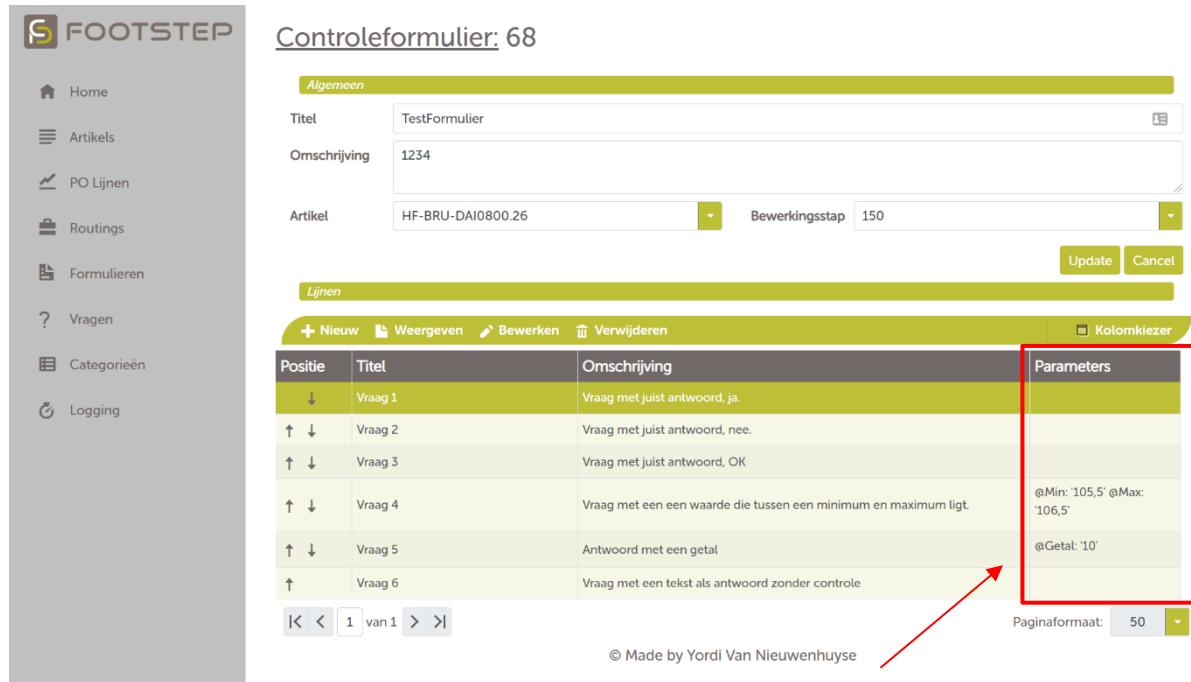
**Vragen**

Formulier bewerken

<b>Titel</b>	Bramen na lassen
<b>Omschrijving</b>	Heeft het onderdeel bramen na het lassen?
<b>Type antwoord</b>	Ja/Nee
<b>Categorie</b>	Title Description juiste antwoord Ja/Nee Antwoord met Ja of Nee. (Nee is het juiste antwoord)

Figuur 6: Enkele mogelijkheden type antwoorden

Op bepaalde vragen moeten de medewerkers ook antwoorden met een tekst of met getallen. Deze vragen worden door het systeem zelf opgevraagd en gecontroleerd. Wanneer ze dus een antwoord invullen dat niet past bij de vraag, geeft het systeem al meteen feedback. Op die manier is de kans dat er fouten bij het invullen van het controleformulier sluipen, heel wat kleiner dan voorheen. Het is op dit moment mogelijk om deze parameters toe te voegen aan het controleformulier.



The screenshot shows the 'Controleformulier: 68' page. On the left is a sidebar with links: Home, Artikels, PO Lijnen, Routings, Formulieren, Vragen, Categorieën, and Logging. The main area has tabs for 'Algemeen' and 'Lijnen'. Under 'Algemeen', fields include Titel (TestFormulier), Omschrijving (1234), Artikel (HF-BRU-DAI0800.26), and Bewerkingstap (150). Buttons for Update and Cancel are at the bottom. Under 'Lijnen', there's a table with columns: Positie, Titel, Omschrijving, and Parameters. A red box highlights the 'Parameters' column for questions 4 and 5. Question 4 has parameters '@Min: '105,5' @Max: '106,5''. Question 5 has parameters '@Getal: '10''. A red arrow points from the text 'Vragen naar parameters' to this red box. At the bottom are navigation buttons (K, <, 1, >, >>) and a Paginaformaat (50) dropdown.

Positie	Titel	Omschrijving	Parameters
↓	Vraag 1	Vraag met juist antwoord, ja.	
↑ ↓	Vraag 2	Vraag met juist antwoord, nee.	
↑ ↓	Vraag 3	Vraag met juist antwoord, OK	
↑ ↓	Vraag 4	Vraag met een waarde die tussen een minimum en maximum ligt.	@Min: '105,5' @Max: '106,5'
↑ ↓	Vraag 5	Antwoord met een getal	@Getal: '10'
↑	Vraag 6	Vraag met een tekst als antwoord zonder controle	

Figuur 7: Vragen naar parameters

## Laksheid

Ook de laksheid bij het invullen van het formulier zal door deze nieuwe manier van werken vermeden worden. De medewerkers kunnen niet zomaar iets invullen bij een vraag. Ze moeten nadenken over het antwoord en deze kiezen uit de lijst of zelf invoeren. Er gebeurt telkens een controle door het systeem zelf. Het systeem gaat na of het antwoord dat gegeven werd, past bij de vraag. Zo niet, wordt er een foutmelding gegeven en moeten ze dit antwoord aanpassen.



Formulier bewerken X

**Titel**

**Omschrijving**

**Type antwoord** Selecteer een soort

**Categorie** Selecteer een category

- Selecteer een soort antwoord
- Selecteer een category
- Vul een titel in

**Update** **Cancel**

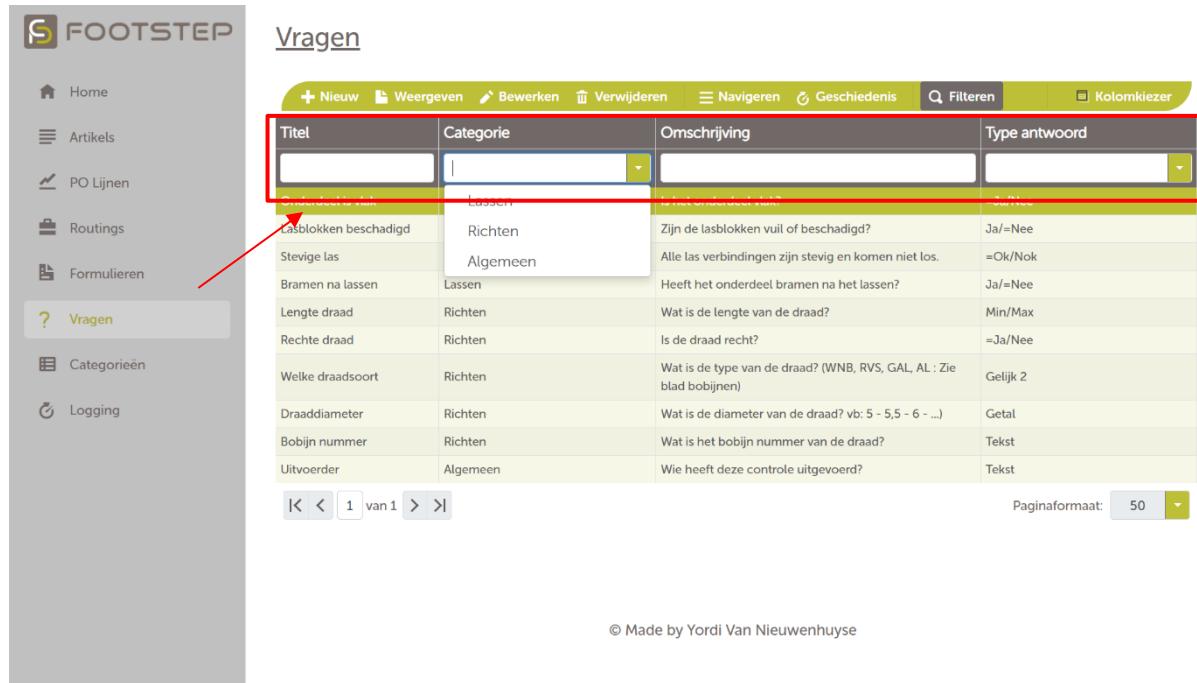
Figuur 8: Foutmelding bij het vergeten van essentiële informatie

## Milieuvriendelijkheid

Door de manier van controleren volledig te digitaliseren, worden de vele afdrukken per jaar vermeden. Ook het bijhouden van de papieren voor 5 jaar lang, wordt op deze manier gedigitaliseerd. Op die manier spaar je niet alleen papier, maar ook een archiefruimte uit.

## Raadpleging

Wanneer er zich nu een probleem voordoet bij een bepaald artikel, kan het controleformulier steeds via het digitaal systeem geraadpleegd worden. Via de filteropties (datum, naam, productieorder, artikel,...) op het webbrowser kan heel snel het juiste formulier teruggevonden worden. Dit spaart heel wat zoekwerk en werkuren uit.



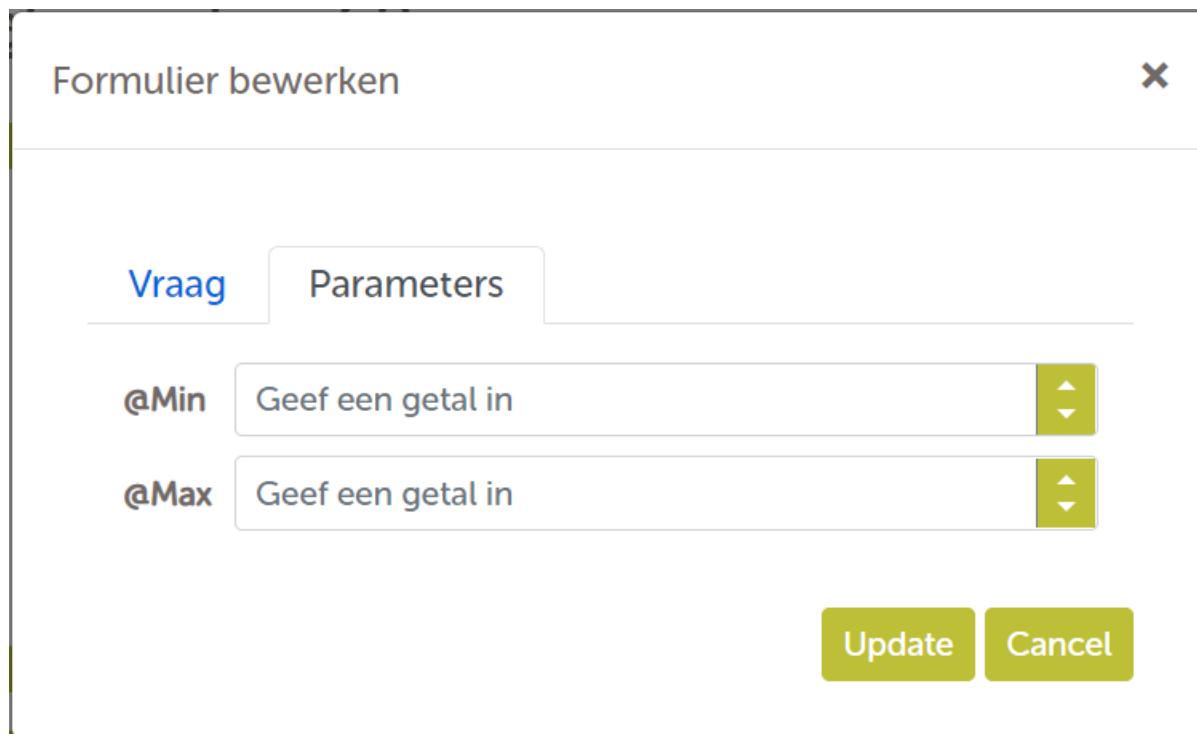
**Vragen**

Titel	Categorie	Omschrijving	Type antwoord
Lassen	Lassen	Zijn de lasblokken vuil of beschadigd?	=Ja/Nee
Lasblokken beschadigd	Richten	Alle las verbindingen zijn stevig en komen niet los.	=Ok/Nok
Stevige las	Algemeen	Heeft het onderdeel bramen na het lassen?	=Ja/Nee
Bramen na lassen	Lassen	Wat is de lengte van de draad?	Min/Max
Lengte draad	Richten	Is de draad recht?	=Ja/Nee
Rechte draad	Richten	Wat is de type van de draad? (WNB, RVS, GAL, AL : Zie blad bobijnen)	Gelijk 2
Welke draadsoort	Richten	Wat is de diameter van de draad? vb: 5 - 5,5 - 6 - ...)	Getal
Draaddiameter	Richten	Wat is het bobijn nummer van de draad?	Tekst
Bobijn nummer	Richten	Wie heeft deze controle uitgevoerd?	Tekst
Uitvoerder	Algemeen		

Paginaformaat: 50

© Made by Yordi Van Nieuwenhuyse

Figuur 9: Mogelijkheid om te filteren op verschillende onderdelen



Formulier bewerken

Vraag Parameters

@Min Geef een getal in

@Max Geef een getal in

Update Cancel

Figuur 10: Invoeren van de parameters

## Codevoorbeelden

Bij het bekijken van onderstaande teksten en afbeeldingen, zal je voorbeelden zien van de codes die ik gebruikte om te komen tot mijn eindproduct. Uiteraard toon ik hieronder niet alle gebruikte codes, maar toon ik hier de belangrijkste fragmenten van.

### De API

#### A. Database

Er zijn twee databases: één die al bestaat van ons ERP-systeem. Daarnaast heb je ook de database die ik zelf maakte. Voor de database van ons ERP-systeem heb ik enkel leesrechten, dus kon ik geen views of stored procedures aanmaken. Hiervoor heb ik dan gekozen om mijn queries in een aparte class op te slaan.

Voor mijn zelfgemaakte database heb ik gekozen voor een SQL Server Database Project in de API Solution. Zo kan je deze makkelijk publishen wanneer nodig. Hierin hebben we een folder waar de tabellen zich in bevinden, deze zijn allemaal geschreven in TSQL. Daarnaast is er ook nog een folder met de stored procedures, deze zijn ingedeeld in create/update/delete.

```
CREATE PROCEDURE [dbo].[spCreateQuestion]
    @Title varchar(30),
    @Description nvarchar(255),
    @RelationType int,
    @QuestionCategoryId int,
    @AnswerTypeId int,
    @Id int = null output
As
Begin
    set nocount on;

    exec _spCreateRelationBase @Title, @Description, @RelationType, @Id output;

    insert into Question(Id,QuestionCategoryId,AnswerTypeId)
    values(@Id,@QuestionCategoryId,@AnswerTypeId);
end
```

Figuur 11: Voorbeeld van create stored procedure

```
CREATE PROCEDURE [dbo].[spDeleteQuestion]
    @Id int
AS
Begin
    set nocount on;
    delete from [EditLog]
    where RelationBaseId = @Id;

    delete from [Question]
    where Id = @Id;

    exec _spDeleteRelationBase @Id;
end;
```

Figuur 12: Voorbeeld van delete stored procedure

```

CREATE PROCEDURE [dbo].[spUpdateQuestion]
    @Id int,
    @Title varchar(30) = NULL,
    @Description nvarchar(255) = NULL,
    @RelationType int = NULL,
    @QuestionCategoryId int = NULL,
    @AnswerTypeId int = NULL
As
Begin
    set nocount on;

    exec _spUpdateRelationBase @Id, @Title, @Description, @RelationType;

    update Question
    set
        QuestionCategoryId = ISNULL(@QuestionCategoryId,QuestionCategoryId),
        AnswerTypeId = ISNULL(@AnswerTypeId,AnswerTypeId)
    where
        Id = @Id;
end

```

Figuur 13: Voorbeeld van update stored procedure

Dan hebben we ook nog een folder met views. Hiervan hebben we één of twee views per tabel. De ene met zo min mogelijk relaties en de andere met de onderliggende items in verwerkt. Om te kunnen samenwerken met slapper moeten de naamgevingen overeenkomen met de classes en hun properties.

Als voorbeeld deze class DO\_Question heeft propertie AnswerType. Dan moet deze in de view ook zo benoemd worden. Indien we de properties van die class (DO\_AnswerType) ook willen benoemen, moeten we de benaming AnswerType volgen met een '\_' en de propertie van de class die we willen benoemen. Zie als voorbeeld de class en view hieronder.

```

[Table("Questions")]
public class DO_Question : DO_RelationBase
{
    #region Constructors
    public DO_Question()
    {
        RelationType = EnumLibrary.Enums.RelationType.Question;
    }
    #endregion

    #region Properties
    public int AnswerTypeId { get; set; }
    public int QuestionCategoryId { get; set; }

    // Navigation
    public DO_AnswerType AnswerType { get; set; }
    public DO_QuestionCategory QuestionCategory { get; set; }
    public IEnumerable<DO_EditLog> EditLogs { get; set; }
    public IEnumerable<DO_File> Files { get; set; }
    #endregion

    #region Methods
    #endregion
}

```

Figuur 14: Class DO\_Question

```

CREATE VIEW [dbo].[QuestionViewAll]
AS
SELECT
dbo.Question.Id,
dbo.RelationBase.Title,
dbo.RelationBase.Description,
dbo.RelationBase.RelationType,
dbo.Question.QuestionCategoryId,
dbo.QuestionCategory.Id AS QuestionCategory_Id,
RelationBase_2.Title AS QuestionCategory_Title,
RelationBase_2.Description AS QuestionCategory_Description,
RelationBase_2.RelationType AS QuestionCategory_RelationType,
dbo.Question.AnswerTypeId,
dbo.AnswerType.Id AS AnswerType_Id,
RelationBase_1.Title AS AnswerType_Title,
RelationBase_1.Description AS AnswerType_Description,
RelationBase_1.RelationType AS AnswerType_RelationType,
dbo.[File].Id AS Files_Id,
RelationBase_3.Title AS Files_Title,
RelationBase_3.Description AS Files_Description,
RelationBase_3.RelationType AS Files_RelationType,
dbo.[File].DirectoryPath AS Files_DirectoryPath,
dbo.[File].Type AS Files_Type,
dbo.EditLog.Id AS EditLogs_Id,
RelationBase_4.Title AS EditLogs_Title,
RelationBase_4.Description AS EditLogs_Description,
RelationBase_4.RelationType AS EditLogs_RelationType,
dbo.EditLog.RelationBaseId AS EditLogs_RelationBaseId,
dbo.EditLog.ChangedProperty AS EditLogs_ChangedProperty,
dbo.EditLog.NewValue AS EditLogs_NewValue,
dbo.EditLog.UserName AS EditLogs_UserName,
dbo.EditLog.DateTime AS EditLogs_DateTime
FROM
dbo.Question

INNER JOIN
dbo.RelationBase ON dbo.Question.Id = dbo.RelationBase.Id
INNER JOIN
dbo.AnswerType ON dbo.Question.AnswerTypeId = dbo.AnswerType.Id
INNER JOIN
dbo.RelationBase AS RelationBase_1 ON dbo.AnswerType.Id = RelationBase_1.Id
INNER JOIN
dbo.QuestionCategory ON dbo.Question.QuestionCategoryId = dbo.QuestionCategory.Id
INNER JOIN
dbo.RelationBase AS RelationBase_2 ON dbo.QuestionCategory.Id = RelationBase_2.Id
LEFT OUTER JOIN
dbo.EditLog ON dbo.RelationBase.Id = dbo.EditLog.RelationBaseId
LEFT OUTER JOIN
dbo.RelationBase AS RelationBase_4 ON dbo.EditLog.Id = RelationBase_4.Id
LEFT OUTER JOIN
dbo.RelationHasComponents ON dbo.RelationBase.Id = dbo.RelationHasComponents.RelationId
LEFT OUTER JOIN
dbo.RelationBase AS RelationBase_3 ON dbo.RelationHasComponents.ComponentId =
RelationBase_3.Id
LEFT OUTER JOIN
dbo.[File] ON RelationBase_3.Id = dbo.[File].Id

```

Figuur 15: View QuestionViewAll

Als eerste testdata heb ik een Post-Deployment Script gemaakt dat bij het publishen al meteen data in de database stopt. Hieronder zie je een deel van de aangemaakte data.

```

Declare @DummyId int;

Declare @Question int = 0, @Form int = 1, @QuestionIndex int = 2, @QuestionGroup int = 3,
@RegisteredQuestion int = 4, @RegisteredForm int = 5, @EditLog int = 6, @File int = 7,
@AnswerType int = 8, @RegistryType int = 9, @QuestionCategory int = 10;
Declare @AnswerType_JaNee int, @AnswerType_NeeJa int, @AnswerType_MinMax int,
@AnswerType_Gelijk1 int, @AnswerType_Gelijk2 int, @AnswerType_Getal int,
@AnswerType_Tekst int, @AnswerType_OkNok int;

if not exists (select * from [AnswerType])
begin
    exec spCreateAnswerType '=Ja/Nee', 'Antwoord met een Ja of Nee. (Ja is het juiste antwoord)', @AnswerType, @AnswerType_JaNee output; -- 1
    exec spCreateAnswerType 'Ja=Nee', 'Antwoord met Ja of Nee. (Nee is het juiste antwoord)', @AnswerType, @AnswerType_NeeJa output; -- 2
    exec spCreateAnswerType '=Ok/Nok', 'Antwoord met Ok of Niet ok. (Ok is het juiste antwoord)', @AnswerType, @AnswerType_OkNok output; -- 3
    exec spCreateAnswerType 'Min/Max', 'Antwoord met een numerieke waarde die tussen min en max valt.', @AnswerType, @AnswerType_MinMax output; -- 4
    exec spCreateAnswerType 'Gelijk 1', 'Antwoord met een tekst die moet overeenkomen met de opgegeven tekst. [Hoofdlettergevoelig]', @AnswerType, @AnswerType_Gelijk1 output; -- 5
    exec spCreateAnswerType 'Gelijk 2', 'Antwoord met een tekst die moet overeenkomen met de opgegeven tekst. [Niet-hoofdlettergevoelig]', @AnswerType, @AnswerType_Gelijk2 output; -- 6
    exec spCreateAnswerType 'Getal', 'Antwoord met een numerieke waarde die moet overeenkomen met de opgegeven waarde.', @AnswerType, @AnswerType_Getal output; -- 7
    exec spCreateAnswerType 'Tekst', 'Antwoord met een tekst zonder controle.', @AnswerType, @AnswerType_Tekst output; -- 8
end;

Declare @Parameter_Min int, @Parameter_Max int, @Parameter_Gelijk1 int,
@Parameter_Gelijk2 int, @Parameter_Getal int;
Declare @ParameterType_Numeriek int = 0, @ParameterType_Tekst int = 1;

If not exists (select * from [Parameter])
begin
    exec spCreateParameter '@Min', @ParameterType_Numeriek, @AnswerType_MinMax,
@Parameter_Min output; -- 1
    exec spCreateParameter '@Max', @ParameterType_Numeriek, @AnswerType_MinMax,
@Parameter_Max output; -- 2
    exec spCreateParameter '@Gelijk1', @ParameterType_Tekst, @AnswerType_Gelijk1,
@Parameter_Gelijk1 output; -- 3
    exec spCreateParameter '@Gelijk2', @ParameterType_Tekst, @AnswerType_Gelijk2,
@Parameter_Gelijk2 output; -- 4
    exec spCreateParameter '@Getal', @ParameterType_Numeriek, @AnswerType_Getal,
@Parameter_Getal output; -- 5
end

```

Figuur 16: Een klein deel van de Dummy data voor de database

## B. Datalayer

Ik heb twee datalayers aangemaakt. Eén voor de data van BC en één voor de data van mijn zelfgemaakte database. In deze voorbeelden ga ik me toespitsen op deze van mijn eigen database.

Voor de meeste van de classes heb ik gekozen voor inheritance. Dit wil zeggen dat veel classes de properties Id, Title, Description en RelationType hebben. Daarvoor heb ik één basisclass DO\_RelationBase gemaakt.

```
[Table("RelationBase")]
public class DO_RelationBase
{
    #region Constructors
    #endregion

    #region Properties
    public int Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public RelationType RelationType { get; private protected set; }
    #endregion

    #region Methods
    #endregion

    #region relations Many To Many
    #endregion
}
```

Figuur 17: Class DO\_RelationBase

Deze wordt onder andere gebruikt in de class DO\_Question.

Zie Figuur 14: Class DO\_Question

Dit zorgt ervoor dat ik bepaalde taken met een generic class kan uitvoeren. Hier kom ik later nog op terug.

Voor de data acces heb ik gekozen voor dapper. Hiervoor heb ik een generic class aangemaakt. Deze zal gebruikt worden om de database aan te spreken via stored procedure ofwel via query met de mogelijkheid om parameters mee te geven. Deze zal in de repositories gebruikt worden om op die manier zo weinig mogelijk code te kopiëren en te plakken.

```
public class SqlDatabase : IDataAcces
{
    private readonly IConfiguration _config;
    private readonly ConnectionStringData _connection;

    public SqlDatabase(IConfiguration config, ConnectionStringData connection)
    {
        this._config = config;
        this._connection = connection;
    }
    public async Task<IEnumerable<T>> LoadData<T, U>(string query, U param)
    {
        using ( IDbConnection con = new
SqlConnection(_config.GetConnectionString(_connection.SqlConnectionString)))
        {
            var result = await con.QueryAsync<T>(query, param, commandType:
CommandType.Text);
            return result;
        }
    }

    public async Task<int> SaveData<T>(string st, T param)
    {
        using ( IDbConnection con = new
SqlConnection(_config.GetConnectionString(_connection.SqlConnectionString)))
        {
            return await con.ExecuteAsync(st, param, commandType:
CommandType.StoredProcedure);
        }
    }
}
```

Figuur 18: Class SqlDatabase

Op de volgende pagina zal je een voorbeeld vinden van een repository. Deze hebben telkens minstens vijf methodes.

- Create
- Read (1 entity)
- Read (alles)
- Update
- Delete

```
public interface IBaseRepository<T>
{
    Task<IEnumerable<T>> GetAll();
    Task<T> Get_ByKey(T entity);
    Task<int> Create(T entity);
    Task<int> Update(T entity);
    Task<int> Delete(T entity);
}
```

Figuur 19: Interface IBaseRepository<T>

Allen volgen onderliggend één interface IBaseRepository<T>.

```

public class QuestionRepository : IQuestionRepository
{
    private readonly IDataAcces _data;
    private readonly IControleFormulierenQueries _query;

    public QuestionRepository(IDataAcces data, IControleFormulierenQueries query)
    {
        this._data = data;
        this._query = query;
    }
    public async Task<int> Create(DO_Question entity)
    {
        DynamicParameters p = new DynamicParameters();
        p.Add("Title", entity.Title);
        p.Add("Description", entity.Description);
        p.Add("RelationType", entity.RelationType);
        p.Add("AnswerTypeId", entity.AnswerTypeId);
        p.Add("QuestionCategoryId", entity.QuestionCategoryId);
        p.Add("Id", DbType.Int32, direction: ParameterDirection.Output);

        await _data.SaveData("spCreateQuestion", p);
        return p.Get<int>("Id");
    }

    public async Task<int> Delete(DO_Question entity)
    {
        return await _data.SaveData("spDeleteQuestion", new { entity.Id });
    }

    public async Task<IEnumerable<DO_Question>> GetAll()
    {
        var exec = await _data.LoadData<dynamic,
dynamic>(_query.Get(CF_QueryViews.QuestionView), new { });
        return Slapper.AutoMapper.MapDynamic<DO_Question>(exec);
    }

    public async Task<DO_Question> Get_ByKey(DO_Question entity)
    {
        var exec = await _data.LoadData<dynamic,
dynamic>(_query.Get(CF_QueryViews.QuestionViewAll, true), new { entity.Id });
        return Slapper.AutoMapper.MapDynamic<DO_Question>(exec).FirstOrDefault();
    }

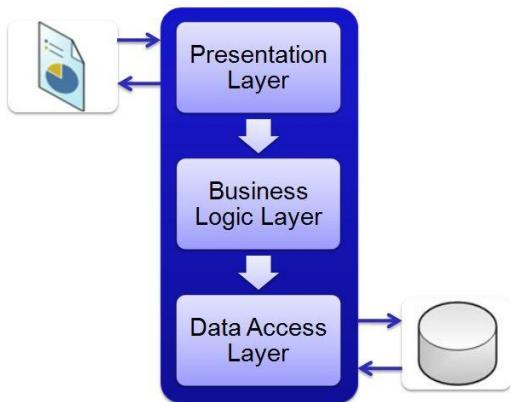
    public async Task<int> Update(DO_Question entity)
    {
        return await _data.SaveData("spUpdateQuestion", new
        {
            entity.Id,
            entity.Title,
            entity.Description,
            entity.AnswerTypeId,
            entity.QuestionCategoryId
        });
    }
}

```

Figuur 20: Class QuestionRepository

### C. Business layer

De business layer spreekt met de datalayer. Zo worden de classes van de database niet aangesproken door de business layer. Dit is een drielaagige architectuur. Dit zorgt voor een grotere vrijheid in het programmeren en het ontwikkelen. In plaats van hier met DO\_Question te werken, werk ik hier met BO\_Question. Deze inherit van de class BusinessObjectBase. Deze zorgt voor de business logic.



Figuur 21: Drielaagige architectuur

```
public class BO_Question : BO_RelationBase
{
    #region Constructors
    public BO_Question()
    {
        RelationType = EnumLibrary.Enums.RelationType.Question;
    }
    #endregion

    #region Properties
    public int AnswerTypeId { get; set; }
    public int QuestionCategoryId { get; set; }

    // Navigation
    public BO_AnswerType AnswerType { get; set; }
    public BO_QuestionCategory QuestionCategory { get; set; }
    public IEnumerable<BO_EditLog> EditLogs { get; set; }
    public IEnumerable<BO_File> Files { get; set; }
    #endregion

    #region Methods
    public override bool AddBusinessRules()
    {
        BusinessRules.Add(new BusinessRule()).IsRequired(nameof(AnswerTypeId),
AnswerTypeId));
        BusinessRules.Add(new BusinessRule()).IsRequired(nameof(QuestionCategoryId),
QuestionCategoryId));

        return base.AddBusinessRules();
    }
    #endregion
}
```

Figuur 22: Class BO\_Question

```

public class BusinessObjectBase
{
    public virtual List<BusinessRule> BrokenRules { get; set; } = null;
    public virtual List<BusinessRule> BusinessRules { get; set; } = null;
    public bool Valid
    {
        get
        {
            BrokenRules = new List<BusinessRule>();
            BusinessRules = new List<BusinessRule>();
            return AddBusinessRules();
        }
    }

    public virtual bool AddBusinessRules()
    {
        foreach (BusinessRule businessRule in this.BusinessRules)
        {
            if (businessRule.Passed == false)
            {
                BrokenRules.Add(businessRule);
            }
        }
        if (BrokenRules.Count != 0)
        {
            return false;
        }
        return true;
    }
}

```

Figuur 23: Class BusinessObjectBase

Via de BusinessRules en de UseCases definiëer ik de business logic. Voor de UseCases heb ik ook gebruik gemaakt van de generic class StandardUseCase waardoor niet alles meerdere malen moet herhaald worden. Op de volgende bladzijde kan je een deel van deze UseCase zien. Deze wordt dan aangeroepen door de andere UseCases.

```

public class UC_Question : IUC_Question
{
    private readonly IQuestionRepository _questionRepository;
    private readonly IStandardUseCase _standard;

    public UC_Question(IQuestionRepository questionRepository, IStandardUseCase standard)
    {
        this._questionRepository = questionRepository;
        this._standard = standard;
    }

    public async Task<BO_Question> UC_Create(BO_Question questionToAdd)
    {
        return await _standard.UC_Create
            <BO_Question, DO_Question, IQuestionRepository>
            (questionToAdd, _questionRepository);
    }
}

```

Figuur 24: Deel van Class UC\_Question

```

public class StandardUseCase : IStandardUseCase
{
    private readonly IMapper _mapper;
    private readonly IUC_ExceptionHandling _uc_ExceptionHandling;
    private readonly IUC_EditLog _editLog;

    public StandardUseCase(IMapper mapper, IUC_ExceptionHandling uc_ExceptionHandling, IUC_EditLog editLog)
    {
        this._mapper = mapper;
        this._uc_ExceptionHandling = uc_ExceptionHandling;
        this._editLog = editLog;
    }

    public async Task<T> UC_Create<T, U, V>(T entityToAdd,
                                                V repository,
                                                [CallerMemberName] string caller = "",
                                                [CallerLineNumber] int lineNumber = 0,
                                                [CallerFilePath] string filePath = "")
                                                where T : BO_RelationBase
                                                where U : DO_RelationBase
                                                where V : IBaseRepository<U>
    {
        try
        {
            if (entityToAdd.Valid)
            {
                entityToAdd.Id = await repository.Create(_mapper.Map<U>(entityToAdd));
                await _editLog.UC_CreationEditLog<T>(entityToAdd);
            }
            else
            {
                _uc_ExceptionHandling.SaveException(entityToAdd, caller, lineNumber, filePath);
            }
            return entityToAdd;
        }
        catch (Exception ex)
        {
            throw _uc_ExceptionHandling.SaveException(ex);
        }
    }
    public async Task<IEnumerable<T>> UC_GetList<T, U, V>(V repository,
                                                               [CallerMemberName] string caller = "",
                                                               [CallerLineNumber] int lineNumber = 0,
                                                               [CallerFilePath] string filePath = "")
                                                               where T : BO_RelationBase
                                                               where U : DO_RelationBase
                                                               where V : IBaseRepository<U>
    {
        try
        {
            return _mapper.Map<IEnumerable<T>>(await repository.GetAll());
        }
        catch (Exception ex)
        {
            throw _uc_ExceptionHandling.SaveException(ex, caller, lineNumber, filePath);
        }
    }
}

```

Figuur 25: Deel van Class StandardUseCase

Voor de exception handling heb ik er voor gekozen om deze op te slaan in de database. Deze slaat de gegevens van de method caller op samen met de exception.

```

public class UC_ExceptionHandling : IUC_ExceptionHandling
{
    private readonly IControleFormulierenExceptionRepository _exceptionRepository;
    private readonly IMapper _mapper;

    public UC_ExceptionHandling(IControleFormulierenExceptionRepository exceptionRepository, IMapper mapper)
    {
        this._exceptionRepository = exceptionRepository;
        this._mapper = mapper;
    }

    public void SaveException(BusinessObjectBase entity,
                                [CallerMemberName] string caller = "",
                                [CallerLineNumber] int lineNumber = 0,
                                [CallerFilePath] string filepath = "")
    {
        entity.BrokenRules.ForEach(br =>
        {
            FrameworkException exception = new FrameworkException(filepath,
                $"{caller} [{lineNumber}]",
                br.failedMessage,
                FrameworkExceptionType.BusinessRuleViolation);

            _exceptionRepository.Create(_mapper.Map<DO_ControleFormulierenExceptions>(exception));
        });
    }

    public FrameworkException SaveException(Exception ex,
                                            [CallerMemberName] string caller = "",
                                            [CallerLineNumber] int lineNumber = 0,
                                            [CallerFilePath] string filepath = "")
    {
        FrameworkException exception = new FrameworkException(filepath,
            $"{caller}
            [{lineNumber}]",
            ex.Message,
            FrameworkExceptionType.Error);

        _exceptionRepository.Create(_mapper.Map<DO_ControleFormulierenExceptions>(exception));
        return exception;
    }
}

```

Figuur 26: Class UC\_ExceptionHandling

## D. Controllers

De API Controllers spreken de business layer UseCases aan. Een voorbeeld van een QuestionController vind je hieronder.

```
[Route("api/[controller]")]
[ApiController]
public class QuestionController : ControllerBase
{
    private readonly IUC_Question _question;
    private readonly IMapper _mapper;

    public QuestionController(IUC_Question question, IMapper mapper)
    {
        this._question = question;
        this._mapper = mapper;
    }

    [HttpGet]
    public async Task<IActionResult> GetQuestions()
    {
        return Ok(await _question.UC_GetList());
    }

    [HttpGet("{id}")]
    public async Task<IActionResult> GetQuestion(int id)
    {
        if (id <= 0) return BadRequest();

        BO_Question result = await _question.UC_GetByKey(id);
        return result == null ? NotFound() : (IActionResult)Ok(result);
    }

    [HttpPost]
    public async Task<IActionResult> CreateQuestion([FromBody] QuestionInput input)
    {
        if (ModelState.IsValid)
        {
            BO_Question result = await
                _question.UC_Create(_mapper.Map<BO_Question>(input));
            return result == null ? NotFound() : (IActionResult)Ok(result);
        }
        return BadRequest();
    }

    [HttpPut]
    public async Task<IActionResult> UpdateQuestion([FromBody] BO_Question update)
    {
        if (ModelState.IsValid)
        {
            BO_Question result = await
                _question.UC_Update(_mapper.Map<BO_Question>(update));
            return result == null ? NotFound() : (IActionResult)Ok(result);
        }
        return BadRequest();
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteQuestion(int id)
    {
        if (id <= 0) return BadRequest();

        await _question.UC_Delete(id);
        return Ok();
    }
}
```

Figuur 27: Class QuestionController

## Blazor Server

### A. Data acces

Voor de data acces heb ik de modellen overgenomen van de business layer van de API. Deze heb ik lichtjes aangepast. Daarna heb ik ze aangevuld met display data annotations. Ook heb ik van elk van deze een input model gemaakt met validation data annotations.

```
public class RelationBase
{
    #region Constructors
    #endregion

    #region Properties
    [Display(Name = "#")]
    public int Id { get; set; }
    [Display(Name = "Titel")]
    public string Title { get; set; }
    [Display(Name = "Omschrijving")]
    public string Description { get; set; }
    [Display(Name = "Type")]
    public RelationType RelationType { get; set; }
    #endregion

    #region Methods
    #endregion
}
```

Figuur 28: Class RelationBase

```
public class RelationBaseInput : NewRowClass
{
    [Display(Name = "#")]
    public int? Id { get; set; }
    [Required(AllowEmptyStrings = false, ErrorMessage = "Vul een titel in")]
    [MinLength(1, ErrorMessage = "Titel moet minsten 1 character hebben")]
    [MaxLength(30, ErrorMessage = "Titel mag niet meer dan 30 characters hebben")]
    public virtual string Title { get; set; }

    [MaxLength(255, ErrorMessage = "Omschrijving mag niet meer dan 255 characters hebben")]
    public virtual string Description { get; set; }
}
```

Figuur 29: Class RelationBaseInput

Om de data uit de API te halen, maak ik gebruik van de class Serializer.

```

public class Serializer : ISerializer
{
    private readonly HttpClient _client;
    private readonly IConfiguration _configuration;

    public Serializer(IConfiguration configuration)
    {
        this._configuration = configuration;
        HttpClient client = new HttpClient
        {
            BaseAddress = new Uri(_configuration[" ApiService:Base"])
        };
        client.DefaultRequestHeaders.Accept.Add(new
        MediaTypeWithQualityHeaderValue("application/json"));
        this._client = client;
    }

    public async Task<T> DeSerialize<T>(string apiUrl) where T : class
    {
        HttpResponseMessage response = await _client.GetAsync(apiUrl);
        string stringData = await response.Content.ReadAsStringAsync();
        JsonSerializerOptions options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        return JsonSerializer.Deserialize<T>(stringData, options);
    }

    public async Task<T> DeSerialize<T>(HttpResponseMessage response) where T : class
    {
        string stringData = await response.Content.ReadAsStringAsync();
        JsonSerializerOptions options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        return JsonSerializer.Deserialize<T>(stringData, options);
    }

    public async Task<HttpResponseMessage> SerializePost<T>(T model, string apiUrl) where T : class
    {
        string stringData = JsonSerializer.Serialize(model);
        StringContent contentData = new StringContent(stringData, Encoding.UTF8, "application/json");
        HttpResponseMessage response = await _client.PostAsync(apiUrl, contentData);
        return response;
    }

    public async Task<HttpResponseMessage> SerializePut<T>(T model, string apiUrl) where T : class
    {
        string stringData = JsonSerializer.Serialize(model);
        StringContent contentData = new StringContent(stringData, Encoding.UTF8, "application/json");
        HttpResponseMessage response = await _client.PutAsync(apiUrl, contentData);
        return response;
    }

    public async Task<bool> Delete(string apiUrl)
    {
        HttpResponseMessage response = await _client.DeleteAsync(apiUrl);
        return response.IsSuccessStatusCode;
    }
}

```

Figuur 30: Class Serializer

Deze wordt dan gebruikt in de andere classes om de API aan te spreken zoals de class QuestionData.

```
public class QuestionData : IQuestionData
{
    private readonly ISerializer _serializer;
    private readonly IMapper _mapper;

    public QuestionData(ISerializer serializer, IMapper mapper)
    {
        this._serializer = serializer;
        this._mapper = mapper;
    }

    public async Task<Question> Load(int id)
    {
        return await _serializer.DeSerialize<Question>(${"ApiUrl.Question}/{id}");
    }

    public async Task<IEnumerable<Question>> LoadAsync(CancellationToken ct =
default)
    {
        return await _serializer.DeSerialize<IEnumerable<Question>>(ApiUrl.Question);
    }
    public async Task<Question> Add(Question item)
    {
        var response = await
_serializer.SerializePost(_mapper.Map<QuestionInput>(item), ApiUrl.Question);
        return await _serializer.DeSerialize<Question>(response);
    }

    public async Task<Question> Update(Question item)
    {
        var response = await _serializer.SerializePut(item, ApiUrl.Question);
        return await _serializer.DeSerialize<Question>(response);
    }

    public Task Remove(Question item)
    {
        return _serializer.Delete(${"ApiUrl.Question}/{item.Id}");
    }
}
```

Figuur 31: Class QuestionData

## B. Client

Voor de client heb ik gekozen voor Blazor Server. Dit heb ik gekozen na het volgen van een cursus van + 12 uur op Udemy. De reden dat ik hiervoor gekozen heb, is omdat de code allemaal op één pagina staat en deze makkelijk toegankelijk is om een API aan te spreken.

Om de gegevens weer te geven, heb ik ervoor gekozen om met de extensie DevExpress te werken. Dit was een heel leerproces om met de min- en pluspunten te leren werken. Ik heb één component gemaakt als template voor de Data Grid die de volgende parameters aanvaardt. De volledige code die deze component gebruikt, geef ik niet mee in dit document aangezien deze te groot is.

```
[Parameter] public RenderFragment Columns { get; set; }
[Parameter] public RenderFragment<TInput> EditFormItems { get; set; }

[Parameter] public string LayoutName { get; set; }
[Parameter] public bool CreateNewEnabled { get; set; } = true;
[Parameter] public bool ShowEnabled { get; set; } = true;
[Parameter] public bool EditEnabled { get; set; } = true;
[Parameter] public bool DeleteEnabled { get; set; } = true;
[Parameter] public bool NavigationEnabled { get; set; } = false;
[Parameter] public bool EditLogEnabled { get; set; } = false;
[Parameter] public bool FilesEnabled { get; set; } = false;
[Parameter] public bool FilterEnabled { get; set; } = true;

[Parameter] public Action OnAddClick { get; set; }
[Parameter] public Action<bool> BeforeValidSubmit { get; set; }
[Parameter] public Action<bool> OnValidSubmitAction { get; set; }
[Parameter] public Action BeforeRowEditStart { get; set; }
[Parameter] public Action OnShowClick { get; set; }
[Parameter] public Action OnEditClick { get; set; }
[Parameter] public Action OnDeleteClick { get; set; }
[Parameter] public Action OnEditLogClick { get; set; }
[Parameter] public Action OnFilesClick { get; set; }
[Parameter] public Action OnNavigationClick { get; set; }
[Parameter] public Action FilterData { get; set; }
[Parameter] public Func<System.Threading.CancellationToken, Task<IEnumerable<TModel>>>
SetDataAsync { get; set; }
[Parameter] public Func<Task> BeforeRefresh { get; set; }
[Parameter] public Func<Task> AfterRefresh { get; set; }
```

Figuur 32: Parameters DataGrid Component

```

<DxDataGrid @ref="grid"
    Data="@data"
    RowEditStartAsync="@((dataItem => OnRowEditStarting(dataItem)))"
    RowInsertStartAsync="@(() => OnRowEditStarting(default(TModel)))"
    EditMode="@CurrentEditMode"
    ShowFilterRow="@ShowFilterRow"
    SelectionMode="@DataGridSelectionMode.SingleSelectedDataRow"
    BindSingleSelectedDataRow="@SelectedRow"
    HorizontalScrollBarMode="@ScrollBarMode.Auto"
    VerticalScrollBarMode="@ScrollBarMode.Auto"
    VerticalScalableHeight="300"
    PagerNavigationMode="@PagerNavigationMode.InputBox"
    CssClass="mygrid"
    PagerAllowedPageSizes="@(new int[] { 10, 20, 30, 40, 50, 100 })"
    PageSize="50"
    ShowPager="true"
    PagerAllDataRowsItemVisible="false"
    PagerPageSizeSelectorVisible="true"
    LayoutChanged="@((async layout => await OnLayoutChange(layout)))"
    LayoutRestoring="@((async layout => await OnLayoutRestore(layout)))"
<HeaderTemplate>
    <DxToolbar ItemRenderStyleMode="ToolbarRenderStyleMode.Plain" CssClass="custtoolbar">
        @if (CreateNewEnabled)
            {<DxToolbarItem Text="Nieuw" BeginGroup="@CreateNewEnabled" Click="@OnAddButtonClick"
                IconCssClass="oi oi-plus" CssClass="custtoolbarbutton"
                Tooltip="Nieuw" />}
        @if (ShowEnabled)
            {<DxToolbarItem Text="Weergeven" BeginGroup="@( !CreateNewEnabled && ShowEnabled)" Click="@OnShowButtonClick"
                IconCssClass="oi oi-file" CssClass="custtoolbarbutton"
                Tooltip="Weergeven" Enabled="@Enabled" />}
        @if (EditEnabled)
            {<DxToolbarItem Text="Bewerken" BeginGroup="@( !CreateNewEnabled && !ShowEnabled && EditEnabled)"
                Click="@OnEditButtonClick"
                IconCssClass="oi oi-pencil" CssClass="custtoolbarbutton"
                Tooltip="Bewerken" Enabled="@Enabled" />}
        @if (DeleteEnabled)
            {<DxToolbarItem Text="Verwijderen" BeginGroup="@( !CreateNewEnabled && !ShowEnabled && !EditEnabled &&
DeleteEnabled)" Click="@OnDeleteButtonClick"
                IconCssClass="oi oi-trash" CssClass="custtoolbarbutton"
                Tooltip="Verwijderen" Enabled="@Enabled" />}
        @if (NavigationEnabled)
            {<DxToolbarItem Text="Navigeren" BeginGroup="@NavigationEnabled" Click="@OnNavigationClick"
                IconCssClass="oi oi-menu" CssClass="custtoolbarbutton"
                Tooltip="Navigeren" Enabled="@Enabled" />}
        @if (EditLogEnabled)
            {<DxToolbarItem Text="Geschiedenis" BeginGroup="@( !NavigationEnabled && EditLogEnabled)"
                Click="@OnEditLogClick" IconCssClass="oi oi-timer" CssClass="custtoolbarbutton"
                Tooltip="Geschiedenis" Enabled="@Enabled" />}
        @if (FilesEnabled)
            {<DxToolbarItem Text="Bestanden" BeginGroup="@( !NavigationEnabled && !EditLogEnabled && FilesEnabled)"
                Click="@OnFilesClick"
                IconCssClass="oi oi-paperclip"
                Tooltip="Bestanden" Enabled="@Enabled" />}
        @if (FilterEnabled)
            {<DxToolbarItem Text="Filteren" BeginGroup="true" GroupName="FilterRow" Click="@OnShowFilterRow"
                IconCssClass="oi oi-magnifying-glass" CssClass="custtoolbarbutton"
                Tooltip="Filter" />}
    <DxDataGridColumnChooserToolbarItem BeginGroup="true" Name="Kolommen kiezen"
    Alignment="ToolbarItemAlignment.Right" CssClass="custtoolbarbutton" />
</DxToolbar>
</HeaderTemplate>
<Columns>
    <Columns>
</Columns>
</Columns>
<EditFormTemplate>
    <EditForm Model="@InputModel" Context="InputContext" OnValidSubmit="@OnValidSubmit">
        <DataAnnotationsValidator />
        <DxFmLayout>
            @EditFormItems(InputModel)
        </DxFmLayout>
    </EditForm>
</EditFormTemplate>
</DxDataGrid>

```

Figuur 33: DataGrid HTML Component

Deze template beschikt over een navigatiebar, CRUD opties, een filterrij, een kolomkiezer, de mogelijkheid om de volgorde van de kolommen te kiezen en een formulier met validatie. De filters en de kolommen worden opgeslagen in de database per gebruiker zodat de volgende keer wanneer je de tabel aanroeft, deze nog steeds dezelfde settings heeft.

Als voorbeeld kan je hier de pagina 'Questions' zien, die deze template component aanroeft.

```
@inject IAnswerTypeData AnswerTypeData
@Inject IQuestionCategoryData QuestionCategoryData

<DxDataGridComboBoxColumn Field="@nameof(Question.QuestionCategoryId)"
    Caption="@Model.DisplayName(nameof(Model.QuestionCategoryId))"
    DataAsync="@QuestionCategoryData.LoadAsync"
    ValueFieldName="@nameof(QuestionCategory.Id)"
    TextFieldName="@nameof(QuestionCategory.Title)"
    FilteringMode="DataGridFilteringMode.Contains"
    Width="200px">
</DxDataGridComboBoxColumn>
<DxDataGridComboBoxColumn Field="@nameof(Question.AnswerTypeId)"
    Caption="@Model.DisplayName(nameof(Model.AnswerTypeId))"
    DataAsync="@AnswerTypeData.LoadAsync"
    ValueFieldName="@nameof(AnswerType.Id)"
    TextFieldName="@nameof(AnswerType.Title)"
    FilteringMode="DataGridFilteringMode.Contains"
    Visible="false"
    Width="200px">
</DxDataGridComboBoxColumn>

@code{
    [Parameter]
    public Question Model { get; set; }
}
```

Figuur 34: QuestionColumns Component

```

@inject IAnswerTypeData AnswerTypeData
@inject IQuestionCategoryData QuestionCategoryData

<DxFormLayoutItem Caption="@Model.DisplayName(nameof(Model.AnswerTypeId))" ColSpanMd="12"
Context="FormLayoutContext">
    <Template>
        <DxComboBox ValueFieldName="@nameof(AnswerType.Id)"
            FilteringMode="@DataGridFilteringMode.Contains"
            EditFormat="{0}"
            DataAsync="@AnswerTypeData.LoadAsync"
            NullText="Selecteer een soort"
            @bind-Value="@InputModel.AnswerTypeId"
            ReadOnly="@ReadOnly">
            <DxListEditorColumn FieldName="@nameof(AnswerType.Title)"/>
            <DxListEditorColumn FieldName="@nameof(AnswerType.Description)"/>
        </DxComboBox>
    </Template>
</DxFormLayoutItem>
<DxFormLayoutItem Caption="@Model.DisplayName(nameof(Model.QuestionCategoryId))" ColSpanMd="12" Context="FormLayoutContext">
    <Template>
        <DxComboBox ValueFieldName="@nameof(QuestionCategory.Id)"
            FilteringMode="@DataGridFilteringMode.Contains"
            EditFormat="{0}"
            DataAsync="@QuestionCategoryData.LoadAsync"
            NullText="Selecteer een category"
            @bind-Value="@InputModel.QuestionCategoryId"
            ReadOnly="@ReadOnly">
            <DxListEditorColumn FieldName="@nameof(QuestionCategory.Title)"/>
            <DxListEditorColumn FieldName="@nameof(QuestionCategory.Description)"/>
        </DxComboBox>
    </Template>
</DxFormLayoutItem>

@code{
    [Parameter]
    public Question Model { get; set; }

    [Parameter]
    public QuestionInput InputModel { get; set; }

    [Parameter]
    public bool ReadOnly { get; set; } = false;
}

```

Figuur 35: QuestionFormLayout Component

```

@page "/Questions"
@page "/Question/{questionId:int}"
@page "/Questions/Category/{categoryId:int}"

@using FS_ControleFormulieren_BazorServerApp.Pages.CF.RelationBase
@inject IQuestionData QuestionData
@inject NavigationManager NavigationManager
<h3 class="pagetitle"><u>Vragen</u></h3>

<DataGridComponent TModel="Question" TInput="QuestionInput" LayoutName="Questions"
    InputModel="InputModel" Data="QuestionData" @ref="grid" FilterData="@FilterData"
    NavigationEnabled="true" OnNavigationClick="@(() => NavigationIsVisible = true)"
    EditLogEnabled="true"
    OnEditLogClick="@OnEditLogClick">
    <Columns>
        <RelationBaseColumns Model="Model" />
        <QuestionColumns Model="Model" />
    </Columns>
    <EditFormItems Context="inputModel">
        <RelationBaseFormLayout Model="Model" InputModel="inputModel" ReadOnly="@ReadOnly" />
        <QuestionFormLayout Model="Model" InputModel="inputModel" ReadOnly="@ReadOnly" />
        <DataGridValidationBase OnCancelButtonClick="@OnCancelButtonClick" ReadOnly="@ReadOnly" />
    </EditFormItems>
</DataGridComponent>

<NavigationTemplate NavigationIsVisible="@NavigationIsVisible">
    <NavigationButtons>
        @{
            Question q = grid.SelectedRow;
            string label = q.DisplayName(nameof(q.QuestionCategoryId));
            <DxButton Click="@(()=>NavigationManager.NavigateTo($"Category/{q.QuestionCategoryId}"))">
                @label
            </DxButton>
        }
    </NavigationButtons>
</NavigationTemplate>

@code {
    [Parameter] public int CategoryId { get; set; }
    [Parameter] public int QuestionId { get; set; }
    public bool NavigationIsVisible { get; set; }

    DataGridComponent<Question, QuestionInput> grid { get; set; }

    Question Model { get; set; } = new Question();
    QuestionInput InputModel { get; set; } = new QuestionInput();

    public bool ReadOnly { get { return grid.ReadOnly; } set { grid.ReadOnly = value; } }

    public void FilterData()
    {
        if (CategoryId > 0)
            grid.data = grid.data
                .Where(q => q.QuestionCategoryId == CategoryId);

        if (QuestionId > 0)
            grid.data = grid.data
                .Where(q => q.Id == QuestionId);
    }

    public void OnCancelButtonClick()
    {
        grid.OnCancelButtonClick();
    }

    public void OnEditLogClick()
    {
        NavigationManager.NavigateTo($"EditLog/{grid.SelectedRow.Id}");
    }
}

```

Figuur 36: Questions Page

Alle pagina's gebruiken de officiële kleuren en lettertypes van Footstep. Deze heb ik gedefinieerd in my-styles.css.

```
@font-face {
    font-family: "Musea Sans Rounded 100";
    src: url(ContentFiles/MuseoSansRounded100.otf)
}

@font-face {
    font-family: "Musea Sans Rounded 300";
    src: url(ContentFiles/MuseoSansRounded300.otf);
}

@font-face {
    font-family: "Musea Sans Rounded 500";
    src: url(ContentFiles/MuseoSansRounded500.otf);
}

@font-face {
    font-family: "Musea Sans Rounded 700";
    src: url(ContentFiles/MuseoSansRounded700.otf);
}

@font-face {
    font-family: "Musea Sans Rounded 900";
    src: url(ContentFiles/MuseoSansRounded900.otf);
}

:root {
    --font1: 'Musea Sans Rounded 900';
    --font2: 'Musea Sans Rounded 700';
    --font3: 'Musea Sans Rounded 500';
    --font4: 'Musea Sans Rounded 300';
    --font5: 'Musea Sans Rounded 100';
    --bodygreen: rgb(189,191,55);
    --logogreen: rgb(169,166,50);
    --bodygrey: rgb(192,192,192);
    --logodarkgrey: rgb(113,105,103);
    --tekstgrey: rgb(94,94,94);
    --button1: rgb(161,163,52);
    --button2: rgb(201,203,93);
    --colorroweven: rgb(247,247,231);
    --colorrowodd: rgb(241,241,225);
}
```

Figuur 37: declaratie variabelen my-styles

Deze konden zo gemakkelijk gebruikt worden in mijn verdere css styles. Hieronder kan je hiervan een voorbeeld terugvinden.

```

Grid algemeen
.mygrid {
    color: var(--tekstgrey);
    background-color: var(--bodygreen);
    font-family: var(--font5);
    border: none;
}

/*Grid rows algemeen*/
.mygrid table tr {
    color: var(--tekstgrey);
    font-family: var(--font3);
    font-size: small;
    /* height: auto;*/
}

/*grid altererende row colors*/
.mygrid table tr:nth-child(even) {
    background-color: var(--colorroweven);
}

.mygrid table tr:nth-child(odd) {
    background-color: var(--colorrowodd);
}

/*Grid header*/
.mygrid table th {
    background-color: var(--logodarkgrey);
    color: white;
    font-family: var(--font4);
    font-size: medium;
}

/*Toolbar*/
.mygrid .custtoolbar {
    background-color: var(--bodygreen);
    border-radius: 30px 0 30px 0;
    font-family: var(--font2);
}

.mygrid .custtoolbar .btn-group:first-of-type button:first-of-type {
    border-radius: 30px 0 0 0;
    padding-left: 20px;
}

.mygrid .custtoolbar .btn-group:last-of-type button:last-of-type {
    border-radius: 0 0 30px 0;
    padding-right: 20px;
}

.mygrid .custtoolbar button:hover {
    color: var(--logodarkgrey) !important;
    background-image: linear-gradient(to top, var(--button1), var(--button2));
}

.mygrid .custtoolbar button:active {
    background-image: linear-gradient(to bottom, var(--button1), var(--button2)) !important;
}

.mygrid .custtoolbar button:visited {
    background-image: linear-gradient(to bottom, var(--button1), var(--button2)) !important;
}

.btn-secondary:not(:disabled):not(.disabled).active,
.show > .btn-secondary.dropdown-toggle {
    background-color: var(--logodarkgrey) !important;
}

```

Figuur 38: Deel van my-styles.css

## Besluit

Voor mij was dit project een mooie uitdaging waar ik heel veel van geleerd heb. Wat ik vooral geleerd heb, is dat ik het project beter eerst volledig uitdenk vooraleer te programmeren. Dit is natuurlijk ook wel moeilijk op dit moment omdat ik nog niet veel ervaring heb. Hierdoor denk ik voordien dat iets mogelijk is, maar merk ik later dat het niet mogelijk is. (Vb: TBT van EF Core 3.1)

Doorheen mijn project heb ik af en toe zaken die ik eerder geprogrammeerd had, moeten herprogrammeren. Dit omdat deze uiteindelijk niet voldoende waren. Door het project eerder grondig uit te denken, zou ik redelijk wat tijd kunnen uitsparen.

Tijdens het uitwerken van het project heb ik gemerkt dat het gebruiken van extensies zowel voor- als nadelen heeft. Het voordeel hiervan is dat je bepaalde zaken veel sneller kan bekomen door minder code te typen. Anderzijds is het nadeel dat een extensie moeilijk uit te breiden of aan te passen is. Zolang je een extensie gebruikt voor enkel hetgeen waarvoor deze dient, werkt het heel goed en kan je veel tijd winnen. Maar vanaf dat je iets meer ermee wil doen, kan het zijn dat je op kleine of grote zaken vastloopt of veel tijd verliest. Dit is natuurlijk wel iets dat je door ervaring kan leren. Hoe vaker je een extensie gebruikt, hoe beter je de voordelen in je eigen voordeel kan gebruiken en de nadelen kan omzeilen of achterwege laten. Het nadeel van extensies blijft wel dat, wanneer er een update is of de extensie wordt niet meer ondersteund, er kans is dat je code niet meer werkt.

Tijdens het programmeren merkte ik af en toe op dat ik bepaalde delen code veel moest herschrijven. Daarvoor heb ik dan uiteindelijk een aparte class of template gemaakt. Op die manier kroop er veel werk in die class of template, maar uiteindelijk won ik er toch tijd mee door deze dan te implementeren in mijn code. Zoals echter eerder geschreven in mijn besluit, mocht ik mijn project beter uitgedacht hebben, kon ik meteen deze class of template maken zonder eerst onnodig de code te schrijven.

Als laatste wil ik nogmaals afsluiten met het feit dat ik enorm veel heb bijgeleerd. Zowel in de lessen als bij zelfstudie. Ik hoop dat ik deze opgedane kennis vaak zal kunnen gebruiken in mijn carrière.

## Bijlagen

Bijlage 1: Kostenprijsberekening

Syntra West Brugge

Opleiding C# programmeur

Avondonderwijs

# Kostenprijsberekening

## Controleformulieren Footstep

Onderdeel van het eindwerk voor het behalen  
van het getuigschrift van C# programmeur

door Yordi Van Nieuwenhuyse

o.l.v. Helena Coppieters

Academiejaar 2019 – 2020

Syntra West Brugge • Spoorwegstraat 14 • B-8200 Brugge • Tel. +32 78 35 36 53 • [www.syntrawest.be](http://www.syntrawest.be)

38

Beste klant,

Allereerst wil ik u bedanken voor de aanvraag tot offerte.

Ik vertrouw erop u met deze offerte een passend voorstel te doen.

**Deze offerte is van toepassing op de genoemde wensen in bijlage.**

Omschrijving	Bedrag	Totaal
Fase A: Database 15 uur	€ 1 200	€ 1 200
- Tabellen - Views - Stored procedures - Testdata		
Fase B: API 55 uur	€ 4 400	€ 4 400
Fase C: Blazor Server 60 uur	€ 4 800	€ 4 800
- Homescreen - Artikels scherm - PO Lijnen scherm - Routings scherm - Formulieren scherm - Vragen scherm - Groepen scherm - Categorieën scherm - Logging scherm		
Fase D: upload naar IIS 10 uur	€ 800	€ 800
- Upload - Testen - Finetunen		
	Totaal	€ 11 200

Indien u nog verdere vragen heeft, hoor ik dit graag.

Met vriendelijke groet,

Yordi Van Nieuwenhuyse

## Bijlage: uiteenzetting werkuren

Nummer	Module/scherm	Ingeschatte tijd (in uren)
1	<b>Database</b>	<b>Totaal: 15</b>
	Aanmaken tabellen	3
	Aanmaken views	4
	Aanmaken stored procedures	3
	Aanmaken Testdata	1
	Optimalisatie	4
2	<b>API</b>	<b>Totaal: 55</b>
	Slapper, dapper en automapper	5
	Repository voor Business Central	8
	Lay out	2
	Business layer	4
	Exception logging	2
	Classes aanmaken	3
	UseCase aanmaken en verbeteren	15
	Verschillende controllers	9
	Optimalisatie	7
3	<b>Blazor Server</b>	<b>Totaal: 60</b>
	Datagrid	10
	Toolbar voor datagrid	6
	Filter opties	4
	Verschuifbare kolommen	6
	Opslaan van datagrid settings	4
	Thema 'Footstep'	11
	Form validaties	9
	Optimalisatie	10

Nummer	Module/scherm	Ingeschatte tijd (in uren)
4	Upload naar IIS	<b>Totaal: 10</b>
	Upload, testen en finetunen	10
	<b>Totale werkuren</b>	<b>140</b>

Bijlage 2: Gepresteerde uren

**Syntra West Brugge**  
**Opleiding C# programmeur**

Avondonderwijs

# **Gepresteerde uren**

## **Controleformulieren Footstep**

Onderdeel van het eindwerk voor het behalen  
van het getuigschrift van C# programmeur

door Yordi Van Nieuwenhuyse

o.l.v. Helena Coppieters

Academiejaar 2019 – 2020

Syntra West Brugge • Spoorwegstraat 14 • B-8200 Brugge • Tel. +32 78 35 36 53 • [www.syntrawest.be](http://www.syntrawest.be)

In onderstaand document omschrijf ik welke stappen ik ondernomen heb om tot het eindproduct van mijn eindwerk te komen. Tijdens het proces ondervond ik soms struikelblokken of zag ik dat bepaalde zaken op een andere manier beter konden. De weg naar mijn eindproduct heeft me veel bijgeleerd, maar zelf heb ik ook heel wat zaken opgezocht om het te verbeteren. Via 'trial and error' kwam ik tot mijn bestaand eindwerk. In onderstaande oplijsting zal ik dus ook een opsomming maken van projecten die mijn eindwerk niet haalden of die ik later verbeterde.

## API Versie Entity Framework Core (EF Core)

Met dit onderdeel startte ik mijn project. Naarmate ik verder kwam, merkte ik meer en meer problemen op. Deze problemen kon ik vaak zelf oplossen, maar uiteindelijk kon ik deze EF Core niet gebruiken omdat de Table Per Type (TPT) bij deze versie nog niet ondersteund wordt. EF Core 5 (komt uit in november 2020), zal dit wel ondersteunen.

Onderstaande werkuren zijn dus gepresteerd, maar uiteindelijk geen onderdeel van het eindwerk.

Datum	Taken	Gepresteerde tijd (in uren)
25 juni '20	Aanmaken project	0
25 juni '20	Item + RoutingLines Class + API (externe database)	3
27 juni '20	Werking van Read API testen (externe database)	1
28 juni '20	Classes uitdenken voor eigen database	2
29 juni '20	Uittypen classen	4
22 juli '20	Classen in afzonderlijke library zetten	2
21 aug. '20	EF Core integreren in context + repositories	6
21 aug. '20	Automapper + services	2
22 aug. '20	EF fluent API voor gewenste database te krijgen	7
22 aug. '20	Simpele controller gemaakt om GET/POST/PUT/DELETE te testen met postman	1
22 aug. '20	Problemen trachten op te lossen 'fluent API → TPT'	6
23 aug. '20	Problemen trachten op te lossen 'fluent API → TPT'	12
23 aug. '20	Oplossingen opzoeken op forums. Gemerkt dat dit niet ondersteund wordt. Beslist om helemaal opnieuw te starten.	1

## Gebruikte API

Na de ondervonden problemen met de API versie met EF Core, heb ik besloten om een andere API aan te maken. Hierbij maakte ik zelf mijn database die hierin kan gebruikt worden. Om hierbij tot mijn eindproduct te komen, leerde ik gaandeweg bij over de extensies 'slapper' en 'dapper'. De andere zaken kon ik gebruiken en toepassen dankzij de aangeleerde kennis uit de lessen.

Datum	Taken	Gepresteerd tijd (in uren)
23 aug. '20	DB aanmaken in DB project – tables	3
23 aug. '20	Aanmaken views DB	3
23 aug. '20	Aanmaken Stored procedures DB	3
23 aug. '20	Aanmaken Testdata in script DB	1
25 aug. '20	Finalising test data input DB	1
25 aug. '20	Toevoegen slapper, automapper + services	2
25 aug. '20	Repositories aanmaken voor BC	2
25 aug. '20	Testen data = mislukt. ‘_’ verwijderen kolom benamingen van de queries → views herschrijven zonder spaties en ‘_’	2
25 aug. '20	Controller aanmaken om de repository van BC te testen	1
25 aug. '20	Hier en daar wat zaken wijzigen zodat alles (slapper, dapper en automapper) samen werkt.	2
27 aug. '20	Ondervonden dat mijn project te onoverzichtelijk werd. → lay-out beginnen gebruiken (vb. Taak dagelijks werk: Invoices)	2
27 aug. '20	Business layer aangemaakt met de nodige mappers	3
29 aug. '20	Exception logging toevoegen aan het project	2
29 aug. '20	Classes voorzien van business rules die gelogd worden bij foutieve items	3
29 aug. '20	Repositories aanmaken voor CF	4

Datum	Taken	Gepresteerde tijd (in uren)
30 aug. '20	UseCase maken voor Question + EditLog + Exception	3
30 aug. '20	Automapper 'for members' toegevoegd	1
30 aug. '20	Controller voor testen Question table aangemaakt met exception handling inbegrepen	3
30 aug. '20	Controller met question table vertoonde hier en daar problemen, dus dan overgeschakeld naar AnswerType table. Deze heeft geen foreign keys of IEnumerables onder zich. Waardoor het debuggen wat makkelijker verliep.	2
30 aug. '20	Business rules verbeterd	1
31 aug. '20	Een enkele UseCase gemaakt voor alles dat inherit is aan relationBase → via het DRY principe	3
31 aug. '20	Solution opgeschoond	2
5 sept. '20	Andere controllers en UseCase aangemaakt	8
7 sept. '20	UseCases en controllers verder afgewerkt. → eerste stabiele versie van de API	4
8 sept. '20	Stored procedures aangepast → fout met EditLog	1
8 sept. '20	De externe database zit met een fout in hun entry nummers: er zaten dubbele entry nummers in. → primary key + foreign keys aanpassen in alle classes + repositories + UseCases,...	2
10 sept. '20	Indexen geplaatst op tables + Queries verbeterd. Ook i.p.v. foreign key meteen op te slaan in hun class, deze als een property bij de class gezet. → views, stored procedures, classes, repositories, UseCases hieraan aangepast	4
19 sept. '20	Upload naar IIS (lang zitten zoeken bij een probleem voor het aanspreken van de database)	5

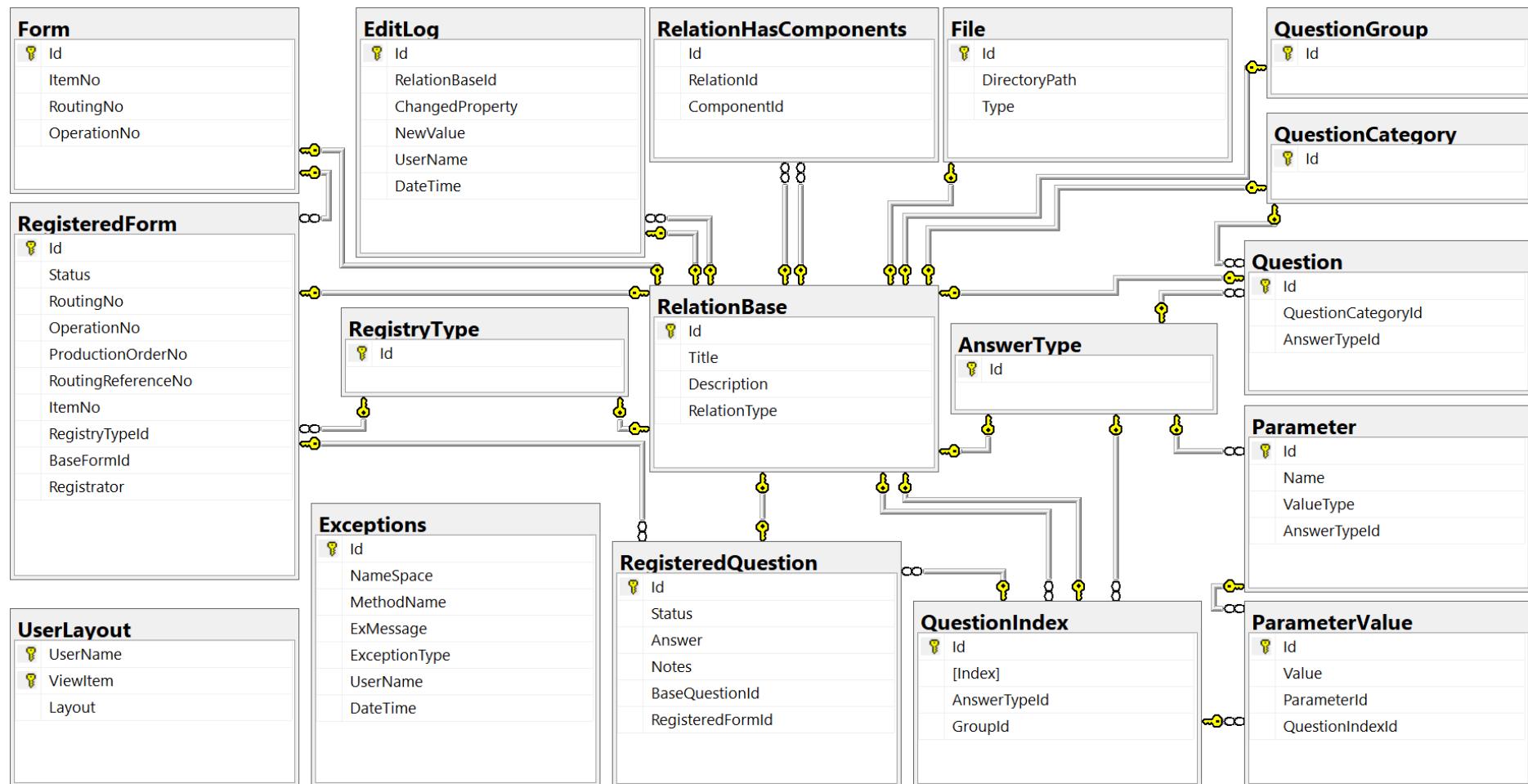
## Blazor Server

Tijdens de lessen hadden we nog niets geleerd over visuals. Voordat ik hier dus voor mijn project mee aan de slag kon, moest ik deze me eigen maken. Ik zocht enkele online cursussen op die me hierbij konden helpen. Door het volgen van deze cursussen en het maken van de bijhorende oefeningen, kon ik ook met de visuals aan de slag. Bij dit onderdeel kroop er dus ook veel tijd in het bijleren van de verschillende functies. Deze heb ik daarna geïmplementeerd in mijn eindwerk.

Datum	Taken	Gepresteerde tijd (in uren)
3 sept. '20	Leren over verschillende mogelijkheden binnen C# E-learning cursus: 12 uren cursus (6 uur oefeningen) <a href="https://www.udemy.com/course/which-aspnet-core">https://www.udemy.com/course/which-aspnet-core</a>	6
4 sept. '20	Leren over verschillende mogelijkheden binnen C# E-learning cursus: 12 uren cursus (6 uur oefeningen) <a href="https://www.udemy.com/course/which-aspnet-core">https://www.udemy.com/course/which-aspnet-core</a>	6
6 sept. '20	Leren over verschillende mogelijkheden binnen C# E-learning cursus: 12 uren cursus (6 uur oefeningen) <a href="https://www.udemy.com/course/which-aspnet-core">https://www.udemy.com/course/which-aspnet-core</a>	6
7 sept. '20	Integreren blazor server	3
7 sept. '20	Project met de nodige API models, input models en Enums toegevoegd	3
8 sept. '20	Testen eerste visuals QuestionList table + API verder aanpassen (zie gebruikte API)	3
8 sept. '20	Zoeken naar een manier om datagrids goed te benutten in mijn views → gebruik DevExpress	3
9 sept. '20	Leren hoe te werken met DevExpress en hoe repositories op te zetten hiervoor	3
10 sept. '20	Aanpassen EntryNo → ItemNo (zie gebruikte API)	3
10 sept. '20	Laden van lijsten ging heel traag, verschillende perfomantie boosts gedaan in webserver + API (zie gebruikte API)	3

Datum	Taken	Gepresteerde tijd (in uren)
11 sept. '20	Datagrid inkleuren volgens de layout van Footstep	2
12 sept. '20	Toolbar uitbreiden en in werking stellen	3
12 sept. '20	Template maken van werkendedatagrid	4
13 sept. '20	Allerhande bugs wegwerken van de template	3
13 sept. '20	Template gebruiken voor andere views	6
14 sept. '20	Template gebruiken voor andere views	6
15 sept. '20	Editforms opmaken met datavalidatie	6
19 sept. '20	Upload naar IIS	1

### Bijlage 3: Database Model



Figuur 39: Eigen Database diagram

Database BC (te groot om een diagram van te maken, daarom deze volgende links)

- Tabel items  
<https://dynamicsdocs.com/nav/2018/w1/table/item>
- Tabel RoutingHeader  
<https://dynamicsdocs.com/nav/2018/w1/table/routing-header>
- Tabel RoutingLine  
<https://dynamicsdocs.com/nav/2018/w1/table/routing-line>
- Tabel ProductionOrderLine  
<https://dynamicsdocs.com/nav/2018/w1/table/prod-order-line>
- Tabel ProductionOrderRoutingLine  
<https://dynamicsdocs.com/nav/2018/w1/table/prod-order-routing-line>

## Geraadpleegde links

<https://useiconic.com/open>

<https://demos.devexpress.com/blazor/GridEditFormTemplateValidation>

<https://www.udemy.com/course/which-aspnet-core/>

[https://www.youtube.com/watch?v=BBFF2l1FIE0&ab\\_channel=CuriousDrive](https://www.youtube.com/watch?v=BBFF2l1FIE0&ab_channel=CuriousDrive)

<https://chrissainty.com/3-ways-to-communicate-between-components-in-blazor/>

[https://www.youtube.com/watch?v=8DNgdphLvag&ab\\_channel=IAmTimCorey](https://www.youtube.com/watch?v=8DNgdphLvag&ab_channel=IAmTimCorey)

<https://www.youtube.com/channel/UCetyodKOWGk5H6FoKoFnkZw>

<https://www.w3schools.com/>

<https://dapper-tutorial.net/dapper>

<https://github.com/SlapperAutoMapper/Slapper.AutoMapper>

<https://www.entityframeworktutorial.net/efcore/fluent-api-in-entity-framework-core.aspx>

<https://www.thinktecture.com/en/entity-framework-core/table-per-type-inheritance-support-part-1-code-first/#:~:text=The%20Entity%20Framework%20Core%20,current%20version%20of%20EF%20Core.>

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/inheritance?view=aspnetcore-3.1>

<https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1>

[https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/cc716693\(v=vs.100\)](https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/cc716693(v=vs.100))

<https://devblogs.microsoft.com/dotnet/announcing-entity-framework-core-ef-core-5-0-preview-8/>

<https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/?fbclid=IwAR0wJpm7ydWO5JghzJmWOKiMdar-ZbVgcZ5Ww0ADSezPjt52abVZu47Z-fg#:~:text=3%2Dtier%20architectures%20provide%20many,independently%20of%20the%20other%20parts>