Bedrijf
Bullit Digital B.V.

Adres:
Barkenkamp 5
7141 EL Groenlo

E-mail:
info@bullit.digital

Isolated Development:

Samenwerken zonder conflicten

Door:
Remco Volkers

Studentnummer:
460318

Telefoon:
0640782841

E-mail:
remco.volkers1@gmail.com

Voorwoord

In mijn voorwoord wil ik graag mijn dank uitspreken aan een aantal mensen die een belangrijke rol hebben gespeeld tijdens mijn afstudeerperiode bij Bullit en mijn studie over het algemeen.

Ik wil Fleur Oudenampsen bedanken voor haar begeleiding en het blussen van brandjes tijdens het maken van keuzes voor mijn opleiding en mijn ontwikkeling als professional. Frank van Doorn wil ik bedanken voor het overnemen van de SLB rol en zijn aandacht voor mij. Daarnaast wil ik Michel Lammertink bedanken voor zijn geduld en tips tijdens het afronden van de afstudeerperiode en voor het aanwakkeren van het vuur in mij tegen het einde van de afstudeerperiode.

Yvonne en Marcel Bomers wil ik bedanken voor hun emotionele support en het mogelijk maken van mijn studie. Mijn lieve zus Wendy wil ik bedanken voor haar steun als mijn #1 cheerleader en mijn moeder voor haar geloof in mij en de liefdevolle woorden. Mijn lieve vriendin Jorien wil ik bedanken voor haar steun tijdens momenten van stress en haar oneindige support. Mijn vrienden wil ik bedanken voor het lezen en aanhoren van mijn presentaties en het stellen van kritische vragen.

Tot slot wil ik Bullit bedanken voor de tijd en resources die zij in mij hebben gestoken als bedrijf en als individuen. Ik waardeer de persoonlijke verbinding die ik met Bullit heb kunnen maken en zal mijn tijd hier niet vergeten. In het bijzonder wil ik Martijn Gemmink bedanken voor zijn geduldige aanpak en goede begeleiding, met name met betrekking tot de documentatie van dit project.

Samenvatting

In de scriptie wordt beschreven hoe het software platform OfficeDog werkt en hoe de huidige architectuur eruit ziet. OfficeDog is gericht op logistieke bedrijven en biedt gebruikers de mogelijkheid om logistieke processen te optimaliseren met behulp van tricks en widgets. In hoofdstuk 3 wordt een stakeholderanalyse uitgevoerd om inzicht te krijgen in de belanghebbenden bij de ontwikkeling van OfficeDog. De stakeholders zijn geïnterviewd en gerangschikt op relevantie voor de opdracht.

In hoofdstuk 4 wordt de huidige situatie van OfficeDog beschreven, waarbij wordt ingegaan op hoe het platform momenteel functioneert en welke problemen er zich voordoen. Er wordt beschreven dat OfficeDog een webapplicatie is die toegankelijk is via een browser, maar dat de huidige architectuur er een monoliet van maakt, wat het onmogelijk maakt om los van het hoofdproject te ontwikkelen. Er wordt ook ingegaan op de technische aspecten van OfficeDog, waaronder de gebruikte tools en problemen die zich hierdoor voordoen.

In hoofdstuk 5 worden de stappen beschreven om de requirements voor de opdracht te bepalen en te prioriteren. De eisen worden opgedeeld in functionele en niet-functionele eisen en er wordt gebruik gemaakt van de MoSCoW-methode om de prioriteiten te bepalen. Er wordt ook aandacht besteed aan onafhankelijk ontwikkelen en de rol van dynamische imports hierbij.

In hoofdstuk 6 wordt besproken hoe externe developers in staat gesteld worden om tricks en widgets toe te voegen aan het OfficeDog-platform zonder toegang tot de broncode van OfficeDog. Dit wordt gedaan door middel van een architecturale verandering aan de huidige frontend, waarbij een (gedeeltelijke) herziening van de huidige opbouw van de frontend noodzakelijk kan zijn. Er wordt gebruik gemaakt van een framework analyse om te bepalen welke oplossing het meest kans maakt om het probleem op te lossen de requirements hiervoor zijn gevalideerd goedgekeurd door Bullit. De oplossing moet ondersteuning bieden voor dynamisch importeren van modules en connectiviteit met de bestaande API behouden.

In hoofdstuk 7 wordt het vervolgtraject besproken. Hierin worden de tekortkomingen van mijn onderzoek besproken en hoe Bullit verder kan gaan met dit onderzoek om bij de gewenste situatie terecht te komen. Hierbij spelen vooral documentatie richting de externe developers en het implementeren van de gewenste situatie een rol.

Inhoudsopgave

Begrippenlijst

Begrip	Definitie
Tricks	Applicaties die gemaakt zijn om in het OfficeDog platform te functioneren.
Widgets	Deelfunctionaliteiten van tricks of andere delen van de applicatie, die op een dashboard kunnen worden weergegeven. Zoals bijvoorbeeld widgets op een Android-telefoon de agenda van Google Calendar op de homescreen weergeven, maar niet de volledige functionaliteit van de app hebben.
Software platform	Een software platform is een omgeving of framework waarbinnen software-applicaties kunnen worden ontwikkeld en uitgevoerd. Het biedt een standaard set van functionaliteiten en services waarmee ontwikkelaars hun software kunnen bouwen, testen en uitvoeren, zonder dat ze de details van de onderliggende technologie hoeven te begrijpen.
Broncode / source code	Versie van een computerprogramma / software product voordat deze is gecompileerd. Wordt gebruikt als basis voor het maken van een uitvoerbaar programma. In het kader van OfficeDog betekent dit de code waar het platform uit opgebouwd is minus de tricks en widgets.
Frontend monoliet	Software architectuur waarbij alle frontend-onderdelen van een applicatie in één enkele codebase worden geïmplementeerd. Dit betekent dat de code voor de interface van de gebruiker, de presentatie van gegevens en de interactie met de gebruiker allemaal in één enkele grote codebase worden geplaatst.
Developer Experience	Developer Experience is de ervaring die een ontwikkelaar heeft bij het werken met een bepaald platform, set tools, API's of frameworks.
Monorepo	Een monorepo is een repository waarin meerdere projecten of onderdelen van een project samen worden beheerd en opgeslagen.

Tabel 1: Begrippenlijst

1. Inleiding

In deze scriptie zal ik behandelen hoe ik geïsoleerde software ontwikkelteams samen kan laten werken aan hetzelfde platform zonder last te ondervinden van elkaar.

Dit platform is OfficeDog. OfficeDog is een start-up die volop in ontwikkeling is en zich voorbereidt om de logistieke wereld te voorzien van een breed en gevarieerd assortiment aan functionaliteit. De reden waarom ik dit onderwerp gekozen heb is omdat ik de probleemstelling interessant vind en de manier van benaderen mij enthousiasmeert. Zo zal ik veel vrijheid gaan krijgen om te kiezen voor een oplossing die het probleem adequaat oplost.

Mijn hoofdvraag is: **Hoe kan Bullit externe software ontwikkelaars gebruiksvriendelijk in staat stellen tricks en widgets toe te voegen aan de OfficeDog frontend?**

In dit onderzoek zal ik eerst de stakeholders signaleren waarna ik passende methodes zal toepassen om informatie te verzamelen die relevant is voor mijn opdracht. Van het resultaat van deze activiteiten zal ik vervolgens requirements opstellen voor een software realisatie fase. Ook zal ik ontdekken hoe ik verschillende software development teams met elkaar kan laten samenwerken aan OfficeDog. Deze teams zijn veelal van verschillende (concurrerende) bedrijven en zullen elkaar potentieel nooit spreken waardoor het essentieel is dat zij los van elkaar kunnen opereren in het toevoegen van functionaliteit aan het platform.

Het eindresultaat zal een combinatie zijn van een software product dat als proof of concept zal fungeren en een adviesrapport over een specifiek vraagpunt van Bullit en eventuele vervolgstappen.

Ik zal dit onderzoek uitvoeren bij Bullit Digital. Zij zijn strategisch software partner en zijn gevestigd in Groenlo. Strategisch omdat zij differentiëren van andere software partners op het vlak van innovatie en de manier hoe zij met klanten samenwerken. Dit samenwerken heeft altijd de invalshoek “wat is het probleem dat u probeert op te lossen” in plaats van “wat moeten we doen”. Door die eerste vraag te stellen zorgt Bullit voor een dialoog over het op te lossen probleem.

Lijst van betrokkenen	
Martijn Gemmink	Bedrijfsbegeleider, mede-eigenaar Bullit Digital
Bjorn Goossens	Mede-eigenaar Bullit Digital
Michel Lammertink	Afstudeerbegeleider Saxion
Sebastian Piest	Onderzoeker Universiteit Twente, kartrekker OfficeDog

Tabel 2: Lijst van betrokkenen

2. Opdracht

Om de vraag van Bullit goed te kunnen begrijpen, is het belangrijk om het idee achter OfficeDog helder te krijgen. In de volgende hoofdstukken ga ik onder andere in op het OpenTripModel (OpenTripModel, 2021) en hoe OfficeDog conceptueel werkt.

2.1. Achtergrond

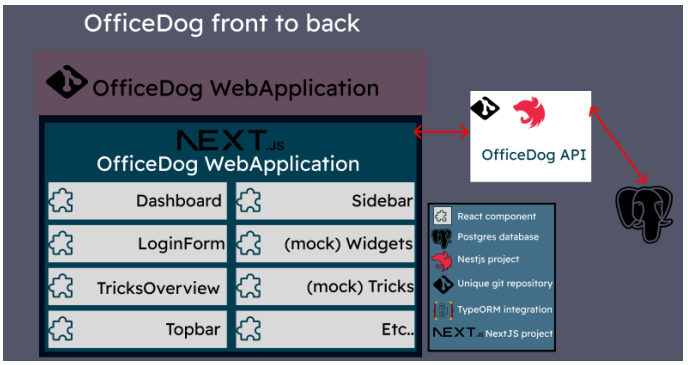
OfficeDog is een product van Bullit Digital, ontwikkeld in samenwerking met Sebastian Piest, onderzoeker bij de afdeling Industrial Engineering en Business Information Systems aan de Universiteit Twente.

2.1.1. OfficeDog

OfficeDog is een software platform dat de logistieke sector wil helpen innoveren door middel van data-uitwisseling en het aanbieden van handige software oplossingen. Logistieke bedrijven leveren gestandaardiseerde data aan OfficeDog, die vervolgens gebruikt wordt om deze oplossingen te creëren. Een doel is dat bedrijven uniforme data kunnen gebruiken om voordeel te behalen.

Het is momenteel nog in een "proof-of-concept" stadium, maar er is al veel interesse getoond vanuit de industrie en er zijn meerdere pilots gedraaid.

Figuur 1 is een weerspiegeling van de huidige architectuur van OfficeDog. Dit figuur zal worden gebruikt om de technische uitdagingen en oplossingen te verduidelijken die in latere hoofdstukken besproken zullen worden. Deze architectuurplaat is van groot belang voor het begrip van de technische uitdagingen en oplossingen die in de scriptie besproken zullen worden.



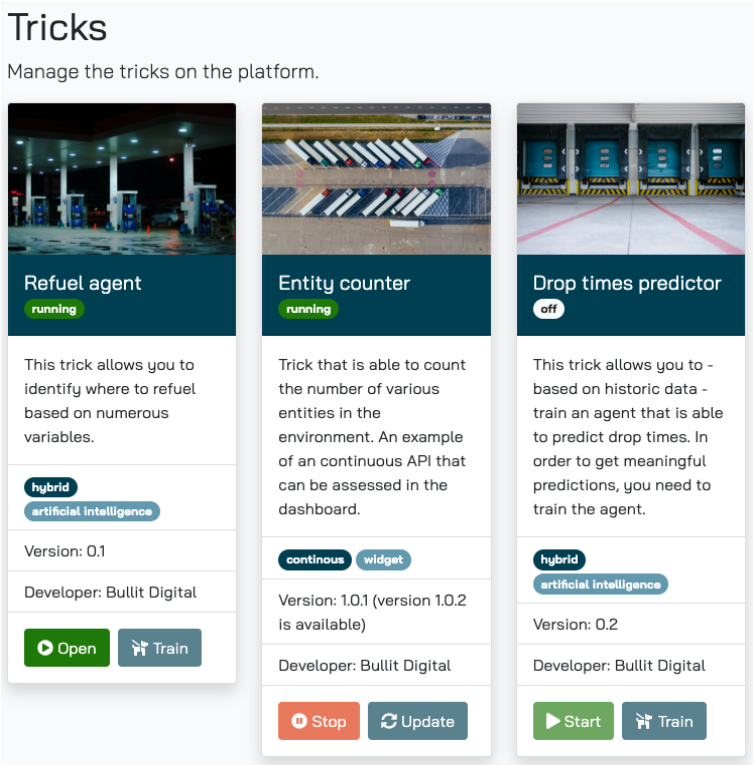
Figuur 1: OfficeDog huidige architectuur

In het geval van OfficeDog betekent “software platform zijn” het huizen van verschillende slimme applicaties.

OfficeDog is een webapplicatie die zich richt op logistieke bedrijven. Het biedt een overzicht van logistieke processen en geeft gebruikers de mogelijkheid om deze processen te optimaliseren door middel van "tricks". Deze tricks zijn kleine oplossingen die specifiek zijn ontwikkeld voor een bepaald logistiek probleem. OfficeDog is momenteel alleen ontwikkeld door Bullit en externe ontwikkelaars kunnen nog niet meeontwikkelen aan de applicatie.

Hieronder zullen er een aantal schermafbeeldingen getoond worden om een overzicht te geven van OfficeDog. Het overzicht is beperkt tot de relevante onderdelen voor dit onderzoek.

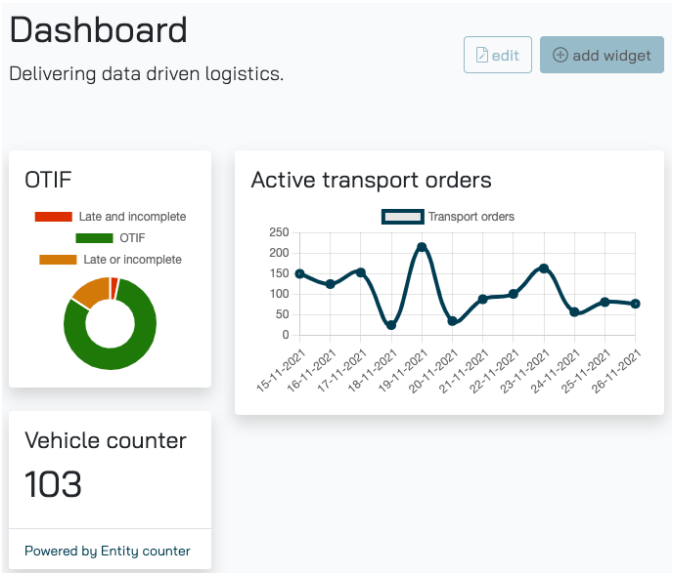
Er is ook een pagina waarop de tricks op het platform kunnen worden beheerd, de TricksOverviewPage (figuur 2), waar labels, namen, acties en uitleg van de tricks te zien zijn.



Figuur 2: TricksOverviewPage

Er is ook een dashboard overview in OfficeDog, dat modulair is en naar de wens van de gebruiker kan worden ingericht. Hier zijn widgets te zien die afkomstig zijn uit andere delen van de applicatie, waaronder tricks. Widgets zijn deelfunctionaliteiten van tricks of andere delen van de applicatie, die op een dashboard kunnen worden weergegeven. **Zoals bijvoorbeeld widgets op een Android-telefoon de agenda van Google Calendar op de homescreen weergeven, maar niet de volledige functionaliteit van de app hebben.**

Een voorbeeld van een widget is de Vehicle Counter die te zien is op het dashboard. Deze is “powered by Entity Counter” omdat de Entity Counter trick deze functionaliteit aanbiedt en het dashboard zo geconfigureerd is dat de widget weergegeven wordt. De vehicle counter is dus de widget, deze ***widget*** is onderdeel van de ***trick*** Entity counter.



Figuur 3: OfficeDog Dashboard

2.1.2. OpenTripModel (OTM)

Het OpenTripModel is een model waarnaar je logistieke data kan inrichten. Elk bedrijf dat data wilt delen met andere logistieke bedrijven zullen bij het OTM of iets soortgelijks uitkomen.

> *"OfficeDog geeft antwoord op een integratievraagstuk. Daar zit echt een horde in de transportwereld. Zij hebben vaak hun datamodellen niet uniform.*
>
> *OfficeDog en het OTM is een manier hoe zij dit kunnen repareren. Als er geen universele datastructuur is kunnen we wel een trick schrijven voor bedrijf A maar dan moet je bij bedrijf B 75% van het werk weer opnieuw doen."*

Contextuele quote Martijn Gemmink uit interview (Bijlage D)

Het OTM is een opkomend initiatief dat wordt gesteund door de Nederlandse Overheid en is een idee van Simacan. Het doel van OTM is om integratie met andere logistieke bedrijven mogelijk te maken zodat er binnen de sector meer samenwerking gerealiseerd kan worden. Het doel van het product is dat elk bedrijf dat wil integreren in de logistieke sector dat doet met behulp van het OTM. Zo zijn zij niet alleen klaar om te integreren met het bedrijf waar zij op dat moment mee willen samenwerken maar ook alle andere bedrijven die de data respectievelijk aan het OTM hebben gemodelleerd.

Als een bedrijf aan de slag wil met OfficeDog dient het een integratie te maken van hun huidige datastructuur naar die van het OTM.

2.2. Opdrachtomschrijving

In de huidige implementatie van de OfficeDog frontend is het niet mogelijk om tricks toe te voegen zonder in de source code aanpassingen te maken. Bullit wil weten hoe zij dit wel mogelijk kunnen maken zodat zij externe developers de mogelijkheid kunnen bieden om tricks te schrijven voor OfficeDog.

Ik ga eerst in op de opdracht en vervolgens op de onderzoeksvragen die ik hieruit heb gehaald. In bijlage A zit de opdrachtomschrijving van Bullit Digital. In bijlage D is een interview afgenomen met Bullit Digital, de opdrachtgever voor deze scriptie. Tijdens dit interview zijn de requirements voor de opdracht verder verkend en in detail besproken. Hierdoor is het duidelijk geworden welke problemen Bullit Digital ervaart bij het in staat stellen van externe ontwikkelteams om mee te ontwikkelen aan OfficeDog en welke oplossing zij hiervoor zoeken.

Bullit zoekt naar een manier om bedrijven die bij OfficeDog aangesloten zijn in staat te stellen gebruiksvriendelijk mee te ontwikkelen aan tricks voor OfficeDog. Om dit te kunnen realiseren is het belangrijk dat er goed geluisterd wordt naar de wensen van developers.

Het OfficeDog platform heeft meer te bieden wanneer de sector zelf helpt mee ontwikkelen. Zo kunnen developers met domeinkennis bouwen aan oplossingen voor de logistieke sector. Terwijl zij op het moment van aansluiten bij OfficeDog al voordelen kunnen halen uit het platform.

Bullit zoekt ook naar een manier om widgets te kunnen ontwikkelen op een vergelijkbare manier als dat zij dat voor tricks voor ogen hebben. Dus zonder dat de source code van OfficeDog aangepast hoeft te worden.

2.3. Onderzoeksvragen

Om de onderzoeksvragen te bepalen zijn er gesprekken gevoerd vooraf aan mijn afstudeerperiode met Bullit. Bullit heeft ook voorzien in een opdrachtomschrijving (bijlage A). Daar komt de hoofdvraag:

Hoe kan Bullit externe software ontwikkelaars gebruiksvriendelijk in staat stellen tricks en widgets toe te voegen aan de OfficeDog frontend?

vandaan.

Bullit wil dat OfficeDog een platform wordt waar door meerdere teams aan ontwikkelt kan worden maar dan wel met de restrictie dat deze teams geen beschikking krijgen tot de source code van het platform.

Om ervoor te zorgen dat de oplossing goed aansluit op de wensen van de stakeholders is het belangrijk om deze in kaart te brengen. Hier komt de deelvraag:

Welke stakeholders zijn er?

vandaan.

Vervolgens moet er onderzocht worden wat de verwachtingen zijn voor dit onderzoek. De stakeholders zijn geïdentificeerd en hun belangen zijn in kaart gebracht.

Om inzicht te verkrijgen in de technische mogelijkheden en beperkingen voor het dynamisch inladen van tricks en widgets in OfficeDog, is het noodzakelijk om de huidige situatie te analyseren. Hier komt de deelvraag:

Hoe ziet de huidige situatie eruit?

vandaan.

Om de wensen van de stakeholders om te zetten in een software oplossing is het belangrijk om requirements te definiëren. Ook wordt hier behandeld wat het betekent om gebruiksvriendelijk te ontwikkelen. Hier komt de deelvraag:

Welke requirements zijn er?

vandaan.

Nu ben ik op een punt aangekomen in mijn onderzoek dat ik weet wat ik moet doen, de requirements zijn gevalideerd. Nu start het ontwikkelen van de oplossing. Hier komt de deelvraag:

Hoe stel ik developers in staat om tricks aan te bieden aan de OfficeDog frontend?

vandaan.

OfficeDog wilt ook widgets blijven aanbieden op dezelfde onafhankelijke manier als dat zij de tricks willen zien. Zij zien graag een advies tegemoet over hoe dit mogelijk geïmplementeerd kan worden. Hier komt de deelvraag:

Hoe stel ik developers in staat om widgets aan te bieden aan de OfficeDog frontend?

vandaan.

Hieronder nog een overzicht van de hoofdvraag, deelvragen en waar deze beantwoordt worden.

Hoofdvraag		
Hoe kan Bullit externe software ontwikkelaars gebruiksvriendelijk in staat stellen tricks en widgets toe te voegen aan OfficeDog frontend?		
Deelvragen		**Beantwoord in**
DV1	Welke stakeholders zijn er?	Hoofdstuk 3
DV2	Hoe ziet de huidige situatie eruit?	Hoofdstuk 4
DV3	Welke requirements zijn er?	Hoofdstuk 5
DV4	Hoe stel ik developers in staat om tricks aan te bieden aan de OfficeDog frontend?	Hoofdstuk 6/7
DV5	Hoe stel ik developers in staat om widgets aan te bieden aan de OfficeDog frontend?	Hoofdstuk 6/7

Tabel 5: Deelvragen

2.4. Aanpak

Voor deelvraag 1 zal er een stakeholderanalyse uitgevoerd worden om te bepalen wie de stakeholders zijn en waarom zij belangrijk zijn voor het project.

Voor deelvraag 2 zal er een overzicht gemaakt worden van de huidige situatie van OfficeDog. Dit zal technisch en conceptueel worden uitgelegd aan de hand van uitleg en screenshots. Dit is belangrijk om te weten omdat dit de basis vormt van het probleem waarmee gewerkt wordt. Momenteel is het niet mogelijk voor externe developers om tricks te ontwikkelen zonder aanpassingen aan de source code te doen.

Voor deelvraag 3 zal er een analyse uitgevoerd worden om inzicht te krijgen in hoe de stakeholders willen dat het platform eruit ziet. Hierbij zal gebruik gemaakt worden van de resultaten van de stakeholderanalyse en interviews met stakeholders, software developers en onderzoek naar gebruiksvriendelijk ontwikkelen.

Voor deelvraag 4 en 5 zal er uitleg gegeven worden over de oplossing die is bedacht. Hierbij zal beschreven worden hoe de beslissingen zijn genomen met behulp van community research en zal er een architectuurplaat worden gepresenteerd met technische en conceptuele uitleg.

Tijdens dit onderzoek zal er gewerkt worden in sprints, zoals beschreven in the Scrum guide (Schwaber & Sutherland, z.d.). Dit is gekozen omdat het een iteratief raamwerk is, wat goed past bij deze opdracht omdat de oplossing niet vooraf gedefinieerd is. Niet alle elementen van Scrum zullen gebruikt worden en dit zal worden uitgelegd waarom dit zo is. De volgende scrum elementen zullen niet of anders geïmplementeerd worden dan volgens de scrum methodiek:

- **Daily Scrum**
  Deze meeting heeft als doel om een team gefocust te krijgen om het sprint doel te halen en biedt dit element overzicht, ook dient het om adequaat en op tijd interventies te verzorgen voor developers die dit nodig hebben. Ik werk niet in een team, ik werk alleen. Ook kan ik de energie die hierbij komt kijken niet vragen van mijn afstudeerbedrijf.

In plaats van “Daily Scrum” toe te passen heb ik zo tijdens het koffie halen mijn afstudeerbedrijf laten weten waar ik mee bezig was die dag. Zo hebben zij genoeg ruimte ervaren om bij te sturen waar zij dat nodig vonden. Hier zat geen vast patroon in maar gebeurde regelmatig.

- **Sprint Retrospective**
  Het doel van een sprint retrospective is om kwaliteit en effectiviteit positief te stimuleren. Ook hier is weer een team-belang. Kwaliteit/effectiviteit verbeteringen gebeurde over het algemeen in 1-op-1 gesprekken tussen mij en Martijn of tijdens zelfreflectie-momenten. Deze gesprekken met Martijn vinden minstens 1 keer per week plaats met als vast moment de woensdagen aan het eind van de dag.

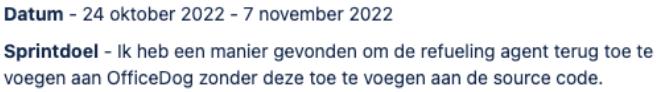
- **Sprint Review**
  Het doel van een sprint review is het inspecteren van de behaalde progressie van het product in een sprint en om te bepalen wat de volgende stappen zijn voor het product. Om elke 2 weken de aandacht van Bullit te vragen kost te veel tijd. Ik heb in plaats van de twee-wekelijkse sprint review wel de voortgang besproken en gepresenteerd op kritieke momenten. Die kritieke momenten waren aan mij om aan te geven en dan werd er een presentatie gepland.

  In hoofdstuk 6 is de oplossing voor het toevoegen van tricks en widgets aan het OfficeDog-platform zonder toegang tot de broncode van OfficeDog beschreven. Deze oplossing is gepresenteerd aan Martijn en Bjorn van Bullit voor feedback. Tijdens deze presentatie is ook een vroege versie van het proof of concept getoond. De oplossing is goed ontvangen en de mogelijkheid om de microfrontends dynamisch te laden is als een oplossing voor hun vraag beschouwd. De refuel agent kan bijvoorbeeld als een externe applicatie worden geladen.

  <u>De rest van scrum evenementen en artifacts gebruik ik wel:</u>

- **Sprint**
  Sprints duren 2 weken, te beginnen op de maandag.

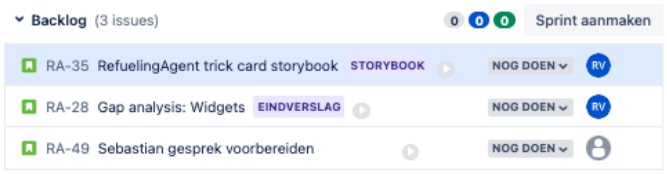
- **Sprint goal**
  Elke sprint heeft een “sprint goal”. Het voorbeeld komt zoals je kan zien van een sprint die gestart is in eind oktober. Zie figuur 4.



Figuur 4: Voorbeeld Jira sprint doel

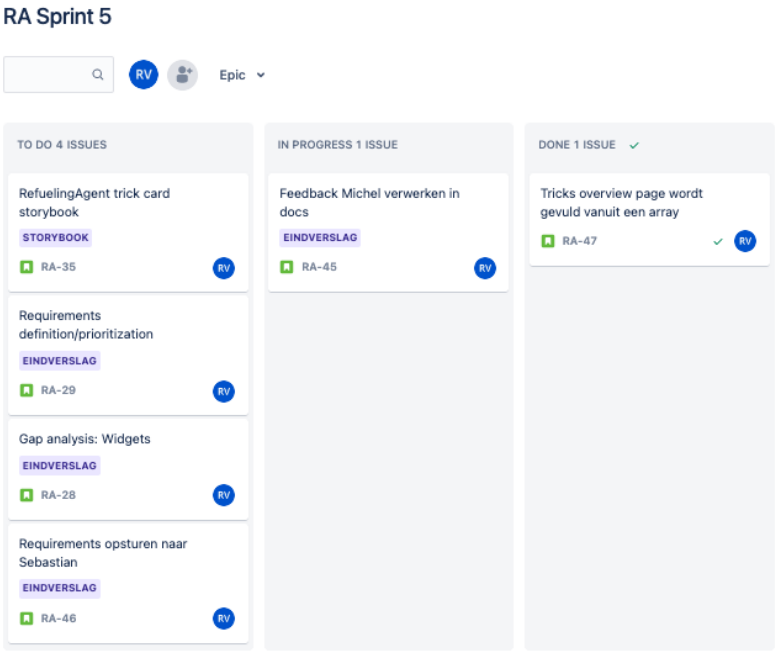
- **Sprint planning**
  Voor elke sprint start deel ik de werkzaamheden op in tickets en plaats ze op het bord. Ook creëer ik hier nieuwe tickets.

- **Product backlog**
  Gebruik ik als combinatie van eerder gemaakte tickets en toekomstige tickets die niet in de huidige sprint verwerkt zijn. Zie figuur 5.



Figuur 5: Voorbeeld Jira product backlog

- **Sprint backlog**
  Hieronder een voorbeeld van mijn sprint backlog. Dit is de status van de backlog op het moment van het schrijven van deze tekst. Zie figuur 6.



Figuur 6: Voorbeeld Jira sprint backlog

3. Stakeholderanalyse

Om inzicht te krijgen in welke stakeholders er zijn in verband met de ontwikkeling van OfficeDog, zullen de volgende stappen worden ondernomen:

1. **Signaleren van stakeholders**: Er worden verschillende groepen mensen of individuen geïdentificeerd die een belang hebben bij het eindresultaat van dit onderzoek.
2. **Interviewen van stakeholders**: De geïdentificeerde stakeholders worden geïnterviewd om hun wensen en verwachtingen in kaart te brengen.
3. **Priotisering van stakeholders:** De stakeholders zullen in kaart gebracht worden met betrekking tot relevantie van de opdracht.

De volgende stakeholders zijn geïdentificeerd:

- **Bullit Digital**: Ontwikkelaar van het platform
- **Sebastian Piest**: Kartrekker van OfficeDog en belangrijke verbinder in de logistieke sector
- **(Management van) Logistieke bedrijven**: Beoogde afnemers van het product
- **Ontwikkelaars:** Software Developers/Data scientists die met OfficeDog van doen krijgen en tricks zullen ontwikkelen voor het platform
- **Logistiek medewerkers**: In gevarieerde rollen wiens werk geautomatiseerd zal worden wanneer OfficeDog een succes wordt
- **OfficeDog gebruikers**: Logistiek medewerkers die gebruik zullen maken van OfficeDog en zullen werken met de functionaliteit die het platform aanbiedt

Bullit Digital, Sebastian Piest, studiegenoten van HBO-ICT software engineering en een senior developer zijn geïnterviewd. Interviews met de andere stakeholders zijn niet uitgevoerd omdat zij geen verschil zouden moeten merken in het gebruik van OfficeDog, en omdat het platform zich momenteel nog in een pilotfase bevindt en er daarom geen representatieve personen beschikbaar zijn. Om rekening te houden met deze doelgroepen wordt er geadviseerd geen veranderingen door te voeren aan de gebruikerservaring met OfficeDog zoals deze nu bestaat tot deze stakeholders wel inspraak kunnen geven op OfficeDog. Deze stakeholders zijn geïdentificeerd omdat zij in een later stadium in de ontwikkeling van OfficeDog wel te maken gaan krijgen met het platform.

Stakeholder	Priotisering	Beredenering
Bullit Digital	Hoog	Opdrachtgever, belangrijke beslissingen rondom ontwikkeling en implementatie van OfficeDog
Sebastian Piest	Hoog	Belangrijk in de ontwikkeling en implementatie van OfficeDog
Developers	Hoog	Belangrijk voor het toevoegen van nieuwe "tricks" en flexibiliteit van het platform
Management Logistiek	Laag	Belangrijk voor de toekomstige groei en implementatie van OfficeDog binnen deze bedrijven, momenteel geen representatief beeld van de doelgroep beschikbaar
Logistiek Medewerkers	Laag	Gebruikers van het platform, momenteel geen representatief beeld van de doelgroep beschikbaar
OfficeDog gebruikers	Laag	Gebruikers van het platform, momenteel geen representatief beeld van de doelgroep beschikbaar

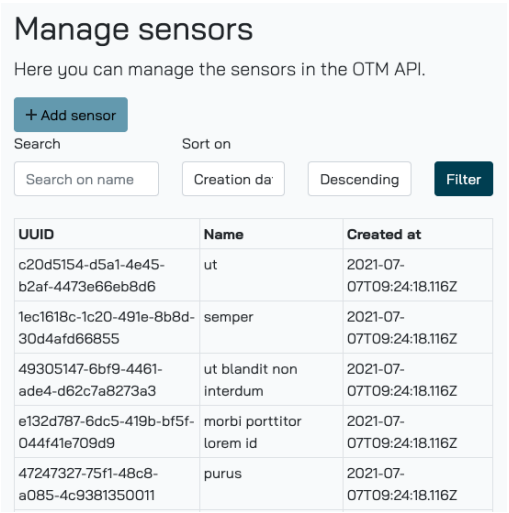
Tabel 6: Stakeholders prioritisatie

4. Huidige situatie

Om een antwoord te geven op ***DV2: "Hoe ziet de huidige situatie eruit?"*** is het noodzakelijk om een inzicht te krijgen in hoe OfficeDog functioneert en welke problemen er zich voordoen. Hiervoor wordt eerst de conceptuele uitwerking uitgediept en vervolgens worden de technische knelpunten geïdentificeerd. Op deze manier kan een duidelijk beeld verkregen worden van de huidige situatie van OfficeDog en welke verbeteringen er nodig zijn om het probleem waar Bullit tegenaan loopt op te lossen.

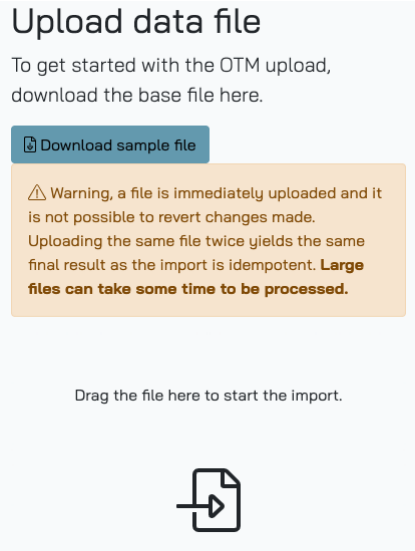
4.1. Conceptueel

OfficeDog is een webapplicatie die toegankelijk is via een browser. Een gebruiker logt in met zijn gegevens en heeft vervolgens toegang tot OfficeDog, mits het bedrijf waar hij bij hoort aangesloten is bij OfficeDog. Een gebruikersprofiel is gekoppeld aan een bedrijf en de gebruiker krijgt de dashboardpagina (figuur 3) van dit bedrijf te zien. Hiermee krijgt de gebruiker een overzicht van het specifieke bedrijf waar hij werkzaam is.

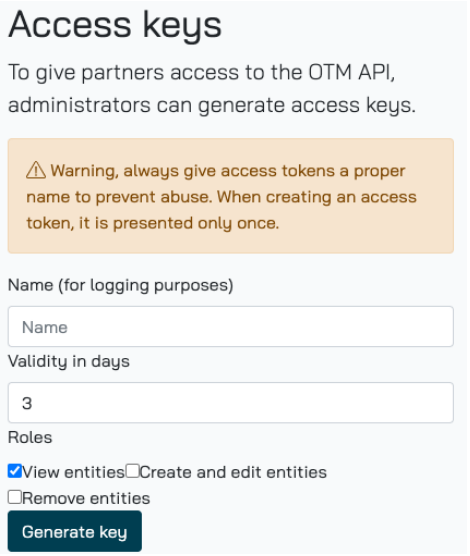


Figuur 7: Entity "Sensor" in OfficeDog

Ook zijn er mogelijkheden om data (figuur 7) aan te leveren aan OfficeDog in csv bestanden (figuur 8). De gebruikers met admin privileges kunnen in OfficeDog API keys genereren zodat concullega's/bedrijven met wie zij samen willen werken toegang kunnen verlenen tot de data van het bedrijf van de gebruiker (figuur 9).



Figuur 8: Data uploading in OfficeDog



Figuur 9: Data sharing in OfficeDog

OfficeDog is op het moment een frontend monoliet. Dat wil zeggen dat de frontend uit 1 project bestaat en er binnen de applicatie afhankelijkheden zijn.

Dit problemen die zich momenteel voordoen die relevant zijn voor dit onderzoek:

1. Onafhankelijk van het hoofdproject ontwikkelen is onmogelijk omdat tricks onderdeel zijn van de source code.
2. Ontwikkelen aan het project vereist dat een ontwikkelaar toegang heeft tot de source code.
3. Aanpassingen aan de applicatie zullen vaak implicaties hebben voor andere onderdelen van de applicatie.

4.2. Technisch

Het huidige project is gebouwd met Next (Vercel, 2023) en is een React-app. React is een library die de developer helpt met het bouwen van user interfaces (Meta Platforms Inc, 2023). Waarom Bullit voor Next heeft gekozen is omdat het een React app robuuster maakt in de vorm van:

- Authenticatie/Autorisatie
- Vrijheid van waar en wanneer je de applicatie wilt renderen (server side rendering, client side rendering, static site generation)
- TypeScript support
- Routing

Deze voordelen van Next komen *“out of the box”.* Next biedt OfficeDog oplossingen voor vraagstukken die zonder Next tijd kosten om robuust te implementeren.

Van de oplossingen die Next aanbiedt maakt OfficeDog gebruik van onder andere routing en authenticatie.

In de backend is OfficeDog gebouwd met het NestJS framework. Hierbij maakt zij ook gebruik van een Object Relational Mapping (ORM) library: TypeORM. De backend is in een veel verder stadium dan dat de frontend is. Daarmee wordt bedoeld dat deze volwassener is, de OfficeDog frontend bestaat op dit moment uit veelal mockdata en mockcomponenten. Het is een tool om potentiële klanten te laten zien waar het product toe in staat is terwijl zij het platform verder ontwikkelen. De backend is meer “production-ready”.

Voor het verschil in volwassenheid tussen de backend en frontend van OfficeDog zijn twee redenen geconstateerd:

1. Een herschrijving van de OfficeDog frontend is voorzien met de visie die Bullit heeft met OfficeDog, namelijk: dat externe ontwikkelteams helpen met het ontwikkelen van tricks.
2. Het product heeft om het te idee te pitchen niet meer functionaliteit nodig dan dat het nu heeft.

5. Requirements

Om de deelvraag ***DV3: "Welke requirements zijn er"*** te beantwoorden te beantwoorden, zullen de volgende stappen ondernomen worden:

- Requirements vaststellen
- Requirements prioriteren (met gebruik van MoSCoW)
- Validatie van de requirements door Sebastian en Bullit

Tijdens de interviews met Sebastian Piest (bijlage F) en Martijn Gemmink (bijlage D) bleek duidelijk wat hun doelstelling is met de opdracht. Ze willen eerst vaststellen of het mogelijk is om aan de frontend binnen het platform onafhankelijke functionaliteit toe te voegen. Ten tweede willen ze weten of het mogelijk is om widgets te ontwikkelen op dezelfde manier als ze voor ogen hebben met de tricks. Schaalbaarheid stond daarbij centraal in het resultaat.

Tijdens de interviews met de doelgroep (de developers, bijlage E), is er verduidelijking ontwikkelt met betreft *developer experience*. Hierin werd vooral duidelijk dat goede documentatie en codevoorbeelden essentieel zijn in het begeleiden van het ontwikkelproces.

De eisen zullen worden opgedeeld in niet-functionele en functionele eisen. Niet-functionele eisen beschrijven **hoe** een systeem zou moeten functioneren, terwijl functionele eisen bepalen **wat** een systeem zou moeten bereiken. Hieruit kan een duidelijke "separatie van concerns" worden gemaakt, wat het overzicht van taken zou moeten verbeteren.

Onafhankelijk ontwikkelen kan beter worden begrepen door kennis te hebben over dynamische imports. Dynamische imports zijn modules die op een dynamische manier geladen kunnen worden, zonder dat deze vooraf gedefinieerd hoeven te zijn. Deze functionaliteit is cruciaal voor de eindoplevering omdat externe ontwikkelaars geen toegang hebben tot de broncode van het project. Hierdoor moet het project in staat zijn om zichzelf te openen voor de dynamisch geladen modules, zodat deze zich kunnen aanmelden en op de juiste plek kunnen worden weergegeven. De modules horen te draaien in hun eigen geïsoleerde omgeving binnen het OfficeDog platform. Hierdoor zal de verantwoordelijkheid van security veelal aan de developer kant liggen.

Non-functioneel

#	Requirement	MoSCoW
NF1	Developers hebben geen toegang tot de source nodig code tijdens het developen van tricks.	MUST
NF2	Er is documentatie beschikbaar met de volgende eigenschappen: - Bevat tenminste hoe zij een trick kunnen ontwikkelen - Legt uit hoe een trick conceptueel in OfficeDog aansluit - Bevat codevoorbeelden voor het maken van tricks	MUST
NF3	Er is een vervolgplan voor het widgets concept **Acceptatiecriteria:** - De oplossing is voorzien in een adviesrapport	MUST
NF4	Developers kunnen hun eigen software tools en frameworks gebruiken bij het ontwikkelen van tricks voor OfficeDog.	MUST
NF5	De developer heeft toegang tot documentatie met codevoorbeelden zodat zij begeleid kunnen worden tijdens het ontwikkelen van tricks. **Acceptatiecriteria:** - Documentatie is gebruiksvriendelijk om in te zien.	MUST

Tabel 8: Non-functionele requirements

Functioneel

#	Requirement	MoSCoW
F1	De refuel agent-trick is uit de source code verwijderd en toegevoegd op een nieuwe manier dat onafhankelijk van de broncode functioneert.	MUST
F2	De bestaande API kan zijn huidige functionaliteit blijven uitvoeren. **Acceptatiecriteria:** - Login werkt met de bestaande API. - Tricks kunnen de backend aanspreken om informatie op te halen.	MUST
F3	Tricks die geschreven zijn door developers moeten dynamisch ingeladen kunnen worden	MUST
F4	Bepaalde functionaliteit uit tricks, “widgets”, kunnen weergegeven worden op de dashboard pagina	COULD

Tabel 9: Functionele requirements

In hoofdstuk 6 worden er framework-specifieke requirements geïntroduceerd. Dit zijn meer technische requirements voor de oplossing. De project requirements en de framework-specifieke requirements zijn goedgekeurd door Bullit. De project requirements zijn ook goedgekeurd door Sebastian Piest.

6. Oplossing

Om de gewenste situatie te bereiken, zijn meerdere mogelijke oplossingen onderzocht. In dit hoofdstuk wordt het gekozen framework besproken. Eerst wordt er een conceptuele uitleg gegeven over de implementatie. Daarna wordt er technisch in detail uitgelegd hoe het framework het product dichterbij de gewenste staat heeft gebracht.

Dit hoofdstuk beantwoord de volgende deelvragen: ***DV3: Hoe stel ik developers in staat om tricks aan te bieden aan OfficeDog frontend?*** en ***DV4: Hoe stel ik developers in staat om widgets aan te bieden aan OfficeDog frontend?*** Dit zal worden gedaan door te beschrijven hoe de eisen uit hoofdstuk 5 zijn behaald.

6.1. Conceptueel

Bullit is op zoek naar manieren om externe softwareontwikkelaars in staat te stellen om tricks en widgets toe te voegen aan het OfficeDog-platform zonder dat zij toegang tot de broncode van OfficeDog nodig hebben. Dit probleem zal worden aangepakt door middel van een architecturale verandering aan de huidige frontend, waarbij een (gedeeltelijke) herziening van de huidige opbouw van de frontend noodzakelijk is.

De frontend van OfficeDog bevindt zich momenteel in een proof of concept fase en Bullit is zich bewust van de noodzaak om de frontend te herzien om aan de gebruikerswensen te voldoen en gebruiksvriendelijkheid te verbeteren.

Het gewenste resultaat is dat de ontwikkelaar een trick kan ontwikkelen via een manier dat de developer experience niet verslechterd en deze vervolgens gebruiksvriendelijk kan toevoegen aan OfficeDog. De trick moet dan kunnen draaien in de OfficeDog webapplicatie en een widget kunnen aanbieden die de gebruiker op de dashboardpagina kan weergeven. Het is vastgesteld dat het in de huidige architectuur niet mogelijk is om tricks toe te voegen op een onafhankelijke manier. Er is een oplossing nodig waarbij externe applicaties zichzelf kunnen aanmelden bij het platform en hierdoor geladen kunnen worden door het platform.

Om dit te bereiken zal een framework analyse worden uitgevoerd om te bepalen of een dergelijke oplossing haalbaar is. Indien dit niet het geval is, zullen alternatieven worden gezocht. De requirements zijn opgezet door eigen inzichten en goedgekeurd door Bullit. De F of NF geven aan of een requirement functional is (**F**) of non functional (**NF**).

Framework requirements:

#F/NF	Features	MoSCoW	Beredenering
R1/F	Dynamisch importeren van modules is ondersteund	MUST	OfficeDog wil een platform zijn waarin applicaties kunnen leven die gemaakt zijn door verschillende bedrijven. Deze bedrijven mogen niet ontwikkelen aan het platform (de source code).
R2/F	Connectiviteit met de bestaande API moet functioneel blijven zonder dat er aanpassingen aan de API gemaakt hoeven worden.	MUST	Bullit wil dat er een oplossing komt die gebruik kan maken van de huidige backend implementatie. Dit is omdat deze al in een verder gevorderd stadium is dan de frontend.
R3/NF	Bevat goede community support, is een populair framework en het framework heeft een gezonde support-omgeving	MUST	Dit garandeert over het algemeen goede langdurige support vanuit een community. Dit betekend dat de kans dat dit product langer onderhouden wordt en er bugs uitgehaald worden en features worden toegevoegd groter is dan bij producten waar dit niet het geval is.
R4/NF	Bevat goede documentatie	MUST	Wanneer Bullit verder gaat met het gekozen framework moeten zij goede documentatie kunnen raadplegen voor ontwikkeling/debugging.
R5/F	Developers kunnen hun eigen software tools en frameworks gebruiken bij het ontwikkelen van tricks en widgets voor OfficeDog.	SHOULD	Om de developer experience optimaal te houden wilt Bullit de developer in de mogelijkheid te stellen om de tools te gebruiken die zij passend vinden voor de trick/widget die zij willen bouwen.
R6/F	Kan gebruik maken van NPM of yarn	SHOULD	Om redelijk in structuur te kunnen blijven met de huidige producten die Bullit ontwikkelt willen zij gebruik maken van packages die npm en/of yarn aanbieden.
R7/F	Is te implementeren in de huidige frontend applicatie	COULD	Bullit wil dat er naar mogelijkheden gekeken wordt om zo min mogelijk code te refactoren. Als de oplossing verschaft kan worden in de huidige codebase is dat ideaal. **De huidige (frontend) codebase is in een beginnend stadium en daardoor is deze requirement een 'COULD'.**

Tabel 10: Framework requirements

Er is onderzoek gedaan naar de plugin-architectuur zoals beschreven door Apple Inc. (2013). De techniek bleek vooral gericht te zijn op backend-producten en een vergelijkbare oplossing als microservices. Een artikel (Shinde, 2021) stelt dat "plugin-architectuur" en "microservice-architectuur" niet echt architecturen zijn, maar architectuur patronen.

> *"Different microservices or plugins can be implemented by different teams pursuing orthogonal goals. These teams might not even know about each other's existence. This is only possible if microservices and plugins are loosely coupled and don't know about each other. (Shinde, 2021)"*

Contextuele quote Shinde

Als je de bovenste quote overweegt kun je conceptueel stellen dat microfrontends de "plugin architecture pattern" respecteren. Door onderzoek te doen naar het plugin-architectuur patroon kwam het concept van "microfrontends" naar voren. Bullit had ook al aangegeven dat zij iets dergelijks zochten.

"Microfrontends zijn bedacht als oplossing voor bedrijven die een steeds moeilijker onderhoudbare frontend monoliet aan het ontwikkelen zijn" (Knothe, z.d.).

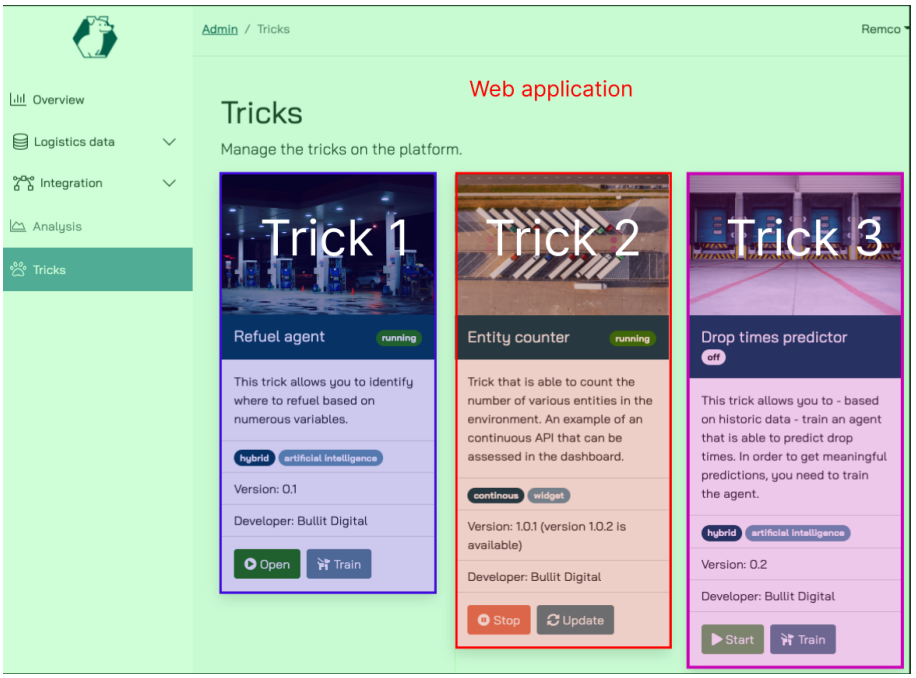
Een frontend monoliet is een softwareproject waarbij de gehele frontend in één grote codebase is gebouwd, met veel afhankelijkheden tussen de verschillende onderdelen. Dit kan leiden tot problemen zoals een groeiend en onoverzichtelijk project, en beperkte mogelijkheden om te innoveren. Een oplossing hiervoor is om de monoliet te splitsen in kleinere, onafhankelijke componenten, bekend als microfrontends of microservices. Alternatieven zijn niet overwogen omdat de probleemstelling van Bullit de use case van microfrontends perfect bewijzen (Plain Concepts, z.d.). Dit proces wordt ook wel "*breaking the monolith*" genoemd, en kan helpen om afhankelijkheden te verminderen en het product schaalbaarder te maken.

Bedrijven die aan één monolithisch frontend project werken kunnen tegen problemen aanlopen die in de backend projecten vaak al opgelost zijn met

microservices, het opbreken van de code in verschillende onafhankelijke projecten.

Microfrontends brengen dat concept naar de frontend.

Wanneer we dit toepassen op tricks binnen OfficeDog krijgen we het volgende beeld:



Figuur 10: Microfrontends in OfficeDog

De webapplicatie en tricks zijn onafhankelijk van elkaar ontwikkeld. Dit betekent dat de tricks kunnen worden overgebracht naar een ander microfrontend-project zonder dat dit problemen veroorzaakt.

Bullit is verantwoordelijk voor het ontwikkelen en beheren van de webapplicatie. Voor het beheer van de tricks is Bullit ook verantwoordelijk, maar niet noodzakelijkerwijs voor hun ontwikkeling. Het proces van het toevoegen van tricks aan OfficeDog valt binnen de scope van de opdracht, maar alleen op een minimal viable product niveau. Dit betekent dat er wordt aangetoond dat een microfrontend kan worden uitgevoerd binnen OfficeDog zonder onderdeel te zijn van de broncode.

Uiteindelijk is het doel dat de microfrontends worden gehost in een virtuele container en dat deze containers draaien in een schaalbare

kubernetes cluster. Dit aspect valt echter buiten de scope van de opdracht en zal later door Bullit worden opgepakt.

Figuur 10 toont een overzichtspagina van de Tricks die momenteel beschikbaar zijn in OfficeDog. De knoppen die worden weergegeven, vertegenwoordigen functionaliteit die specifiek is voor de acties die kunnen worden uitgevoerd met de tricks. Bijvoorbeeld, de EntityCounter kan alleen worden gestart/gestopt en bijgewerkt, terwijl de refuel agent een UI-element heeft dat zichtbaar wordt wanneer er op "open" wordt geklikt.

6.2. Framework vergelijking

Om aan de wensen van Bullit te voldoen, is gezocht naar frameworks die geschikt zijn voor microfrontends. Deze techniek, waarbij kleine onafhankelijke componenten worden gebruikt in plaats van één grote monolithische applicatie, is eerder genoemd in gesprekken met Martijn.

De onderzochte frameworks zijn beoordeeld op basis van de requirements uit de conceptuele uitleg. De MoSCoW-gradering is hierbij gebruikt als richtlijn voor de maximale punten die konden worden behaald. Onderzoek is gedaan naar hoe men een applicatie kan toevoegen aan een platform zonder dat zij volledige toegang hebben tot de broncode van dat platform. In de bron over de plugin architecture (Apple Inc, 2013) is te vinden dat de “plugins” afgestemd zijn op een backend implementatie al werd het duidelijk dat de plugin architectuur een design pattern is en niet onder de backend of frontend valt. Microfrontends respecteert dit design pattern en onderstreept des te meer de keuze voor microfrontends voor deze opdracht.

De volgende microfrontend oplossingen zijn vervolgens onderzocht: Module Federation, Single-spa en Qiankun. Elk van deze frameworks wordt beschreven als een geschikte oplossing voor microfrontends. De oplossing met de hoogste score wordt beschouwd als de meest geschikte oplossing voor de opdracht van Bullit.

Module Federation

Module Federation is een feature van Webpack die het mogelijk maakt om modules van verschillende projecten samen te voegen in één groter project, waardoor ontwikkelaars modules van andere projecten kunnen importeren en gebruiken zonder dat deze fysiek gekopieerd hoeven te worden. Dit biedt voordelen zoals meerdere teams op dezelfde codebase, verminderde duplicatie van code en verbetert flexibiliteit en onderhoudbaarheid. Het stelt ook ontwikkelaars in staat om een microservice-architectuur te implementeren in hun front-end code, waardoor de codebase in kleine, zelfstandige modules kan worden opgedeeld die gemakkelijk kunnen worden geüpgraded of vervangen.

Req	Score	Beredenering
R1	2.5	Module Federation kan complex zijn om op te stellen voor de developer, omdat het vereist dat de developer zich bewust is van de structuur van de verschillende microfrontends en hoe deze op de juiste manier geïmporteerd en geconfigureerd kunnen worden. De reden waarom deze requirement niet maximaal scoort is omdat er vragen open staan met betrekking tot volledige afhankelijkheid van de broncode en hier geen duidelijk antwoord op is gevonden.
R2	5	Module federation heeft geen mening over hoe er gecommuniceerd wordt met een backend. Hierdoor zullen bestaande API calls nog steeds werken en kan de data op dezelfde manier verwerkt worden als dat dat nu gebeurd.
R3	5	Module Federation is algemeen aangenomen. Module Federation komt nu ook mee met de nieuwste versie van Webpack.
R4	2.5	Module Federation is goed gedocumenteerd. Het is een van de populairste oplossingen voor de microfrontend architectuur (adservio, 2022). De zorg met betrekking tot documentatie ten behoeve van het dynamisch importeren blijft wel aanwezig bij Module Federation. Module Federation komt het best tot zijn recht bij het gebruiken van statische microfrontends. **Wel lijkt Module Federation doordat deze geïntegreerd is met Webpack parallel bruikbaar te zijn met andere microfrontend-frameworks die gebruik maken van Webpack.**
R5	2.5	Module Federation is een plugin voor Webpack. Hierdoor ben je gedeeltelijk vrij als developer om te kiezen wat je wilt zolang je framework samen kan werken met Webpack.
R6	1.25	Module Federation vormt geen mening over package managers. De frameworks die de developer kiest om in te ontwikkelen heeft hier wel impact.
R7	0.5	Module Federation kan geïmplementeerd worden in een Next applicatie, echter is hier refactoring nodig en de vraag met betrekking tot dynamische applicaties blijft bestaan.
TOT	**21.75**	

Tabel 11: Beoordeling Module Federation

Single-spa

Single-spa is een JavaScript-framework dat is ontwikkeld om microfrontends te creëren en te implementeren binnen één webapplicatie. Het maakt gebruik van een 'single-page-app' -architectuur, waarbij de webpagina zichzelf zal herladen als een gebruiker tussen verschillende delen van de webapplicatie navigeert. Dit maakt het mogelijk om verschillende onafhankelijke delen van een webapplicatie te ontwikkelen, te testen en te implementeren zonder dat deze delen afhankelijk zijn van elkaar. Afbakening van applicaties binnen een single-page-application is een use case van single-spa.

Req	Score	Beredenering
R1	5	Single-spa is een framework dat gebruik maakt van systemjs-importmaps om modules dynamisch te laden, zonder de noodzaak om deze modules vooraf te definiëren. Dit maakt het mogelijk om microfrontends toe te voegen aan het platform zonder toegang te hebben tot de broncode van OfficeDog.
R2	5	Single-spa vormt geen specifieke mening over hoe er gecommuniceerd moet worden met een backend, waardoor bestaande API's onveranderd kunnen worden gebruikt.
R3	5	Single-spa wordt ook genoemd in het populairste microfrontend framework artikel van adservio (adservio, 2022).
R4	5	Single-spa beschikt over uitgebreide documentatie en een actieve community die zich inzet voor de verdere ontwikkeling en verbetering van het framework. Bovendien biedt het bedrijf cursussen aan voor bedrijven die meer willen weten over het gebruik van het framework. Ook is er een slack-kanaal beschikbaar waar experts ontwikkelaars kunnen helpen bij eventuele problemen. Hoewel er ook een beperking is in de specifieke documentatie voor de use case van Bullit, wordt deze gecompenseerd door de extra ondersteuning en meer uitgebreide documentatie over dynamisch importeren door SystemJS, de module-loader die single-spa gebruikt.
R5	1.25	Single-spa heeft geen mening over welk framework er aangeboden wordt. Zolang er voldaan kan worden aan lifecycle methoden van single spa kunnen de microfrontends gebundeld worden. Soms vereist dit extra onderzoek van de developer als er geen single-spa library beschikbaar is voor zijn gewenste framework. Wel biedt single-spa genoeg documentatie aan de developer om hen te begeleiden in het maken van een microfrontend die aan de eisen van single-spa voldoet.
R6	2.5	Single-spa vormt geen mening over package managers.
R7	0	Single spa vereist een herschrijving van de huidige frontend. Dit komt omdat single-spa gebruik maakt van een root file waaruit gecompileerd wordt, deze file is niet in een Next applicatie te verkrijgen. Andersom werkt een NextJS microfrontend ook niet in single-spa, al zijn hier wel ontwikkelingen (Single-spa.js.org, 2021).
TOT	**23.75**	

Tabel 12: Beoordeling single-spa

Qiankun

Qiankun is gebaseerd op single-spa, echter beweert het framework meer tools aan te bieden voor het ontwikkelen van microfrontends. Hierdoor hebben zij een andere opvatting over hoe microfrontends geïmplementeerd zouden moeten worden.

Een groot nadeel van Qiankun is de documentatie (Qiankun Docs, 2022). Deze is naar Engels vertaald vanuit een Chinees origine, waardoor deze er rommelig uitziet en niet compleet is, met soms vreemde vertalingen. Bovendien is er een gebrek aan relevante informatie. Door de slechte documentatie, halveert of vermindert Qiankun de punten die gegeven worden voor de documentatie requirement. Hierdoor is Qiankun per definitie een minder geschikte keuze dan single-spa voor OfficeDog, ondanks de voordelen die het zegt te bieden.

6.3. Framework conclusie

Uit een vergelijking van verschillende frameworks, is gebleken dat single-spa het framework is met de meeste kans op succes voor de huidige opdracht. Bovendien lijkt het bij module federation dat er extra uitdagingen liggen in het kader van dynamisch laden van applicaties. Single-spa biedt deze functionaliteit "out of the box" met behulp van SystemJS importmaps.

Module Federation heeft als potentie dat de frontend niet ge-reworked hoeft te worden al behoeft dit nog meer onderzoek. Aangezien deze requirement niet belangrijk is voor Bullit is hier minder zwaar aan getild.

SystemJS import maps is een technologie die het mogelijk maakt om dynamisch modules te laden in een webapplicatie. Dit wordt bereikt door middel van het gebruik van import map configuratiebestanden. Deze bestanden bevatten informatie over welke modules beschikbaar zijn voor de applicatie en waar deze modules zich bevinden. Wanneer een developer een module wil importeren in de applicatie, zoekt SystemJS naar de locatie van de module in de import map configuratiebestanden. Hierdoor kan de developer modules laden zonder dat hij handmatig de locatie van de modules hoeft te kennen. Dit maakt het ontwikkelproces efficiënter en flexibeler, omdat de developer niet afhankelijk is van de interne structuur van de applicatie. Bovendien kan de applicatie

gemakkelijk worden uitgebreid met nieuwe functionaliteit zonder dat er aanpassingen aan de broncode hoeven te worden gemaakt. Figuur 11 is een voorbeeld van hoe de applicatie dynamisch gepopuleerd kan worden vanuit een API.

```
import * as SystemJS from 'systemjs';
import * as singleSpa from 'single-spa';

// Interval voor het ophalen van modules van de API
setInterval(() => {
    // Haal de dynamic tricks op uit de API
    fetch('api/dynamic-tricks')
        .then(response => response.json())
        .then((moduleUrls) => {
            // Ga door de lijst met module-URL's heen
            moduleUrls.forEach((moduleUrl) => {
                // Gebruik SystemJS om de module dynamisch te importeren
                SystemJS.import(moduleUrl)
                    .then((module) => {
                        // Voeg de module toe aan single-spa registry
                        singleSpa.registerApplication(moduleUrl, () =>
.                       module.default, location => true);
                    });
            });
        });
}, 30000);
```

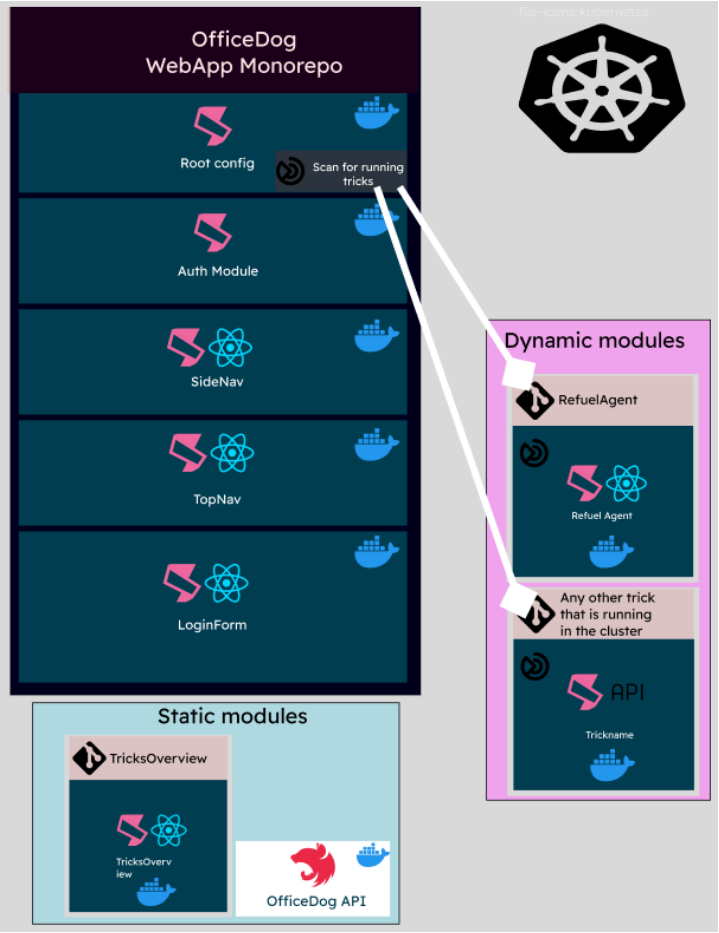
Figuur 11: Dynamisch laden van tricks in single-spa

6.4. Afbakening

In de afstudeerperiode is er duidelijkheid ontstaan over de gewenste eindsituatie voor OfficeDog en het dynamisch mee laten ontwikkelen van externe software teams aan OfficeDog. Hoewel er veel uitdagingen liggen binnen de gewenste eindsituatie, is het niet haalbaar om deze in één afstudeerperiode af te ronden.

De gewenste eindsituatie is dat OfficeDog gebruik maakt van kubernetes voor automatisering van container-orchestratie en dat tricks als docker containers in een kubernetes-cluster worden geplaatst. Het OfficeDog-platform zal vervolgens het cluster scannen op draaiende tricks en deze via de oplossing die na deze afstudeeropdracht ontwikkeld wordt inladen in de frontend van de applicatie. Dit maakt het eenvoudiger om de applicatie te schalen en te onderhouden, aangezien Kubernetes automatisch containers kan toevoegen of verwijderen afhankelijk van de hoeveelheid verkeer en de prestatie van de applicatie.

Figuur 12 is een voorbeeld van hoe deze situatie eruit kan komen te zien. Hoewel de backend van OfficeDog, inclusief de DevOps-aspecten, buiten de scope van deze afstudeeropdracht vallen, biedt het resultaat van dit onderzoek Bullit een antwoord op hoe zij uiteindelijk de tricks dynamisch in de OfficeDog frontend kunnen weergeven.



Figuur 12: Gewenste eindsituatie OfficeDog met Kubernetes

6.5. Technisch

Single-spa is een tool die wordt gebruikt om microfrontends te bouwen. Het biedt een manier om microfrontends te registreren en te beheren in een single page-application, en om deze microfrontends samen te voegen tot één geheel. Dit kan worden gecombineerd met een module loader, zoals SystemJS, om microfrontends te laden en te beheren.

De architectuur van single-spa ziet er als volgt uit:

- Microfrontends worden geregistreerd in het single-spa framework
- De te gebruiken microfrontends worden beschreven in de HTML van de "root"
- De "root" is de orchestrator van de microfrontends
- Microfrontends worden geladen en beheerd met SystemJS
- Microfrontends kunnen dynamisch worden geregistreerd en beheerd tijdens het uitvoeren van de applicatie in single-spa

Met single-spa kun je microfrontends dynamisch registreren en beheren terwijl de applicatie wordt uitgevoerd. Dit is laten zien in figuur 11.

SystemJS biedt ook een manier om modules te definiëren door het gebruik van een json file. Dit geeft de Bullit meer vrijheid bij het initialiseren van de modules (als zij bijvoorbeeld de broncode van OfficeDog willen uitbreiden).

```
{
  "imports": {
    "@bullit-digital/refueling-agent":
"http://localhost:9004/bullit-digital-refueling-agent.js"
  }
}
```

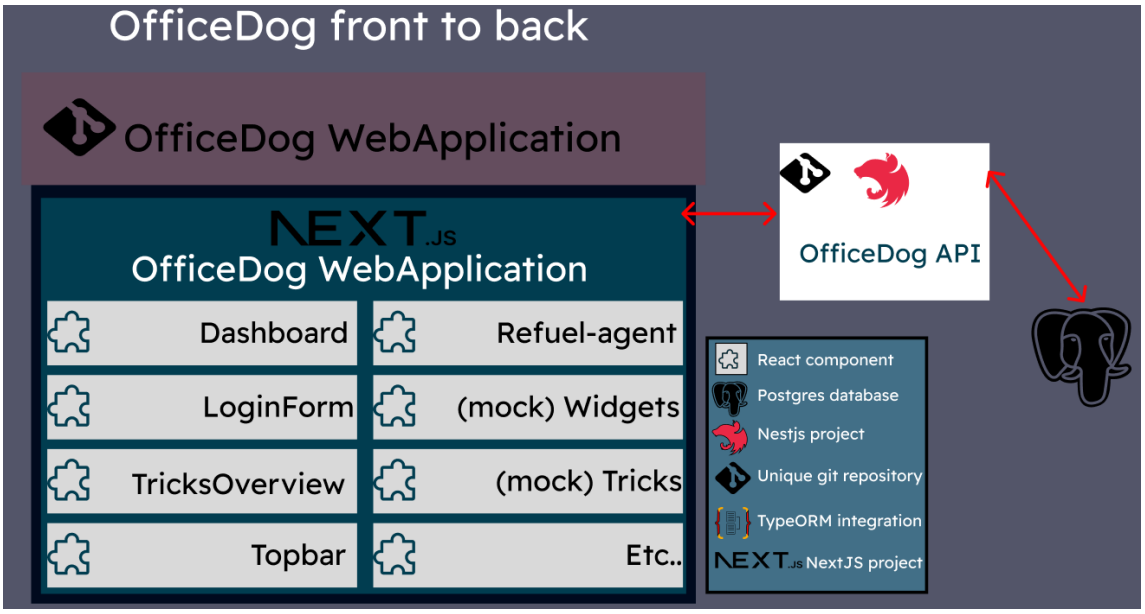
Contextueel voorbeeld van een SystemJS importmap met entry voorbeeld

Webbrowsers zijn in staat om meerdere system-js importmaps elementen te importeren, als de applicaties ook geregistreerd zijn bij single-spa zullen de applicaties laden. Op dit moment wordt alleen de refuel agent geladen via de importmap.json (figuur 13).

```
//index.ejs
<script type="systemjs-importmap"
src="https://officedog.bullit-test.com/officedog-microfrontends/packages/officedog-layout
-routing-engine/importmap.json">
</script>
```

Figuur 13: SystemJS importmap.json voorbeeld

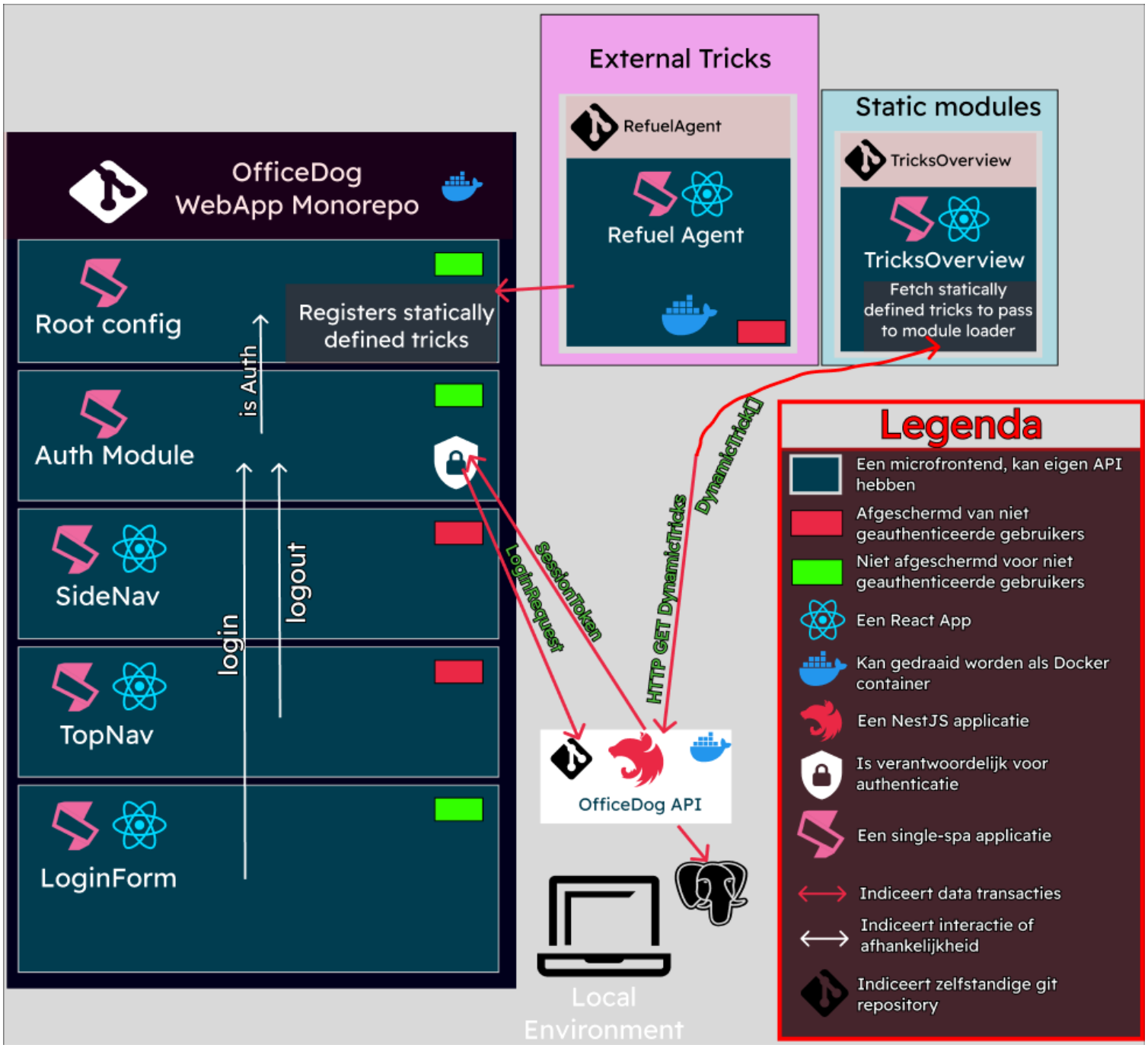
De huidige situatie van OfficeDog (figuur 14) is een monolithische structuur waarbij alle componenten van de applicatie in één enkel project zitten. Dit maakt het moeilijk om specifieke componenten te updaten of te onderhouden zonder de hele applicatie te moeten herladen. externe partijen moeten nu toegang hebben tot de source code om tricks te kunnen toevoegen. Dat is niet gewenst.



Figuur 14: Contextuele afbeelding van huidige situatie OfficeDog

In de nieuwe situatie van OfficeDog is de applicatie opgebouwd uit kleine, onafhankelijke microfrontends. Elk microfrontend heeft zijn eigen codebase waardoor elke applicatie individueel kan worden ontwikkeld, geüpgraded en getest. Dit maakt de applicatie flexibeler omdat nieuwe functies of verbeteringen kunnen worden toegevoegd zonder dat de hele applicatie opnieuw moet worden gebouwd of herladen.

Dit maakt de applicatie ook makkelijker te onderhouden omdat problemen of bugs kunnen worden geïsoleerd en opgelost zonder dat de hele applicatie getroffen wordt. Bovendien kan deze opbouw van microfrontends helpen bij het beter schaalbaar maken van de applicatie omdat microfrontends individueel kunnen worden geüpgraded en beheerd, waardoor de impact van de groei van de applicatie op de prestaties kan worden verminderd. Daarnaast wordt er gebruik gemaakt van een monorepo voor de statische onderdelen van de applicatie, zoals bijvoorbeeld de TopNav, SideNav en AuthModule, met behulp van yarn workspaces. Dit zorgt ervoor dat deze onderdelen, die de "source code" vertegenwoordigen waar externe developers geen invloed op mogen hebben, afgebakend functioneren van de dynamische onderdelen (tricks) van de nieuwe webapplicatie (figuur 15).



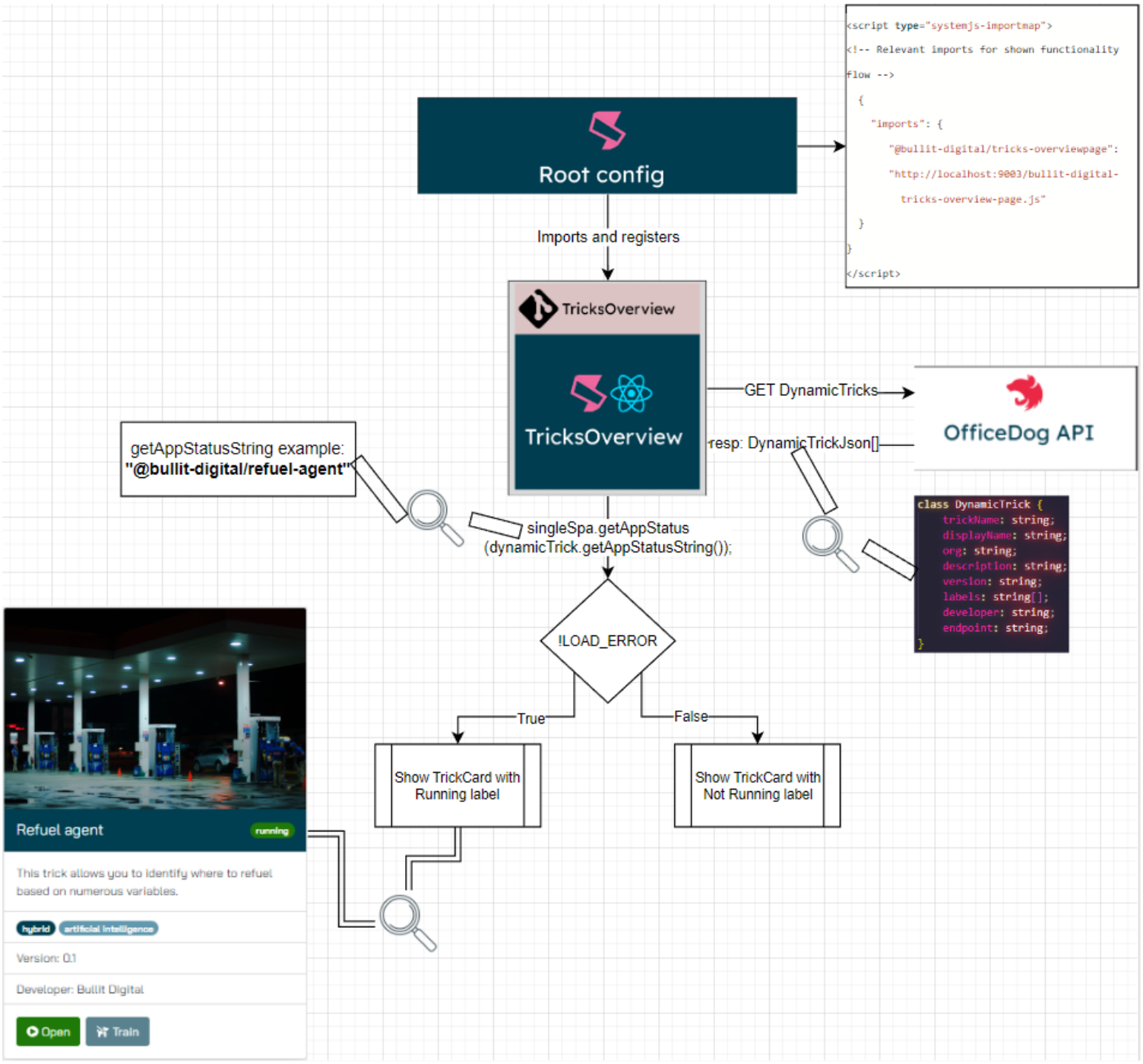
Figuur 15: Gerealiseerde architectuur

Om te laten zien wat er is veranderd in de architectuur van OfficeDog wordt getoond wat er is veranderd tijdens deze opdracht. Figuur 14 laat zien wat de huidige architectuur is van OfficeDog en figuur 15 laat zien hoe de architectuur eruit ziet na de rework. De screenshots van de rework en de huidige situatie is toegevoegd als bijlage G.

In de nieuwe opstelling is van de RefuelAgent een microfrontend gemaakt en deze is gedockerized. Dit is gedaan om te laten zien aan Bullit dat de gewenste eindsituatie van OfficeDog haalbaar is en om de stappen te laten zien die hiervoor gezet zijn.

Het plan van het Bullit is om deze tricks uiteindelijk in een kubernetes cluster te laten draaien en het OfficeDog platform zal vervolgens het cluster scannen op draaiende tricks en deze vervolgens in te laden in de frontend van de applicatie. Hoewel de focus niet op de backend-functionaliteit is gelegd tijdens deze afstudeerperiode, is de OfficeDog API wel aangepast om de dynamische tricks uit een GET-request te laden en in de frontend weer te geven.

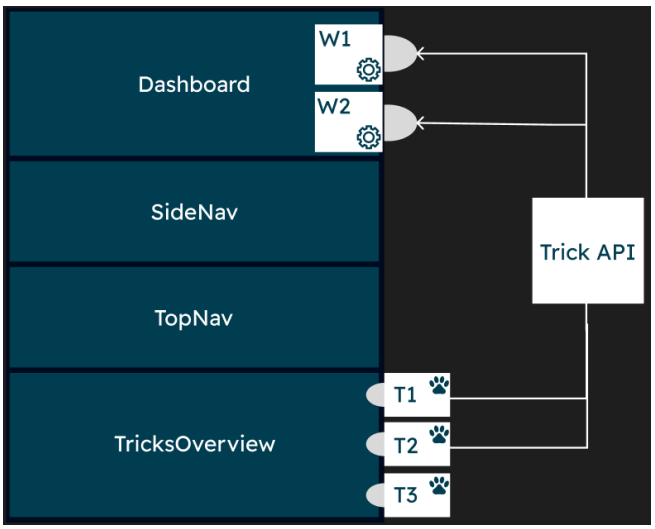
Een TrickCard is een component van het TricksOverviewPage project. Deze card geeft metadata weer en voorziet de gebruiker van opties met de Trick. Zo kunnen zij een trick bijvoorbeeld openen door op "open" te klikken. Zie hiervoor figuur 2. In de API is er een simpele controller gemaakt die een lijst van DynamicTricks teruggeeft. Deze worden vervolgens verwerkt in de TrickCard en dienen als een href naar de functionaliteit van de trick. Figuur 16 laat de flow zien van hoe dit tot stand komt.



Figuur 16: Flowchart dynamisch populeren TricksOverviewPage

7. Vervolg

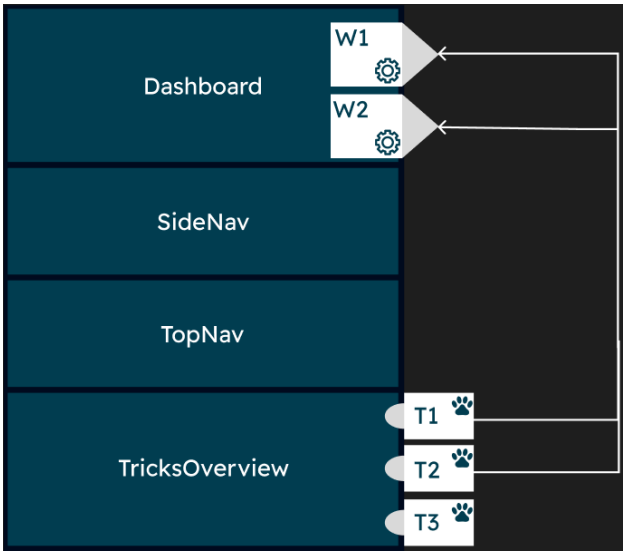
Advies voor Bullit is om single-spa te gebruiken voor de widgets. Dit biedt een losstaande applicatie van de trick applicatie, maar gebruikt dezelfde logica zodat de widget consistent is met de trick. Dit kan echter problemen opleveren voor onderhoudbaarheid, omdat wijzigingen in de trick ook aangepast moeten worden in het widget project. Een mogelijke oplossing hiervoor is door bedrijven hun eigen API te schrijven om de waardes tussen trick en widget gelijk te houden. Een alternatieve oplossing is om widgets te behandelen als kleine tricks binnen single-spa. Dit biedt echter wel development overhead omdat er drie applicaties nodig zijn (api, widget, trick) zodat de applicaties consistent blijven met elkaar. Figuur 17 laat zien hoe zo'n systeem eruit kan zien. De widgets en tricks worden van data voorzien vanuit een "Trick API". Deze kunnen samen aangeboden worden in de gewenste eindsituatie binnen een kubernetes cluster vanuit de externe developers. De trick, widget en Trick API zijn losstaande projecten van elkaar in dit voorbeeld.



Figuur 17: Widgets implementatie met single-spa

Een tweede advies is het gebruik van Webpack en Module Federation. Hiermee kunnen specifieke bestanden gedeeld worden tussen verschillende projecten. De ontwikkelaar moet de widget afbakenen binnen het project en deze bestanden via Module Federation exposeren op het netwerk. Figuur 18 laat zien hoe zo'n systeem eruit kan zien. De driehoeken suggereren dat de functionaliteit een uitbreiding is van de trick. Single-spa en Module Federation zijn te combineren (omdat een single-spa project ook een webpack configuratie kan hebben) maar

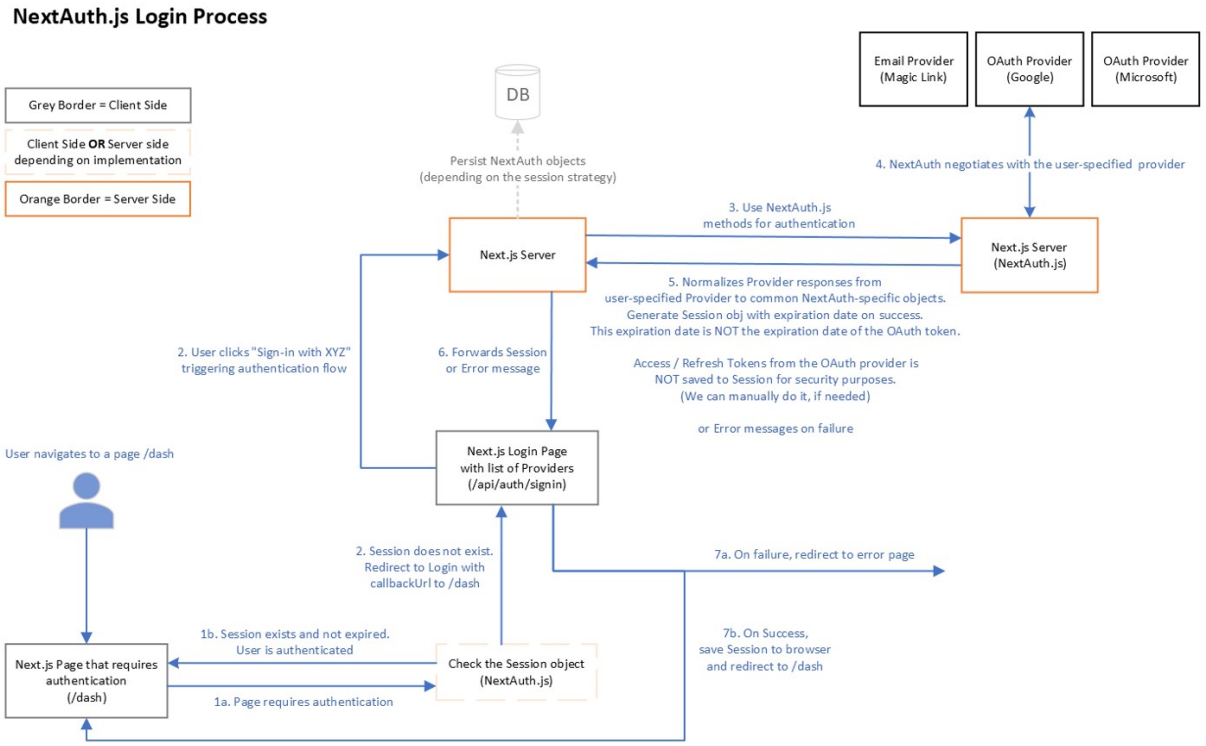
dezelfde problemen blijven bestaan met Module Federation als beschreven in hoofdstuk 6.2.



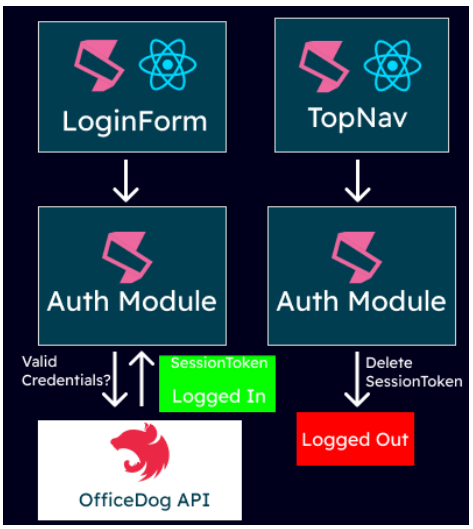
Figuur 18: Voorziene widget implementatie met behulp van Module Federation

De inlogfunctionaliteit is nog niet op hetzelfde niveau als het huidige project maar er wordt verwacht dat dit wel te realiseren is. Er is nog geen implementatie van de manier waarop Next refresh tokens uitdeelt en automatisch invalideert. Het is belangrijk voor Bullit dat de authenticatie op hetzelfde niveau blijft als het huidige platform.

Voor verduidelijking zijn figuur 19 en figuur 20 voorbeelden van NextJS authenticatie en de geïmplementeerde authenticatie module in het single-spa proof of concept respectievelijk. De versie uit het proof of concept is nog simpel maar bewijst wel dat authenticatie kan werken vanuit een microfrontend en gebruikt kan worden in een ander deel van de applicatie.



Figuur 19: NextJS authenticatie logica



Figuur 20: Implementatie single-spa OfficeDog met een AuthModule

Bij het aanmelden van tricks in de OfficeDog frontend moet er nog nagedacht worden over implicaties voor security. Er moet verzekerd worden dat microfrontends functionaliteiten binnen de widget/trick blijven en geen malicious code uit kunnen voeren in de broncode. Aangezien de tricks afgebakend opereren van de broncode worden hier geen implicaties verwacht maar het is verstandig om dit verder te onderzoeken.

Voor het aanbieden van documentatie aan de externe developers is er een start gemaakt met Storybook. Storybook is een open-source tool die ontwikkelaars helpt bij het documenteren en testen van hun componenten in bijvoorbeeld een React- of Angular-applicatie. Door gebruik te maken van Storybook kunnen externe developers eenvoudig de UI-componenten van de applicatie bekijken en begrijpen hoe deze gebruikt moeten worden in hun eigen ontwikkeling van tricks. Daarnaast kan Storybook ook gebruikt worden om de componenten te testen, zodat er zekerheid is dat de componenten werken zoals verwacht. Door het gebruik van Storybook, kunnen externe developers snel en efficiënt documentatie en informatie over de componenten verkrijgen, wat bijdraagt aan een verbeterde gebruiksvriendelijkheid voor de externe developers. De link moet gelegd worden tussen single-spa en Storybook. Naast dat Storybook componenten aanraad of afdwingt dient het een pagina te bevatten die developers uitlegt hoe zij kunnen beginnen met een trick maken met single-spa.

De huidige staat van de OfficeDog-storybook repository bevat een aantal componenten maar mist nog cruciale informatie met betrekking tot tricks en widgets. Om de ideeën over te brengen naar externe developers moeten hier stappen gezet worden om deze developers toegang te geven tot de Storybook documentatie van OfficeDog. Omdat Storybook wordt gebruikt kan deze ook iteratief uitgebreid worden om externe developers tricks de OfficeDog thema mee te geven zodat de tricks passen in de stijl van het platform.

Stappenplan voor Bullit:

1. Denk na over welke implementatie van widgets jullie willen:
    a. Gebruik single-spa voor de widgets. Dit biedt een losstaande applicatie van de trick applicatie, maar gebruikt dezelfde logica met behulp van een bijbehorend API project zodat de widget consistent is met de trick.
    b. Zoek uit hoe Module Federation onafhankelijk en dynamisch kan opereren binnen de nieuwe oplossing en implementeer dat.
2. Bedenk een oplossing voor onderhoudbaarheid:
    a. Bedrijven kunnen hun eigen API schrijven om de waardes tussen trick en widget gelijk te houden (single-spa widgets).
    b. Òf gebruik Webpack en Module Federation voor het delen van widgets. Dit zorgt voor een betere developer experience omdat een widget project niet afzonderlijk van het trick project hoeft te zijn.
3. Zorg dat de inlogfunctionaliteit op hetzelfde niveau is als het huidige project door uitbreiding van de AuthModule. Voor andere “service”-achtige functionaliteiten is het best practice om single-spa modules te maken zoals de AuthModule. Voor volwassenheid van het project is het noodzakelijk dat dit gebied verder onderzocht wordt.
4. Onderzoek implicaties voor security bij het aanmelden van tricks in de OfficeDog frontend. Zorg ervoor dat microfrontends functionaliteiten geen impact hebben op de source code. Bevestig dat deze fungeren als losse applicaties en bouw eventueel guards in om de beperking op te leggen.
5. Implementeer de kubernetes oplossing die in de pijplijn ligt. Maar gebruik SystemJS en single-spa om de opgehaalde microfrontends in te laden in de frontend zoals nu ook is gerealiseerd in de rootconfig waar de fetch methode naar de api/dynamictricks plaatsvindt.
6. Breidt Storybook uit zodat het documentatie bevat over hoe zij in de nieuwe opstelling kunnen beginnen met het ontwikkelen van tricks voor OfficeDog.
7. Breidt de TrickCard component uit in de TricksOverviewPage microfrontend om ook een afbeelding of referentie naar een afbeelding te kunnen bevatten zodat de developers kunnen kiezen welke afbeelding hun trick zal weergeven.

8. Conclusie & discussie

In dit hoofdstuk wordt beschreven hoe de afstudeerperiode is verlopen en hoe de requirements van het project zijn aangepakt. De hoofdvraag die centraal staat is "Hoe kan Bullit externe software ontwikkelaars gebruiksvriendelijk in staat stellen tricks en widgets toe te voegen aan de OfficeDog frontend?". Door middel van de deelvragen uit hoofdstuk 2.3 is de hoofdvraag beantwoord.

Om in het kort antwoord te geven op de hoofdvraag: Er is een duidelijke visie ontstaan tijdens mijn onderzoek voor de oplossing van de vraag. Er is een implementatie gemaakt van de gewenste situatie en hoewel deze nog niet compleet is - heb ik laten zien dat dit een oplossingsrichting is waar Bullit mee verder kan.

Het onderzoek richtte zich op het dynamisch toevoegen van tricks en widgets aan de applicatie. Er is een oplossing gevonden hiervoor, maar er zijn nog geen widgets geïmplementeerd in het proof of concept. Er is wel een visie ontwikkeld voor hoe widgets geïmplementeerd kunnen worden. De documentatie voor externe developers (gebruiksvriendelijkheid) is een start mee gemaakt maar vereist nog wat toevoeging van relevante onderdelen. Er is een duidelijke visie ontwikkeld met behulp van Storybook. Omdat OfficeDog nog in een "proof of concept" stadium is, is dit in overleg met Bullit lager op de prioriteitenlijst gekomen. Tricks kunnen dynamisch toegevoegd worden aan de API, maar ze komen nog niet uit een gewenste situatie van een kubernetes cluster.

Met de kennis uit hoofdstuk 6 kan de volgende conclusie worden getrokken: het platform kan bij starten een extern bestand uitlezen waarin modules zijn gedefinieerd en deze modules worden vervolgens weergegeven in het platform. Dit resulteert in het behalen van requirements NF1 en F3. Door gebruik te maken van single-spa, hebben developers de vrijheid om gebruik te maken van hun voorkeurstools, resulterend in het behalen van NF4.

De oplossing die wordt aangeboden in dit verslag geeft antwoord op de vraag van Bullit met betrekking tot onafhankelijke software ontwikkeling en bewijst dat dit in de praktijk haalbaar is. Er is geprobeerd om de huidige stack van Bullit te gebruiken voor de oplossing, maar dit bleek niet mogelijk vanwege de onvolledige ondersteuning van Next door single-spa. Zoals bijvoorbeeld React en single-spa wel geïntegreerd zijn. Zo is er een library “single-spa-react” beschikbaar. Dit bestaat nog niet voor Next apps.

Hoewel het waardevol is om verder te onderzoeken wat er al is bereikt met betrekking tot deze issue (Single-spa.js.org, 2021), is het momenteel onzeker of de makers van single-spa verder gaan met ontwikkelen van native ondersteuning voor NextJS.

Er werd aandacht besteed aan het implementeren van features die NextJS Bullit boodt in single-spa, zoals authenticatie. Er is gekeken of en hoe deze vraagstukken konden worden opgelost binnen single-spa. Om te bewijzen dat deze oplossing naar behoren werkt, is de login-functionaliteit in het proof of concept geïmplementeerd. Deze maakt nog steeds gebruik van de bestaande login-flow, maar is aangepast aan de manier waarop single-spa dit soort functionaliteit verwacht. Dit behaald F2.

De refuel agent is gebruikt als voorbeeld om te bewijzen dat tricks onafhankelijk van elkaar en van de root configuratie kunnen functioneren. Dit is gelukt zonder aanpassingen te maken aan de manier waarop de refuel-agent communiceert met de API, waardoor ook doelstelling F1 behaald is. Bijlage G laat ook zien hoe de nieuwe refuel agent functioneert in de microfrontend opstelling.

Om een overzicht te geven van de project requirements zullen tabellen 13 en 14 aangegeven worden welke requirements wel en niet behaald zijn. De focus van de opdracht lag vooral op het onderzoeken van de mogelijkheden. Er was nog geen echt plan voor ogen en dit is onderzocht om dat voor Bullit te maken.
In deze scriptie is aangetoond hoe single-spa kan worden gebruikt om microfrontends te implementeren in een webapplicatie. Dit stelt ontwikkelaars in staat om zelfstandig "tricks" toe te voegen aan OfficeDog. Dit kan leiden tot een flexibelere en schaalbare webapplicatie.

Daarnaast is aangetoond hoe dynamische importmaps kunnen worden gebruikt om module namen naar module URLs te mappen, en hoe deze importmaps kunnen worden opgehaald van een server terwijl de applicatie draait. Dit geeft Bullit de mogelijkheid om "live adding" toe te passen.

Voor F4 en NF2/NF5 is er een stappenplan ontwikkeld die beschreven staan in hoofd 7: Vervolg. Hierin kan Bullit een duidelijk vervolgplan vinden voor de requirements van het project. Hoofdstuk 7 geldt ook als adviesrapport en behaald daarmee NF3.

✅	#	Requirement	MoSCoW
	NF1	Developers hebben geen toegang nodig tot de source code tijdens het developen van tricks.	MUST
	NF2	De documentatie heeft de volgende eigenschappen: - Bevat tenminste hoe zij een trick kunnen ontwikkelen - Legt uit hoe een trick architecturaal in OfficeDog aansluit - Bevat codevoorbeelden voor het maken van tricks	MUST
	NF3	Er is een vervolgplan voor het widgets concept **Acceptatiecriteria:** - De oplossing is voorzien in een adviesrapport	MUST
	NF4	Developers kunnen hun eigen software tools en frameworks gebruiken bij het ontwikkelen van tricks voor OfficeDog.	MUST
	NF5	De developer heeft toegang tot documentatie met codevoorbeelden zodat zij begeleid kunnen worden tijdens het ontwikkelen van tricks. **Acceptatiecriteria:** - Documentatie is gebruiksvriendelijk om in te zien.	MUST

Tabel 13: Non-functionele requirements status tabel

✅	#	Requirement	MoSCoW
	F1	De refuel agent-trick is uit de source code verwijderd en toegevoegd op een nieuwe manier dat onafhankelijk van de broncode functioneert.	MUST
	F2	De bestaande API kan zijn huidige functionaliteit blijven uitvoeren. **Acceptatiecriteria:** - Login werkt met de bestaande API. - Tricks kunnen de backend aanspreken om informatie op te halen.	MUST
	F3	Tricks die geschreven zijn door developers moeten dynamisch ingeladen kunnen worden	SHOULD
	F4	Bepaalde functionaliteit uit tricks, “widgets”, kunnen weergegeven worden op de dashboard pagina	COULD

Tabel 14: Functionele requirements status tabel

	Behaald		**Deels behaald**

Tabel 15: Legenda requirements tabel

Bibliografie

Dot-framework. (2012). Een kader voor het ontwikkelen van vaardigheden en competenties voor de arbeidsmarkt. Geraadpleegd op 9 december 2022 van https://dot-framework.com/

Piest, J.P.S. (2022). Persoonlijke website van John P.S. Piest. Geraadpleegd op 9 december 2022 van https://people.utwente.nl/j.p.s.piest

CGI. (2022). Digitalisering en innovatie binnen de logistieke wereld. Geraadpleegd op 9 december 2022 van https://www.cgi.com/nl/nl/artikelen/logistiek/digitalisering-en-innovatie-binnen-de-logistieke-wereld

Codecademy.com. (n.d.). What JavaScript Framework to Learn [Beginner]. Geraadpleegd op 12 december 2022, van https://www.codecademy.com/resources/blog/what-javascript-framework-to-learn-beginner/

Outsystems. (2021, januari 12). How to build frameworks for developer experience. Geraadpleegd op 12 december 2022, van https://www.outsystems.com/blog/posts/how-to-build-frameworks-developer-experience/

Hackr.io. (n.d.). Web development frameworks. Geraadpleegd op 12 december 2022, van https://hackr.io/blog/web-development-frameworks

Schwaber, K., & Sutherland, J. (n.d.). The Sprint. In Scrum Guide. Geraadpleegd op 12 december 2022, van https://scrumguides.org/scrum-guide.html#the-sprint

SUTC, (n.d.) OpenTripModel. Geraadpleegd op 15 december 2022, van https://www.sutc.nl/en_US/open-trip-model

Knothe, R. (z.d.). Micro Frontends. Geraadpleegd op 19 december 2022 van https://micro-frontends.org/

Single-SPA. (z.d.). Getting started overview. Geraadpleegd op 19 december 2022 van https://single-spa.js.org/docs/getting-started-overview

(OpenTripModel, 2021) OpenTripModel. Geraadpleegd op 23 december 2022 van https://otm5.opentripmodel.org/

Apple Inc. (2013). Plugins. In Cocoa Developer Library. Geraadpleegd op 28 december 2022 van https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/LoadingCode/Concepts/Plugins.html

(Webpack 2020) Webpack release notes. Geraadpleegd op 02 januari 2023 van https://webpack.js.org/blog/2020-10-10-webpack-5-release/#module-federation

Top Microfrontend Frameworks (2022). Geraadpleegd op 02 januari 2023 van https://www.adservio.fr/post/top-micro-frontend-frameworks

(Chameera Dulanga, 2021) Top 5 Javascript Module Bundlers. Geraadpleegd op 02 januari 2023 van https://enlear.academy/top-5-javascript-module-bundlers-for-2021-e6fe17657fb

(Vercel, 2023) Next.js by Vercel. Geraadpleegd op 02 januari 2023 van https://nextjs.org/

(Meta Platforms Inc., 2023) React - A JavaScript library for building user interfaces. Geraadpleegd op 02 januari 2023 van https://reactjs.org

(Qiankun Docs, 2022) Qiankun readme. Geraadpleegd op 02 januari 2023 van https://qiankun.umijs.org/guide

Shinde, P. (z.d.). Plugin vs Microservice Architecture. LinkedIn. Opgehaald op 3 januari 2023 van https://www.linkedin.com/pulse/plugin-vs-microservice-architecture-pradip-shinde

Single-spa.js.org (2021). single-spa/single-spa.js.org #306: Next.js compatibility. Geraadpleegd op 14 januari 2023 van https://github.com/single-spa/single-spa.js.org/issues/306

Plain Concepts. (z.d.). Micro-frontends. Geraadpleegd op 16 januiari 2023 van https://www.plainconcepts.com/micro-frontends

Bijlagen

Alle bijlagen zijn samen met de scriptie ingeleverd. Hieronder geef ik alleen de titel weer van het bestand waar ik naar refereer in mijn scriptie.

Bijlage A: Opdrachtformulering Bullit
Bijlage B: Deskresearch “Gebruiksvriendelijk Ontwikkelen”
Bijlage C: Developer Interviews
Bijlage D: Interview Bullit Digital
Bijlage E: Developer Interviews Conclusie
Bijlage F: Interview Sebastian Piest
Bijlage G: Screenshots OfficeDog voor en na rework

Inhoudelijke reflectie

Situatie: In deze scriptie is onderzocht hoe externe ontwikkelaars mee kunnen helpen ontwikkelen aan het OfficeDog platform.

Taak: Door de refuel-agent uit als trick toe te voegen aan OfficeDog en deze geen onderdeel meer te laten zijn van de source code bewijzen dat de architectuur die Bullit voor ogen had haalbaar is.

Actie: Er is onderzocht hoe de refuel-agent als trick toegevoegd kan worden aan OfficeDog en hoe deze niet meer onderdeel is van de source code.

Resultaat: De architectuur die Bullit voor ogen had is haalbaar bewezen. Er zijn echter nog wel wat vraagstukken met betrekking tot de implementatie van OfficeDog bij de klant. Hier moet verder onderzoek gedaan worden naar de volgende onderdelen: Hoe realiseert Bullit dat een klant zijn tricks daadwerkelijk draaiend kan krijgen in OfficeDog? Hoe deelt een klant een trick met een concullega? Hoe maakt Bullit OfficeDog production-ready? Verder moet er nog meer van de huidige opzet overgezet worden naar de implementatie met single-spa op de nieuwe manier, met microfrontends. Module Federation lijkt de schoonste optie voor het implementeren van widgets in OfficeDog, single-spa is een sterk alternatief. Verder onderzoek is hier nodig.