

Stage NRDS

Ruben Heurckmans

Bachelor in de Toegepaste Informatica
keuzerichting Application Development

Inhoud

1. INLEIDING	4
1.1. Het stagebedrijf	4
1.2. Mijn opdrachten	4
1.2.1. DTOS	4
1.2.2. SFPP	5
2. PLANNING	6
3. ONDERZOEK	7
3.1. DTOS	7
3.1.1. Vue	7
3.1.2. .NET	7
3.2. SFPP	8
3.2.1. Frameworks	8
4. REALISATIE	9
4.1. DTOS	9
4.1.1. Urenregistratie	9
4.1.2. Refactor Back-end	11
4.1.3. Refactor Front-end	12
4.1.4. Bugfixes	12
4.2. Projectmanagementmodule SFPP	17
4.2.1. Soft delete functionaliteit	17
4.2.2. Soft delete testen	19
4.2.3. Soft delete implementatie	20
5. CONCLUSIE	22

Figurenlijst

Figuur 1: Api endpoints HourRegistrationController	9
Figuur 2: Urenregistratie tabel	10
Figuur 3: Api endpoint voor urenregistratie dashboard	10
Figuur 4: Dashboard urenregistratie	11
Figuur 5: ServerMinimalDto	11
Figuur 6: ServerDto	11
Figuur 7: Project dashboard	13
Figuur 8: Project tabel	13
Figuur 9: Resources organigram	13
Figuur 10: Applicaties tabel	14
Figuur 11: Organigram overzicht	14
Figuur 12: Datacenters tabel	14
Figuur 13: Servers tabel	15
Figuur 14: Netwerkbereiken tabel	15
Figuur 15: Switches tabel	15
Figuur 16: Firewalls tabel	16
Figuur 17: Locaties tabel	16
Figuur 18: EntityState.Delete tegenhouden	17
Figuur 19: Functie SoftDelete	18
Figuur 20: Functie SetSoftDelete	18
Figuur 21: Functie SoftDeleteChildren	18
Figuur 22: Functie GetQueryAbleForParentcascade	18
Figuur 23: Voorbeeld van GetQueryAbleForParentcascade	18
Figuur 24: SetSoftDelete test	19
Figuur 25: GetQueryAbleForParentCascade test	20
Figuur 26: SoftDelete test	20
Figuur 27: baseDelete functie in BaseApiService	21
Figuur 28: delete functie in ModelxApiService	21
Figuur 29: remove functie in ModelxStore	21
Figuur 30: removehandlers in ModelxComponent.vue	21
Figuur 31: canDelete functie in ModelxComponent.vue	21
Figuur 32: Button en dialoog venster in ModelxComponent.vue	21

1. Inleiding

Een van de laatste onderdelen van de opleiding, is de stage. Hierbij worden verschillende vaardigheden getest om te kijken of de student klaar is voor het werkveld. Men leert tijdens de stage het werkveld kennen. Hierbij hoort communicatie met klanten, werken met collega's, kennis opdoen en nog veel meer.

1.1. Het stagebedrijf

Het bedrijf waar ik stage bij heb gelopen is NRDS. Dit is een klein IT bedrijf dat websites op maat en webapplicaties maakt maar. Daarnaast bieden ze ook advies en begeleiding aan. Ze werken met start-ups en scale-ups, en herstructureren ook software als de projecten interessant zijn. NRDS faciliteert ook de digitale transformatie voor lokale overheden met robuuste ICT, geïntegreerde oplossingen en sterke beveiliging, waardoor een soepele overgang naar een digitale werkomgeving mogelijk wordt.

Hierbij heb ik de rol aangepakt als full stack developer die meedraaide in het team.

Door mee te draaien in hun team had ik niet echt het gevoel dat ik een stagiaire was maar eerder een gewone werknemer. Hierdoor was de sfeer automatisch beter.

1.2. Mijn opdrachten

Aangezien ik meewerkte in het development team werkten we samen aan de applicaties.

Hier werd ik ingezet om features toe te voegen of bugs te fixen in bestaande features. Daardoor kwam ik met zowel de front- als de back-end in contact.

Bij deze features had ik alle verantwoordelijkheid om het op tijd klaar te krijgen en duidelijke communicatie te hebben met de andere developers om de stand van zaken door te geven en eventueel vragen te stellen.

De applicaties waarop ik een toegevoegde waarde heb kunnen leveren, zijn DTOS (DTOS (Digitale transformatie openbare sector) en een module van de SFPP (Semper Fi Productivity Platform) samenwerking. Ik zal deze applicaties hieronder verder beschrijven.

Naast deze opdrachten heb ik de mogelijkheid gekregen om verschillende business meetings bij te wonen. Dit omdat ik aangaf dat de business kant van IT me ook nog wel aansprak.

Uit deze meetings heb ik veel geleerd rondom communicatie met de klanten en demomeetings, enzoverder.

1.2.1. DTOS

Dit project wordt gebruikt om gemeentes te helpen met het digitaliseren van hun systemen.

Het project bestond uit twee grote functionaliteiten. Namelijk het inventariseren van de digitalisatie en een project registratie met opvolging.

Dit was een bestaand project dat al op twee locaties in productie was.

Mijn taken hierin waren het toevoegen van een urenregistratie feature, herwerken van de backend en de front-end.

De urenregistratie zou door de medewerkers van NRDS worden gebruikt om bij te houden hoeveel uren er aan een gemeente besteed zijn of nog besteed moeten worden. Zo weten ze precies hoeveel uren gefactureerd moeten worden.

Toen alles daarrond klaar was ben ik begonnen aan heel wat bugfixes.

1.2.2. SFPP

Dit is de afkorting voor de samenwerking met het bedrijf Semper Fi. De afkorting staat voor Semper Fi - Productivity Platform.

In deze samenwerking maakt NRDS modules aan om de applicatie van Semper Fi uit te breiden. Deze applicatie wordt gebruikt voor de productiviteit in een bedrijf te verbeteren door processen te automatiseren. Bij de projectmanagement module werden bijvoorbeeld de budgetten bijgehouden waardoor er werd aangegeven wanneer een actie onder een project boven het budget ging.

De applicatie ging uiteindelijk gepitcht worden voor een bestek met verschillende gemeentes.

Doorheen mijn stage was er vooral sprake van twee modules: een kennisdatabank en een project management module.

Ik kreeg de kans om aan de project management module te werken.

In dit project was het mijn taak om de delete-functionaliteit zowel in de front- als in de back-end te implementeren. Dit moest echter niet gewoon een delete zijn, de deletes moesten soft deletes zijn die ook cascaderend werkten. Namelijk, als een meerjarenplan werd verwijderd, moest alles onder dat meerjarenplan ook verwijderd worden.

De data die onder een meerjarenplan kon staan waren: meerjarenplannen, beleidsdoelstellingen, kerndoelstellingen, projecten, acties en subacties.

Natuurlijk geldt het ook zo dat als er een kerndoelstelling verwijderd werd dat alles daaronder ook verwijderde en dat het niet alleen zo bij meerjarenplan werkte.

2. Planning

Aangezien we met meerderen aan het project werkten, was het niet altijd mogelijk om een duidelijke planning op voorhand te maken. Ik kreeg namelijk een feature die ik moest implementeren en als die af was wist ik pas wat mijn volgende taak ging zijn.

Voor mijn eerste opdracht (de urenregistratie) werd er gezegd dat ik deze af moest hebben binnen 3 weken. Daardoor vond ik het wel nuttig om hier een planning voor te maken. Dit zodat ik wat structuur had voor de feature en zodat ik op tijd hulp kon vragen als ik achter kwam te liggen of ergens mee vastzat.

Taak beschrijving	Week 1	Week 2	Week 3
Inlezen code			
Back-end functionaliteit maken			
Front-end overzichttabel			
Front-end samenvatting			
Bugfixes			

Tabel 1: Planningsschema

Uiteindelijk was ik hier sneller mee klaar en kon ik vanaf de derde week beginnen met de herwerking van heel de backend van de applicatie.

Vanaf dit punt kreeg ik een nieuwe opdracht als de vorige af was en was er geen vaste planning. Ik moest dus heel flexibel kunnen zijn. De communicatie hierrond gebeurde vooral tijdens de dagelijkse stand-ups die plaatvonden om 1 uur in de middag. Als een opdracht meer duidelijkheid of gewoon meer uitleg nodig had dan werd dit vaak vlak na de stand-up gedaan waardoor de andere developers ook verder konden met hun werk.

3. Onderzoek

De eerste weken dienen normaal om het plan van aanpak te maken en kennis te maken met de werkplek. Bij mij liep dit heel anders.

Op de eerste dag was mijn eerste opdracht nog niet helemaal uitgeschreven. De tijd die ik daardoor kreeg heb ik gebruikt om kennis op te doen voor de opdrachten die ik ging krijgen.

3.1. DTOS

Ik begon met het bekijken van de applicatie om te analyseren welke frameworks ze in de front-end en back-end gebruiken, zodat ik kon bepalen of ik nog nieuwe vaardigheden moest aanleren. Omdat DTOS al bestond, kon ik de basic styling overnemen zonder eerst een prototype in een designtool zoals Figma te maken. Bovendien zou daar, gezien de deadline van drie weken voor de eerste opdracht, geen tijd voor zijn geweest.

3.1.1. Vue

In de front-end werd er gebruik gemaakt van Vue.js. Hier had ik voor mijn stage nog nooit mee gewerkt maar gelukkig al wel van gehoord. Daarom dat ik op de eerste dag van de stage gebruik heb gemaakt van mijn skillshare abonnement om daarmee een crash course in Vue te volgen.

Hierbij heb ik me voornamelijk gericht op het begrijpen van hoe de states werken in Vue. Dit bleek uiteindelijk het enige aspect te zijn dat ik daadwerkelijk nodig had voor de taak die voorhanden was.

De rest van de nodige kennis kon ik halen uit de bestaande code doorheen de applicatie.

Uiteindelijk was het niet zo moeilijk om Vue te gebruiken maar dit kwam natuurlijk grotendeels doordat er al een deel van de applicatie was waardoor ik vaak kon kijken hoe andere dingen gedaan waren.

3.1.2. .NET

In .NET had ik al veel ervaring opgedaan, zowel tijdens lessen op school als door enkele projectjes. Daarnaast maakt .NET gebruik van C# als programmeertaal, waarin ik de meeste ervaring heb dankzij mijn studie in het middelbaar onderwijs en mijn kleine projecten in Unity, dat ook C# gebruikt.

Vervolgens heb ik de code grondig doorgenomen om de gebruikte structuren en patronen te analyseren, zodat ik de volgende dagen direct aan de slag kon. Tijdens deze analyse ontdekte ik dat ze geen gebruik maakten van het Unit of Work-patroon. In plaats daarvan kozen ze voor een eenvoudiger architectuur om de complexiteit te verminderen en een snellere oplevering te bevorderen. Dit betekende dat elke repository zelf verantwoordelijk was voor het beheren van transacties, wat mogelijk minder abstractie bood maar wel de ontwikkelsnelheid ten goede kwam. Hierdoor kon ik beter begrijpen hoe data management en transactionele operaties werden afgehandeld binnen de applicatie, en kon ik mijn aanpak hierop afstemmen.

Vervolgens heb ik de code grondig doorgenomen om de gebruikte structuren en patronen te analyseren, zodat ik de volgende dagen direct aan de slag kon. Tijdens deze analyse ontdekte ik dat ze geen gebruik maakten van het Unit of Work-patroon. In plaats daarvan kozen ze voor een eenvoudiger architectuur om de complexiteit te verminderen en een snellere oplevering te bevorderen. Dit betekende dat elke repository zelf verantwoordelijk was voor het beheren van transacties, wat mogelijk minder abstractie bood maar wel de ontwikkelsnelheid ten goede kwam. Hierdoor kon ik beter begrijpen hoe data management en transactionele operaties werden afgehandeld binnen de applicatie, en kon ik mijn aanpak hierop afstemmen.

3.2. SFPP

Bij de projectmanagement module moest ik soft deletes mogelijk maken die cascerend werkten. Het toevoegen van soft deletes zelf is niet zo moeilijk maar het laten casceren had ik nog niet gedaan. Hierbij werd ook de opdracht gegeven dat dit met Entity Framework gedaan moest worden. Dit door de context te configureren en de entiteiten aan te passen. Door artikels te lezen en het gebruik van AI tools (Chat-GPT en Phind) kwam ik erachter hoe ik hieraan zou moeten beginnen.

3.2.1. Frameworks

In deze applicatie werden dezelfde frameworks gebruikt als bij DTOS dus ben ik gewoon door de code gegaan om te kijken of de structuren ook hetzelfde waren. Dit kwam grotendeels overeen dus ben ik al vrij snel begonnen aan het schrijven van de code voor de functionaliteit.

4. Realisatie

Hieronder volgt alles wat ik op de stage gedaan heb op chronologische volgorde.

4.1. DTOS

Zoals eerder vermeld was het eerste project waar ik op gezet werd DTOS. Dit met als eerste opdracht de toevoeging van een urenregistratie.

4.1.1. Urenregistratie

Voor de urenregistratie ben ik begonnen met de back-end. Voor deze feature moest elke klant een wallet krijgen met de urenregistraties in.

Door deze feature moest de gebruiker door een nieuw submenu naar een dashboard of een overzichtstabel kunnen gaan.

In zowel het dashboard als de tabel moest de gebruiker uren kunnen verhogen of verlagen. Hierdoor werd duidelijk gemaakt hoeveel uren er besteed werden aan een gemeente.

Dus een gebruiker kon uren opladen (uren gewerkt voor de gemeente), of uren crediteren (werkuren reserveren voor de gemeente).

Om dit te kunnen doen heb ik twee models toegevoegd. Dit was HourWallet en HourRegistrations.

Daarna heb ik code geschreven zodat als een gebruiker uren toevoegt (oplading of creditering) er wordt nagegaan of de klant waarvoor de registratie gebeurt al een wallet heeft of niet. Als die wallet nog niet bestaat wordt deze aangemaakt. Als deze al wel een wallet in het systeem heeft wordt de registratie aan deze wallet verbonden.

Daardoor zijn er enkel api calls voor de HourRegistrations en niet voor HourWallet. De front-end moet geen data rechtstreeks van de wallet krijgen.

HourRegistration			^
GET	/api/{customerId}/HourRegistration/all		⌵ 🔒
GET	/api/{customerId}/HourRegistration/{id}		⌵ 🔒
DELETE	/api/{customerId}/HourRegistration/{id}		⌵ 🔒
POST	/api/{customerId}/HourRegistration		⌵ 🔒

Figuur 1: Api endpoints HourRegistrationController

De delete is de enige dat nog niet geïmplementeerd is in de front-end maar dat was nog niet nodig.

Na het maken van de back-end ben ik gestart met de front-end. In het begin was dit lastig omdat ik nu echt voor de eerste keer in Vue werkte. Gelukkig had ik al redelijk wat code om op terug te vallen als er iets niet duidelijk was.

Ik ben begonnen met het maken van de tabel waarin alle registraties zouden staan.

In deze tabel moest de volgende data komen te staan:

- Datum
- Beschrijving
- Uren
- Gebruiker
- Type

De tabel moest gesorteerd worden op datum.
 Boven de tabel moest er een samenvatting komen van de uren.
 Dit ziet u allemaal in onderstaande afbeelding.
 Urenregistratie lijst + samenvatting

Datum	Beschrijving	Uren	Gebruiker	Type
22/04/2024	Testing	11	raf@nrds.be	✓
11/04/2024	Test 009 datum check	4	ruben@nrds.be	✓
29/03/2024	Test 008 Registration	5	ruben@nrds.be	✓
29/03/2024	Tipoff date test 001	2	bernad@nrds.be	✓
26/03/2024	Test 001 Registratie	10	ruben@nrds.be	✓
23/03/2024	Test 002 Registratie	3	ruben@nrds.be	✓
01/01/01	test	4	raf@nrds.be	✓
01/01/01	test delete hours	5	raf@nrds.be	✓
01/01/01	test 25	5	raf@nrds.be	✓
01/01/01	voeding gedaan	8	raf@nrds.be	✓
01/01/01	test	5	ruben@nrds.be	✓

Figuur 2: Urenregistratie tabel

Daarna ben ik begonnen aan een extra opdracht: het ontwikkelen van het dashboard voor de urenregistraties. Dit dashboard moest de volgende functionaliteiten bevatten:

- Een filteroptie waarmee gebruikers de gegevens konden sorteren tussen twee datums.
- Een samenvatting die dezelfde informatie toont als de overzichtstabel.
- Een doughnut chart die het aantal credits en opgeladen uren weergeeft.
- Een overzicht van de laatste 10 registraties.

Voor de front-end ben ik gaan kijken hoe het dashboard van de projecten eruit zag en heb ik deze layout overgenomen.

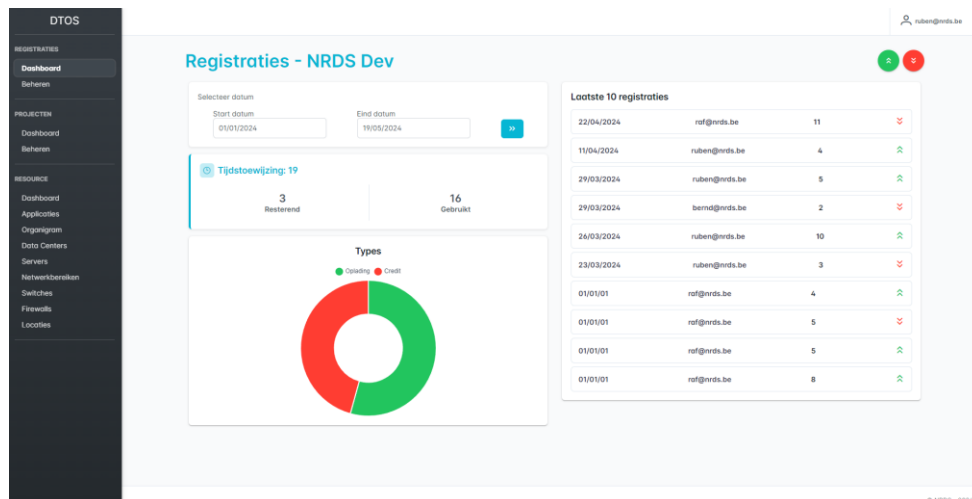
In de back-end is hier een extra API endpoint voor aangemaakt. Deze doet voorlopig hetzelfde als de Get All in de HourRegistrationController (vorige API screenshot).

Als ze later meer data in het dashboard willen toevoegen, kunnen ze dit doen zonder de andere code aan te passen.

GET `/api/{customerId}/Dashboard/all_hourregistrations`

Figuur 3: Api endpoint voor urenregistratie dashboard

De filtering op datums gebeurt in de front-end, zodat er niet elke keer opnieuw een call naar de database moet gaan.



Figuur 4: Dashboard urenregistratie

4.1.2. Refactor Back-end

Bij de eerste code check, kwam mijn stagebegeleider er achter dat de back-end niet optimaal gestructureerd was.

Er was nog geen engine laag aanwezig waardoor heel veel functionaliteit in de controllers zelf zat. Hier is dan eerst een werknemer mee begonnen met dit aan te maken. Toen hij klaar was moest ik mijn urenregistratie updaten zodat het ook gebruik maakte van de nieuwe laag.

Daarna werd er gevraagd om nog extra DTOs (Data transfer objects) aan te maken voor verschillende entiteiten en deze te gebruiken bij verschillende calls. Een voorbeeld hiervan is de "ServerDto" en de "ServerMinimalDto".

Aangezien niet altijd alle data zichtbaar moet zijn is het niet nodig om al de data van de entiteiten op te vragen.

```

7 references
public class ServerDto: DtoBase
{
    0 references
    public string Label { get; set; }
    0 references
    public string IPAddress { get; set; }
    0 references
    public string Cpu { get; set; }
    0 references
    public string Memory { get; set; }
    0 references
    public string DiskSpace { get; set; }
    0 references
    public ServerType ServerType { get; set; }

    0 references
    public Guid NetworkScopeId { get; set; }
    0 references
    public NetworkScopeMinimalDto NetworkScope { get; set; }
    0 references
    public Guid DatacenterId { get; set; }
    0 references
    public DatacenterMinimalDto Datacenter { get; set; }
    0 references
    public string Comment { get; set; }
}

```

Figuur 6: ServerDto

```

2 references
public class ServerMinimalDto: DtoBase
{
    0 references
    public string Label { get; set; }
}

```

Figuur 5: ServerMinimalDto

4.1.3. Refactor Front-end

Nadat de back-end klaar was met de toevoeging van de engine laag, besloten we ook om de front-end ook bij te werken. Ook had hier een werknemer een deel al van gedaan en moest ik vooral focussen op mijn functionaliteit.

Verder heb ik de hele applicatie vertaald. Oorspronkelijk was deze in het Engels ontwikkeld, maar gezien het gebruik ervan in Nederlandstalige gemeentes, werd besloten om de applicatie naar het Nederlands te vertalen.

4.1.4. Bugfixes

Nadat de refactor klaar was, werd ik ingezet om heel de applicatie nog eens te testen en eventuele bugs/errors op te lossen. Dit waren er vrij veel, hierdoor ben ik wel met alle code van heel de applicatie in contact gekomen wat het interessant maakte voor mij.

Een andere bug zat in het urenregistratie gedeelte waarbij de datum op de foute dag opgeslagen werd. Dit kwam door tijdzones. Dit kon ik uiteindelijk oplossen door de npm package “calendar-date” te gebruiken. Dan hebben we de registraties aangepast zodat het DateOnly is in de plaats van DateTime. En dan door het gebruik van calendar-date nemen we de datum van de lokale tijdzone van de gebruiker, zodat deze altijd klopt afhankelijk van de gebruiker.

Doorheen de rest van de applicatie waren er door de refactor een paar attributen waarbij de null-allowed instellingen niet meer correct waren, zowel in de front-end als in de back-end. Dit waren snelle oplossingen eenmaal dat ze gelokaliseerd waren.

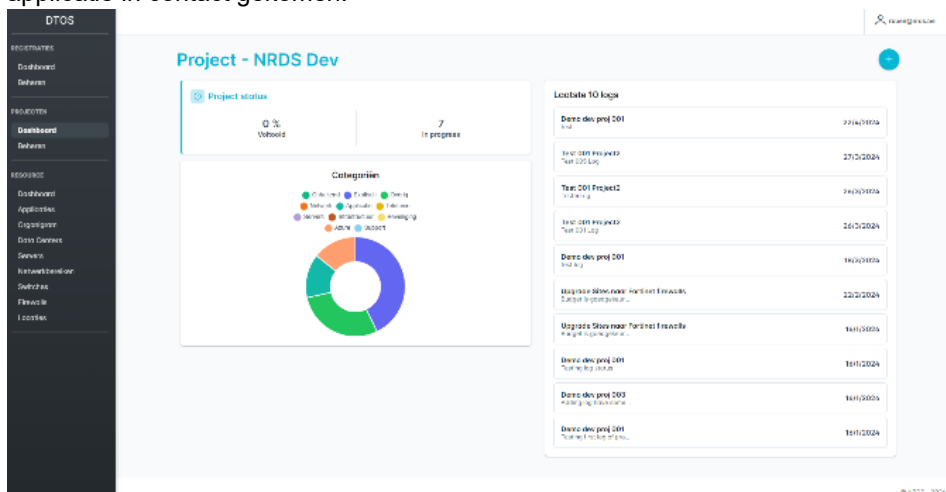
In de front-end waren er nog bepaalde styling bugs. Bijvoorbeeld bij de resource dropdowns dat afhankelijk van de labels de styling wel, of niet toegepast werd.

Aangezien we primevue gebruikte om componenten makkelijk te stylen was dit soms wat uitzoeken hoe deze componentjes werkten. Uiteindelijk was dit vrij simpel te doen door de variabele uit het labelField te halen en deze dan terug te geven als string. Deze konden we dan gebruiken in het component.

Ook heb ik in alle formulieren doorheen de applicatie validatie moeten steken omdat dit nog niet het geval was. Hierdoor wist de gebruiker soms niet waardoor er een item niet werd aangemaakt.

Een ander probleempje was dat de doughnut chart in zowel het project als het urenregistratie dashboard niet automatisch een update deed als er iets aangepast werd. Dit gebeurde pas na een reload. Dit kwam omdat er niet automatisch een nieuwe call gedaan werd gedaan naar de ProjectStore/HourRegistrationStore na het aanmaken/bewerken van een item.

Hieronder zijn wat screenshots van de DTOS applicatie. Buiten het login gedeelte ben ik met de gehele applicatie in contact gekomen.

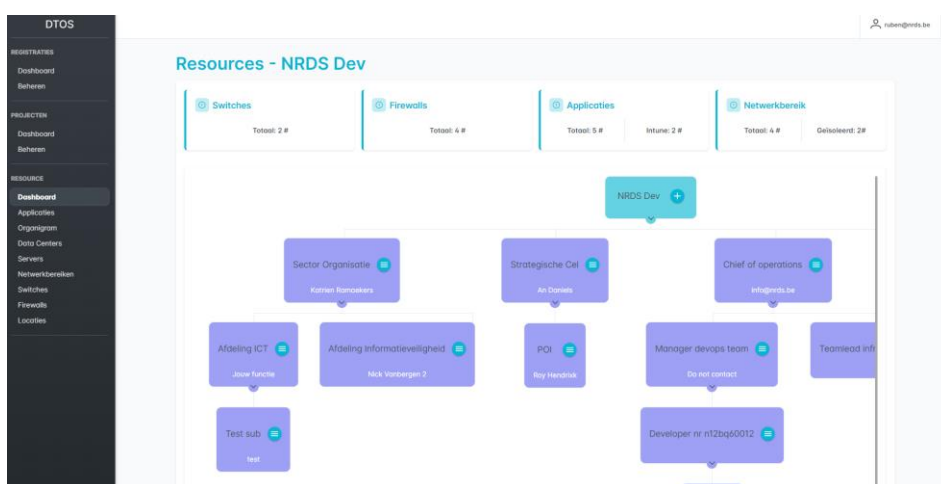


Figuur 7: Project dashboard

Project - NRDS Dev

Label	Beschrijving	Info	Categorie	Status	Acties
Demo dev proj 001	Project for demo purpose	No specific content	Project	In planning	[+][+][+]
Demo dev proj 002	demo info about proj 002	None	Change	On hold	[+][+][+]
Demo dev proj 003	No specific descriptions	None provided	Deploy	On hold	[+][+][+]
Test 001 Project2	TestBeschrijving2	TestInfo002	Applicatie	In planning	[+][+][+]
Test Project	Test Description2	Test Content	Acties	On hold	[+][+][+]
Upgrade Silex naar Fortinet Firewall	Verwijzen van alle info o met veranderingen naar Wayne + Fortinet Silex na Silex verandering	Verwijzen van alle info o met veranderingen naar Wayne + Fortinet Silex na Silex verandering	Project	Project	[+][+][+]
demo dev proj 004	verwijzing van servers	nothing specific	Change	On hold	[+][+][+]

Figuur 8: Project tabel



Figuur 9: Resources organigram

DTOS

REGISTRATES

Dashboard

Beharen

PROJECTEN

Dashboard

Beharen

RESOURCE

Dashboard

Applications

Organigram

Data Centers

Servers

Netwerkbereiken

Switches

Firewalls

Locaties

Search...

Label

Verkoper

SLA

Prijs

Type applicatie

Server

Intune

Acties

Nrds-Testing applicatie 01

Raf

Geen

120000

Overig

Testing server Nrds 01

Test

testverkoper

sla

1000

Onbekend

Server 12 mainframe

Test 003 Application

Raf

TestSLA

120000

Onbekend

Testing server Nrds 02

Raf

geen

120000

Onbekend

Testing server Nrds 01

Testing server Nrds 03

Raf

geen

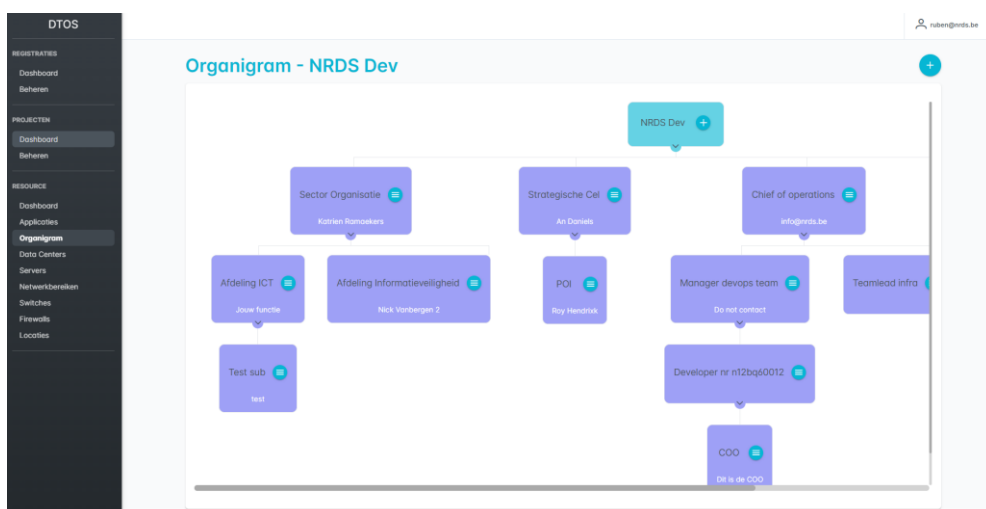
12000

Onbekend

nrds@nrds.be

© 140715 - 2024

Figuur 10: Applicaties tabel



DTOS

REGISTRATES

Dashboard

Beheer

PROJECTEN

Dashboard

Beheer

RESOURCE

Dashboard

Applicaties

Organogram

Data Centers

Servers

Netwerkbereiken

Switches

Firewalls

Locaties

ruben@nrds.be

Servers - NRDS Dev

Q Search...

Label ↓	IP adres	Processor	Geheugen	Opslagruimte	Server Type	Netwerkbereik	Datacenter	Opmerking	Acties
Server 12 mainframe	127.0.0.1	Veel	32 gb	12t	Exotisch	Vlan kerkstraat	Datacenter kerkstraat		<div><div></div><div></div></div>
Syntegro Server	10.202.34.57	4 cores	16 gb	100 gb	Onbekend	vlan technische dienst	Cipal Datacenter		<div><div></div><div></div></div>
Test server	10.133.13.12	Scpu	600	1000	Infrastructuur	vlan dienst vrije tijd	Securitas Datacenter	test	<div><div></div><div></div></div>
Testing server Nrds 01	127.0.1.125	I-5	32gb	2Tb	Infrastructuur	Vlan kerkstraat	Testing Datacenter after migration bundle 001	geen	<div><div></div><div></div></div>

Figuur 13: Servers tabel

DTOS

REGISTRATES

Dashboard

Beheer

PROJECTEN

Dashboard

Beheer

RESOURCE

Dashboard

Applicaties

Organogram

Data Centers

Netwerkbereiken

Switches

Firewalls

Locaties

ruben@nrds.be

Netwerkbereiken - NRDS Dev

Q Search...

VLAN	IP-bereik	Geïsoleerd	Acties
vlan technische dienst	10.133.15/24	<div></div>	<div><div></div><div></div></div>
vlan dienst vrije tijd	10.133.13/24	<div></div>	<div><div></div><div></div></div>
Vlan binnentuin	255.255.255.100 - 255.255.255.255	<div></div>	<div><div></div><div></div></div>
Vlan kerkstraat	0.0.0.0 to 255.255.255.255	<div></div>	<div><div></div><div></div></div>

Figuur 14: Netwerkbereiken tabel

DTOS

REGISTRATES

Dashboard

Beheer

PROJECTEN

Dashboard

Beheer

RESOURCE

Dashboard

Applicaties

Organogram

Data Centers

Servers

Netwerkbereiken

Switches

Firewalls

Locaties

ruben@nrds.be

Switches - NRDS Dev

Q Search...

Label ↓	IP adres	Merk	URL	Netwerkbereik	Locatie	Acties
Core switch 1	10.133.13.1	Aruba	https://10.133.13.1/manage	vlan technische dienst	Kerkstraat Datacenter	<div><div></div><div></div></div>
Core switch 2	10.133.13.5	Aruba	http://10.133.13.5	vlan dienst vrije tijd	De Ploek	<div><div></div><div></div></div>

Figuur 15: Switches tabel

DTOS

ruben@nrds.be

REGISTRATIES

Dashboard

Behenen

PROJECTEN

Dashboard

Behenen

RESOURCE

Dashboard

Applicaties

Organigram

Data Centers

Servers

Netwerkberreken









Switches

Firewalls

Locaties

Firewalls - NRDS Dev

Q Search...

Label ↓	IP adres	Locatie	Netwerkbereik	Merk	Url	Acties
Firewall dienst burgerzaken	127.0.0.1	Kierstraat Datacenter	Vlan kerkerstraat	Norton	https://www.google.com	 
Fortigate 100F	10.133.13.2	Kierstraat Datacenter	vlan technische dienst	Fortinet	https://10.133.13.2	 
Test FW	10.133.15.5	Technische dienst	Vlan binrentuin	Fortigate	http://10.133.15.5	 
Test firewall	127.0.0.1	Kierstraat Datacenter	Vlan binrentuin	geen	www.google.be	 

Figuur 16: Firewalls tabel

DTOS

ruben@nrds.be

REGISTRATIES

Dashboard

Behenen

PROJECTEN

Dashboard

Behenen

RESOURCE

Dashboard

Applicaties

Organigram

Data Centers

Servers

Netwerkberreken







Switches

Firewalls

Locaties

Locaties - NRDS Dev

Q Search...

Label ↓	Adres	Acties
De Plak		 
Gemeentehuis	Dorpstraat 14	 
Technische dienst		 

Figuur 17: Locaties tabel

4.2. Projectmanagementmodule SFPP

In dit project kreeg ik de opdracht om soft deletes toe te voegen. Hoewel ik eerst dacht dat dit niet zo moeilijk ging zijn, bleek dit toch een grotere uitdaging te zijn dan verwacht.

4.2.1. Soft delete functionaliteit

Met deze opdracht ben ik het langste bezig geweest. Dit omdat het veel uitzoeken en testen was. In het begin werd mij gevraagd om de soft deletes te doen via Entity Framework, dit omdat de soft deletes cascading moesten hebben. De bedoeling was dus om Entity Framework Deleted state over te nemen en dan de entiteit niet te verwijderen maar de variabelen rondom soft delete aanpassen waardoor het soft deleted wordt. Toen ik hier een werkende oplossing voor had was dit toch niet precies wat ze zochten. Dit kwam omdat de oplossing vrij omslachtig was en als er een nieuwe entiteit bij zou komen moesten we er best veel code voor schrijven.

Daarna moest ik het mogelijk zien te maken via de repositories.

In deze foto ziet u hoe de delete wordt vervangen door de aanpassing van de attributen IsDeleted en SoftDeletedAt. De CascadeSoftDeleteChildren(entity), diende voor de cascading ervan te doen maar dit is was dus uiteindelijk niet wat ze zochten.

```
public new virtual async Task<int> SaveChangesAsync(CancellationToken cancellationToken = default)
{
    FillBaseFields();

    foreach (var entry in ChangeTracker.Entries())
    {
        if (entry.Entity is SoftDeleteCascadeEntityBase entity)
        {
            switch (entry.State)
            {
                case EntityState.Deleted:
                    entry.State = EntityState.Modified;
                    entity.IsDeleted = true;
                    entity.SoftDeletedAt = DateTime.UtcNow;
                    //CascadeSoftDeleteChildren(entity);
                    break;
                default:
                    break;
            }
        }
    }

    return await base.SaveChangesAsync(cancellationToken);
}
```

Figuur 18: EntityState.Delete tegenhouden

Natuurlijk moest er door de soft deletes ook een standaard filter staan op de get methodes zodat deze de verwijderde items niet mee gaf met de queries.

En aangezien er later misschien de optie moet komen voor een admin om wel de verwijderde items te zien, een methode gemaakt die deze filter negeert.

```
base.OnModelCreating(modelBuilder);
modelBuilder.Entity<ActionPoint>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<KeyObjective>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<MultiYearPlan>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<PolicyObjective>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<Project>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<SubAction>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<ProjectExpenditure>().HasQueryFilter(m => !m.IsDeleted);
modelBuilder.Entity<User>().HasQueryFilter(m => !m.IsDeleted);
```

```
public IQueryable<TEntity> GetIncludingDeleted<TEntity>() where TEntity : EntityBase
{
    return Set<TEntity>().IgnoreQueryFilters();
}
```

Toen ik het via de repositories moest doen ben ik begonnen om het eerst te testen met twee entiteiten. Toen dit werkte heb ik de functies die ik bij deze entiteiten had generiek gemaakt en doorgetrokken naar de andere entiteiten.

In de onderstaande afbeeldingen ziet u al de generieke functies.

```
public virtual async Task SoftDelete(Guid id, string tenantId)
{
    var entity = await GetById(id, tenantId);

    if (entity != null)
    {
        await SetSoftDelete(entity);
        await softDeleteCascadeRepository.SoftDeleteChildren(entity.Id, entity.TenantId);
    }

    await _context.SaveChangesAsync();
}
```

Figuur 19: Functie SoftDelete

```
public virtual async Task SetSoftDelete(ISoftDeleteCascadeEntity entity)
{
    entity.IsDeleted = true;
    entity.SoftDeletedAt = DateTime.UtcNow;
}
```

Figuur 20: Functie SetSoftDelete

```
public virtual async Task SoftDeleteChildren(Guid parentId, string tenantId)
{
    var query :IQueryable<ISoftDeleteCascadeEntity>? = GetQueryAbleForParentCascade(parentId, tenantId);

    if (query == null)
    {
        return;
    }

    var entitiesToSoftDelete :List<ISoftDeleteCascadeEntity> = await query.ToListAsync();

    foreach (var entityToSoftDelete in entitiesToSoftDelete)
    {
        await SetSoftDelete(entityToSoftDelete);
        await softDeleteCascadeRepository.SoftDeleteChildren(entityToSoftDelete.Id, entityToSoftDelete.TenantId);
    }
}
```

Figuur 21: Functie SoftDeleteChildren

Als laatste hebben we deze functie die in elke repository een body krijgt.

```
public abstract IQueryable<ISoftDeleteCascadeEntity>? GetQueryAbleForParentCascade(Guid parentId, string tenantId);
```

Figuur 22: Functie GetQueryAbleForParentcascade

Dit is een voorbeeld van een body, in dit geval is het de body van de policyObjectiveRepository.

```
public override IQueryable<PolicyObjective>? GetQueryAbleForParentCascade(Guid parentId, string tenantId)
{
    return Table.Where(x :PolicyObjective => x.MultiYearPlanId == parentId && x.TenantId == tenantId);
}
```

Figuur 23: Voorbeeld van GetQueryAbleForParentcascade

4.2.2. Soft delete testen

Toen de functionaliteit erin zat, kreeg ik de opdracht om hiervoor testen te schrijven. Aangezien de soft delete cascading ook getest moest worden was dit ook weer niet zo eenvoudig als dat ik gedacht had. Dit omdat er database functies opgeroepen werden in de functies.

Één enkele entiteit testen ging gemakkelijk met Moq. Dit is een populair mocking framework voor .NET dat wordt gebruikt bij het schrijven van unit tests. Moq stelt ontwikkelaars in staat om objecten te maken die het gedrag van echte objecten nabootsen (mocks), zodat je de afhankelijkheden van de te testen entiteit kunt isoleren. Dit is vooral nuttig om te controleren hoe je code omgaat met verschillende scenario's zonder dat je echte afhankelijkheden hoeft aan te roepen, zoals databases of externe services. Moq maakt het eenvoudig om deze mock-objecten op te zetten, verwachtingen te definiëren en het gedrag van de afhankelijkheden te verifiëren.

Door de entiteiten te mocken en dan de SetSoftDelete functie te callen waren de testen al operationeel voor één entiteit. Maar vanaf dat de cascading getest moest worden was dit niet meer zo eenvoudig. Uiteindelijk hebben we er eentje met Moq gedaan om de rest toch met een andere technologie te doen. Dit was omdat ze op mijn stageplaats zoveel mogelijk verschillende dingen wilden laten zien voor zover het mogelijk was binnen de projecten.

Deze andere technologie was InMemory testing.

Deze techniek gebruikt een in-memory versie van een database in de plaats van de echte database.

```
[Fact]
public async Task SetSoftDelete_Should_Set_Entity_IsDeleted_To_True()
{
    //Arrange
    var dbContextMock = new Mock<IAppDbContext>();
    var keyObjectiveRepositoryMock = new Mock<IKeyObjectiveRepository>();

    var policyObjectiveMock = new Mock<PolicyObjective>();
    var repository = new PolicyObjectiveRepository(dbContextMock.Object, keyObjectiveRepositoryMock.Object);

    //Act
    await repository.SetSoftDelete(policyObjectiveMock.Object);

    //Assert
    Assert.True(policyObjectiveMock.Object.IsDeleted);
    Assert.NotNull(policyObjectiveMock.Object.SoftDeletedAt);
}
```

Figuur 24: SetSoftDelete test

```
[Fact]
public async Task GetQueryAbleForParentCascade_Should_Return_List_Of_PolicyObjectives()
{
    var multiYearPlanId = new Guid();
    // Arrange
    var data = new List<PolicyObjective>
    {
        new PolicyObjective { MultiYearPlanId = multiYearPlanId, TenantId = "tenant1", Title = null, Description = null, ExtraInformation = null, Id = default },
        new PolicyObjective { MultiYearPlanId = multiYearPlanId, TenantId = "tenant1", Title = null, Description = null, ExtraInformation = null, Id = default },
        new PolicyObjective { MultiYearPlanId = Guid.NewGuid(), TenantId = "tenant2", Title = null, Description = null, ExtraInformation = null, Id = default },
    }.AsQueryable();

    var mockSet = new Mock<DbSet<PolicyObjective>>();
    mockSet.As<IQueryable<PolicyObjective>>().Setup(expression => mockSet.Provider).Returns(data.Provider);
    mockSet.As<IQueryable<PolicyObjective>>().Setup(expression => mockSet.Expression).Returns(data.Expression);
    mockSet.As<IQueryable<PolicyObjective>>().Setup(expression => mockSet.ElementType).Returns(data.ElementType);
    mockSet.As<IQueryable<PolicyObjective>>().Setup(expression => mockSet.GetEnumerator()).Returns(new Enumerator(data.GetEnumerator()));

    var mockContext = new Mock<AppDbContext>();
    mockContext.Setup(expression => mockContext.Set<PolicyObjective>()).Returns(mockSet.Object);

    var keyObjectiveRepoMock = new Mock<IKeyObjectiveRepository>();

    var repository = new PolicyObjectiveRepository(mockContext.Object, keyObjectiveRepoMock.Object);
    // Act
    var result = repository.GetQueryAbleForParentCascade(multiYearPlanId, "tenant1");

    // Assert
    Assert.Equal(2, result.Count()); // Assuming you expect 2 items to match the criteria
}
```

Figuur 25: GetQueryAbleForParentCascade test

```
[Fact]
public async Task SoftDelete_Should_Set_IsDeleted_To_True()
{
    // Arrange
    var options = new DbContextOptionsBuilder<AppDbContext>()
        .UseInMemoryDatabase(databaseName: "PolicyObjectiveDb")
        .Options;

    var policyObjective = new PolicyObjective
    {
        Title = "Title", Description = "", ExtraInformation = "", TenantId = "tenant 1", Id = Guid.NewGuid(), MultiYearPlanId = Guid.NewGuid()
    };

    using (var context = new AppDbContext(options))
    {
        context.PolicyObjectives.Add(policyObjective);
        await context.SaveChangesAsync();

        var policyObjectiveRepository = new PolicyObjectiveRepository(context, new Mock<IKeyObjectiveRepository>().Object);
        // Act
        await policyObjectiveRepository.SoftDelete(policyObjective.Id, policyObjective.TenantId);
    }

    // Assert
    Assert.True(policyObjective.IsDeleted);
}
```

Figuur 26: SoftDelete test

4.2.3. Soft delete implementatie

Toen de testen volledig werkten ben ik begonnen met de implementatie van de delete functionaliteit in de front-end. Dit hield in dat ik bij alle componenten die verwijderd konden worden, een knop kwam te staan om het te verwijderen. Als er op deze knop gedrukt werd moest er een dialoog venster openen waarbij er een bevestiging gegeven moest worden. Pas als er hier bevestigd werd moest de entiteit verwijderd worden, en een redirect gedaan worden naar het parent element. Als dit element geen parent had, bijvoorbeeld een meerjarenplan. Dan werd er een redirect gedaan naar de lijst met alle meerjarenplannen.

Hieronder zie je de code om te delete in de front-end beschikbaar te maken.

Enkele van deze componenten zaten al in de applicatie waaronder de baseDelete, delete en remove functies. Deze heb ik voor de zekerheid maar toegevoegd als screenshot aangezien dit allemaal samenhangt.

In de bijschriften staat soms Modelx. Dit staat in dit geval voor PolicyObjective. Maar deze files zijn er ook voor MultiYearPlan, KeyObjective, Project, ...

```

async baseDelete(id: string): Promise<ApiResponseModel<string>> {
  const url = `${this.BuildUrlForController()}/${id}`;
  const response = await this.ApiClient.delete<string>(url)
    .then((response) => {
      return this.CreateResponseModel<string>(response, 200);
    })
    .catch((error) => {
      return this.CreateFailedResponseModel<string>(error);
    });
  return response;
}

```

Figuur 27: baseDelete functie in BaseApiService

```

async delete(id: string): Promise<ApiResponseModel<string>> {
  const response = super.baseDelete(id);
  return response;
}

```

Figuur 28: delete functie in ModelxApiService

```

async function remove(model: PolicyObjectiveModel): Promise<void> {
  const result = await PolicyObjectiveApiService.delete(model.id);
  if (!result.succes) {
    toast.error(`Beleidsdoelstelling: ${model.title} kan niet verwijderd worden.`);
  } else {
    toast.success(`Beleidsdoelstelling: ${model.title} werd succesvol verwijderd!`);
  }
}

```

Figuur 29: remove functie in ModelxStore

```

const handleDeletePolicyObjectiveClicked = () => {
  deletePolicyObjectiveDialog.value = true;
};

const handleDeletePolicyObjectiveDialogResponse = async () => {
  deletePolicyObjectiveDialog.value = false;
  if (policyObjectiveDto.value != undefined){
    await policyObjectiveStore.remove(policyObjectiveDto.value);
    if (router) {
      router.push(`/meerjarenplan/` + policyObjectiveDto.value.multiYearPlanId);
    }
  }
};

```

Figuur 30: removehandlers in ModelxComponent.vue

```

const canDelete = function (): boolean {
  return userStore.hasRolePermission([RoleDefinition.Beheerder, RoleDefinition.Superadmin]);
}

```

Figuur 31: canDelete functie in ModelxComponent.vue

```

<Button v-if="canDelete()" v-tooltip.bottom="Verwijderen" icon="pi pi-trash" rounded outlined class="text-red-500 hover:text-white hover:bg-red-500" @click="handleDeletePolicyObjectiveClicked()" />
<Dialog v-model:visible="deletePolicyObjectiveDialog" :style="{ width: '450px' }" header="Meerjarenplan verwijderen" :modal="true">
  <div class="flex align-items-center justify-content-center">
    <span>Ben je zeker dat je de huidige doelstelling wilt verwijderen?</span>
  </div>
  <template #footer>
    <Button label="Annuleren" icon="pi pi-times" severity="danger" @click="deletePolicyObjectiveDialog = false" />
    <Button label="Bevestigen" icon="pi pi-check" @click="handleDeletePolicyObjectiveDialogResponse" />
  </template>
</Dialog>

```

Figuur 32: Button en dialoog venster in ModelxComponent.vue

5. Conclusie

In deze stage heb ik aanzienlijke groei doorgemaakt, met name op het gebied van testing. Bij DTOS heb ik met plezier aan een volledig nieuwe feature binnen een bestaande applicatie gewerkt, gevolgd door een herwerking van de gehele applicatie. Dit bood mij waardevolle ervaring in het upgraden van verouderde software. Het implementeren van soft deletes in de projectmanagementmodule leek aanvankelijk eenvoudig, maar door de toevoeging van cascading werd het al snel complexer dan verwacht.

Daarnaast heb ik de kans gehad om veel business meetings bij te wonen, wat mijn begrip van de zakelijke aspecten van een softwarebedrijf heeft vergroot. Hierdoor heb ik geleerd over effectieve communicatie, het geven van demo's en het ontvangen van feedback.

Hoewel ik aanvankelijk vond dat ik veel tijd besteedde aan de softdeletions, hebben mijn collega's me gerustgesteld door te benadrukken dat dit een normaal en complex proces is. Over het geheel genomen is de stage verlopen zoals ik had gehoopt, en heb ik waardevolle ervaring opgedaan die mijn ontwikkeling als softwareontwikkelaar ten goede zal komen.