Software ontwikkeling
en professionele vaardigheden

2de bachelorjaar informatica

Rapporteringsysteem Federale Politie: Analyseverslag

Groep 1:
Chass Beerts (1746388),
Mathias Jans (1848820),
Dieter Warson (1745609)

Begeleider
Mariano Di Martino

Coördinerend verantwoordelijke
prof. dr. Wim Lamotte

Jaar 2019-2020

Inhoudsopgave

1 Diepgaande beschrijving van het onderwerp

Het einddoel van het project is om het opmaken van wachtverslagen te vergemakkelijken voor de politie.
Een fiche is een verslag dat wordt opgemaakt wanneer er een oproep gedaan wordt naar de politie en zij dan in actie moeten komen. Dit verslag bevat dan persoonlijke informatie van de beller, zoals de naam van de persoon en de locatie van het voorval. Maar er kunnen ook andere dingen in het verslag voorkomen zoals of er een arts aanwezig was en of er gebruik is gemaakt van een helikopter. Elk fiche heeft een uniek PL-nummer. Van elke oproep wordt een digitale fiche aangemaakt, elk zo een fiche heeft dan een uniek nummer, een PL-nummer.
In de meldkamer van de politie moet de supervisor tijdens elke shift wachtverslagen opmaken, deze bevatten verschillende fiches van tijdens de shift. In deze verslagen wordt er informatie bijgehouden van een aantal rubrieken: operationeel, personeel en technisch. Op het einde van de shift is er een briefing om iedereen op de hoogte te brengen van wat er doorheen vorige shift allemaal is gebeurd, dit wordt met behulp van de wachtverslagen gedaan. Ook worden de wachtverslagen gebruikt om op het einde van het jaar statistieken op te maken.
De applicatie moet enkel beschikbaar zijn in de browser van de supervisors die werken op een lokaal gesloten netwerk. Ten eerste moet er een overzichtelijke en gemakkelijke manier zijn om verslagen te maken. In het verslag zijn er verschillende categorieën met hun specifieke velden die de nodige informatie bevatten. Het toevoegen van informatie in het verslag moet zo gebruiksvriendelijk mogelijk zijn. Men wil dit bekomen door de gewenste vakjes die zij willen zien aan te vinken. Men kan de informatie van een fiche automatisch toevoegen aan het verslag door het PL-nummer in te geven.
Bij het opstarten van de applicatie moet men inloggen, op deze manier kan er bijgehouden worden wie welke verslagen heeft gemaakt. Ook zorgt dit ervoor dat alleen de supervisors verslagen kunnen maken. Het secretariaat kan dan de verslagen bekijken en eventueel ook de opvolging zien. Om nieuwe velden toe te voegen aan een bepaalde categorie, wordt er gebruik gemaakt van een administrator-account. Deze wordt beheerd door iemand hogerop, dus niet door de supervisors.
Het is ook mogelijk om het verslag naderhand nog aan te passen. Vervolgens moet het systeem op het einde van de shift een pdf sturen naar alle personen uit een maillijst met alle gebeurtenissen van de vorige shift. Deze pdf bevat enkel de velden van het verslag die ingevuld zijn, dus als er bijvoorbeeld niemand ziek is geweest, dan zal dit veld ook niet toegevoegd worden. Ook is het belangrijk om een vorig verslag gemakkelijk terug te vinden met behulp van een zoekfunctie en eventueel filters voor bepaalde velden.
Er moet ook een manier zijn om een verslag op te vragen dat alle data van het jaar bevat, zoals bijvoorbeeld hoeveel keer er een achtervolging was waarbij een helikopter ingezet is geweest. Een functie voor opvolging kan ook toegevoegd worden. Dit wordt gebruikt om te vermijden dat er meerdere keren een bepaald probleem gemeld moet worden. Er kan dan bij een bepaald probleem informatie aangevuld worden en als het probleem opgelost is kan dit aangeduid worden.

Extra functies

Autosave kan ook een mogelijke toevoeging zijn, tijdens dat men een verslag maakt zal het programma dan bijhouden wat er wordt ingegeven. Pas als alle nodige informatie ingevuld is, dan kan het verslag ingediend worden om in de database gezet te worden. Ook kan er een visuele voorstelling gemaakt worden van het verslag op het einde van de shift en voor de statistieken. Dit om ervoor te zorgen dat alles overzichtelijker is en dat bepaalde gebeurtenissen benadrukt worden, bijvoorbeeld als iemand ziek is of als er een technisch defect is.

2 Gerelateerd werk: software, libraries en literatuur

2.1 Software

- ERP-software
  De software die wij moeten ontwikkelen is vaak geïntegreerd in ERP-software. Enterprise Resource Planning (ERP) wordt gebruik ter ondersteuning van alle processen binnen een bedrijf. Met deze software kunnen rapporteringen gemaakt worden.

- Microsoft Access
  Access kan gebruikt worden om gemakkelijk een database te maken en daarmee een front-end te verbinden, maar ook bijvoorbeeld om statistieken en rapporten mee op te maken. Dit werd deels al gebruikt bij de opdrachtgever, maar het werkte niet goed genoeg. Indien hij de statistieken wou opvragen, werd niet alle beschikbare data verstuurd, hierdoor moest een deel van de data manueel nagekeken worden.

2.2 Libraries

- Node-Cron
  Deze module wordt gebruikt om een Cron Job binnen Node.js te maken. [10] Dit hebben we nodig bij het automatisch verzenden van de wachtverslagen op het einde van de shift.

- Vue-chartjs
  Deze library kan gebruikt worden om de visualisaties te maken. Deze maken het overzichtelijk om te vergelijken tussen verschillende jaren. [11]

- pdf-creator-node
  De verslagen worden op het einde van de shift doorgestuurd in pdf-formaat. Om een html template om te vormen naar een pdf-bestand gebruiken we pdf-creator-node. Hier kunnen we alle nodige voorwaarden aan meegeven, bijvoorbeeld dat het verslag liggend moet zijn. [12]

3 Evaluatiecriteria

3.1 Functionele Vereisten

- De gebruiker kan inloggen op het systeem met zijn of haar account
- Een verslag kan toegevoegd worden aan de database
- Bij foute invoer in een van de velden van het verslag krijgt de gebruiker een foutmelding
- Informatie van een fiche wordt automatisch in het verslag toegevoegd bij het ingeven van een PL-nummer
- Op het einde van de shift worden de gebeurtenissen van die shift doorgestuurd naar alle personen van een maillijst
- Verslagen kunnen terug opgehaald worden met behulp van de zoekfunctie
- De inhoud van verslagen kan naderhand aangepast worden
- Er kunnen nieuwe categorie-velden toegevoegd worden aan de categorieën door de administrator
- De statistieken van verschillende categorieën kunnen rechtstreeks opgehaald worden
- De applicatie zorgt ervoor met autosave dat er tijdens het maken van een verslag geen ingegeven data verloren gaat
- Er is een visualisatie voor de statistieken van het huidige jaar

- Men kan het verslag op het einde van de dag ook laten visualiseren in plaats van een pdf te gebruiken
- Er is opvolging voor problemen die niet dadelijk opgelost kunnen worden, dit probleem kan extra informatie krijgen en er kan aangeduid worden als het opgelost is

3.2 Niet-functionele Vereisten

- Het maken van een verslag kan op een gebruiksvriendelijke manier door middel van checkboxes
- Instellingen en voorkeuren moeten gemakkelijk aan te passen zijn
- Er moet een zo laag mogelijke leercurve zijn om de applicatie te gebruiken

4 Analyse

4.1 Algoritmen

- makeStatistics(data)
  De parameter data bevat de gevraagde velden waarvan er een overzicht met statistieken gemaakt moet worden en de jaren van de gewenste overzichten. In een overzicht is te zien hoe vaak bepaalde gebeurtenissen zijn voorgekomen in een bepaald jaar. Het algoritme gaat dan het aantal voorkomens van een gebeurtenis in het bepaald jaar optellen, door het aantal keer dat het veld van die gebeurtenis voorkomt op te tellen. Dit kan gedaan worden met behulp van een query. De query zoekt alle rapporten waarvoor het bepaalde veld geldig is en gaat vervolgens enkel de rapporten van de aangegeven jaren teruggeven. Dit kan dan in JSON gezet worden en teruggestuurd worden naar de view.
- mailOverview()
  Deze functie zorgt ervoor dat alle supervisors en het personeel van het secretariaat voorzien worden van een samenvatting van alle gebeurtenissen (operationeel, personeel en technisch) van de vorige shift. Met behulp van queries gaan we deze data uit de database ophalen. De query gaat dus van elke categorie alle data verzamelen die in de vorige shift gebeurd is. Dit doet het aan de hand van de datum waarop de rapportering gebeurd is. Al deze data wordt dan in een pdf gegoten zodat het gemaild kan worden. Deze wordt aan het einde van elke shift uitgevoerd door een Cron Job binnen Node.js. [10]

4.2 Datastructuren

- Linked list
  Als er meerdere afwezigheden, technische defecten of operational events zijn, is er een lijst nodig. We voegen enkel elementen toe aan deze lijst en er worden geen elementen verwijderd. Dit betekent dat een linked list de meest efficiënte lijst is in dit geval. De complexiteit om een element toe te voegen is bij een linked list O(1) en bij een list O(n). Een queue en stack zijn ook vormen van linked lists en zij hebben beiden ook bij het toevoegen van data een tijdscomplexiteit van O(1). Een queue en een stack hebben bij het verwijderen van data ook een tijdscomplexiteit van O(1) en de gewone linked list een tijdscomplexiteit van O(n). De reden dat wij linked list verkiezen is omdat we de functionaliteit om data te verwijderen niet nodig hebben en queue en stack dus geen voordeel bieden omtrent tijdscomplexiteit.

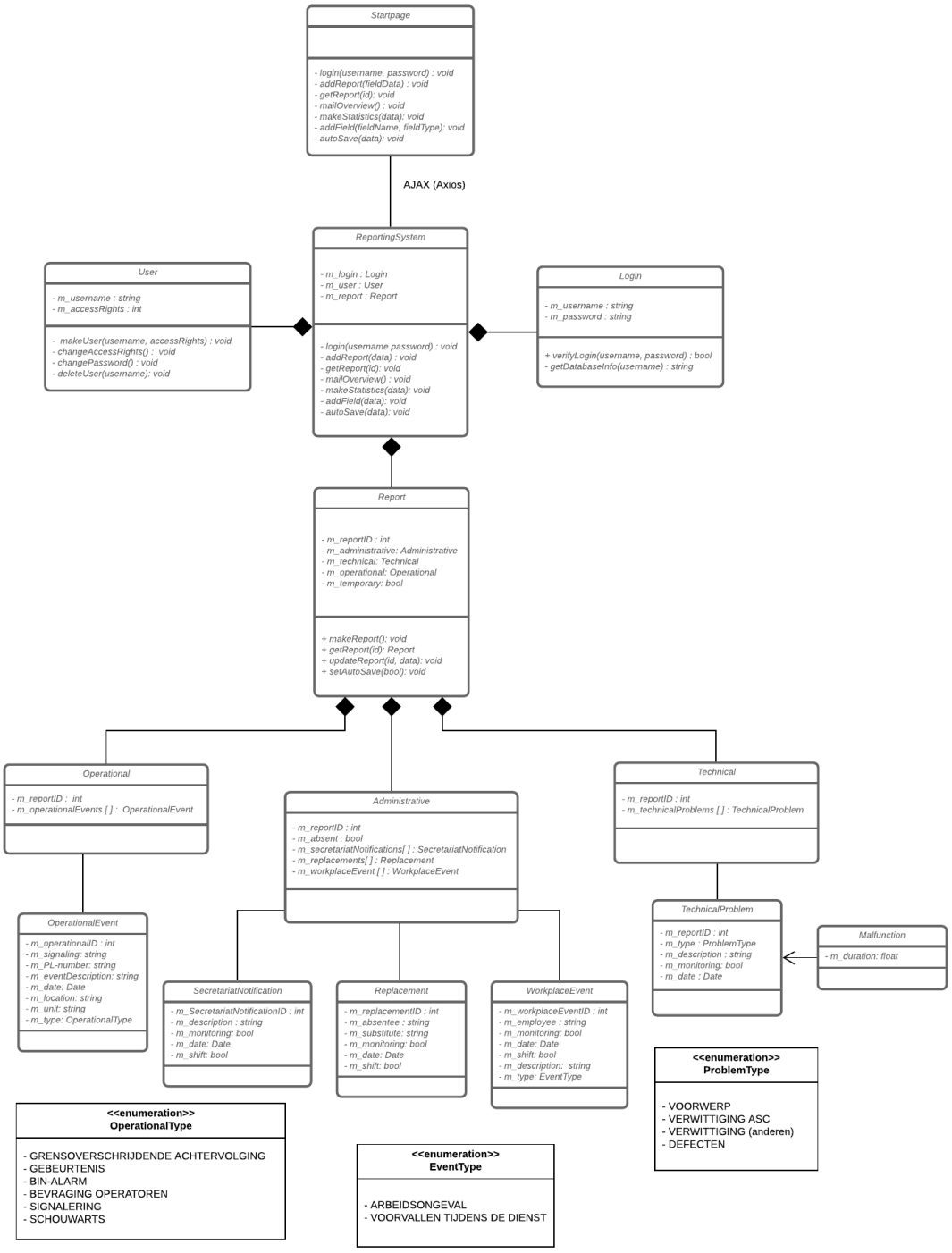
4.3 Klassendiagram en ontwerp

De basisklasse van het model is ReportingSystem, deze bevat de functies die opgeroepen worden bij een aankomende AJAX call. Deze klasse wordt dus als een controller gebruikt zodat men niet in de rest van de klassen AJAX calls moet gaan doen naar de view. Startpage stelt de view voor. Reportingsystem bevat een User object dat gebruikt wordt om users aan te maken en informatie over de gebruikers aan te passen. Vervolgens heeft het een Login object om te controleren of de logingegevens die doorgestuurd geweest zijn correct zijn. Ten slotte heeft Reportingsystem ook nog het Report object dat dan gebruikt wordt om de database te updaten met nieuwe verslagen. ReportingSystem bevat de functies die aangesproken worden bij calls van de view. Een voorbeeld hiervan is wanneer men een verslag wil zoeken met behulp van een PL-nummer. Hiervoor is getReport(id) toegevoegd, deze geeft dus een verslag terug indien deze het PL-nummer bevat.

Report bevat dan de verschillende categorieën, deze bevatten de kolommen die nodig zijn in de database met Sequelize. Ook heeft het een veld om bij te houden of het report voorlopig is om de autosave functionaliteit te voorzien.
Operational bevat een lijst van OperationalEvents, hiermee kunnen alle mogelijke operationele gebeurtenissen bijgehouden worden.
Technical heeft een TechnicalProblem klasse die gebruikt wordt om technische problemen op te slaan. Malfunction wordt afgeleid van TechnicalProblem omdat het dezelfde functionaliteit heeft, maar wel een tijdsduur moet bijhouden. Dit respecteert het Liskov substitution principle omdat TechnicalProblem en Malfunction allebei kunnen gebruikt worden door dezelfde functies zonder dat er een aanpassing gemaakt moet worden voor Malfunction.
Administrative bevat drie klassen die de nodige events bijhouden. Dit zijn SecretariatNotification voor de meldingen aan het secretariaat, Replacement voor afwezigen die vervangen worden en WorkplaceEvent om voorvallen tijdens de dienst bij te houden.

De enum ProblemType wordt gebruikt door TechnicalProblem om het type van het probleem aan te duiden. OperationalType is nodig bij OperationalEvent om de soort gebeurtenis aan te geven. Ten slotte is er ook nog de EventType enum om verschillende WorkplaceEvents aan te duiden. Single responsibility is goed toegepast omdat alle klassen een duidelijke functie hebben binnen de applicatie. Ook hebben we de Report duidelijk opgedeeld om te vermijden dat dit een grote klasse wordt die alles doet.

Om de functionaliteit van nieuwe velden aan te maken gaan we de objecten uit een JSON file importeren zodat als men een field aanmaakt we deze ook in Typescript kunnen gebruiken en zodat deze consistent is doorheen de code.



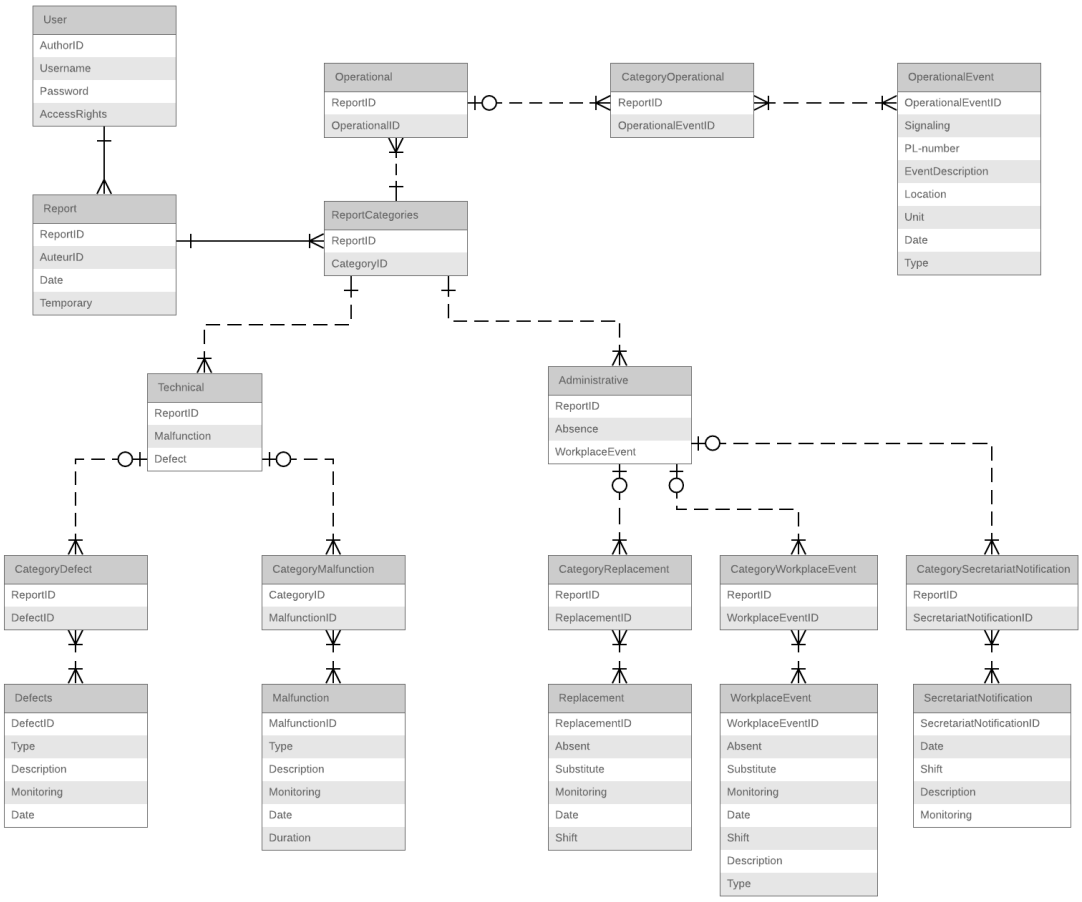
Figuur 1: UML-klassendiagram

4.4 Data en databaseschema

We hebben een lokale database die gebruikt wordt om de verslagen op te slaan. Het ontwerp van deze database is zichtbaar in het schema hieronder toegevoegd. Er is gekozen om het verslag in de verschillende categorieën op te delen zodat er niet te veel kolommen zijn. Bij Administrative is er nog een extra tabel toegevoegd om de vervangingen in te zetten aangezien dit er meerderen per dag kunnen zijn. Dit is ook toegepast bij Technical en Operational.

In de uiteindelijke applicatie gebruiken we de database die lokaal aanwezig is bij de politie. Deze bevat de nodige informatie over de fiches zodat men de data kan importeren in het verslag. Omdat deze database enkel lokaal beschikbaar is in een gesloten netwerk, moeten we een dummy database gebruiken. Deze dummy database bootst tijdelijk de database van de politie na tijdens het testen.

Bij het maken van het verslag is het mogelijk om het PL-nummer in te geven zodat de informatie van een fiche geïmporteerd kan worden. Hier kunnen we vanuit Vue.js met behulp van Ajax de back-end server van de politie aanspreken om de data op te halen. Deze data wordt vervolgens automatisch ingevuld in de velden. Het verslag wordt uiteindelijk in onze lokale database opgeslagen.



Figuur 2: ERD

4.5 Software componenten en programmeertalen

- TypeScript
  TypeScript voorziet static typing in vergelijking met JavaScript dat dynamic typing voorziet. Door static typing wordt de geschreven code leesbaarder en is het eenvoudiger om bepaalde bugs op te lossen. We gebruiken TypeScript in dit project zodat het mogelijk is om een duidelijke structuur aan te houden en alle functionaliteit in klassen op te delen.
  Dit gebruiken we beide front-end en back-end om continuïteit te hebben. [5]

4.5.1 Requests en responses

- Axios.js
  Door middel van AJAX kan de data van een fiche asynchroon ingevuld worden. Hierdoor moet de pagina niet elke keer herladen worden indien er bijvoorbeeld een fiche opgevraagd wordt. We gebruiken hiervoor Axios zodat we niet rechtstreeks met een XMLHttpRequest moeten werken.

4.5.2 Back-end

- MySQL
  Doordat MySQL bepaalde SQL functies niet opneemt, wordt snelheid hier geprioriteerd. Met name gelijktijdige read-only functies worden sneller.[1] Ook hebben we de extra features niet nodig, aangezien we enkel eenvoudige queries zullen moeten uitvoeren. [2] Om deze reden verkiezen wij MySQL voor dit project.

- PostgreSQL
  PostgreSQL is een andere mogelijkheid voor onze database. Het voordeel van PostgreSQL is dat het bijna alle features van SQL bevat. Voor onze applicatie hebben we echter enkel eenvoudige queries nodig en zijn deze features dus overbodig. [2] Daarom kiezen wij om niet met PostgreSQL te werken.

- Django
  Django is gemaakt om alles te bevatten wat een developer nodig zou hebben, hierdoor is alles redelijk consistent en werkt alles goed samen. Er is een goede integratie van relationele databases met een hoge schaalbaarheid. Verder heeft Django standaard-bescherming tegen SQL-injection en cross site scripting. [6]

- Laravel
  Laravel is een PHP framework om webapplicaties te maken volgens het MVC-patroon. Er is ORM ondersteuning ingebouwd om de objecten te mappen op de database. [3]

- Node.js
  Node.js heeft de node package manager (NPM) die ervoor zorgt dat men gemakkelijk packages kan gebruiken die eventueel nodig zijn. Men kan ook Typescript gebruiken in Node.js, dit maakt dat we voor de front-end en back-end dezelfde interface kunnen gebruiken. Ook is er in Node.js snelle en eenvoudige afhandeling van gelijktijdige aanvragen van gebruikers. Hierdoor lijkt ons Node.js de meest interessante optie als back-end software. [3] [4]

- Express.js
  Express is een Node web framework dat de functionaliteit die Node niet heeft beschikbaar maakt. Dit kunnen we bijvoorbeeld gebruiken om de login te maken. Hier kunnen we dan voor elke request aparte functionaliteit implementeren. Ook kunnen antwoorden naar de front-end automatisch gegenereerd worden.

- Sequelize.js
  Sequelize is een promise-based ORM voor Node.js en io.js. Het ondersteunt PostgreSQL, MySQL, MariaDB, SQLite en MSSQL. Deze library gebruiken we om onze databasestructuur mee te maken en onze queries mee uit te voeren. [8]

Voor onze back-end kiezen we Node.js zodat we beide front-end en back-end JavaScript kunnen gebruiken. Meer specifiek gebruiken we dan overal Typescript.

4.5.3 Front-end

- JQuery
  Indien we geen extra functies implementeren is JQuery voldoende als front-end library. Hiermee kunnen we alle invoer implementeren en met de server communiceren. Voor de niet-functionele vereisten is JQuery niet uitgebreid genoeg.

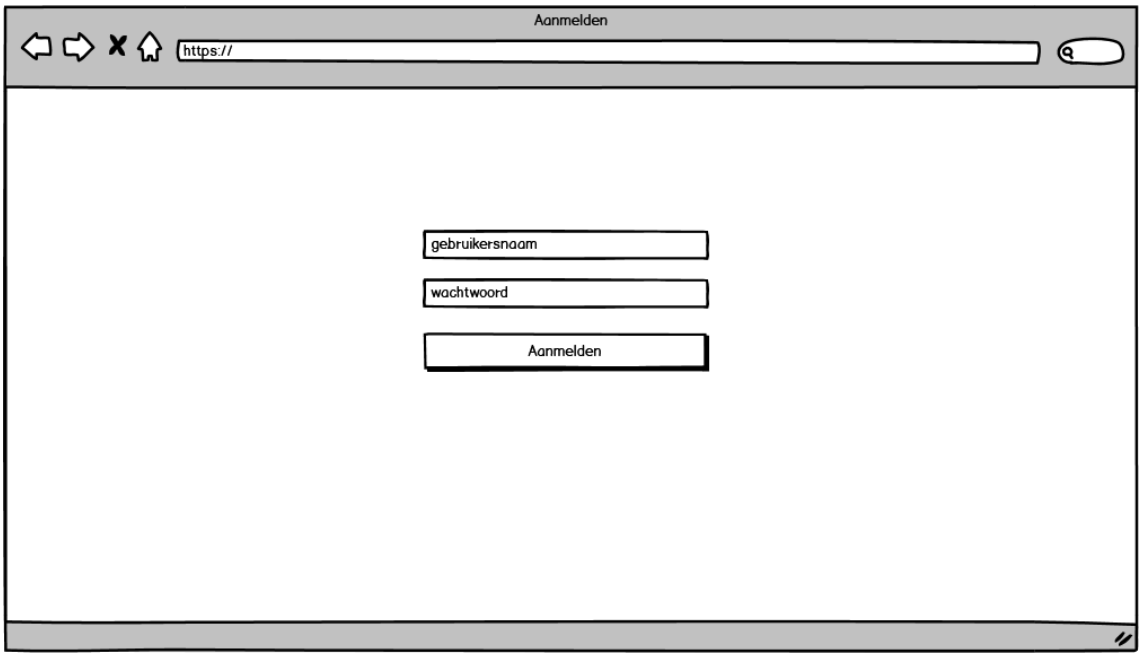
- Vue.js
  Met Vue.js kunnen we de applicatie een single page application maken en de extra functies zoals visualisatie gemakkelijker implementeren. Dit is ook allemaal mogelijk met JQuery, maar hiermee zouden we dan alles zelf terug moeten implementeren terwijl dit al gedaan is in de JavaScript libraries. Vue.js gebruiken we volledig in Typescript zoals hierboven bij backend aangegeven.

- Bootstrap
  Met Bootstrap kunnen we de lay-out van onze website intuïtief opmaken. Bootstrap is ook makkelijk te implementeren met Vue.js. Ook heeft het plug-ins die gebouwd zijn op JQuery. Bootstrap laat ook toe om alleen hun CSS te linken aan ons project, dus we moeten geen overbodige files inladen. [7]

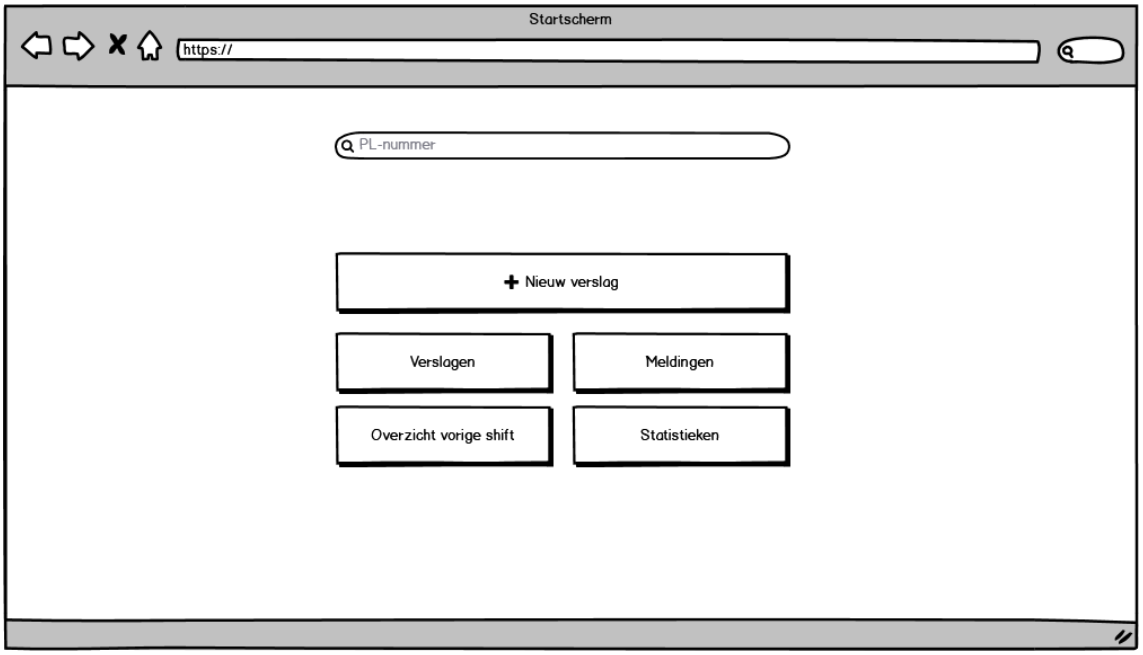
4.6 Bestandsformaat

We gebruiken een environment file (.env) om de logingegevens van de databases op te slaan zodat dit niet rechtstreeks in de code staat. [9]

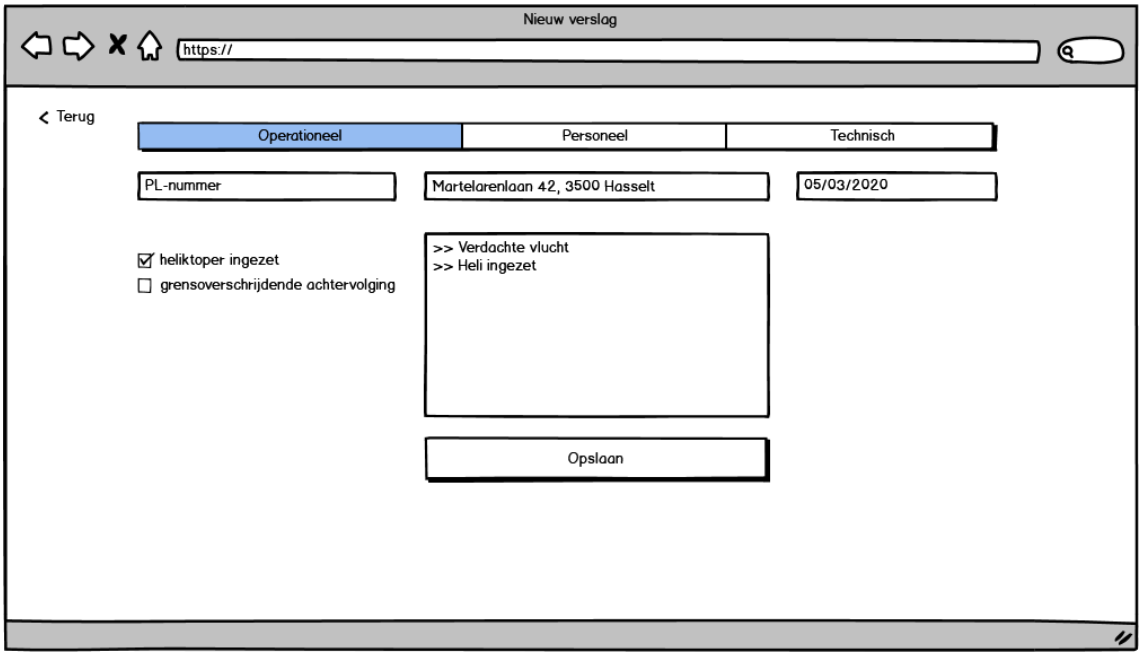
5 Mockups



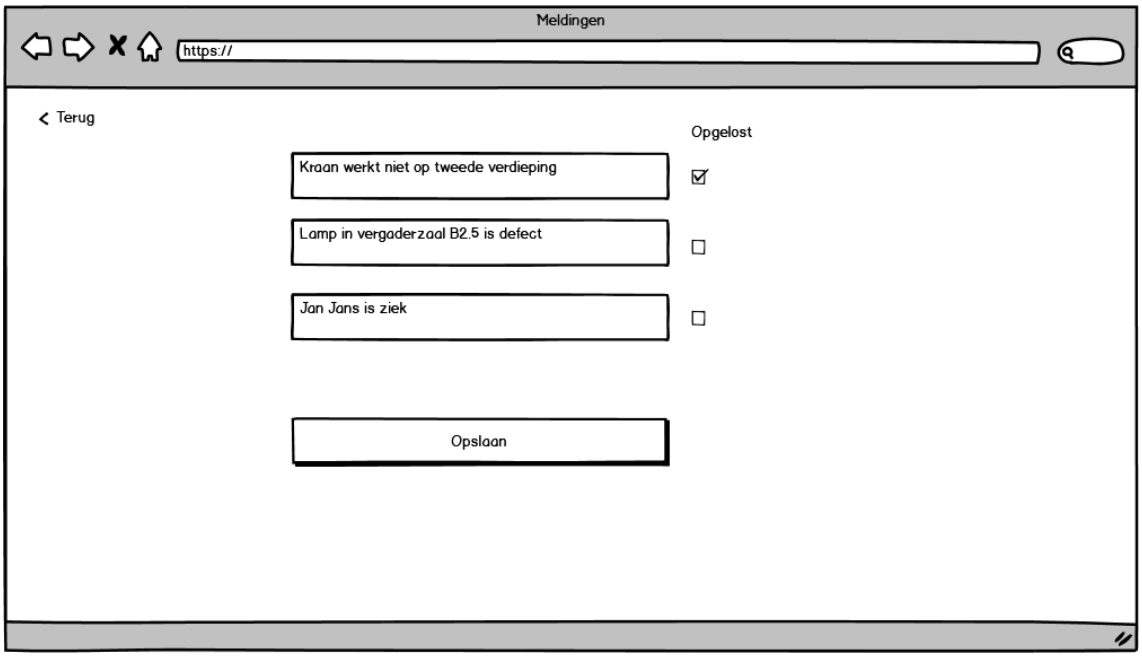
Figuur 3: Aanmeldscherm



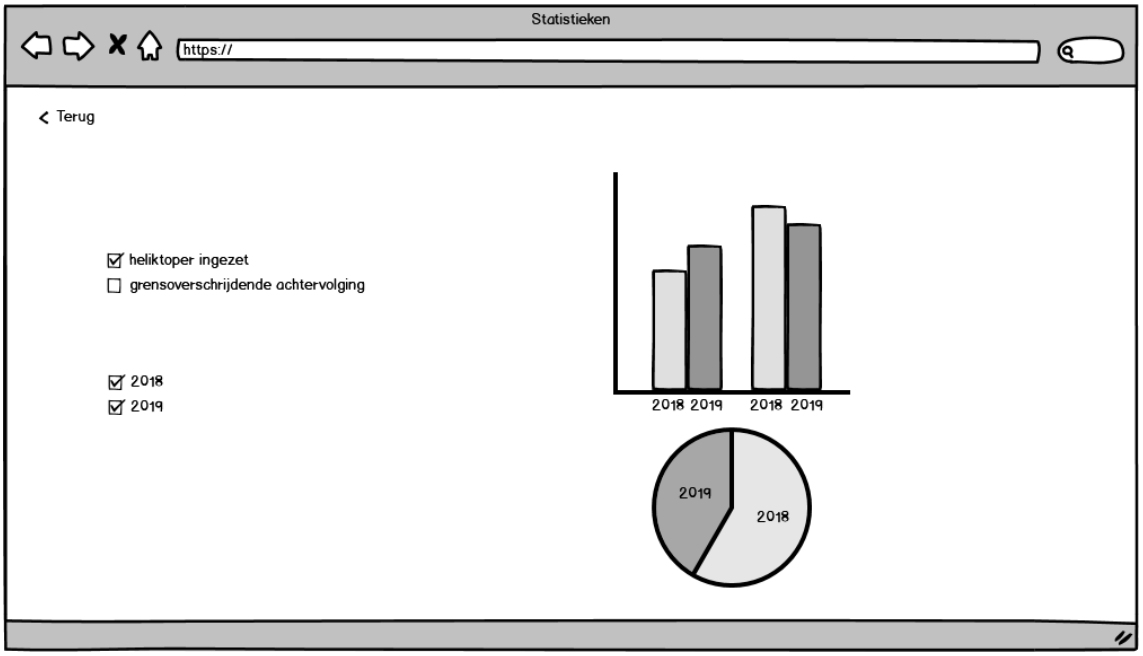
Figuur 4: Startscherm voor de supervisor



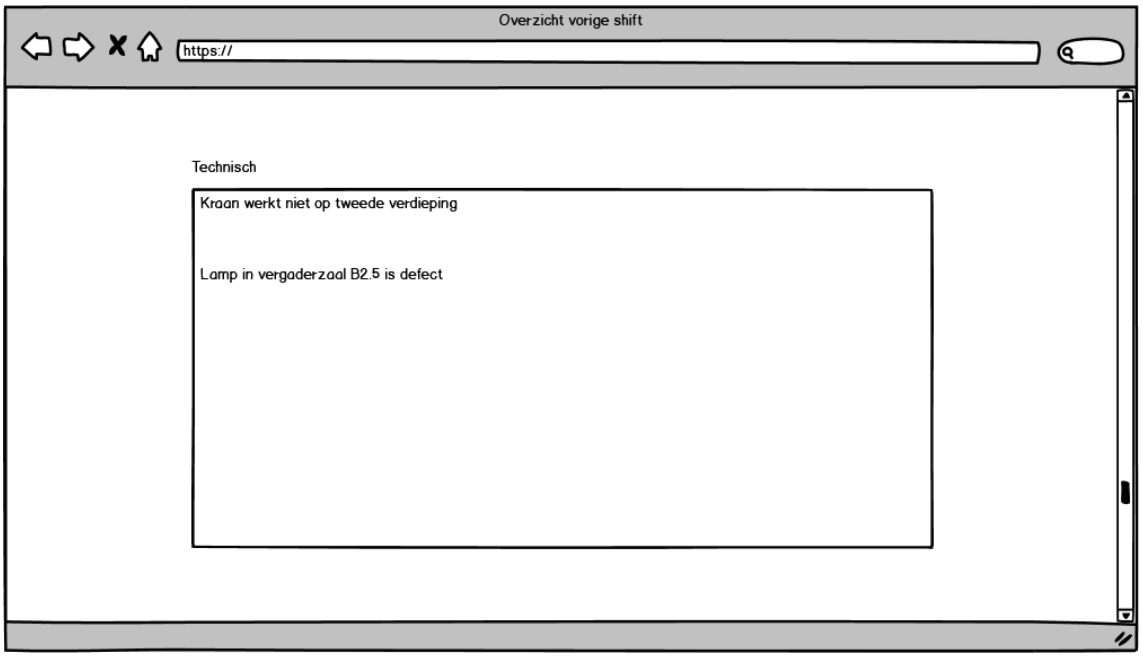
Figuur 5: Scherm om een nieuw verslag te maken



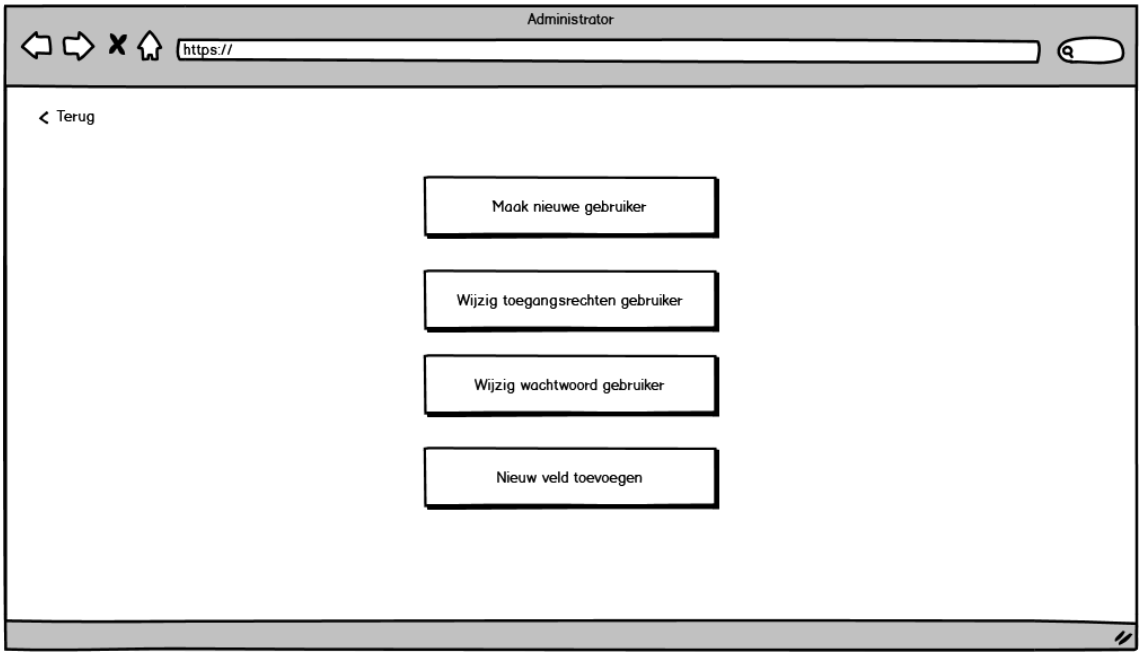
Figuur 6: Opvolgingsfiche



Figuur 7: Statistieken



Figuur 8: Overzicht van de vorige shift



Figuur 9: Extra functies administrator

6 Taakverdeling en planning

- 27/3: Docker omgeving instellen, startscherm, formulier om nieuw verslag te maken
  - Dieter werkt van 20/3 tot 27/3 aan de Docker omgeving, ReportingSystem aanmaken
  - Chass werkt van 20/3 tot 27/3 aan het formulier om een nieuw verslag te maken
  - Mathias werkt van 20/3 tot 27/3 aan het startscherm
- 3/4: structuur database geïmplementeerd, Login werkend
  - Dieter werkt van 27/3 tot 3/4 om Report, Operational en Workforce met Sequelize te implementeren
  - Chass werkt van 27/3 tot 3/4 aan de login (front-end + back-end)
  - Mathias werkt van 27/3 tot 3/4 om Technical en Malfunction met Sequelize te implementeren
- 10/4: nieuw verslag kunnen maken, verslagen ophalen, overzicht van vorige shift
  - Dieter werkt van 3/4 tot 10/4 aan het overzicht van vorige shift
  - Chass werkt van 3/4 tot 10/4 aan het aanmaken van nieuwe verslagen in de database, ophalen van data uit dummy database politie aan de hand van PL-nummer
  - Mathias werkt van 3/4 tot 10/4 aan het ophalen van verslagen uit de database
- 17/4: statistieken, admin functies, gebeurtenissen doorsturen naar maillijst
  - Dieter werkt van 10/4 tot 17/4 aan gebeurtenissen doorsturen naar maillijst
  - Chass werkt van 10/4 tot 17/4 aan de admin functies
  - Mathias werkt van 10/4 tot 17/4 aan de statistieken
- 24/4: verslagen aanpassen, velden toevoegen, opvolging
  - Dieter werkt van 17/4 tot 24/4 aan opvolging
  - Chass werkt van 17/4 tot 24/4 aan velden toevoegen
  - Mathias werkt van 17/4 tot 24/4 aan verslagen aanpassen
- 1/5: visualisatie statistieken, testen van al beschikbare functionaliteiten, velden toevoegen
  - Dieter werkt van 24/4 tot 1/5 aan visualisatie statistieken, testen
  - Chass werkt van 24/4 tot 1/5 aan velden toevoegen
  - Mathias werkt van 24/4 tot 1/5 aan visualisatie resterende statistieken, testen
- 8/5: visualisatie vorige shift, autosave
  - Dieter werkt van 1/5 tot 8/5 aan autosave: databasetabel en terug ophalen concept
  - Chass werkt van 1/5 tot 8/5 aan visualisatie vorige shift
  - Mathias werkt van 1/5 tot 8/5 aan autosave: de ingegeven data in de database overzetten
- 15/5: testen, autosave, eindverslag maken (ruimte voor eventuele problemen of ziekte)
  - Dieter werkt van 8/5 tot 15/5 aan autosave afmaken, testen, eindverslag
  - Chass werkt van 8/5 tot 15/5 aan refactoring en eindverslag
  - Mathias werkt van 8/5 tot 15/5 aan autosave afmaken, testen
- 22/5: applicatie volledig functioneel, refactoring, presentatie, eindverslag (ruimte voor eventuele problemen of ziekte)
  - Dieter werkt van 15/5 tot 22/5 aan refactoring, presentatie en eindverslag

  - Chass werkt van 15/5 tot 22/5 aan refactoring, presentatie en eindverslag
  - Mathias werkt van 15/5 tot 22/5 aan refactoring, presentatie en eindverslag
- 27/5: indienen eindproduct
  - Dieter werkt van 22/5 tot 27/5 aan refactoring, presentatie en eindverslag
  - Chass werkt van 22/5 tot 27/5 aan refactoring, presentatie en eindverslag
  - Mathias werkt van 22/5 tot 27/5 aan refactoring, presentatie en eindverslag

7 Bibliografie

Referenties

[1] Smallcombe Mark, 2019. *PostgreSQL vs. MySQL: Which One Is Better for Your Use Case?*, https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/

[2] ostezer + Mark Drake, 2019. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems*, https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[3] Chandresh Patel, 2019. *Laravel Vs. Django Vs. Node – Which one is better?*, https://www.bacancytechnology.com/blog/laravel-vs-django-vs-nodejs

[4] *Express web framework (Node.js/JavaScript)*, https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

[5] *Why You Should Use TypeScript for Developing Web Applications*, https://dzone.com/articles/what-is-typescript-and-why-use-it

[6] *Django Web Framework*, https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django

[7] *Bootstrap*, https://getbootstrap.com/

[8] *Sequelize — The Node.js ORM*, https://sequelize.org/v3/

[9] *dotenv* https://github.com/motdotla/dotenv

[10] *node-cron* https://github.com/kelektiv/node-cron

[11] *vue-chartjs* https://github.com/apertureless/vue-chartjs

[12] *pdf-creator-node* https://github.com/hajareshyam/pdf-creator-node