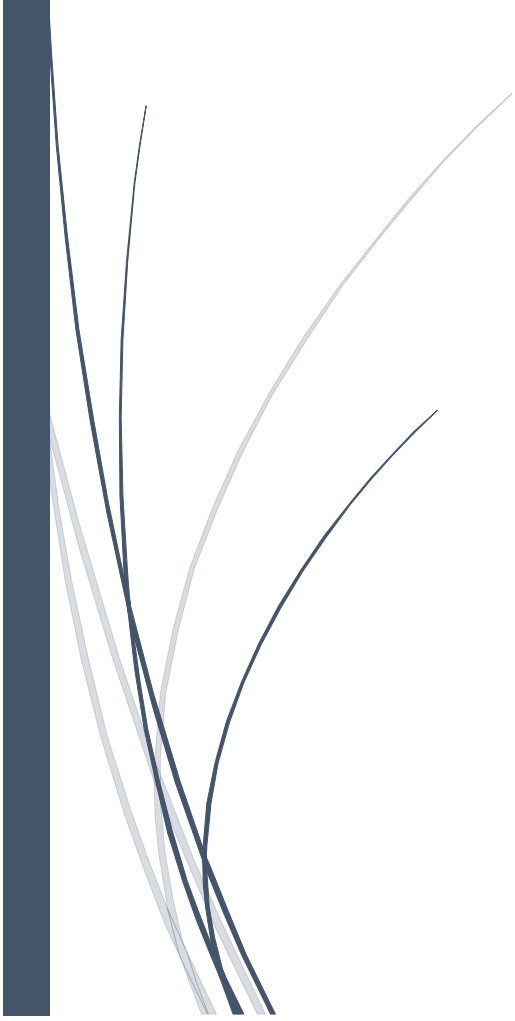
15-6-2017

Technisch Ontwerp

Omgevingswet



Database Factory 5

Informatieblad

ISE-Project Omgevingswet
Namens de Hogeschool van Arnhem en Nijmegen uitgevoerd voor CGI Nederland.

Het project wordt uitgevoerd door Database Factory 5.

Projectleden:

- Michiel Bos 561485
- Ricardo van Burik 569129
- Johan Heij 584299
- Duncan Luiten 553272
- Tristan de Roo 563431

Begeleiders:

- Tim de Goede
- Pim Haenen

Datum: 19 mei 2017
Locatie: Ruitenberglaan 31, 6826 CC Arnhem
Versie 2.0

Versiebeheer

Versie	Datum	Aanpassingen
1.0		
1.1	8 juni 2017	● Kleine aanpassingen Non Functional Requirements. ● Extra informatie Website toegevoegd.
1.2	13 juni 2017	● Indexen en Lost Updates toegevoegd. ● Nieuw hoofdstuk Integrations toegevoegd.
1.3	14 juni 2017	● Kleine aanpassingen PDM op basis van wijzigingen in feitenbeschrijvingen.
2.0	15 juni 2017	● Stored procedures, triggers en check constraints beschrijvingen aangepast.

Inhoudsopgave

1. Inleiding

Dit document geeft een beeld van de technische aspecten van dit project. Het laat een overzicht zien van de gemaakte technische keuzes, de gekozen frameworks en de gekozen programmeertalen.

In het hoofdstuk ‘Netwerkarchitectuur’ wordt beschreven welke componenten via het netwerk moeten communiceren, welke database is gekozen en waar de server is gelokaliseerd.

Vervolgens worden de gekozen front-end en back-end frameworks beschreven en wordt toegelicht hoe die gebruikt gaan worden. Ook wordt er uitgelegd hoe deze frameworks werken en waarom wij de keuze gemaakt hebben om ze te gebruiken.

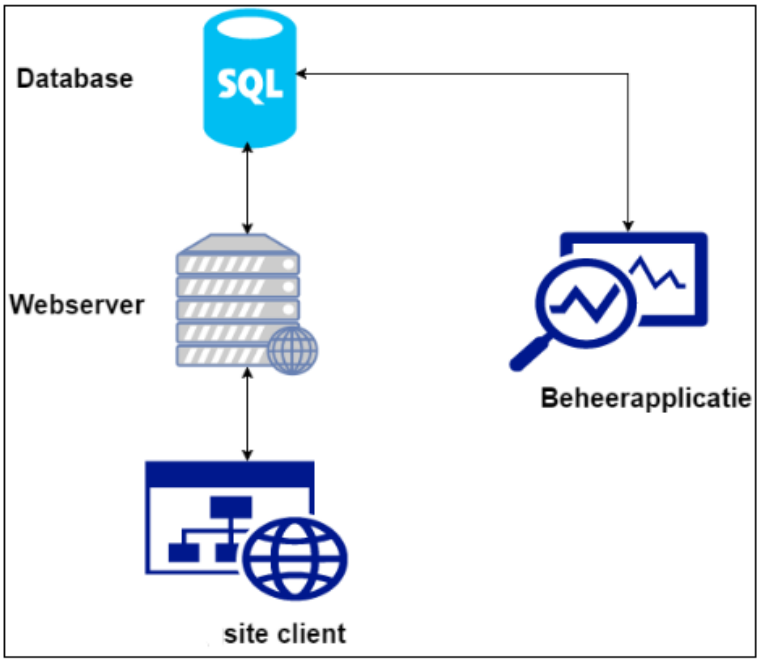
Daarna wordt toegelicht hoe de kwaliteit van de data in de database wordt gewaarborgd, hoe het PDM is ontstaan en hoe de webapplicatie gaat worden beveiligd. Ook worden de indexen toegelicht die gebruikt zijn om de performance te handhaven. Vervolgens wordt een overzicht weergegeven van alle triggers, stored procedures en checks die geïmplementeerd zijn om de constraints af te vangen.

Tot slot worden nog enkele niet-functionele requirements behandeld.

2. Netwerkarchitectuur

Dit project heeft vier componenten die via het netwerk moeten communiceren:

- Microsoft SQL Server
    - De database die gebruikt wordt heet Microsoft SQL Server. Hierin wordt alle informatie opgeslagen. De database communiceert via de webserver met de website, en direct met de beheerapplicatie.
- Website
    - De website draait op de webserver. De website communiceert alleen met de webserver. Als de website iets wilt opslaan in de database, wordt dit via de webserver gecommuniceerd.
- Beheerapplicatie
    - Met de beheerapplicatie wordt alle data in de database beheert. Alle data die hier word bewerkt, wordt direct naar de database gecommuniceerd.
- Webserver
    - De webserver is het middelpunt van de communicatie tussen de website en de database. De website die op de webserver staat, is zichtbaar met behulp van het internet.



Figuur 1, Netwerkarchitectuur

3. Programmeertalen

3.1. Beheerapplicatie

Voor de beheerapplicatie is uiteindelijk de keuze gemaakt voor C#. Als eerst is MS Access gevallen. Hoewel MS Access goed is om een demo/prototype applicatie te maken, is C# beter voor dit project omdat de opdrachtgever de intentie heeft om het project uit te breiden. Dit komt omdat C# meer functionaliteiten tot zijn beschikking heeft.

3.2. Website

- De website wordt opgebouwd met HTML en CSS. Om hier niet te veel tijd aan te besteden is er gekozen om te zoeken naar een goede template die sterk samenhangt met Bootstrap. Zo kan er meer aandacht worden besteed aan de functionaliteiten van de webapplicatie, in plaats van de opmaak. Naast het besparen van tijd hebben we Bootstrap gekozen om de volgende redenen:
    - Alle pagina's hebben een consistent ontwerp.
    - Ervaring binnen de groep.
    - Makkelijk te integreren.
    - Responsive.
- Om de webapplicatie dynamisch te maken, wordt er verbinding gemaakt met een MSSQL Database. Deze database zal overigens ook in verbinding staan met de beheerapplicatie.
- Data wordt met behulp van het PHP Framework Laravel uit de database gehaald. Ook maakt Laravel het makkelijk om verschillende onderdelen van een site te hergebruiken. Verdere punten die mee hebben gespeeld in het besluit zijn:
    - Goede meegeleverde documentatie.
    - Veel online learning sessions (Laracast, Pluralsight).
    - Veelgebruikte bestanden aanmaken via Artisan commands (Models, Controllers).
    - Blade Templates, hiermee kan er overzichtelijk gemaakte Laravel functies in de html worden verwerkt.
    - Makkelijke Dependency Injection.
    - Automated Authentication.

3.3. Database

De keuze is op Microsoft SQL Server gevallen, omdat alle projectleden hier de meeste ervaring mee hebben. Vanuit school is MSSQL aanbevolen en wordt ook het meest ondersteund. Vergeleken met een database zoals MySQL heeft MSSQL het voordeel dat *CHECKS, STORED PROCEDURES en FUNCTIONS* uitgeschreven kunnen worden. Voor de web- en beheerapplicatie, waar zich veel van deze constraints in bevinden, is dit dus ideaal.

4. Front-End Framework

In dit hoofdstuk worden de keuzes met betrekking tot het front-end framework toegelicht voor zowel de beheerapplicatie als de webapplicatie.

4.1. Beheerapplicatie

Voor de beheerapplicatie is de keuze gemaakt om het Graphical User Interface (GUI) framework Windows Presentation Foundation (WPF) te gebruiken. WPF is een grafisch subsysteem dat deel uitmaakt van het DotNet framework versie 3 of hoger van Microsoft. De opmaaktaal waarin gewerkt wordt heet Extensible Application Markup Language (XAML). XAML is een mix van de opmaaktalen HTML, CSS en XML. Wat in de XAML code wordt gemaakt, wordt real time bijgewerkt in het designer scherm dat, in Visual Studio althans, standaard boven de XAML code staat.

WPF wordt gebruikt omdat:

- Het een modernere GUI oplevert dan de nog veelgebruikte standaard front-end frameworks voor C# (winforms).
- Met WPF een programmeur op een hoger niveau aanpassingen kan maken aan een grafisch element dan met een winforms framework. Dit betekent dat de sub-elementen van een grafisch element kunnen worden aangepast.
- Een programmeur met het grid systeem op gemakkelijke wijze een consistent formulier kan maken.
- Een programmeur kan met een resource dictionary file genaamd ‘app.xaml’ alle stijlen aanpassen die voor de hele applicatie gelden. De programmeur kan in de app.xaml bijvoorbeeld aangeven dat:
    - Alle knoppen blauw moeten zijn.
    - Alle knoppen in de applicatie 80 pixels hoog en 80 pixels breed moeten zijn.

4.2. Website

Het front-end framework dat wordt gebruikt voor de webapplicatie is Bootstrap. Zoals eerder aangegeven heeft Bootstrap een aantal voordelen die erg toepasselijk zijn op het project Omgevingswet. De webapplicatie gaat gebruikt worden door bewoners van een gemeente. Het moet dus het ‘visitekaartje’ worden van de gemeente. Met het gebruik van Bootstrap kan er snel een goed uitziende basis worden neergezet, waar vervolgens de benodigde functionaliteiten in ingebouwd kunnen worden.
Ondanks dat de huidige applicatie eerst gebruikt gaat worden als demo, is het goed om na te denken over de toekomst. Wij denken dat met het gebruik van Bootstrap de ‘overdracht’ van het project naar een uiteindelijk development team van CGI makkelijker wordt. Bootstrap wordt namelijk wereldwijd veel gebruikt (Arsenault, 2017).

5. Back-End Framework

In dit hoofdstuk worden de keuzes met betrekking tot het back-end framework toegelicht voor zowel de beheerapplicatie als de webapplicatie.

5.1. Beheerapplicatie

Aangezien voor de beheerapplicatie het front-end framework WPF is gekozen, is het gebruik van het back-end framework DotNet verplicht. WPF heeft functionaliteiten van DotNet nodig om goed te kunnen functioneren. Dit zijn voornamelijk functionaliteiten die het afhandelen van data betreffen.

DotNet wordt gebruikt omdat:

- WPF gebruik maakt van DotNet om te functioneren.
- Het DotNet framework helpt bij:
    - Het ophalen van data uit de database.
    - Het afhandelen van data die is opgehaald uit de database.
    - Het verbinden van de data aan een datagrid.

5.2. Website

Het PHP framework Laravel wordt gebruikt als back-end framework voor de webapplicatie. Om de functionaliteiten binnen de applicatie te kunnen gebruiken, moeten gebruikers een account aanmaken. Dit is een cruciaal onderdeel van de applicatie. Echter is het inloggen van een gebruiker niet aan bod gekomen tijdens het eerste gesprek. Alhoewel inloggen ‘vanzelfsprekend’ is, kan het realiseren van de functionaliteit veel tijd kosten. Door gebruik te maken van Laravel kan er al snel automatisch een basic authentication gemaakt worden. Dit was niet de voornaamste reden om Laravel te gebruiken. Na het inventariseren van de benodigde functionaliteiten konden we concluderen dat de website kan worden gebouwd met behulp van PHP. Echter, om onze eigen kennis te verbreden, hebben we gekozen voor Laravel. Hierbij kan er worden vastgesteld dat naarmate het project vordert, we meer profijt van Laravel krijgen. Het framework maakt bijvoorbeeld het gebruik van queries makkelijker, aangezien het “denken” op de achtergrond van Laravel plaatsvindt.

De connectie met MSSQL binnen Laravel is ook gemakkelijk te leggen, aangezien de basis al ingebouwd is. Enkel servernaam, databasenaam, gebruikersnaam en wachtwoord moeten worden ingevuld, Laravel doet de rest.

5.3. Integrations

Binnen de webapplicatie worden verschillende integrations gebruikt. Door gebruik te maken van integrations is er tijd bespaard aangezien functionaliteiten niet opnieuw gemaakt hoeven te worden. In onderstaande tabel is een opsomming te vinden waarin ook elke integration kort wordt toegelicht.

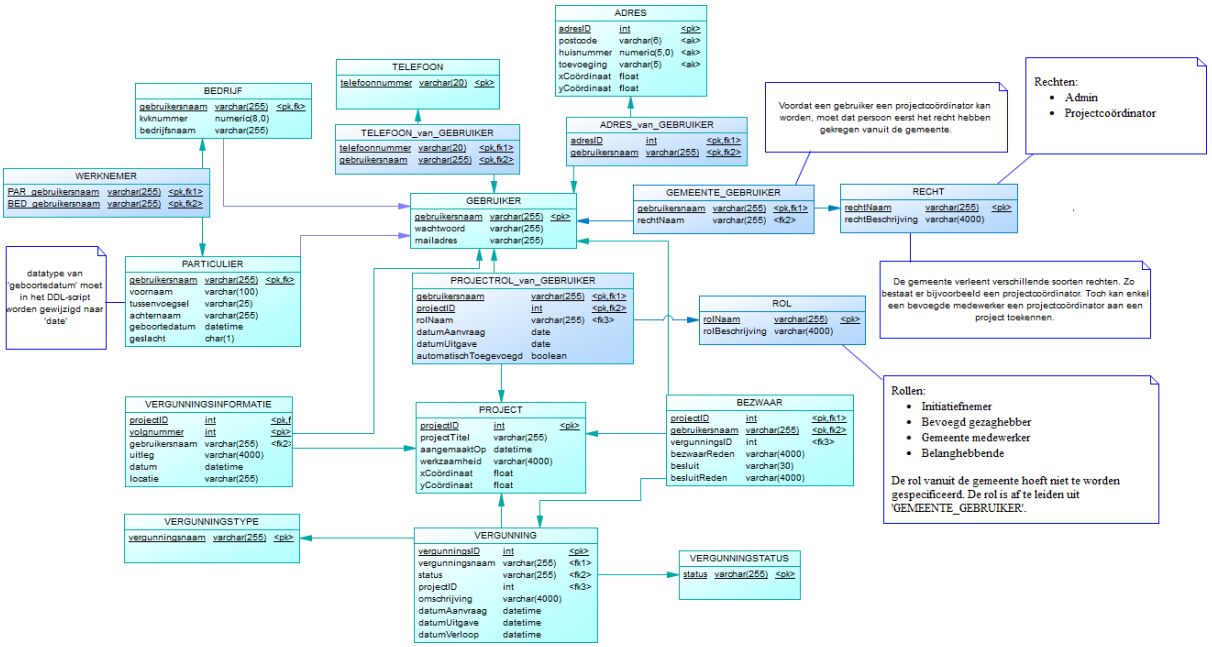
Integration	***Uitleg***	***Referentie***
GooglMapper	Alle functionaliteiten van Google Maps, binnen Laravel. Binnen de applicatie wordt GooglMapper gebruikt om kaarten met markers weer te geven.	(bradcornford, 2017)
Geocoder	Geocoder wordt gebruikt bij het aanmaken van een nieuw user, waarbij de gebruiker zijn locatiegegevens in moet vullen. Het zet gegevens om in coördinaten.	(geocoder-php, 2017)
OpenLayers	Met een klik op de map, kunnen de geo coördinaten worden opgehaald en opgeslagen in de database. OpenLayers wordt gebruikt bij het aanmaken van een nieuw project, waarbij de gebruiker een locatie moet selecteren.	(OpenLayers, 2017)

6. Database

De database is misschien wel het belangrijkste product dat opgeleverd gaat worden. Het is in elk geval het meest belaste product. Dat wil zeggen dat er veel gebruikers tegelijkertijd indirect gebruik van maken, via de webapplicatie en de beheerapplicatie. Daarnaast is veiligheid een belangrijk en actueel thema, en is datakwaliteit ook van belang.

6.1. Ondernomen stappen tot PDM

Het PDM is automatisch gegenereerd vanuit het CDM.



Figuur 2, Physical Data Model

6.1.1. Toelichting PDM

In onderstaande tabel worden de verschillende entiteiten, die aanwezig zijn in het PDM, uitgelegd.

Entiteit	**Uitleg**
Adresgegevens	In de entiteit worden alle gegevens welke betrekking hebben tot een adres opgeslagen. De postcode samen met het huisnummer en de toevoeging zijn de alternative key. Deze drie kolommen zijn uniek genoeg om één unieke locatie op een kaart te krijgen. Daarnaast worden de geo coördinaten opgeslagen.
Adres van gebruiker	Elke gebruiker woont op een adres. De tabel ‘Adres van gebruiker’ is een koppeltabel, aangezien meerdere gebruikers op hetzelfde adres kunnen wonen.

Bedrijf	Een gebruiker kan een bedrijf zijn. Bedrijven moeten namelijk ook invloed uit kunnen oefenen op hun leefomgeving.
Bezwaar	Belanghebbenden van een project kunnen bezwaar maken. Een bezwaar wordt gebruikt om aan de gemeente te laten weten dat mensen in de omgeving van het project het niet eens zijn met de gestelde plannen.
Gebruiker	Een persoon die de applicatie wil gebruiken moet zich registreren. Er wordt dan een gebruiker binnen het systeem aangemaakt.
Gemeente gebruiker	Een gebruiker kan ook werknemer zijn van de gemeente. De gebruiker zou eventueel speciale functies kunnen uitvoeren. Middels deze tabel kan er worden vastgelegd welke werknemer bij de gemeente werkt.
Particulier	Een gebruiker kan bestaan uit een particulier of een bedrijf. De gebruiker van de applicatie logt altijd in als particulier met een wachtwoord en een mailadres.
Project	Alle basisinformatie met betrekking tot een project wordt opgeslagen in de tabel Project.
Projectrol van gebruiker	Elke gebruiker kan een rol hebben binnen een project. Zo is de aanvrager van een project altijd een initiatiefnemer. Zijn buurman zou dan een belanghebbende kunnen zijn.
Recht	De gemeente verleent verschillende soorten rechten. Zo bestaat er bijvoorbeeld een projectcoördinator. Enkel een bevoegde medewerker kan een projectcoördinator aan een project toekennen.
Rol	De verschillende rollen welke een gebruiker kan hebben.
Telefoon	Telefoon is de hoofdtabel waarin alle telefoonnummers van particulieren en bedrijven worden opgeslagen.
Telefoon van gebruiker	Elke gebruiker is verplicht een telefoonnummer op te geven. Echter is de tabel Telefoon van gebruiker een koppeltabel, omdat meerdere gebruikers hetzelfde telefoonnummer kunnen hebben en een gebruiker ook meerdere nummers

	kan bezitten.
Vergunning	Aan een project kunnen verschillende vergunningen gekoppeld worden. Zo kan er makkelijk worden gespecificeerd wat er precies nodig is om een project te laten voltooien.
Vergunningsinformatie	Alle informatie die wordt aangeleverd bij een project of vergunning wordt opgeslagen in de vergunningsinformatie. Denk hierbij aan documenten met informatie en links naar relevante webpagina's.
Vergunningstatus	Elke vergunning heeft een status wanneer deze wordt aangevraagd. De status kan gedurende het project wijzigen. De verschillende statusmogelijkheden zijn opgeslagen in de tabel 'Vergunningstatus'.
Vergunningstype	Vergunningen hebben verschillende types. Zo is er bijvoorbeeld een evenementvergunning en een monumentvergunning. De verschillende vergunningstypen worden opgeslagen in deze tabel.
Werknemer	Werknemer is een koppeltabel tussen bedrijf en particulier. Elke gebruiker binnen een bedrijf kan handelingen namens het bedrijf ondernemen. Belangrijk hierbij is dat het altijd mogelijk is om na te gaan welke werknemer een handeling namens het bedrijf heeft gedaan.

6.2. Datakwaliteit

Het beschermen van de datakwaliteit wordt gedaan door middel van constraints. Het gaat hierbij om basisfunctionaliteiten zoals primary keys, alternative keys en foreign keys, maar ook om check constraints, triggers, stored procedures en functions. Voor een aantal gemaakte constraints wordt uitgelegd of hierbij problemen kunnen optreden met betrekking tot concurrency en lost updates.

6.2.1. Lost updates

Lost updates vinden plaats wanneer twee gebruikers min of meer tegelijkertijd een update uitvoeren op dezelfde rij. Hierdoor kunnen wijzigingen van de eerste danwel tweede gebruiker verloren gaan. Dit is een onwenselijke situatie, mede omdat niet alle gebruikers dubbelchecken of de zojuist door hun ingevoerde informatie enkele (tientallen) seconden na het opslaan plotseling is gewijzigd. Om deze reden wordt er gecontroleerd waar lost updates kunnen plaatsvinden, en hoe deze verholpen kunnen worden.

Spoc_Set_ProjectCoordinator
Hieronder ziet u een voorbeeld opgenomen in een tabel hoe een lost update wordt voorkomen. U moet zich voorstellen dat de statements in connectie één en twee in deze volgorde ook uitgevoerd worden en allebei ongeveer tegelijkertijd begonnen zijn.

De lost update die wordt voorkomen zit in stap drie. Deze wordt voorkomen omdat de oude projectcoördinator die meegegeven wordt in allebei de connecties nog hetzelfde zijn. Maar wanneer in connectie één de projectcoördinator gewijzigd wordt in de database, komt deze niet meer overeen met de oude projectcoördinator in connectie twee. Waardoor de check in connectie twee false returned en een error geeft. Hierdoor wordt niets overschreven wat in connectie één door wat in connectie twee wordt meegegeven.

Stap	Connectie 1	Connectie 2
1	IF EXISTS (SELECT * FROM GEMEENTE_GEBRUIKER WHERE GEBRUIKERSNAAM IN (SELECT GEBRUIKERSNAAM FROM PROJECTROL_VAN_GEBRUIKER WHERE UPPER(ROLNAAM) = 'GEMEENTE') AND GEBRUIKERSNAAM = ‘OudeGebruiker1’) AND ‘OudeGebruiker1’ != ‘NieuweGebruiker1’ ***Returned True***	
2	UPDATE PROJECTROL_VAN_GEBRUIKER SET GEBRUIKERSNAAM = ‘NieuweGebruiker1’ WHERE Projectid = 1 AND GEBRUIKERSNAAM = ‘OudeGebruiker1’ ***Update database regel***	
3		IF EXISTS (SELECT * FROM GEMEENTE_GEBRUIKER WHERE GEBRUIKERSNAAM IN (SELECT GEBRUIKERSNAAM FROM PROJECTROL_VAN_GEBRUIKER WHERE UPPER(ROLNAAM) = 'GEMEENTE') AND GEBRUIKERSNAAM = ‘OudeGebruiker1’) AND ‘OudeGebruiker1’ != ‘NieuweGebruiker2’ ***Returned false***
4		ELSE RAISERROR('Het toevoegen of aanpassen van een projectcoördinator is mislukt.',16,1) ***Raised error en gaat naar Catch***
5	***Commit succes***	
6		***Rollback***

6.3. Beveiliging

De beveiliging van de database wordt gerealiseerd op twee manieren. Aan de ene kant is er de beveiliging tegen doelbewust hacken (zoals SQL injectie). Aan de andere kant kunnen gebruikers verschillende permissies hebben. Als deze permissies niet goed gekoppeld worden aan de gebruikers, dan kan er mogelijk data gelekt worden.
Een oplossing, om gebruikersdata toch enigszins veilig te stellen, is om wachtwoorden te hashen. Het hashen van wachtwoorden wordt opgevangen in de front-end van de applicatie. *Bcrypt* wordt standaard geleverd met Laravel Authentication. Het voordeel van bcrypt is dat de developer zelf kan aangeven hoe sterk een wachtwoord gehashed moet worden. Ook wordt *salt* gebruikt. Dit zorgt ervoor dat de kans dat het wachtwoord gekraakt zal worden minimaal is.

6.4. Indexen

Het leggen van indexen op de kolommen van een tabel kan de performance versnellen van bepaalde SELECT-queries. Hierdoor wordt de druk op de database verlicht, waardoor de performance van de beheer- en webapplicatie erop vooruit gaat. Vanwege tijdgebrek in dit project worden niet alle SELECT-queries behandeld.

De query die hier behandeld wordt is van de beheerapplicatie.

```
SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY p.projectid) AS RowIndex,
p.PROJECTID, CONVERT(VARCHAR(10),p.AANGEMAAKTOP, 103) aangemaaktop, p.WERKZAAMHEID, p.PROJECTTITEL,
g.GEBRUIKERSNAAM, g.MAILADRES, pa.VOORNAAM, pa.TUSSENVOEGSEL, pa.ACHTERNAAM
FROM project p
INNER JOIN (SELECT PROJECTID, GEBRUIKERSNAAM, rolnaam FROM PROJECTROL_VAN_GEBRUIKER) PvG ON p.PROJECTID = pvg.PROJECTID AND UPPER(pvg.rolnaam) = 'INITIATIEFNEMER'
INNER JOIN (SELECT GEBRUIKERSNAAM, MAILADRES FROM GEBRUIKER) g on pvg.GEBRUIKERSNAAM = g.GEBRUIKERSNAAM
INNER JOIN (SELECT VOORNAAM, TUSSENVOEGSEL, ACHTERNAAM, GEBRUIKERSNAAM FROM PARTICULIER) pa on g.GEBRUIKERSNAAM = pa.GEBRUIKERSNAAM
WHERE LOWER(pa.voornaam) like  '%%'
AND LOWER(pa.achternaam) like  '%%'
AND LOWER(g.gebruikersnaam) like  '%%'
AND LOWER(g.mailadres) like '%%') sub
WHERE sub.RowIndex > 0 AND sub.RowIndex <= 50
```

Figuur 3, Select query

De query hierboven is zeer complex en de performance van de query zou erop vooruit kunnen gaan. Door de volgende Non-Clustered index te leggen gaat de performance erop vooruit:

```
CREATE NONCLUSTERED INDEX NCIX_Project_MetGebruikerRol
on dbo.projectrol_van_Gebruiker(projectid,rolnaam)
```

Figuur 4, Non-clustered index query

Door het leggen van deze index wordt het volgende resultaat behaald:

	Trial 7	Trial 6	Trial 5	Trial 4	Trial 3	Trial 2	Trial 1	Average
Bytes sent from client	2082 →	2082 →	2082 →	2082 →	2082 →	2082 →	2082 →	2082.0000
Bytes received from server	52301 →	52301 ↓	61769 →	61769 →	61769 →	61769 ↑	52301 →	57711.2900

Figuur 5, Resultaten

Trials 2,3,4 en 5 zijn de trials ***ZONDER*** de Non-Clustered index.
Trials 1,6 en 7 zijn de trials ***MET*** de Non-Clustered index.
Zoals hierboven te zien is, gaat de ‘bytes received from servers’ met ca. 9.500 omlaag. Dit is een verlaging van ongeveer 16%.

6.5. Stored Procedures

In onderstaande tabel worden de Stored Procedures weergegeven, welke functie ze hebben en waar ze in het programma worden gebruikt.

Naam	**Functie**	**Gebruik**
spInsertUser	Het inserten van een user in de database. Het adres en het telefoonnummer wordt direct gecontroleerd of deze al bestaan. Als deze nog niet bestaan worden ze eerst toegevoegd, waarna ze als Foreign Key aan de gebruiker worden gekoppeld.	Webapplicatie registratieformulier.
spAddProject	Het toevoegen van een project. Hierin wordt de datum en tijd op het moment van opslaan automatisch ingevuld. Verder zal de x-coördinaat en y-coördinaat worden meegegeven, net zoals de gebruiker die het project toevoegt.	Webapplicatie. De Stored Procedure wordt aangeroepen op het moment dat de gebruiker een locatie heeft aangewezen en een omschrijving heeft toegevoegd.
spAlterPhonenumber	Het wijzigen van een telefoonnummer door alle niet-numerieke symbolen te verwijderen van de invoer.	Hij hoort uitgevoerd te worden in de webapplicatie bij het registreren van een gebruiker. Maar dit is niet geïmplementeerd.
spAdd_License	Slaat het toevoegen van een vergunning op in de database.	Wordt gebruikt in de beheerapplicatie op de opslaan knop wanneer er een vergunning is toegevoegd.
spAddBevoegdGezag	Slaat het toevoegen van een bevoegd gezaghebber op in de database.	Wordt gebruikt in de beheerapplicatie op de opslaan knop wanneer er een bevoegd gezaghebber is toegevoegd.
spAddUserToProject	Voegt een rol toe aan een project.	Webapplicatie, wordt gebruikt bij het abonneren en project toevoegen.
spMakeObjection	Voegt een bezwaar toe aan een vergunning onder een project. Wordt een check	Webapplicatie, bij het maken van een bezwaar.

	uitgevoerd dat een gebruiker geen bezwaar maakt op zijn eigen project.	
Spoc_Set_ProjectCoordinator	Voegt een projectcoördinator toe als er geen onder een project staat, update de projectcoördinator als er een projectcoördinator bestaat onder een project. Voert ook verschillende checks uit of de opgegeven gebruiker niet gelijk staat aan de oude gebruiker en of de gebruikersnaam als gemeente rol ‘Coördinator’ staat geregistreerd.	Beheerapplicatie, wordt gebruikt bij het toevoegen/wijzigen van een bevoegd gezaghebber.
spPermitDecision	Update een vergunning regel wanneer het behandeld wordt. Bij het goedkeuren van een vergunning wordt de datumverloop en datumuitgave als waarde opgegeven, als de vergunning is afgekeurd worden deze als null meegegeven.	Beheerapplicatie bij het behandelen van een vergunning onder de ‘Opslaan’ knop.

6.6. Triggers

In onderstaande tabel worden de Triggers weergegeven, welke functie ze hebben en waar ze in het programma worden gebruikt.

Naam	**Functie**	**Gebruik**
trgBezwaarMakenVanaf18	Een gebruiker mag wel een account hebben maar mag niet bezwaar maken op een project als zijn/haar leeftijd onder de 18 ligt.	Webapplicatie, maken van bezwaar op een project.
trgAutomatischAbonnerenAdres	Abonneert een gebruiker automatisch wanneer er adresgegevens toegevoegd worden. Automatisch toegewezen abonnementen worden ook automatisch verwijderd wanneer het adres verwijderd wordt.	Bij registratie via website of het handmatig toekennen van een nieuw adres.
trgAbonnementBijwerkenAdresgegevens	Abonneert de gebruiker automatisch wanneer zijn/haar adresgegevens gewijzigd worden. Verwijdert ook automatisch toegewezen abonnementen die niet meer in zijn/haar leefomgeving liggen.	Bij het aanpassen van de adresgegevens van een gebruiker.
trgAutomatischAbonnerenProject	Abonneert de gebruikers in de omgeving wanneer een project toegevoegd of bijgewerkt wordt. Omdat er geen abonnee-acties verricht worden tijdens het bijwerken van een project, worden er **geen** huidige abonnees verwijderd.	Bij het aanmaken van een project (handmatig of via website).

6.7. Check constraints

In onderstaande tabel worden de Check constraints weergegeven, welke functie ze hebben en waar ze in het programma worden gebruikt.

Naam	**Functie**	**Gebruik**
CKC_KVKNUMMER_BEDRIJF	Het KVK nummer van een bedrijf mag niet langer dan 8 karakters lang zijn.	Webapplicatie, registratieformulier.
CKC_GEBRUIKERSNAAM_GEBRUIKE2	De gebruikersnaam moet tenminste 4 characters lang zijn.	Webapplicatie registratieformulier.
CKC_GESLACHT_GEBRUIKE	De gebruiker kan een man, vrouw of onbekend zijn.	Webapplicatie registratieformulier.
CKC_TELEFOONNUMMER_GEBRUIKE	Het telefoonnummer moet tenminste 8 characters lang zijn.	Webapplicatie registratieformulier.
CKC_STATUS_VERGUNNI2	De vergunning moet verschillende statussen kunnen hebben, namelijk: Aangevraagd, Afgewezen, Uitgegeven, Verlopen, Bezwaar.	Webapplicatie aanvragen project, bekijken projecten. Beheerapplicatie.

7. Opsomming niet-functionele requirements

De requirements die niet kunnen worden onderverdeeld in een use case, staan in de tabel hieronder beschreven, volgens de ISO-25010 standaard. De requirements die onder de 'functional suitability' vallen zijn terug te vinden in de functionele requirements in het functioneel ontwerp. De onderdelen 'Reliability' en 'Performance efficiency' zijn niet opgenomen omdat er aan deze onderdelen geen eisen waren gesteld.

Code	Omschrijving
Usability	
	Een ingelogde gebruiker moet met één muisklik zijn projectenoverzicht kunnen bekijken.
Compatibility	
	De web- en beheerapplicatie moeten via dezelfde database kunnen communiceren.
Security	
	Persoonlijke gegevens van gebruikers (met uitzondering van gebruikersnaam, voornaam, tussenvoegsel en achternaam), mogen alleen inzichtelijk zijn voor de gebruikers zelf, het systeembeheer en de bevoegde gezagen.
Maintainability	
	De geschreven triggers en stored procedures (met uitzondering van de procedures die niet in de productiedatabase opgenomen moeten worden) moeten voorzien zijn van testscripts.
Portability	
	De webapplicatie moet ondersteund worden op de platformen Windows (7 en hoger), Linux (in ieder geval Ubuntu 16.04), Android (2.6 en hoger) en iOS (10.0 en hoger).
	De beheerapplicatie moet draaien op Windows 7 en hoger.
	De webapplicatie moet te gebruiken zijn vanuit de browsers Mozilla Firefox (53.0 en hoger), Google Chrome (58 en hoger), Microsoft Edge (15 en hoger) en Safari (10.1 en hoger).

Literatuurlijst

Arsenault, C. (6 juni 2017). *Top 10 Front-End Frameworks of 2016.* Geraadpleegd op 18 mei 2017, van https://www.keycdn.com/blog/front-end-frameworks/#Front-End-Frameworks-Statistics

bradcornford. (16 maart 2017). *An easy way to integrate Google Maps with Laravel.* Geraadpleegd op 13 juni 2017, van https://github.com/bradcornford/Googlmapper

geocoder-php. (17 april, 2017) *Geocoder for Laravel.* Geraadpleegd op 13 juni 2017, van https://github.com/geocoder-php/GeocoderLaravel

OpenLayers. (z.d.) *A high-performance, feature-packed library for all your mapping needs.* Geraadpleegd op 13 juni 2017, van https://openlayers.org/