Dataopslag Vermissingen in Sarea

Ontwerp

10.2.e.

Versie 0.8
1 december 2020

Gegevens

Afstudeerder: 10.2.e.
Opleiding: Hanzehogeschool, HBO-ICT, Software Engineering
Studentnummer: 10.2.e.

Afstudeerbedrijf: Innovatiehuis Politie Noord-Nederland
Digital Society Hub - Zernikepark 10 –
Groningen
10.2.g @politie.nl

Bedrijf begeleider: 10.2.e.
10.2.e. @sarea-samenzoeken.nl
06 10.2.e.

Docenten: 10.2.e.
10.2.e. @pl.hanze.nl
06- 10.2.e.

10.2.e.

10.2.e.
10.2.e. @pl.hanze.nl

Inhoudsopgave

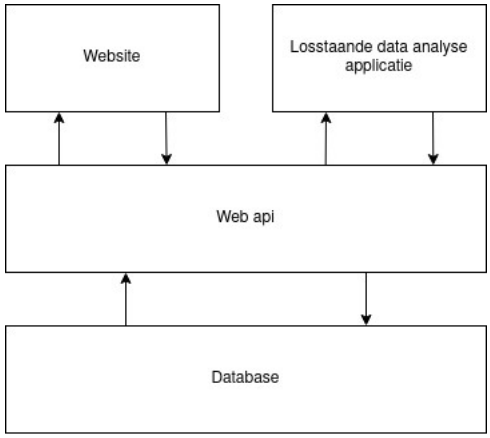
1 Inleiding

In dit document wordt het ontwerp van de vermissing opslag beschreven. Dit ontwerp dient inzicht te geven in de gemaakte ontwerpbeslissingen vanuit de gebruikskant en technisch perspectief. Hierbij zullen de API en gebruikersinterface worden beschreven, de gekozen technologieën en database structuur. De beveiliging zal apart worden besproken. Als laatst zal er worden beschreven hoe de uiteindelijke oplossing in productie kan worden gezet.

2 Context

De service bestaat uit drie hoofdzakelijke componenten. Elk component heeft alleen een directe afhankelijkheid van de laag hieronder. Deze drie lagen zijn de website, web API en database. In deze structuur kan een bovenliggende laag vervangen worden zonder dat de onderliggende laag iets hoeft te veranderen (mits gestelde eisen niet veranderen). Omdat de web API niet afhankelijk is van de gebruiker kan deze ook door andere applicaties worden gebruikt, zoals het algoritme dat door 10.2.g wordt ontwikkeld.

Figuur 1: GDV communicatie tussen onderdelen

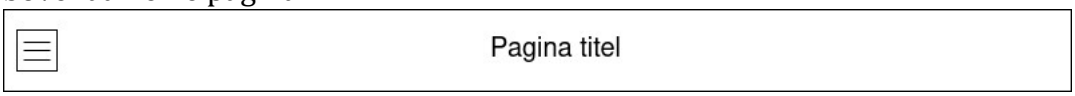


3 Gebruikersinterface

In dit hoofdstuk wordt de gebruikersinterface omschreven. Dit wordt gedaan aan de hand van mockups van de verschillende onderdelen. Componenten die op meerdere pagina's te zien zijn zullen apart worden besproken. Deze zullen daarom niet terug te zien zijn bij andere componenten. Het uiterlijk is geïnspireerd op Google zijn Material Design. Material design is terug te zien in veel Google apps zoals YouTube.

3.1 Boven balk

In figuur 2 is de boven balk van de website te zien. Deze balk is aanwezig bovenaan elke pagina.

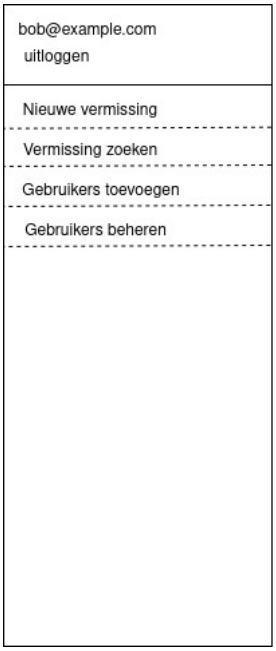


Figuur 2: Site header

In deze balk zijn slechts twee dingen te zien. Als eerst een menu knop linksboven. Deze knop dient om het zij menu te openen. Ten tweede een paginatitel. Deze titel veranderd afhankelijk van de pagina waarop de gebruiker zich bevindt.

3.2 Menu

In figuur 3 is het menu te zien. Dit menu wordt geopend met de knop van de boven balk die te zien is in figuur 2. Dit menu is op elke pagina toegankelijk.

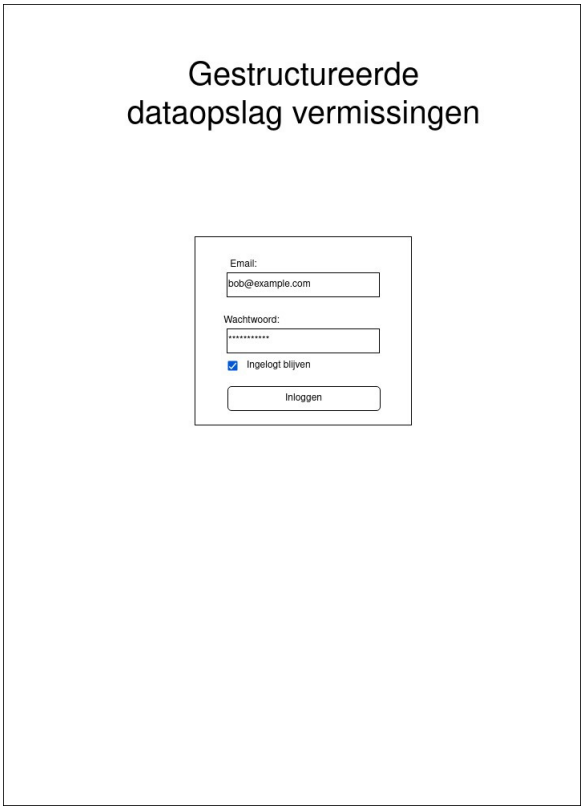


Figuur 3: Menu

Bovenaan in het menu staat welk account is ingelogd en een knop om uit te kunnen loggen. Wanneer de gebruiker niet is ingelogd zal er alleen een inlog knop staan. Onder de gebruiker staan menu knoppen om een nieuwe vermissing aan te maken, een bestaande vermissing te vinden en gebruiksaccounts toe te voegen en of aan te passen. Welke knoppen zichtbaar zijn hangt af van de rechten van de gebruiker.

3.3 Startpagina

De start pagina biedt de gebruiker de mogelijkheid om in te loggen. Wanneer de gebruiker is ingelogd laat deze pagina dezelfde navigatie opties zien als het menu.

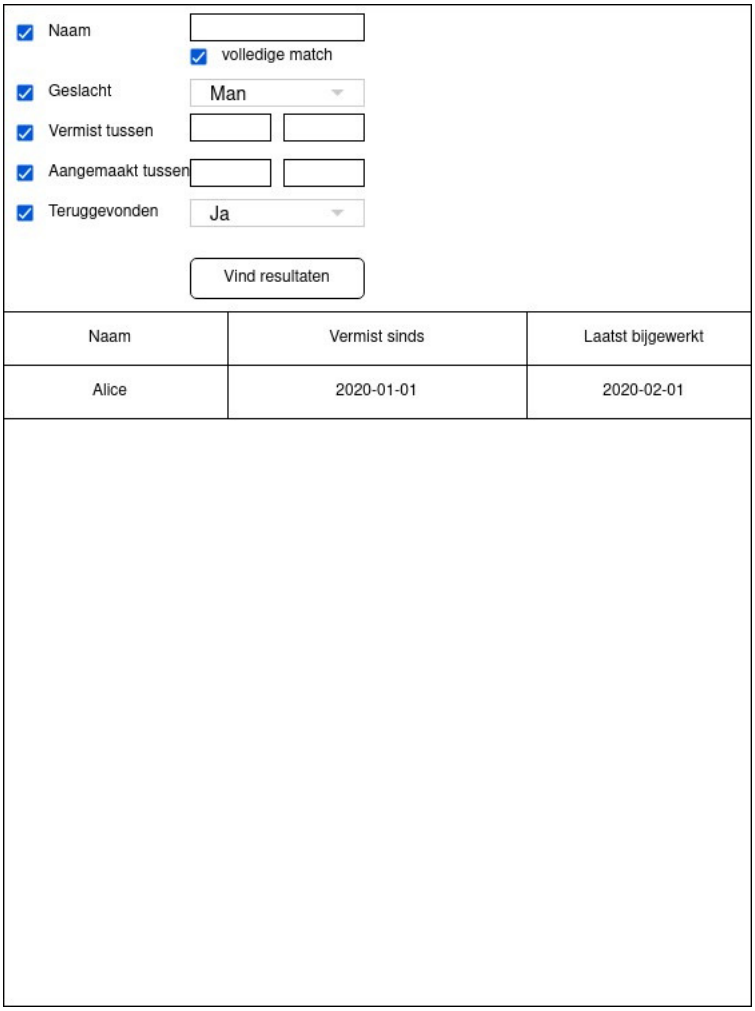


Figuur 4: Start pagina

3.4 Vermissing vinden

Er is een aparte pagina voor het terugvinden van bestaande vermissingszaken. Hier kunnen verschillende eigenschappen worden ingevuld om vermissingen terug te vinden. Elk van deze eigenschappen kan aan en uit worden gezet met behulp van de aanliggende checkboxen.

Onder de zoek parameters zijn de resultaten te zien. Elk van de rijen is een link naar de desbetreffende vermissingscasus. Om de bedoelde vermissing te herkennen zijn er een aantal extra eigenschappen te zien.

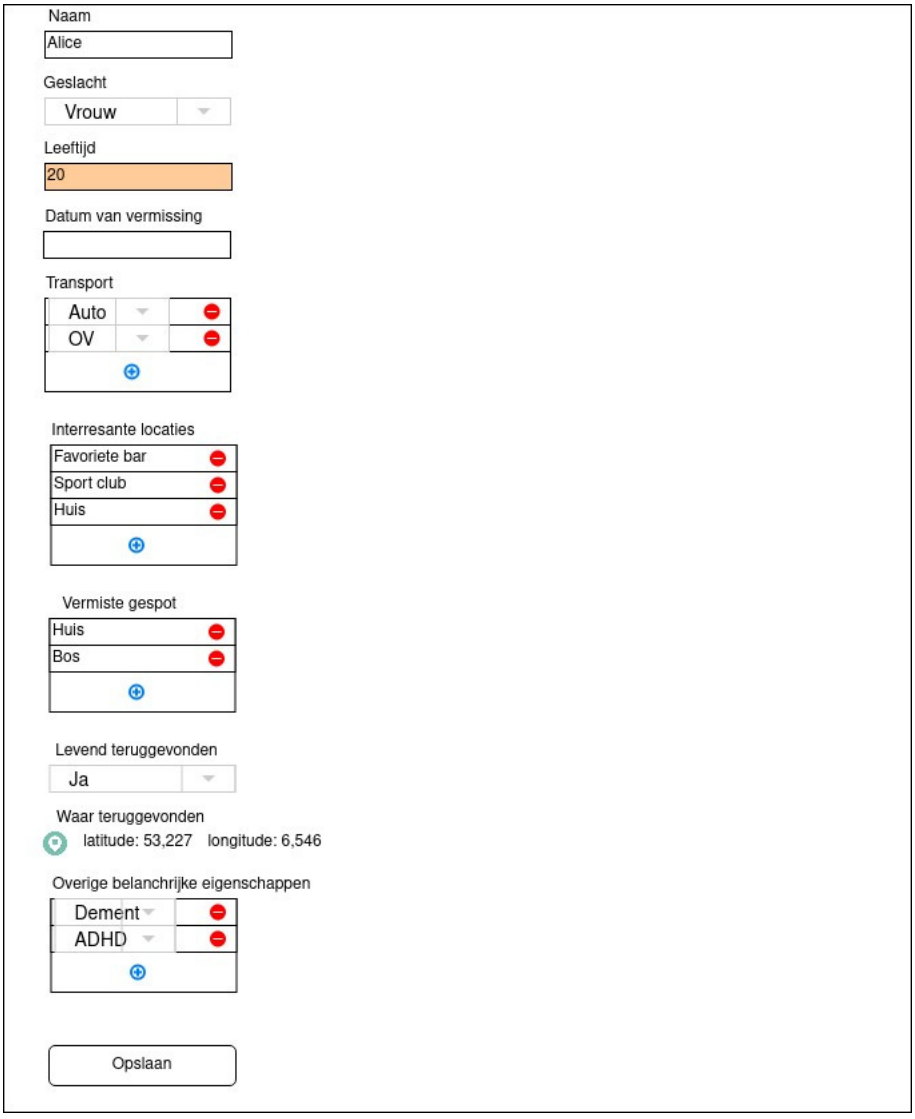


Figuur 5: Vermissingen Terugvinden

3.5 Vermissing overzicht

De vermissing pagina wordt gebruikt voor het aanmaken van nieuwe vermissingen en het bewerken van bestaande. Hierin zijn alle eigenschap groepen te vinden. Eigenschappen zoals de locaties waar de vermiste is gespot zijn kort samengevat met behulp van steekwoorden. Om de volledige informatie hierover te vinden moet erop geklikt worden.
Als de gebruiker een wijziging maakt aan een bestaande vermissing dan wordt het veld gekleurd. Dit dient de gebruiker bewust te maken van de veranderingen. Dit moet helpen onbedoelde wijzigingen te verminderen.

Als de gebruiker tevreden is met mogelijke wijzigingen kan er op de opslaan knop worden gedrukt.



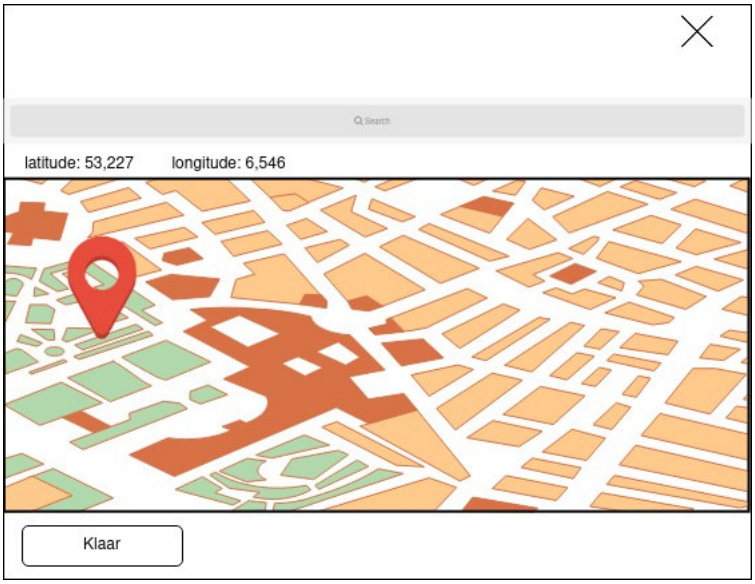
Figuur 6: Vermissing bewerken

Er zijn invoermogelijkheden voor de volgende onderdelen.

- Naam van de vermiste
- Geslacht van de vermiste
- Leeftijd van de vermiste

- Datum van vermissing
- Points of interrest van de vermiste
- Locaties waarop de vermiste is gezien
- Is de vermiste levend teruggevonden
- Locatie waarop de vermiste is teruggevonden
- Overige belangrijke eigenschappen

De points of interrest en locaties waarop de vermiste is gezien opent een nieuw venster waar meer informatie kan worden toegevoegd. Deze worden verder uitgelicht in de volgende paragraaf.
Voor het selecteren van een locatie waar de vermiste is teruggevonden moet op de knop worden geklikt. Dit opent het venster dat te zien is in figuur 7. Hier kan de gebruiker met behulp van de kaart een locatie selecteren.

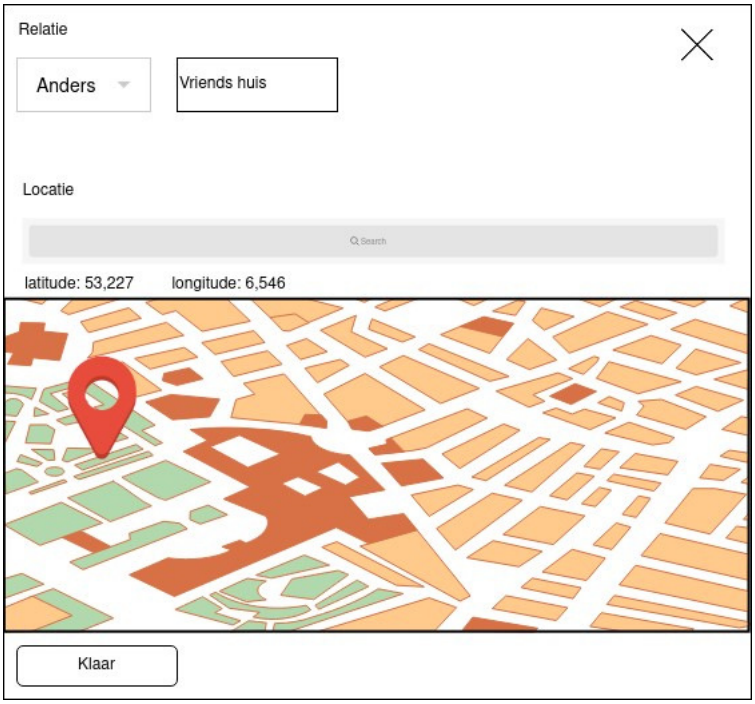


Figuur 7: Locatie selecteren

3.5.1 Point of interest

Voor het invoeren van een point of interrest is een apart venster, zoals te zien in figuur 8. In dit venster kan de gebruiker de bijbehorende relatie selecteren uit een dropdown menu. Als de relatie er niet tussen staat kan

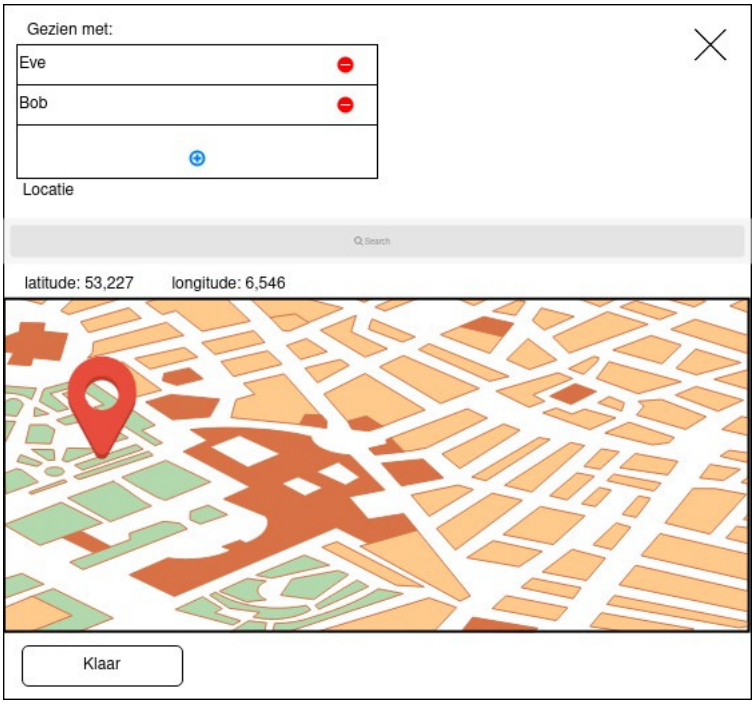
“anders” worden geselecteerd en geeft de gebruiker een eigen omschrijving. Voor het invoeren van de locatie is er een interactieve kaart waarmee de locatie kan worden geselecteerd. Om locaties makkelijker te kunnen vinden is er een zoekbalk. Wanneer tevreden kan er op de klaar knop worden gedrukt. Het is op elk moment mogelijk terug te komen voor verdere wijzigingen.



Figuur 8: Point of interrest toevoegen

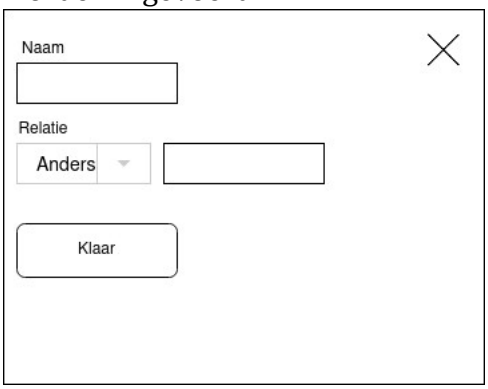
3.5.2 Gezien op locaties

Er is een apart venster voor het invoeren van locaties waar de vermiste is gezien. Dit is te zien in figuur 9. In dit venster kan de gebruiker een locatie selecteren met een interactieve kaart. Verder is er toegang tot een zoekfunctie voor het snel kunnen vinden van plaatsen. Verder kunnen er personen worden toegevoegd waarmee de vermiste is gezien.



Figuur 9: Locatie waar vermiste is gezien toevoegen

Voor het invoeren van de eigenschappen van deze personen is een ander venster, zoals te zien in figuur 10. Hier kan de naam en relatie tot de vermiste worden ingevoerd.

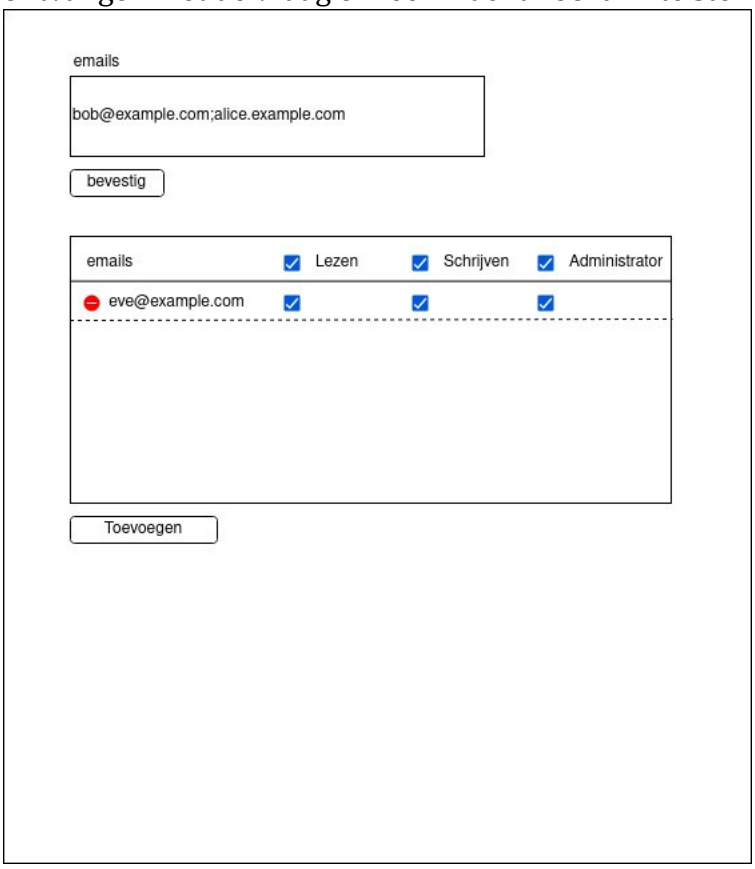


Figuur 10: Persoon toevoegen

3.6 Account toevoegen

Voor het toevoegen van nieuwe gebruikers is een apart scherm zoals te zien in figuur 11. E-mails kunnen in een tekstveld worden ingevoerd waarna

deze in de lijst verschijnen. Er kunnen meerdere e-mails tegelijk worden ingevoerd door deze te scheiden met een ”;”. Wanneer de e-mails zijn ingevoegd kan de gebruiker de rechten voor individuele gebruikers zetten door het aanvinken van de checkboxen rechts van de e-mails. Als alle gebruikers dezelfde rechten moeten hebben kan de checkbox bovenaan worden gebruikt om het in een keer voor alle gegeven e-mails in te stellen. Wanneer de instellingen voldoen kan erop toevoegen worden geklikt om deze gebruikers aan te maken. Elk ingevoerd adres zal hierna een email ontvangen met de vraag om een wachtwoord in te stellen.



Figuur 11: Accounts toevoegen

4 Web interface

In dit hoofdstuk worden de essentiële web API endpoints voor het verwerken van vermissingen besproken. Deze dienen als interface voor de website om informatie op te vragen, plaatsen en bewerken. Deze endpoints worden in eerste instantie gebruikt voor de website maar zijn op geen enkele manier afhankelijk van de website. Deze kunnen op een later moment voor andere applicaties worden gebruikt.

De web API is gemaakt met REST als architectuur style. Alle data wordt gerepresenteerd in het JSON-formaat. Voor het maken van een REST ontwerp zijn de volgende onderdelen meegenomen. Er wordt geen sessie status onthouden door de webserver. Alle informatie voor het adresseren van resources (bijv. individuele vermissingen) zijn encoded in de URI. Verder wordt er gebruik gemaakt van http-methodes (post get en put) en http-status codes voor het overbrengen van informatie.

Alle JSON-berichten verderop zijn opgeschreven met TypeScript syntax om de mogelijke waarden van de velden uit te drukken. Alle JSON-berichten gebruiken camel case als naam conventie voor de sleutel. Dit is omdat dit aansluit bij de naam conventies van de gebruikte programmeertalen, waarover meer valt te lezen in hoofdstuk 5. Al de endpoints in dit hoofdstuk vereisen authenticatie en dit zal dan niet verder worden besproken bij de individuele endpoints. Voor meer informatie over de authenticatie zie hoofdstuk 7.

4.1 Vermissing aanmaken

Voor het aanmaken van een nieuwe vermissing moet de nodige informatie met POST worden verstuurd. De te versturen data moet in valide JSON zijn met de volgende structuur.

Buiten de standaard JSON-hiërarchie is hier een relatie tussen seenAt en persons waar de naam als sleutel wordt gebruikt. Dit is omdat de vermiste meerdere keren kan zijn gezien met dezelfde persoon. Wanneer een vermissing succesvol is aangemaakt geeft deze het aangemaakte UUID van de vermissing terug. Dit UUID kan worden gebruikt voor het linken naar de URL van de aangemaakte vermissing.

4.2 Vermissing opvragen

Het opvragen van de data van een vermissing kan met de bijbehorende URL. Deze URL ziet er als volg uit: /missing/{uuid}. Hier is het UUID hetzelfde als dat is teruggegeven bij het aanmaken van de vermissing. Wanneer er een vermissing bestaat geeft het een JSON-resultaat in de volgende vorm:

4.3 Vermissing bewerken

Voor het bewerken van een vermissing moet een volledig nieuwe versie worden geplaatst. Dit kan met de PUT methode bij de URL /missing/{uuid}. De structuur van de nieuwe vermissingsdata is grotendeels hetzelfde als dat bij het aanmaken van een nieuwe vermissing. Deze is hieronder te zien.

Er hier één veld meer dan bij het aanmaken van een nieuwe vermissing namelijk het lastModified veld. Door het meegeven van wanneer volgens de gebruiker de laatste wijziging is gedaan kan gecontroleerd worden of er wijzigingen zijn geweest die overschreven kunnen worden. Als het lastModified veld niet overeenkomt met wat in de database staat krijgt de gebruiker een 409 Conflict reactie terug.

4.4 Vinden van vermissingen

Voor het terugvinden van vermissingen is een aparte route */missing/search* die te bereiken is met de GET methode. Voor het meegeven van zoek criteria wordt gebruik gemaakt van query syntax in de URI. Voor het vinden van een vermissing van Bob die 20 jaar is zal de URI eruit zien als volgt: */missing/search*?*name* = *Bob*&*age* = 20

De resulterende response van de server heeft de hieronder beschreven structuur.

De response is een array met hierin een object voor elke overeenkomende vermissing. In dit object zit het UUID om de individuele vermissing mee op te vragen. Verder zit er data bij om menselijke gebruikers te helpen het identificeren van de juiste vermissing.

5 Technologieën

In dit hoofdstuk worden de gebruikte technologieën besproken voor het maken van de applicatie. Hieronder vallen gebruikte programmeertalen, frameworks en essentiële dependencies.

5.1 Website

Hier zal verder worden ingegaan op de eigenschappen van de frond-end van de website. De website wordt gemaakt als een single page application. Hiermee staat de frontend los van de server. Verder opent dit extra mogelijkheden voor gebruik als een PWA (progressive web app).

5.1.1 Programmeertaal

Voor het vormgeven van de frontend worden standaard HTML en CSS gebruikt. Voor het afhandelen van alle logica wordt gebruik gemaakt van TypeScript. TypeScript is een programmeertaal ontwikkeld door Microsoft en transpiled naar JavaScript, wat native draait in de webbrowser. TypeScript is een superset van JavaScript wat betekend dat alle functionaliteit van JavaScript ook in TypeScript beschikbaar is. Wat TypeScript toevoegt aan JavaScript zijn data type annotaties. Dit stelt de transpiler instaat om type fouten te vinden zodat deze niet plaatsvinden bij het uitvoeren van de applicatie. Verder geven de type annotaties een vorm van documentatie wat beter overbrengt wat alle variabelen zijn. Mocht een volgende ontwikkelaar besluiten niet verder te werken met TypeScript dan kan de configuratie worden aangepast om standaard JavaScript weer toe te staan wat nog steeds met de TypeScript kan samenwerken.

5.1.2 Framework

Voor het maken van een single page application moet een framework worden gebruikt. Er is hier een keuze gemaakt tussen de drie grootste frameworks op het moment, dit zijn: Angular, React en Vue. Angular valt af op basis van een relatief grote ontwikkelaar ontevredenheid wat terug te zien is in de 2019 versie van de state of js survey. Hier gaf 33,9% van de mensen aan Angular gebruikt te hebben en niet weer te willen gebruiken. Ter vergelijking deze nummers waren 8,6% voor React en 6% voor Vue. Er is gekozen om de community ondersteunde Vue te gebruiken in plaats van React. Beide technologieën hebben veel gemeen. De keuze voor Vue

over React is door de manier waarop de structuur van de code werkt. Waar React alles in JavaScript schrijft met JSX maakt Vue gebruik van standaard HTML en CSS-syntax. Dit maakt het makkelijker voor iemand bekend met standaard web technologie om van start te gaan met Vue. Met Vue kan er nog steeds met JSX geschreven worden het is alleen niet de enige manier. Verdere voordelen van Vue zijn het integreren van een aantal essentiële bibliotheken in het project zoals de router en state management.

5.2 Server

Voor de server zijn een groot aantal aan technologieën mogelijk. Hieronder wordt de programmeertaal keuze beschreven. Verder worden een aantal essentiële dependencies genoemd.

5.2.1 Programmeertaal

Voor het kiezen van een programmeertaal voor de server zijn veel mogelijkheden. Er zijn twee eigenschappen waar hoofdzakelijk naar is gekeken. Deze eigenschappen zijn betrouwbaarheid en gebruikersgemak. De programmeertaalkeuze is hierbij onlosmakelijk verbonden met beschikbare webframeworks.

Er is gekozen voor de programmeertaal Go. Het grote voordeel van Go is dat het snel te leren valt. De syntax is sterk geïnspireerd op C en het heeft een kleine set aan features om te leren. Voor het schrijven van een webserver is geen framework nodig wat veel complexiteit weghaalt. Go is een gecompileerde taal en type safe. Het geven van errors in Go is onderdeel van de functie signature wat deze makkelijker te zien maakt en daardoor af te handelen. Go maakt gebruik van een garbage collector zodat de programmeur niet aan memory mangement hoeft te denken. Voor het gebruik van externe dependencies heeft Go een ingebouwd modulesysteem. Hoe Go zichzelf omschrijft is als volgt: ”Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.‘ Een laatste reden voor het kiezen van Go is dat er een grote hoeveelheid aan voorbeelden bestaan voor het schrijven van een webapplicatie.

5.2.2 Dependencies

Go heeft een uitgebreide standaard library en zodoende weinig externe dependencies nodig. Een plek waar dit wel nodig is voor webrequest routing.

De standaard Go bibliotheek voldoet hier niet door onder andere het gebrek aan variabelen embedded in de URI. Hiervoor zal de bibliotheek gorilla/mux worden gebruikt. Een verder voordeel aan het gebruik van mux is de mogelijkheid voor het toevoegen van middleware. Verder als database driver wordt pgx gebruikt. Hier wordt direct met de driver gewerkt in plaats van via de database/SQL package van Go. Het nadeel van deze methode is dat de code niet database en driver onafhankelijk is. Het voordeel van deze methode is dat er toegang is tot meer PostgreSQL exclusieve features die niet beschikbaar zijn via de generieke interface.

5.3 Database

Als database wordt PostgreSQL gebruikt. Er is onderzoek gedaan naar de verschillende mogelijkheden voor databasen. Hierin zijn een graaf, document en relationele databasen vergeleken. Hierin was de relationele database (PostgreSQL) betere geschikt voor het modeleren van de data. Verder is het sneller in het doorzoeken van data ten opzichte van MongoDB. Er is gekozen voor een open source oplossing. Hierbij is PostgreSQL gekozen over MySQL/MariaDB omdat het meer features heeft om data weer te geven.

6 Dataopslag

In dit hoofdstuk wordt het datamodel besproken voor het opslaan van de vermissingsdata en overige informatie. Er zijn twee verschillende stukken data die worden opgeslagen. Als eerst de gegevens van vermissingen en ten tweede de accountgegevens van de gebruikers.
Al deze data gaat opgeslagen worden in PostgreSQL. PostgreSQL en een open source relationele database waarvan de eerste versie is uitgekomen in 1996.

6.1 Database beheer

Om change mangagement te kunnen doen met de database wordt een migrations tool gebruikt. Deze applicatie regelt het doorvoeren van nieuwe toevoegingen aan de database en het mogelijk terugdraaien van wijzigingen als die toch niet voldoen. De voor dit project gebruikte migration tool is migrate. Deze is gekozen omdat het in Go is geschreven en dus makkelijker in de sever is te integreren indien nodig. Standaard kan deze migratie tool als CLI (command line interface) applicatie worden gebruikt.

6.2 Schema

Er zal hier meer worden uitgelegd over het database schema. Er is een ERD te zien in figuur 12.

Figuur 12: Entity relation diagram

De users tabel staat los van de vermissingen data omdat wijzigingen niet worden geassocieerd met de gebruiker die deze heeft gemaakt. De users tabel heeft de email van de gebruiker en een hash van het wachtwoord plus informatie over het gebruikte algoritme en salt. Verder bevat de users tabel boolean waardes om aan te geven of de gebruiker lees, schrijf en administrator rechten heeft. De users tabel heeft een relatie met de password changes tabel. Hierin wordt bijgehouden op welke adressen er een nieuw wachtwoord voor een account kan worden ingesteld. Elke wachtwoord verandering heeft een UUID ter identificatie of deze al gebruikt is en een datum waarna deze niet meer werkt.

Vanaf hier gaat het over de opslag van vermissingsdata. Als eerst is het entry point missing_persons. Alle relaties maken gebruik van integers als primary key. Geen van deze sleutels moeten aan de buitenwereld weergegeven worden. Voor het communiceren met externe applicaties moet gebruik worden gemaakt van het UUID als herkenningspunt. De

voordelen van een UUID zijn als volgt. Door het gebruik van een UUID in plaats van een nummer optellen kan iemand niet alle vermissingen een bij een langs zoeken door het nummer met één te verhogen. Een tweede voordeel is dat in het geval dat bij een migratie waarbij alle primary keys aanpast worden de gebruikers URL's niet hoeven te veranderen omdat deze alleen vertrouwen op het UUID. Omdat een UUID overal uniek is kunnen twee instanties naast elkaar draaien en later samengevoegd worden zonder dat de identifier veranderd hoeft te worden. Het nadeel van het gebruik van een UUID in deze manier is dat elke actie op een vermissing eerst in de missing persons tabel moet kijken wat de primary key is die intern wordt gebruikt.
De tabel find_location is afgesplitst in een aparte tabel zodat er kan worden geforceerd om bij longitude en latitude allebei mee te geven wanneer de vermiste is teruggevonden. Alle andere relaties zijn one to many relaties. De transportation tabel heeft alleen een tekstveld waarin het soort transport staat. Dit is een tekst veld omdat er geen definitieve set aan waarden is om uit te laten kiezen. Deze eigenschap zal later ook terug te zien zijn in de type relatie velden. Er is ook de points_of_interrest tabel waar betekenisvolle locaties voor de vermiste in staan. Naast de relaties en coördinaten is er een naam veld om de betekenis beter over te brengen aan mensen die de waarden lezen. Dan is er nog de last_seen tabel met de momenten waarop de vermiste is gezien. Deze tabel heeft een many to many relationship met de persons tabel. Dit is omdat verschillende waarnemingen met dezelfde persoon kunnen zijn. Deze persoon heeft mogelijk meerdere soorten relaties tot de vermiste. Als laatste is er de additional_properties tabel. Hierin staan eigenschappen gelinkt aan de vermiste die van belang zijn maar niet onder de andere categorieën vallen.

7 Security

Voor het beveiligen van de gegevens moet er authenticatie en autorisatie plaatvinden. Voordat een gebruiker kan inloggen moet deze over een account beschikken. Deze accounts kunnen niet door een nieuwe gebruiker worden aangemaakt, in plaats daarvan moet een gebruiker met de administrator rechten het email adres en bijbehorende rechten toevoegen. In dit hoofdstuk wordt de authenticatie flow beschreven, mitigaties voor een aantal aanvallen en aanvallen die niet worden tegengehouden.

7.1 Authenticatie en autorisatie

Voor het gebruik maken van de applicatie moet de gebruiker kunnen aangeven dat deze toegang heeft tot de service. Om dit te kunnen doen moet de gebruiker inloggen met een email adres en wachtwoord. De authenticatie flow is te zien in figuur 13. Als eerst moet de gebruiker zijn email en wachtwoord naar de login route sturen. Hier wordt deze gevalideerd tegen de database. Wanneer de gegeven credentials overeenkomen wordt er een JWT gemaakt en als cookie gezet bij de gebruiker.
Wanneer de gebruiker naar één van de beveiligde routes gaat wordt de cookie meegestuurd. Voordat de request bij de route logica komt gaat deze door een middleware laag. Deze middleware valideert dat de gebruiker ingelogd is. Als de JWT een bepaalde leeftijd heeft dan wordt in de database opgezocht of tussen de uitgave en nu veranderingen zijn geweest in de gebruikers credentials. Als dit het geval is dan wordt de JWT geïnvalideerd en toegang geweigerd. Als de gebruikte credentials nog hetzelfde zijn wordt de JWT vernieuwd door een nieuwe cookie te zetten op de response van de route. Als de JWT meer dan een maand oud is dan wordt deze alsnog geweigerd en niet vernieuwd.
Als laatst voor het uitloggen kan de gebruiker een request sturen naar de hiervoor bedoelde route. Deze route verwijderd alle aanwezige cookies.

Figuur 13: Authenticatie flow

7.2 Aanval mitigaties

Er zullen hier een aantal instanties van aanvallen worden genoemd en wat voor maatregelen ertegen genomen zijn. Als eerst het geval dat de database gekraakt is. Om de gebruikers wachtwoorden alsnog te beschermen worden deze opgeslagen als hash. Voor het hash algoritme wordt Argon2 gebruikt. Dit algoritme heeft in 2015 de Password Hashing Competition gewonnen. Verder wordt voor elk wachtwoord een uniek salt gebruikt.

Vervolgens de beschermingen die zijn geplaatst bij de JWT tokens. Deze tokens bevatten informatie over de rechten van de gebruiker. Om wijzigingen te voorkomen zijn deze ondertekend met een private key. Bij wijzigingen klopt de signature niet meer en wordt de token als ongeldig beschouwd. De tokens worden als cookie gezet zodat deze door de webbrowser alleen naar het domein van de webserver worden gestuurd. De requests zijn verder voorzien van de eigenschappen secure en HttpOnly. Deze zeggen de webbrowsers om de cookies alleen over een versleutelde verbinding te versturen en JavaScript geen toegang te geven tot de cookie. Door het verplichten van encryptie wordt het onderscheppen van de token moeilijker gemaakt. Verder door het toegang weigeren van JavaScript tot de token voorkomt het dat het deze naar een derde partij kan doorsturen.

Om ervoor te zorgen dat er geen token worden uitgedeeld aan het verkeerde domein door namaaksites wordt gebruik gemaakt van de CORS functionaliteit. Als een webbrowser een request naar een ander domein stuurt vraagt het eerst of het domein van de website toegang hoort te hebben, wanneer de server zegt van niet dan wordt het netwerkverkeer geblokkeerd door de webbrowser.

Aan de server kant wordt bij de login route timing en dos aanvallen moeilijker gemaakt. Bij het opstarten van de server wordt de tijd gemeten dat het kost om een wachtwoord te hashen. Nu wanneer iemand inlogt met een incorrect email adres wordt voor de gemeten tijd gewacht voordat het een reactie naar de gebruiker stuurt. Door deze wachttijd is het niet aan de responsetijden te zien of het email adres wel of niet bestaat. De reden dat er gewacht wordt in plaats van het gegeven wachtwoord te hashen is om dos aanvallen geen makkelijke plek te geven om intensieve reken operaties te forceren.

Als laatst is een maatregel voor het beperken van schade als een gebruikersaccount zijn wachtwoord is gekraakt. Als hulp bij het verminderen van schade krijgt de gebruiker elke keer bij het inloggen een email met de tijd en locatie van inloggen. Als de gebruiker zich hier niet in herkend kan deze acie ondernemen om schade te beperken.

7.3 Niet gemitigeerde risico's

Nu worden de risico's berspoken waar geen maatregelen tegen gebouwd zijn. Als eerste detectie van cookie hijacken. Zoals in de vorige deel besproken zijn er maatregelen om het aanval oppervlak te beperken maar die zijn er niet voor het detecteren van alsnog gelekte tokens. Deze kunnen bijvoorbeeld lekken doordat het systeem van de eindgebruiker zelf niet meer vertrouwd is. Ook kan het door het vertrouwen van verkeerde certificaten of dat de server een verouderde versie van TLS gebruikt.
Voor het veilig houden van individuele account moet vertrouwd worden op de eindgebruikers om veilige unieke wachtwoorden in te stellen en deze niet te delen met andere.

Een ander manier van toegang krijgen is wanneer een derde toegang heeft tot de key die wordt gebruikt voor het encrypten van de tokens. Hier zal volledig worden vertrouwd op de veiligheid van de servers om de private key geheim te houden. Met de private key kan iemand zijn eigen authenticatie tokens maken en zichzelf daarbij alle rechten geven.

8 Deployment

Hier zal worden besproken hoe de applicatie in een productie omgeving geplaatst kan worden. Voor de ondersteunde methode wordt Docker met Docker Compose gebruikt. Wanneer het gebruik van deze container technologieën niet gewenst is kan alles ook direct op de server worden geïnstalleerd. Echter om zo weinig mogelijk aannames over de productieomgeving te hoeven nemen wordt Docker gebruikt met Docker Compose als configuratie middel gebruikt. De aannames die worden gedaan over de server is dat dit een Linux distributie draait met hierop Docker en Docker Compose geïnstalleerd. Bij gebruik van de volledige Docker stack worden in totaal vijf containers gebruikt. De eerste twee hiervan zijn build containers die de website en backend compileren. Deze compileer resultaten worden door twee andere containers gebruikt om het resultaat beschikbaar te stellen aan de buitenwereld. De vijfde container is een die de database draait. Het is aan te raden deze geen toegang te geven van buitenaf wanneer dit niet nodig is.
De containers zullen de backend en website op twee verschillende poorten beschikbaar stellen. Deze poorten zijn handmatig te configureren. Er vindt geen encryptie plaats binnen deze containers. Deze containers moeten daarom beschikbaar worden gemaakt aan de buitenwereld via een reverse proxy zoals Nginx met daarop TLS ingesteld. Het is aangeraden alleen TLS-versies 1.2 en 1.3 te gebruiken. De website en frontend moeten op twee verschillende (sub)domeinen beschikbaar worden gesteld.
Wanneer de rest van de omgeving is ingesteld kan de service gestart worden. Voor het starten van de service moet de configuratie correct in het .env bestand worden ingevuld. Hierna moet *docker–compose up* worden uitgevoerd en na enkele minuten zal de service beschikbaar zijn. Als dit de eerste keer is dat de service draait dan moeten er gebruikers worden toegevoegd. Voor het aanmaken van de eerste gebruikers kan er worden ingelogd op het account [redacted] @localhost met het wachtwoord [redacted]. Met dit account moeten er direct nieuwe gebruikers met de administrator rechten worden aangemaakt. Na het maken van de nieuwe administrators moeten alle rechten van het originele account worden weggehaald.