Software guidebook

Het software guidebook is bedoeld als een handleiding voor developers die duidelijk maakt hoe het project is opgebouwd. Zo kan elke developer verder met dit project ongeacht uitleg van projectleden. Dit software guidebook is opgesteld aan de hand van de reader software guidebook (Tijsma et al., 2016).

Inhoudsopgave

1. Context

In dit hoofdstuk wordt beschreven wat de context van de applicatie is, wie er gebruik van gaat maken en waar de applicatie uit bestaat.

1.1 Het project

Het team heeft de opdracht gekregen om te kijken naar de digitalisering van het rekenlogboek van basisschool de Wamel.

1.2 Huidige situatie

Voor elk blok zijn er bepaalde leerdoelen bepaald door de rekenmethode. Over deze leerdoelen wordt een pretoets afgenomen. Na het afnemen van de pretoets vullen de leerlingen een logboek in waarin wordt gevraagd, per leerdoel, of ze het snapten.

Op basis van de reactie van de leerling krijgt die een instructieles hierover. Na deze instructieles wordt nog maals voor dat specifieke leerdoel gevraagd of de leerling het nu wel snapt.

Op dit moment vullen de leerlingen de rekenlogboeken schriftelijk in. Hierdoor moet de leerkracht elke keer alle logboeken langs om de voortgang van de leerlingen te zien en moeten de logboeken er telkens worden bijgepakt om in te vullen.

De bedoeling is dat het rekenlogboek online kan worden ingevuld via Microsoft Teams en ook dat leerlingen automatisch worden toegevoegd worden aan de instructielessen.

1.3 De applicatie

De applicatie wordt gebouwd als ondersteuning voor het rekenlogboek dat gebruikt wordt in groep 5 tot en met 8 van basisschool de Wamel om leerlingen te laten inschatten waar ze extra mee nodig hebben.

Het bestaat uit een omgeving voor de logboekontwerper om logboeken te maken/klaarzetten voor een groep en blok. En een gedeelte binnen een Microsoft Teams tab waar de leerling het logboek kan invoeren. Ook kunnen leraren bekijken wat leerlingen hebben ingevuld.

1.4 Rollen

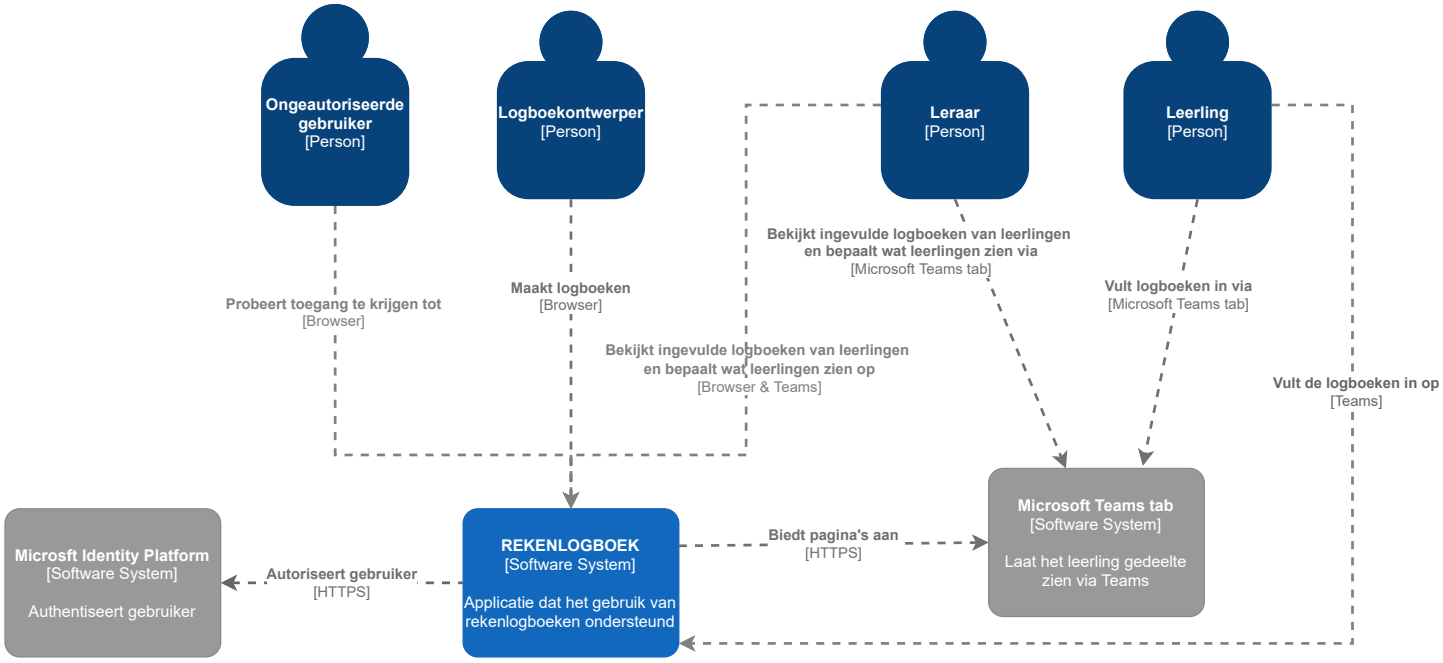
Leraren en leerlingen zijn de gebruikers van de applicatie.
Hieronder een tabel met taken per rol.

Rol	Taken
Leraar	Evalueert de logboeken van leerlingen
	Bepaalt wat de leerlingen te zien krijgen
	Bekijken van (oude) logboekresultaten
Leerling	studentlogboeken invullen (na pre-toets, instructies en evaluatie pagina's)
Logboekontwerper	Maakt logboeken voor verschillende groepen en blokken

Tabel 1: Rollen verdeling.

1.4 Context diagram

Op de afbeelding hieronder is het context diagram te zien, gebaseerd op het C4 model.



Afbeelding 1: Context diagram.

Het blauwe blok geeft de webapplicatie aan die het ontwikkelteam gemaakt heeft. Deze applicatie maakt gebruik van het Microsoft Identity Platform om gebruikers te autoriseren en van een Microsoft Teams Tab om de applicatie werkend te krijgen op Microsoft Teams applicatie.

Verder zijn er nog 4 verschillende gebruikers. Voor de compleetheid staan ze al weergegeven in het context diagram, maar ze zullen in het container en component diagram pas echt van belang worden voor de c4-modellen.

2 Functionele overzicht

Dit hoofdstuk bevat een functioneel overzicht van de gehele applicatie per gebruiker.

2.1 Logboekontwerper

Een logboekontwerper moet in staat zijn om een nieuw logboek aan te maken en die te koppelen aan een groep en blok. Daarna moet de logboekontwerper in staat zijn de verschillende kolomnamen, invulmethoden en leerdoelen in te vullen.

2.1.1 User stories

Hieronder staan de user stories met betrekking tot de logboekontwerper. Niet alle user stories zullen worden uitgevoerd, maar geven wel een goed idee wat de verschillende gebruikers willen.

- Als logboekontwerper wil ik een logboek kunnen aanmaken zodat ik de leerkracht kan helpen met het voorbereiden van een blok

- Als logboekontwerper wil ik per blok leerdoelen kunnen toevoegen zodat ik de leerkracht kan helpen met het voorbereiden van een blok
- Als logboekontwerper wil ik een logboek een aan leerjaar kunnen koppelen zodat ik een logboek vaker kan gebruiken

2.2 Leraar

Een leraar moet een logboek kunnen aanpassen als er een fout instaat. De leraar moet ook live mee kunnen kijken wat zijn/haar leerlingen in hun logboek invullen om hulp te kunnen geven aan degene die dat nodig hebben.

2.2.1 User stories

Hieronder staan de user stories met betrekking tot de leraar. Niet alle user stories zullen worden uitgevoerd, maar geven wel een goed idee wat de verschillende gebruikers willen.

- Als leerkracht wil ik een logboek kunnen toevoegen aan mijn teams omgeving zodat ik gebruik kan maken van het logboek
- Als leerkracht wil ik een snel en makkelijk overzicht van de ingevulde logboeken zodat ik in de les snel kan kijken of een leerling assistentie nodig heeft
- Als leerkracht wil ik aangeven welke pagina de leerling op dat moment ziet zodat ik de leerling de vragen kan laten beantwoorden
- Als leerkracht wil ik een groepsoverzicht per leerdoel zodat ik kan zien welke leerlingen instructie nodig hebben

2.3 Leerling

Een leerling moet via het team van zijn groep op een tab kunnen drukken en automatisch terechtkomen op de juiste pagina om het logboek in te vullen. Alleen leerdoelen die op dat moment van toepassing zijn worden op dat moment getoond aan de leerling. Wanneer er geen logboek beschikbaar is moet een leerling hier een aparte pagina voor te zien krijgen.

2.3.1 User stories

Hieronder staan de user stories met betrekking tot de leerling. Niet alle user stories zullen worden uitgevoerd, maar geven wel een goed idee wat de verschillende gebruikers willen.

- Als leerling wil ik in mijn bestaande Microsoft Teams omgeving gebruik maken van de applicatie zodat ik geen nieuwe programma's en accounts hoef te gebruiken
- Als leerling wil ik kunnen invullen hoe ik de toets heb gemaakt zodat ik aan mijn leerkracht kan aangeven hoe de toets ging
- Als leerling wil ik na de rekengesprekken in tweetallen kunnen aangeven of ik instructie nodig heb per leerdoel uit de pretoets
- Als leerling wil ik na een rekenles de les kunnen evalueren in het logboek

3 Kwaliteitsattributen

In dit hoofdstuk bevinden zich de verschillende kwaliteitseisen waaraan de software moet voldoen.

3.1 Data voor de applicatie

De applicatie heeft leerdoelen nodig, maar die worden uit de rekenmethode gehaald die op dit moment gebruikt wordt. Het team is niet verantwoordelijk voor het aanleveren of beschikbaar stellen van deze leerdoelen, daarnaast is het team ook niet verantwoordelijk voor afbeeldingen voor het lesmateriaal en vooraf gemaakte logboeken. Wij leveren geen database mee, alleen maar de schemas daarvoor zodat de applicatie de data juist kan opslaan in een MongoDB database.

3.2 Documentatie

Documentatie dat gefocused is op de eindgebruiker valt buiten de scope van dit project en zal daarom ook niet gemaakt worden. Wel wordt er documentatie gemaakt voor de product owner (plan van aanpak) en mede ontwikkelaars (het software guidebook). Hierdoor kan een volgend ontwikkelteam verder op de applicatie die het huidige team opgezet heeft.

3.3 Talen en wetgeving

De documentatie en user-interface wordt gemaakt in het Nederlands, de code zelf (en behorende comments) in het Engels.
Namen van componenten en andere code-specifieke onderdelen kunnen wel in het Engels weergegeven worden om de consistentie met de code te behouden. Hierdoor kan een nieuw ontwikkelteam ook gemakkelijker de documentatie en code als een geheel zien.

Er zal geen rekening worden gehouden met (data)wetgeving omdat dit buiten de scope van het project valt en de ontwikkeling te erg zal vertragen.

4. Beperkingen

In dit hoofdstuk worden de beperkingen van dit project opgenoemd. Veel van deze beperkingen vloeien voort vanuit het feit dat dit een schoolproject is.

4.1 Tijd en budget

Het team is beperkt tot 3 sprints van twee weken voor dit project en nog een pre game week (voor plan van aanpak) en een post game week (ter afronding van code en aanvulling van documentatie).

Ook zal er voor schoolopdrachten tijd moeten worden vrijgemaakt. Zo moet elk teamlid een logboek bijhouden, leerdoelen voor zichzelf zetten en per sprint een retrospective invullen.

Omdat de product owner een leraar is, zal deze niet de gehele dag beschikbaar zijn omdat hij/zij ook andere klassen moet onderwijzen. Daarnaast is de product owner alleen beschikbaar tijdens schooltijden.

Alles binnen dit project kan worden gedaan met gratis software of een proefabonnement en hiervan zal ook optimaal gebruik worden gemaakt, zodat wij kunnen testen op een manier die zoveel mogelijk lijkt op de uiteindelijke situatie voor een gebruiker.

4.2 Technologieën

Omdat dit een schoolopdracht is, is het gebruik van React-Redux met een Node backend verplicht. Dit is een beperking, maar omdat het hele team voor dit project een course heeft gehad over het gebruik van deze technologieën zal dit niet al te erg zijn. Daarnaast is voor de database een MongoDB database gekozen, ook dit was vanuit de HAN een verplichting en vanwege onze kennis kunnen we daar ook mee overweg.

Een andere technologie is de Microsoft Teams API waar het team nog niet intensief mee heeft gewerkt. Er is een onderzoek gedaan naar het gebruiken van Teams en dit onderzoek is beschikbaar op de repo van het project.

Hetzelfde geldt eigenlijk voor de Microsoft Authentication Framework waarmee gebruikers moeten inloggen. Hierover is een onderzoek gedaan wat beschikbaar is in de repo.

4.3 SCRUM

Tijdens het project zal gebruik worden gemaakt van SCRUM. Dit betekent dat elke ochtend een daily stand-up wordt gehouden en elke sprint een planning, review en retrospective. Dit zijn momenten die het team niet kan besteden aan ontwikkelen, maar zijn wel waardevolle momenten omdat deze ervoor zorgen dat het proces goed gaat.

4.4 Definition of done

Tijdens het project zal er ook gebruik worden gemaakt van een definition of done waarin staat waar werk aan moet voldoen voordat het af is. Dit is ook een beperking omdat al het werk moet voldoen aan deze definition of done. Maar de definition of done helpt wel met het hoog houden van de kwaliteit van het werk.

5 Principes

Onze principes staan beschreven in de definition of done. Dit bestand wordt tijdens de sprints ook bijgewerkt indien dit nodig is. Ook maken we gebruik van KISS of Keep It Simple Stupid, en DRY of Don't Repeat Yourself als software principes.

6 Software architectuur

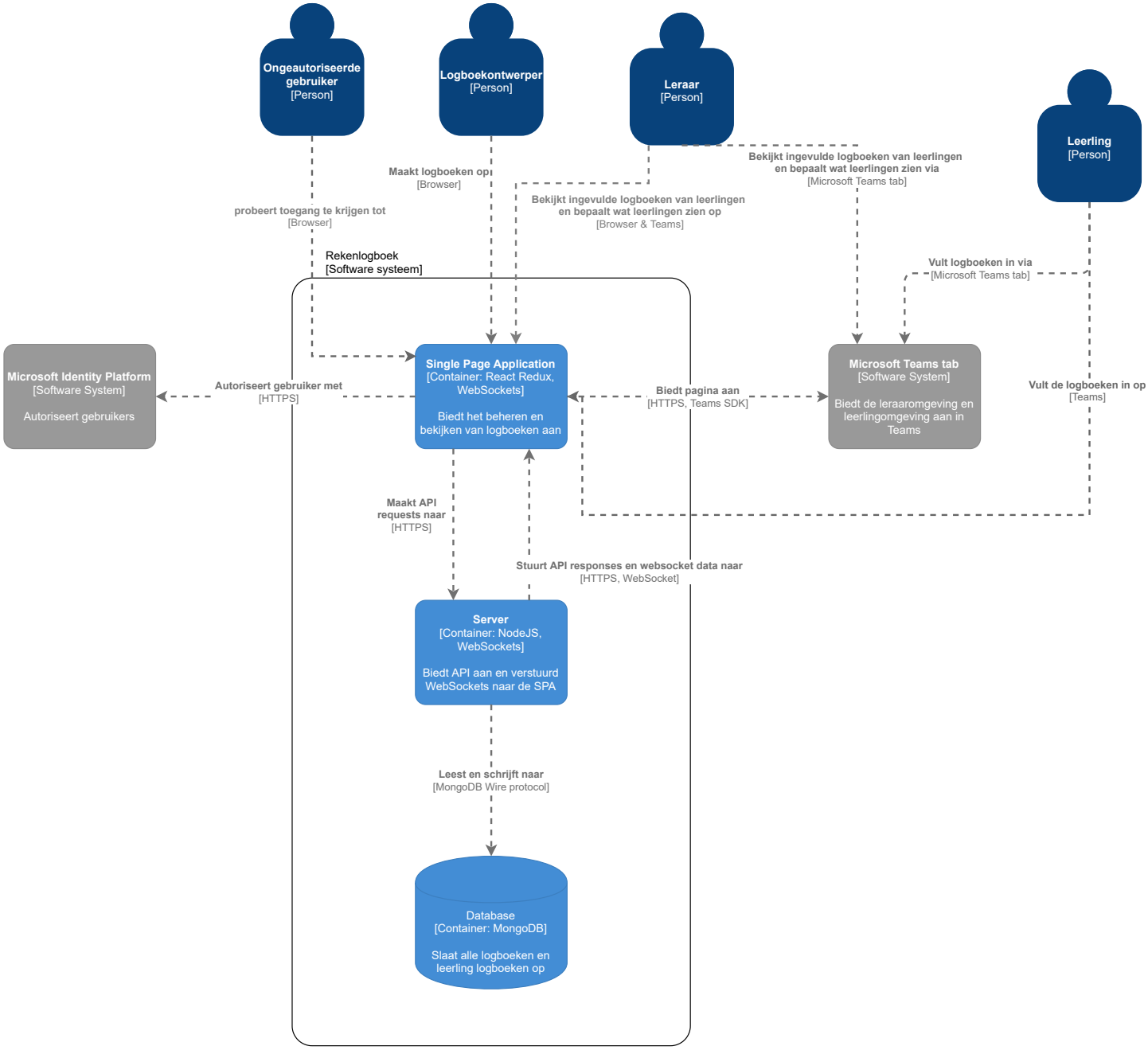
Dit hoofdstuk gaat over de "big picture" van de software architectuur, dit laten we zien aan de hand van de container en component diagrammen van het C4 model

6.1 Diagrammen

Hieronder staan 3 verschillende diagrammen die de software architectuur duidelijk zullen moeten maken.

6.1.1 Container diagram

In de afbeelding hieronder is het container diagram te zien, gebaseerd op het C4 model.



Afbeelding 2: Container diagram.

In het container diagram is te zien hoe de verschillende systemen en gebruikers met elkaar communiceren. Het systeem bestaat uit 3 containers: de Single Page Application, de Server en een database.

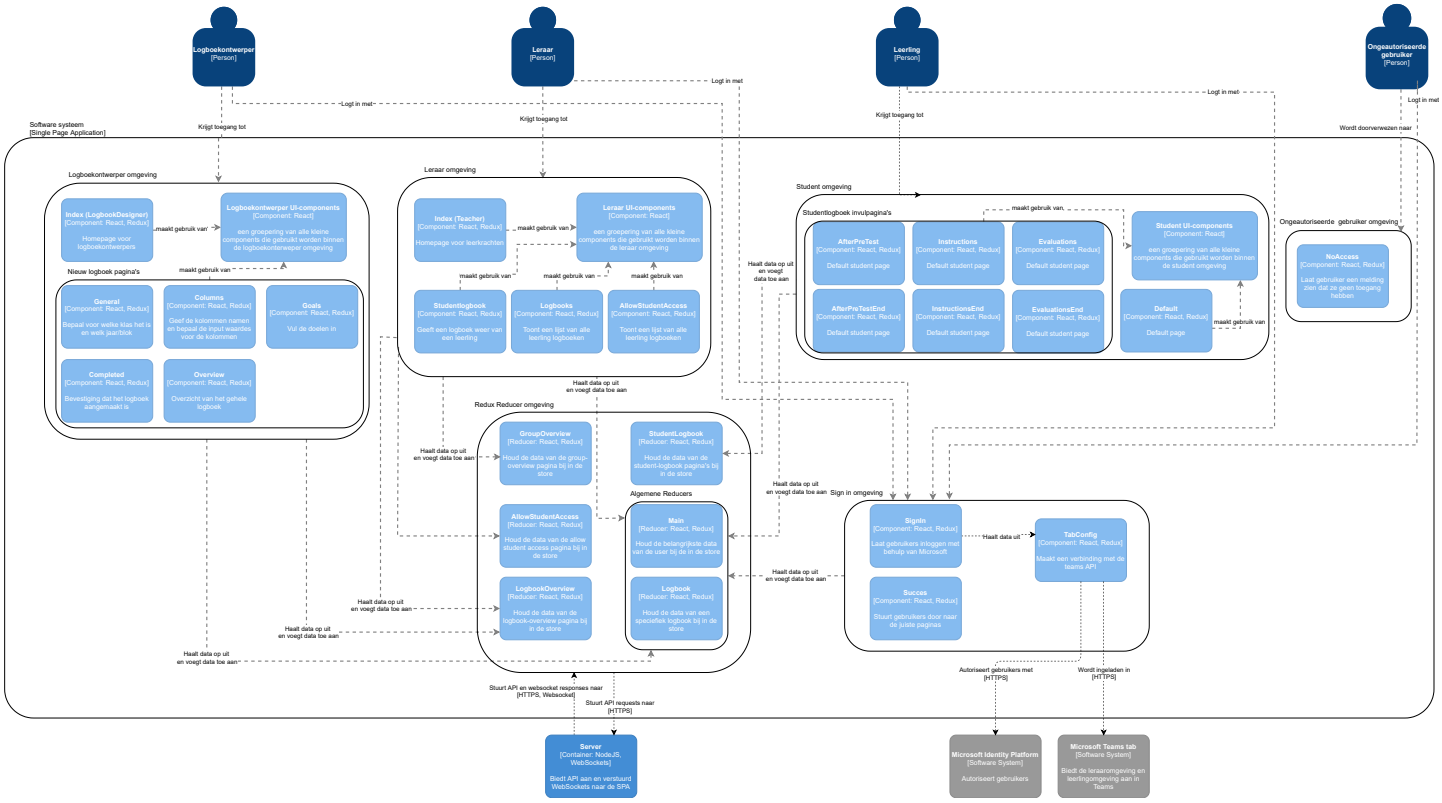
De Single Page Application is een React app die gebruikt maakt van Websockets en Redux. Alhoewel de SPA zelf geen data verstuurd via de websockets ontvangt hij dit wel vanuit de server. Wat de SPA wel verstuurd naar de server zijn de API requests via fetch. De server kan vervolgens met data via HTTPS of websockets een response geven. Wanneer er een PUT/POST request werd verstuurd naar de server zal de server een response geven op basis van de status van het request.

Verder is hier te zien dat de leeraar en leerling gebruikers niet alleen direct via de SPA werken, maar dat zij ook de optie hebben om vanuit de Microsoft Teams tab de applicatie kunnen benaderen.

In de component diagram Single Page Application staan meer details over de verschillen tussen gebruikers.

6.1.2 Component diagram: Single Page Application

In de afbeelding hieronder is het component diagram voor de Single Page Application te zien (ook wel de client-side genoemd), gebaseerd op het C4 model.



Afbeelding 3: Component diagram - Single Page Application.

Aangezien de SPA redelijk uitgebreid is kwam het team al snel tot de conclusie om hier meerdere groeperingen te maken binnen het component diagram. Dit zorgt ervoor dat er veel minder pijlen nodig zijn en daardoor het diagram ook een stuk overzichtelijker is.

De groeperingen zijn vooral gebasseerd op welke omgevingen er binnen de applicatie beschikbaar zijn. Deze omgevingen geven onder andere aan voor welke gebruikers ze bedoeld zijn of voor welke functie ze dienen.

Alle gebruikers hebben dus een eigen omgeving waarvoor specifieke component(s) gemaakt zijn. De logboekontwerper, leraar en student omgevingen hebben ook allemaal hun eigen UI-components. Dit is een groepering aan kleinere components die per pagina ingeladen kunnen worden. Ze zijn echter niet van heel groot belang voor het component diagram en voor het overzicht hebben we ze daarom ook weggelaten.

De omgeving die er dan uitspringt is die van de ongeautoriseerde gebruiker. Aangezien gebruikers zonder de juiste rechten niet toegestaan zijn tot de belangrijke delen van de applicatie, worden ze doorgeleid naar een 'geen toegang' pagina.

Verder hebben de andere drie gebruikers dus allemaal belangrijke pagina's. Hieronder een simpel overzicht per gebruiker wat deze pagina's en hun bijbehorende subgroepen inhouden.

Logboekontwerper

- Index (LogbookDesigner): de hoofdpagina voor de logboekontwerper. Vanuit hier kan hij/zij kiezen om een nieuw logboek aan te maken.
- Subgroep "Nieuw logboek pagina's": de pagina's die in deze subgroep staan moeten door een logboekontwerper worden doorlopen om een nieuw logboek te kunnen maken.

Leraar

- Index (Teacher): vanuit hier kan een leraar een overzicht zien van zijn/haar mogelijkheden binnen de applicatie. De leraar kan kiezen om studentenlogboeken in te zien, een groepsovericht te zien of om te bepalen welke pagina's beschikbaar zijn voor zijn/haar groep.
- Studentlogbook: de pagina waarin de leraar een logboek van zijn/haar leerlingen kan kiezen en inzien.
- Logbooks: de pagina waarin een overzicht staat van alle leerling logboeken.
- AllowStudentAccess: de pagina waarin de leraar kan aangeven wat de leerlingen van zijn/haar groep mogen zien.

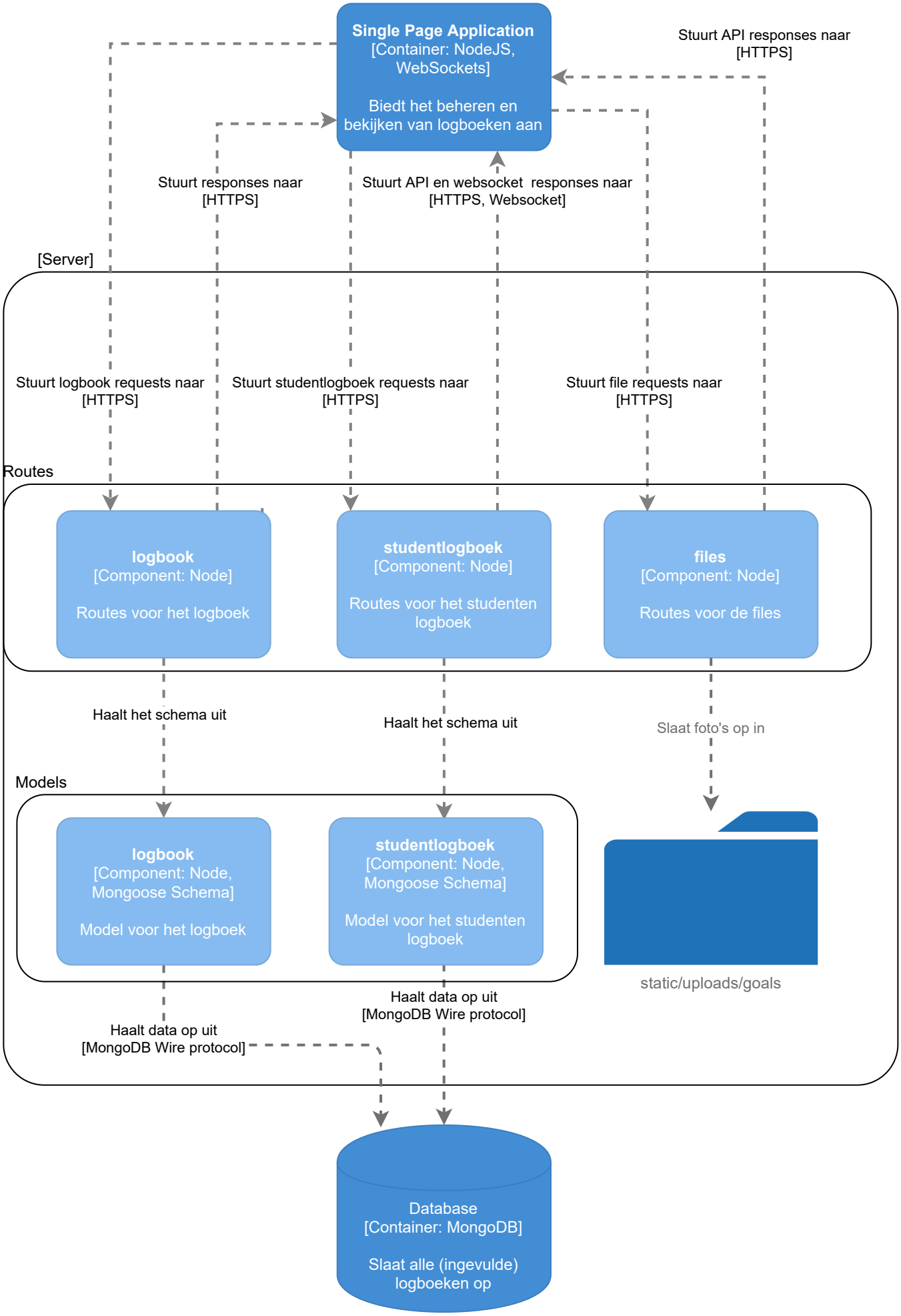
Student

- Subgroep "Studentlogboek invulpagina's": in deze subgroep staan alle pagina's die opengezet kunnen worden door de leraar van een leerling, met een bijbehorende eind pagina voor wanneer een leerling klaar is met het invullen van zijn/haar logboek.
- Default: wanneer een leraar geen logboek heeft openstaan zullen de leerlingen dit zien aan de hand van deze infopagina.

Naast de omgevingen die de gebruikers visueel kunnen zien is er ook nog een Redux en een Sign in omgeving. De Redux omgeving bevat in dit overzicht alle reducers en per reducer wordt er met pijlen aangegeven door wie ze gebruikt worden. De Sign in omgeving bevat alleen een Signin en succespagina voor het inloggen. De Sign in pagina maakt gebruik van Microsoft Teams data via een component genaamd tabConfig. Tabconfig vraagt namelijk data op uit het Microsoft Identity Platform. De Microsoft Teams tab die ook aan de tabConfig gekoppeld staat geld alleen voor de leraren en leerlingen die op Teams werken. In principe heeft dit direct effect op hun complete omgeving, maar omdat in de sign in omgeving de autorisatie wordt gedaan voor een gebruiker en wordt bepaald of ze op Microsoft Teams werken of niet leek het ons handig de Teams Tab hieraan te koppelen.

6.1.3 Component diagram: Server

In de afbeelding hieronder is het component diagram voor de Server te zien, gebaseerd op het C4 model.



Afbeelding 4: Component diagram - Server.

in "6.1.2 Component diagram: Single Page Application" staat beschreven hoe de client-side eruit ziet, maar het is natuurlijk ook belangrijk om te weten hoe de client-side met de server-side communiceerd en hoe deze in elkaar zit. Om dit op te lossen hebben we een 2de component diagram gemaakt voor de server-side. Zoals u kan zien in zijn hier weer 2 groeperingen gemaakt voor een duidelijk overzicht.

De routes groepering bevat alle mogelijke route components waar de client-side requests naar kan sturen. De logbook en studentlogbook components kunnen vervolgens de schemas aanspreken uit de bijbehorende model files. Al is een request vanuit de client-side geldig dan kan er via het MongoDB Wire protocol een verzoek om data gedaan worden naar de MongoDB.

De file route is de enige route file zonder eigen schema. Dit komt, omdat deze file ook niks van de database vraagt. Hij slaat namelijk zelf images op op de server. De path naar deze images wordt door een andere request in de logbook route file afgehandeld.

Als laatst is het ook nog belangrijk om de verbinding met de client-side even kort aan te kaarten. Alle requests die via de client-side naar de server-sider gaan lopen over HTTPS en worden aangeroepen vanuit fetch. De responses van de server kunnen over zowel HTTPS als websockets zijn. Alle responses, inclusief de status codes worden terug gestuurd in een JSON format. De enigste route component dia websockets responses terugstuurt is de studentlogbook. Dit is namelijk van belang, omdat leraren zo direct kunnen zien wanneer logboeken worden gepdate

6.2 Bestandsstructuur

In onze project map zijn er verschillende mappen, namelijk:

- docs, hierin zitten de documenten die met het project te maken hebben
- opdracht-rekenlogboek, wat de opdracht bevat die we uitvoeren voor de PO
- rekenlogboek, dit bevat de code van zowel de client-side als de server-side

Binnen in de map `rekenlogboek` staat de client-side en server-side app in twee aparte folders met elk hun eigen node modules, EsLint, Prettier, Etcetera.

De client-side is opgedeeld in verschillende mappen voor verschillende onderdelen, met het mapje `src/js` de javascript code en de `src/redux` map alles voor de reducers bevattend. Afbeeldingen worden in het mapje `src/img` gestopt. Het hoofdbestand is het bestand App.js waarin de router staat.

```
/src
    /img    -> Bevat afbeeldingen voor de paginas.
    /js     -> Bevat de containers en componenten voor de pagina's.
        /redux  -> Bevat de Redux store, reducers en action creators.
    /scss   -> Bevat de styling voor de paginas.
```

De server-side is opgedeeld in een map voor mongoose models en express routes. Het hoofdbestand is het bestand app.js.

```
/models                 ->Hier staan de database models
/routes                 ->Hier worden de endpoints beschreven
/static/uploads/goals   ->Hier worden afbeeldingen voor de goals naar toe geüpload
app.js                  ->Hoofdbestand waarmee de server gestart kan worden
```

Afbeeldingen worden opgeslagen en geserved onder de `static` folder op de server, die een map bevat met uploads waarin bestanden geupload door gebruikers staat. Deze uploads maken gebruik van express-files en de `files` route. De path van de afbeelding wordt opgeslagen in de database base en kan worden opgehaald door de URL van de server door bijv. `http://localhost:3000` ervoor te zetten.

6.3 Real-time

In onze applicatie wordt gebruik gemaakt van real-time updates in het logboekoverzicht van de leraar. Wanneer een leerling een antwoord verandert of toevoegt wordt er een WebSocket bericht met gestuurd naar de leraar via `Socket.io` waarin het id van het geüpdatete studentenlogboek, het id van het logboek waar deze bij hoort en de naam van de student wordt meegegeven. Deze informatie is nodig om te bepalen of er data gefetcht moet worden door de leraar. Dit Websocket bericht kan door de leraar worden opgevangen in het groepsoverzicht, tijdens het bekijken van antwoorden en tijdens het bekijken van hele logboeken. Als de nieuwe data relevant is voor de leraar fetcht hij/zij alle nieuwe data die nodig is.

6.4 Bekende problemen

Onze app heeft een aantal bekende problemen die niet zijn opgelost tijdens de sprints of postgame:

- gebruik van dark mode in Teams veranderd niks
- privacy & terms of use pagina zijn verplicht voor Teams apps maar die hebben we niet
- De app werkt niet in de Teams App maar wel als je Teams gebruikt in de browser

7 Infrastructuur architectuur

In dit hoofdstuk wordt de infrastructuur getoond van de applicatie. Bij het maken van dit project wordt gebruik gemaakt van Git, dus als er iets misgaat kan met Git de data hersteld worden.

7.1 Lokale ontwikkelomgeving

Tijdens het ontwikkelen van de applicatie, wordt er gebruik gemaakt van een lokale ontwikkelomgeving. De applicatie zal draaien op `localhost` en zal live updaten wanneer er veranderingen worden gemaakt via een Webpack server. De client-side en server-side zullen gebruik maken van twee verschillende poorten. Het testen gebeurt ook binnen de lokale ontwikkelomgeving.

Het werken met Teams in een lokale ontwikkelomgeving is beschreven in het Teams integratie onderzoek.

Voordat nieuwe features gemerged worden in de development branche worden deze aan testen ondervonden. In onderstaande alinea's wordt er ingegaan op de verschillende testen die gehanteerd worden. Alle testen wordten gedraaid met behulp van het test framework Jest.

7.1.1 Unit-testen

De unit-testen bevinden zich op de server-side van de applicatie. Hier worden de verschillende routes van de API getest.

De unit-testen kunnen gestart worden op de back-end server door de dbName in app.js te veranderen naar `testrekenlogboek` en met het volgende commando `$ npm start test` uit te voeren.

7.1.2 E2E-testen

De client-side van de applicatie wordt ook getest door middel van end-to-end testen (e2e). Om deze testen te kunnen draaien wordt er gebruik gemaakt van Puppeteer (https://pptr.dev/) i.c.m. met Jest.

Puppeteer maakt het mogelijk om bepaalde flows binnen de applicatie geautomatiseerd te testen via de Chromium browser. In onderstaande tabel bevinden zich de verschillende testen.

Bij sommige testen bevind zich ook een seed file. Deze seed files bevinden zich op de back-end server en zijn nodig om de test te laten slagen. Seed files kunnen geinstalleerd worden door het bijgevoegde commando in te voeren op de back-end.

#	Test	Omschrijving	Uitvoeren test
1.	Inloggen met Microsoft	Automatisch inloggen op Microsoft via applicatie	client: `$ npm run e2e-auth`
2.	Aanmaken nieuw logboek	Aanmaken nieuw logboek	client: `$ npm run e2e-new-logbook`
3.	Bekijken groepsoverzicht	Bekijken van een groepsoverzicht + websocket implementatie i.c.m. studentenpagina's.	client: `$ npm run e2e-group-overview` server: `$ node seeds/e2e/group-overview.js`

#	Test	Omschrijving	Uitvoeren test
4.	Bekijken studentenlogboeken	Inzien van logboeken per student.	client: `$ npm run e2e-student-logbook` server: `$ node seeds/e2e/student-logbook.js`
5.	Wijzigen studententoegang	Wijzig het formulier waar de student toegang tot heeft.	client: `$ npm run e2e-student-access` server: `$ node seeds/e2e/student-access.js`
6.	Studentpagina's	Testen van het invullen van invoervelden en versturen van vragen.	client: `$ npm run e2e-student-pages` server: `$ node seeds/e2e/student-pages.js`
7.	Inloggen app via teams	Inloggen via teams omgeving en rekenlogboek openen	client: `$ npm run e2e-teams-login`

Tabel 2: E2E-test overzicht.

7.2 Deployed ontwikkelomgeving

Het was wenselijk om het project te deployen op een Azure server, helaas is het het team niet gelukt om dit voor elkaar te krijgen. Mocht een volgend ontwikkelteam dit alsnog voor elkaar willen krijgen dan staat voor hun in het volgende hoofdstuk een deployment diagram die weergeeft hoe het geheel gedeployed zou kunnen worden via Azure.

Daarnaast wordt ook beschreven hoe de server-side bijvoorbeeld op een VPS kan draaien met Ubuntu 20.04.

Het ondersteunen en onderhouden van de applicatie tijdens deployment is niet de verantwoordelijkheid van het team, zoals beschreven in hoofdstuk 4.

8 Deployment

In dit hoofdstuk wordt besproken hoe de applicatie precies wordt gedeployed. De applicatie bestaat eigenlijk uit 2 delen die apart zouden moeten worden gedeployed: de client-side en de server-side.

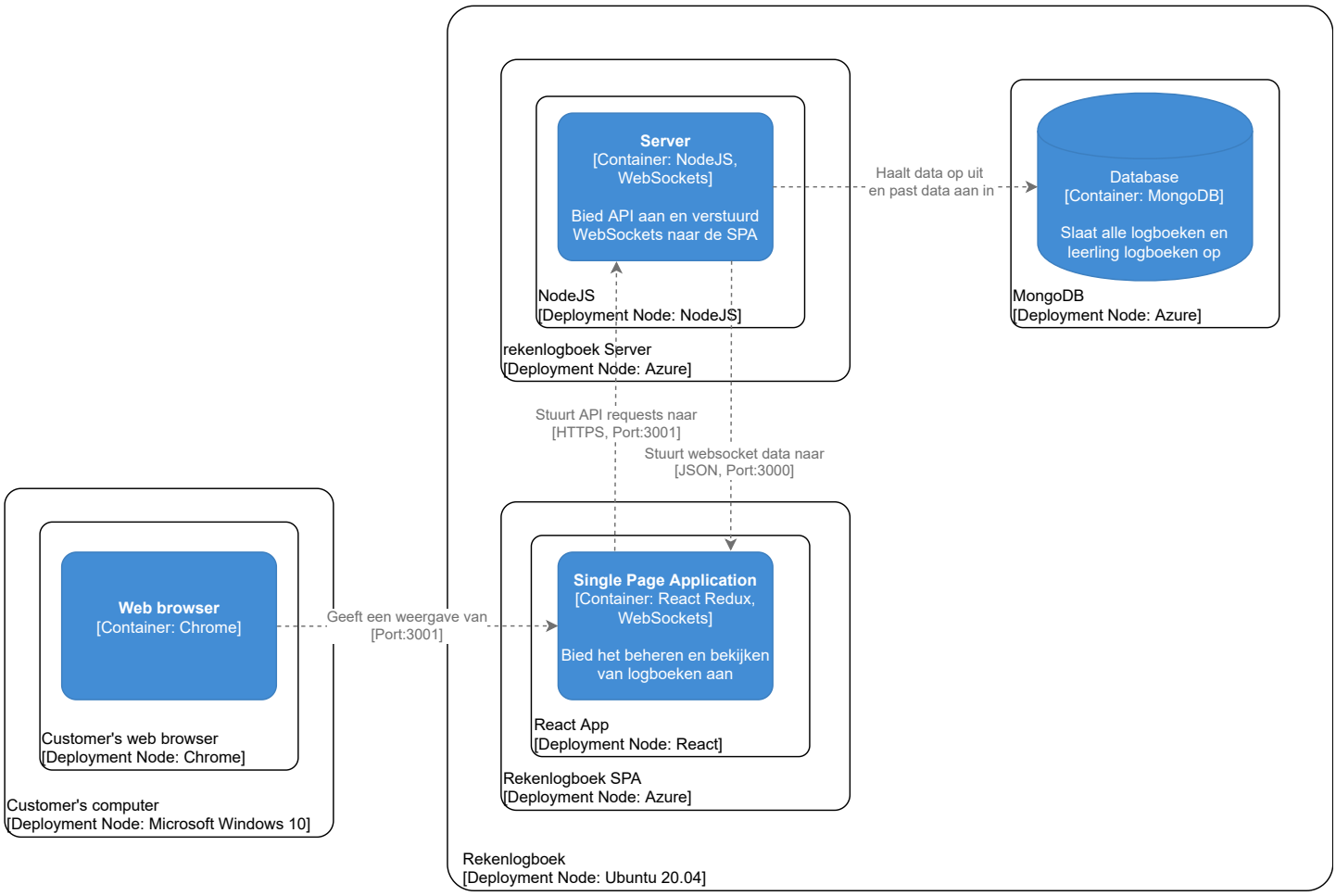
Om de applicatie te deployen moeten de volgende stappen worden uitgevoerd in elk van de folders (dat zijn dus rekenlogboek/client-side en rekenlogboek/server-side):

- npm install

- npm start

8.1 Deployment diagram

Hieronder staat het deployment diagram waarin weergeven wordt hoe het project kan worden gebruikt.



Afbeelding 5: Deployment diagram.

Aan het begin van het project kreeg het ontwikkelteam te horen dat het wenselijk was om via Azure het project te deployen. Alhoewel het ons zelf niet gelukt is om dit realistisch te maken hebben we ervoor gekozen om het deployment diagram alsnog op basis van een Azure-omgeving weer te geven zodat een toekomstige developer hier mogelijk gebruik van kan maken.

Per container staat in het deployment diagram weergegeven waar ze voor bedoeld zijn en waar ze "op draaien". Het gedeelte dat door ons gemaakt is valt binnen de groepering "rekenlogboek". Dit houd dus in dat wij aan het einde van het project een client-side, server-side en de bijbehorende database schemas opleveren. De web browser, database host en server host zijn echter onderdelen waar wij geen verdere verantwoording over hebben en deze zijn alleen verwerkt in het deployment diagram voor een duidelijk overzicht.

8.2 Deployen van de Microsoft Teams app

Microsoft Teams is een dominante speler binnnen educatie en ook onze opdrachtgever maakt gebruik van Teams. De applicatie moet dan ook (gedeeltelijk) beschikbaar zijn op Microsoft Teams. Dit kan gedaan worden door de volgende stappen te volgen.

8.2.1 Microsoft 365 Developer

Om apps te kunnen uploaden naar Teams heb jee Microsoft 365 Developer account nodig. Voor onze doelstelling was het logisch om één apart account te kunnen gebruiken voor het gehele team. De volgende stappen moeten hiervoor uitgevoerd worden:

1. Je hebt een bestaand of nieuw Microsoft account nodig. Aan dit account is een naam en telefoonnummer gekoppeld
2. Daarna moet je je aanmelden voor het Microsoft 365 Developer Program. Hier is tijdens het ontwikkelen gekozen om een gratis proefversie van 92 dagen te nemen van Microsoft E5 subscription, maar elk ander abonnement dat Teams heeft zal moeten werken
3. Een administrator account met bijbehorende email en inloggegevens is aangemaakt. Met dit account wordt ingelogd op Teams

Microsoft beveelt aan om ook nog een voorbeeld populatie toe te voegen via het Microsoft 365 Dev Center. Hier log je in met je Microsoft account. Dit kan best lang duren (voor ons duurde het 30+ minuten).

Via het administrator account kan je accounts aanmaken in het Office365 admin center. Voor onze applicatie zijn wij ervan uitgegaan dat leerlingen, logboekontwerper en leraren een *jobTitle* krijgen met of `Leraar`, `Logboekontwerper` of `Leerling`. Omdat alle leraren automatisch logboekontwerper zijn checken we of ze logboekontwerper of leraar zijn om te bepalen of ze logboeken mogen maken en bewerken.

8.2.2 Teams instellen

Voor de applicatie zijn we er vanuit gegaan dat alle leraren in hun eigen Team zitten met hun leerlingen.

Om de juiste permissies te hebben voor Teams moet je naar het Teams admin panel te gaan en de permissies aan te passen zodat leraren apps kan toevoegen. Dit doe je door in de zijbalk naar `Teams apps` -> `setup policies` te gaan. Maak een nieuwe policy aan genaamd `leraar` of zo en dan bovenaan `upload custom apps` aanzetten.

Geef elke leraar deze policy door naar `Users` -> leraar naam -> `policies` -> `Edit` knop naast assigned policies in te drukken. Selecteer nu bij App setup policy de policy die je net hebt gemaakt.

8.2.3 App toevoegen aan organisatie

De app in Teams is eigenlijk gewoon een iframe waarin de React app wordt getoond binnen Teams. Je moet wel een zogenaamd "App Manifest" maken om het aan Teams te kunnen toe te voegen. Dit

kan je doen de App Studio app te installeren en daar naar `manifest editor` te gaan en te drukken op `new app`.

Vul hierin de gevraagde gegevens in het app de pagina `App details` helemaal in en klik daarna aan de linkerkant op `Tabs` en selecteer je config bestand (waarschijnlijk website /config). Scroll aan de linkerkant helemaal naar beneden en selecteer `Test and distrubute`. Nu kan je op `Download` drukken om een zip te downloaden met het manifest.

Om de app te uploaden, ga naar `Teams apps` -> `Manage apps` en druk dan op upload. Upload hier de zip die je hebt gedownload uit de App Studio. Als het goed is kan iedereen met de juiste permissies de app nu zien.

Het kan enkele uren duren voordat deze veranderen worden toegepast en leraren daadwerkelijk apps kunnen installeren.

8.2.4 App toevoegen aan team

Als leraar, navigeer naar je team en druk op het plusje. Zoek naar de naam van je app en klik erop. Je komt nu op de config pagina, druk op op opslaan om de app toe te voegen aan je team.

Nu kunnen alle leden van het team de app zien in hun bovenbalk en ook gebruiken. Als je een foutmelding krijgt, kan het zijn dat je permissies niet goed ingesteld zijn of je enkele uren moet wachten totdat de veranderen in permissies zijn toegepast.

8.3 SSL

Tijdens het ontwikkelen van de app is er vanuit gegaan dat de client-side HTTPS zou gebruiken, omdat dit een vereiste is voor het gebruik van Teams. Alle site addressen staan in een env variabel, dit betekent dat je ze maar op één plek hoeft aan te passen (het `.env` bestand in de root van de client en server).

9 Operatie en ondersteuning

In dit hoofdstuk wordt het opereren van de applicatie genoemd en de ondersteuning.

9.1 Het project lokaal laten draaien

In dit kopje staat beschreven hoe je het project lokaal kunt laten draaien door een aantal stappen te volgen.

Let op: de server en de client moeten op aparte terminals draaien en zorg ervoor dat de poorten 3000 en 3001 beschikbaar zijn.

Stap 1 - Vooraf

Dit zijn de programma's die vooraf moeten zijn geïnstalleerd:

- Node.js v14.15.3
- MongoDB v3.6.3
- Robo 3T (aanbevolen), Studio 3T of een andere MongoDB client

Als je een van deze programma's niet hebt, download deze dan eerst. De versie die hierboven staan aangegeven zijn gebaseerd op de versies die het ontwikkelteam heeft gebruikt. Wij kunnen niet garanderen dat de applicatie werkt op eerdere versies.

Clone daarna repository naar je lokale machine en open de repository in een code editor.

Stap 2 - Server opstarten

Open een terminal en navigeer in de terminal naar de folder `server-side`. Run daarna de volgende commands om de server correct te laten starten:

```
npm install
node app.js
```

De server draait nu op poort `3000`.

Het kan handig zijn om wat testdata in de database te hebben staan. Hiervoor kun je de de volgende command runnen in de terminal:

```
node seed.js
```

Stap 3 - Client opstarten

Open een nieuwe terminal en navigeer in de terminal naar de folder `client-side`. Run daarna de volgende commands om de client correct te laten starten:

```
npm install
npm start
```

De browser wordt nu geopend en `localhost:3001` wordt getoond.

9.2 Operatie

In principe kan de site zodra hij gedeployed is, gelijk worden gebruikt. Als de office 365 omgeving goed is ingesteld, dus dat elke leerkracht en leerling de goede *job title* heeft en er al teams bestaan waaraan de applicatie kan worden toegevoegd hoeft er niks handmatig worden ingevoerd.

9.3 Ondersteuning

Zoals besproken in hoofdstuk 4, geeft het team geen ondersteuning na het opleveren. Maar het Software Guidebook is zo ingericht dat iedere developer met behulp van dit software guidebook de applicatie goed genoeg zou kunnen moeten begrijpen om er in de toekomst mee verder te kunnen werken. Ook zorgt het team voor comments bij de code die complexere delen van de code uitleggen.

Vanwege de gelimiteerde tijd en scope worden op dit moment errors gelogged in de server-side op de console. Er is dus niet een specifieke error afhandeling in plaats op dit moment, maar op zo veel mogelijk locaties binnen de code hebben we ons best gedaan om fouten op te vangen en goed te verwerken zodat beveligingslekken worden voorkomen en de gebruiker ongehinderd de applicatie kan gebruiken.