ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

MeldAPp

Technische documentatie

ondersteund door de

Artesis Plantijn Hogeschool

Academiejaar 2019-2020
2de semester

Inhoudstafel

Termen en Afkortingen

Term	Omschrijving
AP	Artesis Plantijn Hogeschool
Azure AD	Azure Active Directory
Defect	Iets dat niet werkt of kapot is
EG	Externe Gebruiker
ELL	Ellermansstraat
FA	Facilitaire Administrator
FM	Facilitaire Medewerker
LA	Logistieke Administrator
LM	Logistieke Medewerker
M	Medewerker
Melding	Overkoepelende term voor zowel defecten als opdrachten
NOO	Noorderplaats
OH	Opleidingshoofd
Opdracht	Iets dat gedaan moet worden/klaargezet worden
SA	Superadministrator
SPA	Single Page Application
TI	Toegepaste Informatica
VSC	Visual Studio Code

Samenvatting van de opdracht

Hoewel de campussen ELL en NOO relatief nieuw zijn, komen de studenten en medewerkers van AP Hogeschool nog dagelijks nieuwe defecten tegen. Het melden van deze defecten gebeurt via verschillende kanalen (verbaal, via e-mail, via de telefoon, etc.).

Aangezien er geen structuur is, is het voor de medewerkers niet duidelijk of een bepaald defect reeds gemeld is of niet. Hierdoor worden sommige defecten meerdere malen gemeld en andere weer helemaal niet. De defecten die wel gemeld worden, worden vaak vergeten of gaan verloren.

De studenten TI van de AP Hogeschool werden gevraagd om een applicatie te ontwikkelen om al deze problemen op te lossen.

Het doel van de applicatie is om het management van defecten en taken te centraliseren voor de twee campussen, met ook de andere campussen van AP Hogeschool in gedachte.

Zo zullen de duplicaat meldingen drastisch verminderen en zullen er minder menselijke fouten plaatsvinden. Ook zal de applicatie het verwerken van de defecten en taken vergemakkelijken.

De applicatie vormt namelijk een systeem dat de werkdruk van de facilitaire-/logistieke dienst en campusverantwoordelijke verlicht.

Om dit te realiseren, heeft de applicatie de volgende functionaliteiten:

- Inloggen
- Overzicht pagina
- Aanmaken van defecten/taken
- Verwerking van defecten/taken
- Het toewijzen van defecten/taken
- Het exporteren van gewenste defecten-/takenlijst
- Het toewijzen van rollen
- Noodcontacten

Impact op de infrastructuur

Als eerste is het belangrijk te weten dat we ervan uitgaan dat er per dag ongeveer 5 a 10 defecten en/of opdrachten per dag ingediend zullen worden. Op basis van deze veronderstelling zullen volgende stellingen gedefinieerd zijn.

Hardware

Hieruit kunnen we vaststellen dat de bandbreedte en het datagebruik van het schoolnetwerk niet drastisch zullen stijgen. Dit omdat deze meldingen slechts maximum 10 tot 11 mb groot kunnen zijn. Dit voornamelijk dankzij een optionele foto.

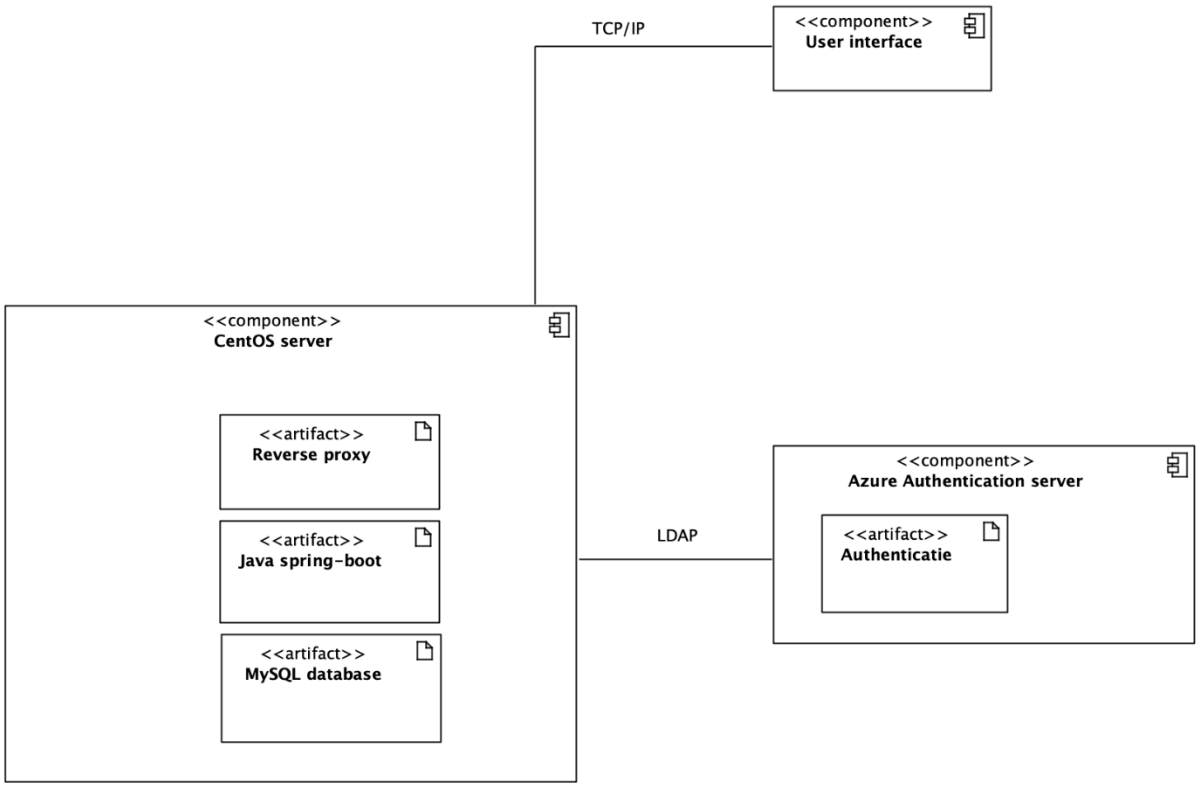
Stel dat we een worstcasescenario volgen. Elke dag gedurende 2 jaar worden er 10 defecten gemeld met elk een individuele grote van pakweg 12mb. Dit zou neerkomen op een cumulatieve grootte van 85,44 GB (12*10*356*2/1000). Naast dit hebben we de Linux CentOS server die een gemiddelde grootte van 11 a 21GB (10GB minimaal + 1GB voor de OS) zou innemen wat zorgt voor ongeveer 100GB opslag aan data in de server. Relatief gesproken is dit niet veel en zouden er dus geen nieuwe servers door AP aangekocht moeten worden.

Software

Voor de software maken we gebruik van opensource. Hieronder een lijst van gebruikte software.

1. Linux CentOS server
2. Docker
3. Docker-compose
4. GitLab runner
5. GitLab (webplatform)

Component/Deployment diagram



Release plan

Installatie Server

Wij gebruiken een Linux CentOS server voor het hosten van onze webapplicatie. Op deze server draaien wij diverse docker containers en maken we gebruik van docker-compose om zo een multi-container applicatie op te stellen. Daarnaast maken we gebruik van GitLab runner om onze CI/CD te configureren.

1. Installeer Linux CentOS server
    a. Versie: CentOS Linux 7 (Core)
    b. Kernel: Linux 3.10.0-1062.18.1.el7.x86_64
2. Installeer docker
    a. Versie: version 19.03.8
    b. Build: afacb8b
3. Installeer docker-compose
    a. Versie: 1.18.0
    b. Build: 8dd22a9
4. GitLab runner
    a. Versie: 12.9.0
    b. Build: 4c96e5ad

Docker containers

Wij maken gebruik van 3 verschillende docker containers. Op de eerste container draait de front-end onder een reverse-proxy. Hiervoor maken wij gebruik van Nginx. De tweede container draait de middleware, m.a.w. onze Java (spring-boot) app. Als laatste hebben we een container waarop een MySQL database staat. Al deze containers staan geconfigureerd in een handige docker-compose.yml file. Dit bestand staat in versie 3 en is meegeleverd in ons project.

1. Nginx
2. Java (spring-boot)
3. MySQL database

GitLab configuratie

Zoals eerder vermeld maken we gebruik van GitLab en GitLab-runner om een eenvoudige CI/CD pipeline te runnen. Op GitLab staan twee projecten die met elkaar verbonden zijn. Als eerste hebben we een front-end branche die gepulled wordt door de backend. Als tweede hebben we de back-end branche die de master branch via een pipeline pushed naar de server.

Zorg er zeker voor dat zowel de develop als master branch “protected” zijn.

1. Front-end
    a. Een develop branch
        i. Protected
    b. Een master branch
        i. Protected
2. Back-end
    a. Een develop branch
        i. Protected
    b. Een master branch
        i. Protected
    c. Gitmodules bestand (zit in project)
        i. Configureer de front-end als een submodule
    d. GitLab-ci.yml bestand met 4 fases (zit in project)
        i. Test
        ii. Prebuild
        iii. Build
        iv. Deploy

Daarnaast is het noodzakelijk om jouw SSH-key toe te voegen aan het project.

GitLab-runner

Als volgt configureren we onze GitLab runner op de server registreren.

GitLab

1. Ga naar GitLab
2. Ga naar het project en vervolgens naar ‘Settings’ > ‘CI/CD’
3. Klap hierbij de ‘Runners’ sectie open.
4. Sta vervolgens ‘shared Runners’ toe
5. Copy de ‘URL’ en de ‘Token’

Server (registreer de 'Runner')

1. Voer hetvolgende commando uit:
    a. 'sudo gitlab-runner register'
2. Plak hier de 'URL' die je eerder van GitLab hebt gekopiëerd.
3. Plak hier de 'Token' die je eerder van GitLab hebt gekopiëerd.
4. Geef een beschrijving voor de runner.
5. Nu vraagt het voor tags. Deze zijn niet nodig dus dit mag je blanco laten.
6. Een 'Runner executor' wordt gevraagt. Bij ons project hebben we gekozen voor 'docker'.
7. Aangezien we voor docker kozen, moeten we hier nog iets extra invullen. Geef hier de default image 'maven:latest' mee. Dit is noodzakelijk aangezien we dit niet meegeven in de "gitlab-ci.yml" file.

Indien dit proces correct verlopen is, zal je op GitLab nu jouw Runner erbij zien staan.

MySQL database

Als laatste configureren we onze database op de server. Zoals eerder vermeld gaat het docker-composy.yml bestand de MySQL container configureren. In deze file kan je ook jouw eigen administrator profiel aanpassen.

Bij de "environment" variabele kan je volgende 2 lijnen code aanpassen:

1. MYSQL_ROOT_USER: root
    a. Deze bepaalt de naam van de administrator/root gebruiker.
2. MYSQL_ROOT_PASSWORD: root
    a. Deze bepaalt het wachtwoord van desbetreffende gebruiker.

Het is aangeraden om enkel het wachtwoord aan te passen aangezien "root: root" niet een sterke beveiliging is en dit dus voor kwetsbaarheden in het systeem kan zorgen.

Als het project draait op de server gaat dit enkele errors geven. Dit is normaal, hij probeert de datamigraties uit te voeren naar de database terwijl de database nog niet gecreëerd is. We kunnen echter de database niet creëren voordat het project op z'n minst één keer gedraaid heeft.

Nadat het project gedraaid heeft moet je volgende stappen uitvoeren:

Ga naar de server en voer volgend commando in: "docker exec -it mysql-db mysql -uroot -p". Dit commando gaat de MySQL instantie aanroepen met de root gebruiker. Daarna geef je het wachtwoord in. Als laatste voer je het MySQL commando uit om de database aan te maken: "create database meldapp_db". Eens je dit uitgevoerd hebt kan je de terminal sluiten en het project opnieuw draaien. Deze keer gaan de datamigraties correct worden uitgevoerd en zou er geen enkele error plaats moeten vinden.

Opmerking

Het is mogelijk dat na verloop van tijd de pipeline van Master (back-end) naar de server niet draait wegens "onvoldoende ruimte". Indien dit het geval is moet je volgend commando op de VPS uitvoeren: "docker image rm $(docker images -q)". Dit verwijdert alle overbodige images en maakt zo ruimte vrij.

Technisch design

Front-end

Voor front-end hebben wij ReactJS gekozen. ReactJS is een opensource JavaScript library die gebruikt wordt voor het creëren van gebruikersinterfaces specifiek bedoeld voor SPA's. Dit laat ons toe om grote webapplicaties te creëren waarbij data verandert zonder de pagina zelf telkens te herladen. Enkel de desbetreffende componenten worden gewijzigd. (Nithin, 2017)

We hebben voor de front-end gebruik gemaakt van de IDE VSC. Dit bood ons zeer veel ondersteuning en was makkelijk te gebruiken voor de front-end. Daarnaast zijn we allemaal al vertrouwd met deze IDE.

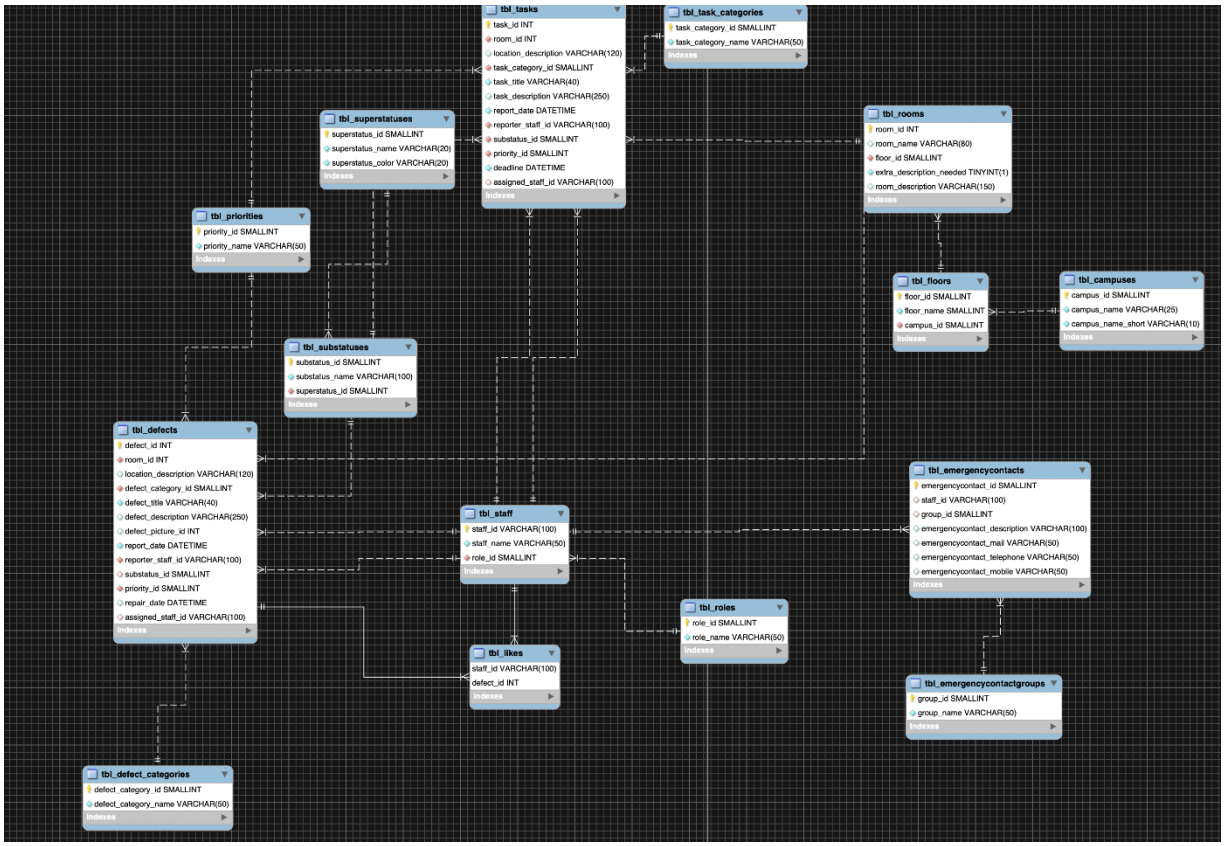
Back-end

Voor de back-end hebben we ervoor gekozen Java te gebruiken als programmeertaal. Dit op verzoek van onze klant. Java is een machine onafhankelijke taal en kan dus door eender welk apparaat gebruikt worden. Daarnaast is Java zeer toegankelijk en eenvoudig om mee te werken. Het maakt gebruik van Object-georiënteerd programmeren en laat toe uw code te hergebruiken. Bovendien heeft Java zeer veel en zeer goede documentatie ondersteund door een een grote community. Tot slot heeft het zeer grote libraries en wordt het door bijna alles ondersteund.

Voor het ontwikkelen van de applicatie maken we gebruik van Visual Studio Code als IDE. Dit programma is een ideale omgeving voor het ontwikkelen. Daarnaast vinden we de IntelliSense en de omgeving zeer gemakkelijk en gebruiksvriendelijk. Soms maakten we gebruik van IntelliJ IDEA aangezien deze specifiek voor java dient. Dit kwam soms erg van pas aangezien niet alle errors correct werden weergeven bij Visual Studio Code.

Database

We hebben onze database in MySQL gemaakt. MySQL is ideaal voor het bijhouden van een gestructureerde database voor webapplicaties doordat het gebruik maakt van een relationele database. Voor ons doel was dit dus uitermate geschikt. We hebben MySQL Workbench gebruikt voor het ontwikkelen van deze database. (EDUCBA., 2019) (Dataedo., 2019) (Klazar, 2017)



Testen

We hebben diverse testen uitgevoerd om ervoor te zorgen dat ons product kwalitatief in orde is.

Als eerste maakten we gebruik van JUnit om unit testen te schrijven. Dit deden we in VSC.

Daarnaast schreven we 2 geautomatiseerde testen voor de webpaginas “Defect melden” en “Opdracht melden”. Hiervoor gebruikten we Selenium Webdriver en Eclipse. De testen zijn geschreven in Java-versie 8.

Tot slot hebben we de performantie van de webapp getest met Google Lighthouse.

Externe systeeminterfaces

Interfaces

- Gebruiker
- Azure
- Applicatie (React-Single page app + spring boot API + MySQL Database)

Communicatie

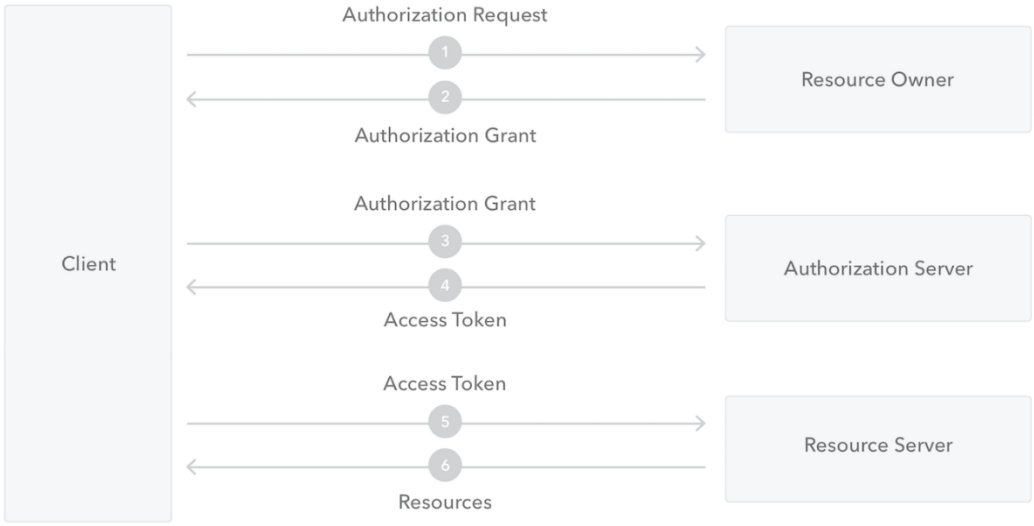
Bij het inloggen wordt de gebruiker doorgestuurd naar het inlog systeem van Azure. Hierbij wordt gebruik gemaakt van het OAuth 2.0 protocol. Vervolgens kan de gebruiker inloggen met zijn AP-inloggegevens. Op deze manier komt de applicatie nooit in aanraking komt met het wachtwoord van de gebruiker.

Eenmaal de gebruiker is ingelogd, krijgt de applicatie een token terug waar allerlei gegevens inzitten van de gebruiker. Hiermee wordt de gebruiker een rol gegeven in de applicatie wat ervoor zorgt dat de gebruiker enkel de geautoriseerde pagina's kan bezoeken en niet beland waar hij niet hoort.

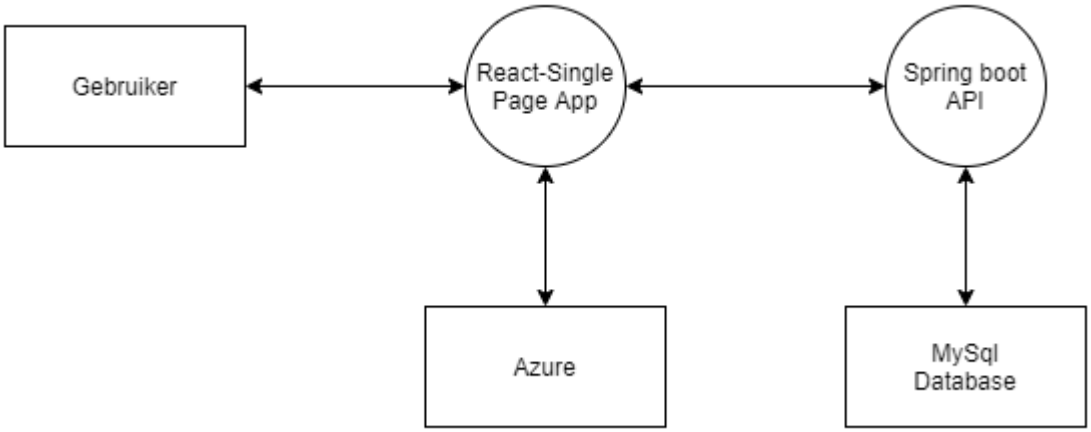
Protocollen

OAuth 2.0 protocol flow zoals te zien in onderstaande afbeelding

1. De applicatie vraagt autorisatie aan bij de "resource owner" om toegang te hebben tot de gegevens.
2. Als de "resource owner" de toegang autoriseert krijgt de applicatie een "authorization grant". Dit is een vergunning om toegang te hebben op de "authorization server".
3. De applicatie vraagt, door een "authorization grant" te geven en te authentiseren, een "access token" aan bij de "authorization server".
4. Als de applicatie wordt geautoriseerd en de "authorization grant" is geldig, stuurt de "authorization server" een "access token" naar de applicatie.
5. Nu kan de applicatie beveiligde gegevens opvragen bij de "resource server" met behulp van de "access token".
6. Als de "access token" geldig is, zal de "resource server" de gevraagde gegevens doorsturen naar de applicatie. (inc., 2020)



DFD



Datamigratie

Aangezien er geen huidig digitaal systeem is, zal er niet veel datamigratie moeten gebeuren. Enkel de openstaande defecten en opdrachten zullen in het systeem geplaatst moeten worden. We krijgen hiervoor een Excelbestand van de opdrachtgever, met daarin de nodige gegevens. Omdat dit niet veel defecten en opdrachten zijn, zullen we de data manueel in ons systeem zetten. Dit zal makkelijker en sneller zijn dan hier een script voor te schrijven.

De vaste gegevens worden met een migratiescript in de database gestoken. Dit wordt automatisch uitgevoerd eenmaal de back-end wordt gerund. De vaste gegevens bestaan uit: defectcategorieën, opdrachtcategorieën, locaties (campus, verdieping, lokaal), prioriteiten, rollen en statussen.

De noodcontacten hebben we echter manueel in de database gezet. Dit is niet inbegrepen in het migratiescript aangezien deze data meer kans heeft om te veranderen.

Security en autorisatierollen

Security

De authenticatie gebeurt met Azure AD via de authorization code flow. De authenticatie gebeurt in de back-end. Indien deze geslaagd is, zal er een token gestuurd worden die in de front-end wordt opgeslagen. Deze token wordt bij elke request naar de API meegestuurd. In deze token zit ook de rol van de gebruiker. Via die rol wordt bij elke request nagekeken of de gebruiker wel toegang heeft tot die request.

Hiernaast hebben we ervoor gezorgd dat gebruikers geen foutieve of ‘maliscious’ input kunnen geven op onze site. M.a.w. zijn zo de textfields beschermt tegen sql-injecties en dergelijke terwijl een bestand dat je kan indienen zo bijvoorbeeld enkel een afbeelding kan zijn.

Autorisatierollen

Er zijn 7 rollen in onze applicatie: Superadmin, Facilitaire Admin, Logistieke Admin, Facilitaire Medewerker, Logistieke Medewerken, Opleidingshoofd en Medewerker. Elke gebruiker heeft een rol in de applicatie. De eerste keer dat de gebruiker inlogt zal dit “Medewerker” zijn, buiten als dit al op voorhand is meegegeven en aangepast in de database.

Onze applicatie heeft een aantal pagina’s. Deze zijn niet voor iedereen beschikbaar: afhankelijk van de rol kan de gebruiker een aantal pagina’s wel of niet zien. Op onderstaande tabel ziet u een overzicht van de pagina’s en wie wat mag zien. Er zijn een paar uitzonderingen die u kan lezen onder de tabel.

Pagina	SA	FA	LA	FM	LM	OH	M
Login	X	X	X	X	X	X	X
Defect melden	X	X	X	X	X	X	X
Opdracht melden	X	X	X			X	
Overzicht	X	X	X	X[2]	X[2]	X	X[2]
Mijn meldingen	X	X	X	X	X	X	X
Mijn taken	X	X	X	X	X		
Toe te wijzen	X	X	X				
Rollen	X						
Noodcontacten	X	X	X	X	X	X	X
Defect details	X	X	X	X	X	X	X
Opdracht details	X	X	X		X	X	

1. Een externe gebruiker kan de loginpagina wel bekijken, maar zal niet kunnen inloggen.
2. Een facilitaire, logistieke en gewone medewerker zal bij overzicht enkel de defecten kunnen zien, niet de opdrachten.

Documentatie

Bij het opleveren van de applicatie aan de klant, zullen er verschillende documenten worden meegegeven. Dit zal het gebruik van de applicatie gemakkelijk en aangenaam maken.

Documentatie code

De documentatie voor onze back-end is in Javadoc, voor de front-end wordt dit in een Markdown bestand voorzien.

Zo is er per stukje code uitleg over zijn functie, gebruik en doel.

Hiermee willen we ervoor zorgen dat onze opvolgers direct aan de code kunnen beginnen zonder al te veel tijd te verliezen aan het begrijpen van de bestaande code.

Handleiding

Voor de user manual hebben we filmpjes voorzien met bijhorende teksten die uitleggen hoe u de applicatie kan gebruiken. De filmpjes staan beschikbaar op YouTube en zijn enkel via een link bereikbaar.

Testplan

In het testplan staat beschreven welke testen uitgevoerd zouden worden en waarom.

Test document

In het test document staat beschreven welke testen we hebben uitgevoerd, wie deze testen heeft uitgevoerd en wanneer. Ook het resultaat van deze testen is duidelijk vermeld.

Bibliography

Dataedo. (2019). *dataedo.com*. Retrieved from Generate documentation for MySQL database in 5: https://dataedo.com/tutorials/mysql/generate-documentation-for-mysql-database

EDUCBA. (2019). *MySQL vs NoSQL - Which One Is More*. Retrieved from www.educba.com: https://www.educba.com/mysql-vs-nosql/

Klazar, R. (2017, December 13). *medium.com*. Retrieved from Three Reasons To Use a NoSQL Document Store for Your Next Web Application: https://medium.com/statuscode/three-reasons-to-use-a-nosql-document-store-for-your-next-web-application-6b9eabffc8d8

Nithin, P. (2017, May 26). *What Is ReactJS and Why Should We Use*. Retrieved from c-sharpcorner: https://www.c-sharpcorner.com/article/what-and-why-reactjs/