Extended Abstract

IAM BINNEN MICROSERVICES

Finn Alberts, Lorenzo Clermonts, Giorgio Peerboom,
Bram Verheijen en Erik Wingers
Begeleiders: Daniël Heynen, Maurice Wollersheim en Marcel Perdok

ZUYD HOGESCHOOL | HBO ICT
IN SAMENWERKING MET CGI MAASTRICHT

Abstract

Broken access control wordt een steeds groter security concern. Zeker bij microservice-systemen, waarbij functionaliteiten gescheiden zijn over meerdere componenten is dit een steeds groter wordende uitdaging. Vanuit CGI Maastricht is daarom de wens om meer kennis en ervaring op gebied van identity and access management (IAM) op te doen. Dit vormt het doel van dit project. Binnen dit project is onderzoek gedaan naar IAM middels OAuth 2.0 en OpenID Connect. Hierbij is een vaccinatieregister als voorbeeld gebruikt. Een microservice-architectuur is hierbij als uitgangspunt genomen, maar de realisatie hiervan valt buiten de scope van dit project.

Om de kennis en ervaring op te doen is middels Kruchten 4 + 1 een ontwerp gemaakt op basis van de eisen voor een vaccinatieregister. Hierdoor is kennis en ervaring opgedaan op theoretisch niveau. Daarnaast is een prototype ontwikkeld, waarbinnen ook een stuk IAM is gerealiseerd. Dit prototype is ontwikkeld in Java met Quarkus en de open source IAM-provider Keycloak. Dit prototype is nog beperkt in functionaliteit, maar kan als basis dienen voor verder praktijkonderzoek.

1 Introductie

1.1 Aanleiding

Waar vroeger applicaties voornamelijk monolithisch ontwikkeld werden, wordt tegenwoordig steeds meer gekozen om applicaties modulair te ontwikkelen. Zo worden applicaties steeds meer ontworpen en gebouwd als een verzameling samenwerkende microservices. Binnen een monolithisch systeem zitten alle functionaliteiten in één applicatie. Bij microservices worden deze functionaliteiten juist opgesplitst in kleine stukjes waarbij elke microservice zich focust op een specifieke functionaliteit. Elke microservice kan hierdoor onafhankelijk van elkaar functioneren. Deze microservice-architecturen worden ook steeds vaker in de cloud gedeployed. (Richards & Ford, 2020)

Deze nieuwe manier van systeemontwikkeling brengt ook nieuwe uitdagingen met zich mee, ook op gebied van security. Eén van deze securityaspecten is authenticatie en autorisatie, ofwel identity and access management (IAM). IAM is een belangrijk securityaspect. Volgens OWASP was broken access control, dus het niet goed functioneren van IAM, het grootste beveiligingsrisico in 2021 (OWASP, 2021).

Bij monolitische applicaties is IAM een stuk eenvoudiger als bij modulaire applicaties. Bij monolitische applicaties wordt vaak gebruik gemaakt van een perimeter-based model, wat wil zeggen dat zodra een gebruiker door de beveiligingslaag heen is, er geen verdere controle meer plaatsvindt (Melony, 2021). Bij een microservice-architectuur is een zero-trust model gebruikelijker. Bij dit model wordt bij iedere actie opnieuw gecontroleerd of de gebruiker bevoegd is (Melony, 2021).

Ook bij CGI zien ze deze ontwikkeling. CGI is een consultancy bedrijf voor informatietechnologie en systeemintegratie en binnen dit project opdrachtgever. CGI is ontstaan in 1976 en heeft tegenwoordig kantoren over de hele wereld (CGI, 2022). Bij CGI in Maastricht is er de wens om aan de slag te gaan met IAM binnen microservices, om op deze manier meer te leren over hoe dit op een goede en vooral veilige manier kan worden geïmplementeerd. Daarnaast wil CGI graag bijdragen aan het overbrengen van kennis aan studenten. CGI wil daarom graag studenten betrekken bij dit onderwerp.

Om dit onderwerp tastbaar te maken en een concreet voorbeeld te hebben om IAM voor te realiseren, is gekozen om dit uit te werken voor een vaccinatieregister. Een vaccinatieregister heeft als voordeel dat er verschillende soorten gebruikers zijn met verschillende rechten en dat dit systeem daarnaast veel privacygevoelige informatie bevat. Dit vaccinatieregister zal dus middels een microservice-architectuur gebouwd worden en als systeem dienen, waarvoor IAM wordt uitgewerkt. Belangrijk om daarbij op te merken is dat het implementeren van IAM de focus heeft en het vaccinatieregister daarom niet compleet functioneel zal zijn.

1.2 Doelstelling

Het doel van dit project is het vergaren van kennis en ervaring over IAM met een zero-trust model en de implementatie hiervan. Hierbij is het uiteindelijke doel om kennis en ervaring hierover te vergaren voor een microservice-architectuur. Dit laatste valt echter buiten de scope van dit project, maar moet wel in acht worden genomen.

2 Theoretisch kader

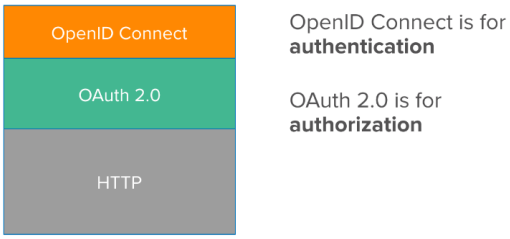
Binnen dit project zijn een aantal technieken en concepten, die van belang zijn bij de uitvoering van dit project. Hieronder zullen deze worden toegelicht, om een basisbegrip hierover vast te stellen.

2.1 OAuth 2.0 en OpenID Connect

OAuth 2.0 is een protocol voor autorisatie. Het wordt gezien als dé standaard binnen de industrie (OAuth, sd). Binnen OAuth 2.0 zijn er verschillende “flows”. De verschillende flows zijn (op technisch gebied) verschillende manieren om te autoriseren. Deze flows hebben ieder hun eigen use-cases.

OpenID Connect is een laag bovenop OAuth 2.0, welke gebruik wordt voor authenticatie. Dit wordt geïllustreerd in Figuur 1 (Raible, 2018). Deze laag maakt het mogelijk voor gebruikers om zich te authentiseren. De eindapplicatie heeft daarnaast de mogelijkheid om wat basis gebruikersinformatie op te halen over deze gebruiker, middels OpenID Connect. (OpenID, sd)

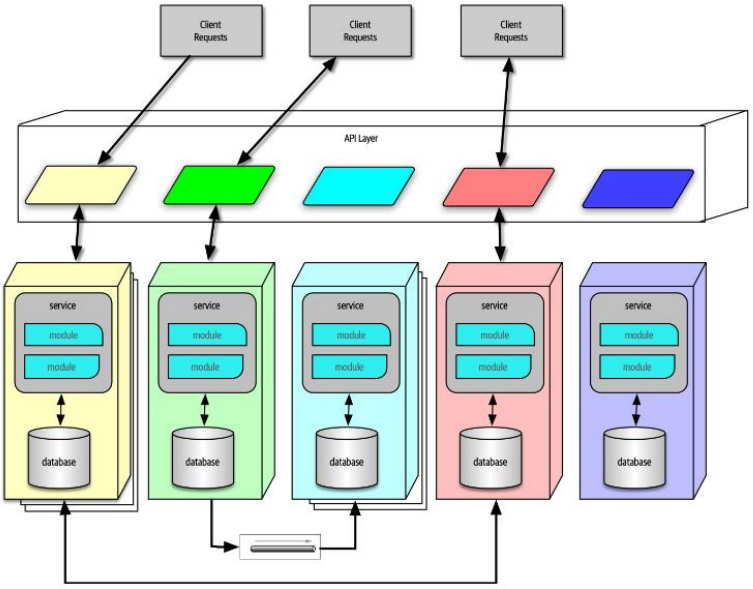
OAuth 2.0 en OpenID Connect vormen dus samen protocollen voor IAM.



Figuur 1 OpenID en OAuth2.0 structuur (Raible, 2018)

2.2 Microservice-architectuur

Een microservice-architectuur is een architectuur voor systemen, waarbij het systeem in meerdere componenten wordt gesplitst, die ieder slechts een klein deel uitvoeren. Doordat ieder van deze microservices erg klein is, kunnen deze snel worden geüpdatet, vervangen en getest. De microservices worden aangestuurd vanuit één (of meerdere) centrale API-gateway(s). De topologie van deze architectuur is te zien in Figuur 2. (Richards & Ford, 2020)



Figuur 2 Topologie van een microservice-architectuur (Richards & Ford, 2020)

3 Methode

Binnen dit project is gewerkt met een agile denkwijze. Om te borgen dat het project aan de wensen van de opdrachtgever voldoet, wordt wekelijks met deze samengezeten om vervolgstappen te nemen en ook een aantal vervolgstappen uit te zetten. Daarnaast vindt binnen deze gesprekken ook een stuk kennisoverdracht plaats.

Globaal zijn er binnen dit project drie stappen uitgevoerd. Als eerste zijn requirements vastgesteld, welke zijn geprioriteerd met de MoSCoW-methode (Wikipedia-community, 2021). Er is gekozen voor deze methode, omdat deze methode relatief eenvoudig is en een complexere methode niet noodzakelijk is voor dit project. Daarnaast is deze methode niet erg tijdsintensief.

De requirements beschrijven waaraan een volwaardig vaccinatieregister moet voldoen. Dat wil zeggen dat deze requirements het ontwikkelen van een vaccinatieregister als uitgangspunt hebben. Bij het realiseren van een prototype hiervoor is het ontwikkelen van dit vaccinatieregister echter niet het uitgangspunt. Het uitgangspunt is namelijk het onderzoeken hoe IAM binnen een microservice-architectuur hiervoor kan worden gerealiseerd. Door het vaccinatieregister wel als uitgangspunt te nemen voor de requirements, wordt echter wel duidelijk wat de belangrijkste elementen hiervan zijn en waar de beveiliging dus het belangrijkst is. Daarnaast geeft dit weer wat de belangrijkste processen zijn, waardoor het duidelijk wordt waar rekening mee gehouden moet worden bij het implementeren van beveiligingsmaatregelen.

Een vaccinatieregister is een privacygevoelig systeem. Het is daarom belangrijk dat dit systeem secure by design[1] is. Er is daarom een uitgebreid ontwerp gemaakt in de vorm van een architectuurbeschrijving. Voor het opstellen hiervan is gebruik gemaakt van de Kruchten 4 + 1-methode (Wikipedia-community, 2021). Centraal hierbij staan de scenario's (use-cases), waaromheen de vier views zijn ontworpen. Normaliter bestaan de views binnen Kruchten 4 + 1 uit een logic view voor de data in het systeem, een development view voor de verschillende componenten, een process view voor de processen en een deployment view voor de uitrol naar de fysieke hardware. Er is echter gekozen om geen deployment view te maken, omdat een uiteindelijke uitrol richting de cloud met Kubernetes (Kubernetes, sd) de voorkeur heeft voor de opdrachtgever. Hierdoor is een deployment view minder van belang binnen dit project. In plaats van deze view is een security-view opgesteld, waarin cross-cutting security aspecten zijn uitgewerkt. De overige views zijn uitgewerkt middels UML-diagrammen (Bennett, McRobb, & Farmer, 2010) met bijbehorende toelichting.

Voor het prototype is gebruik gemaakt van de programmeertaal Java met het Quarkus-framework. Het Quarkus-framework is gekozen, omdat dit framework geschikter is voor gedistribueerde en cloud-gerichte applicaties dan traditioneel Java (Quarkus, sd). Om het implementatieproces te vereenvoudigen wordt gebruik gemaakt van een kant-en-klare OAuth en OpenID Connect server. Er is hierbij gekozen voor Keycloak, omdat dit een open source product is, wat relatief eenvoudig is te implementeren (Keycloak, sd).

[1] Security by design houdt in dat tijdens het gehele ontwikkelproces, van ontwerp tot realisatie, wordt nagedacht over de security van een systeem. (Havinga, 2021)

4 Resultaten

4.1 Beschrijving artefact

De architectuurbeschrijving in combinatie met het prototype vormt het artefact binnen dit project. De architectuurbeschrijving laat zien hoe IAM met OAuth en OpenID Connect geïmplementeerd kan worden voor een vaccinatieregister. Het vaccinatieregister is hier een voorbeeld, waaruit lessen getrokken kunnen worden voor het implementeren van IAM in andere systemen.

Het prototype laat de basis zien van hoe OAuth en OpenID Connect geïmplementeerd kunnen worden. Het voornaamste doel van de applicatie is het aantonen hoe permissies gecontroleerd kunnen worden aan de hand van een token. De applicatie bevat verschillende API-endpoints. Verschillende rollen hebben toegang tot deze verschillende endpoints.

Omdat de front- en backend in het prototype niet gescheiden zijn, is het niet mogelijk om de authorization code flow te realiseren voor dit prototype (welke het best geschikt is voor autorisatie van de gebruiker). Daarom is gekozen voor de password code flow van OAuth (Auth0, sd). Deze flow is veel minder veilig en moet dus in een productieapplicatie niet worden overgenomen (Brady, 2019). Een verdere uitwerking van de te gebruiken flows is te zien in hoofdstuk 4.4 Security view.

4.2 Requirements en use-cases

Voor het vaccinatieregister zijn een groot aantal requirements opgesteld, welke een volwaardig vaccinatieregister beschrijven, en schetsen daarmee de context, waarvoor IAM gerealiseerd moet worden. Deze context is verder weergegeven in een use-casediagram (ofwel de scenario's binnen Kruchten 4 + 1), welke te zien is in Bijlage A – Use-casediagram. Dit diagram laat alleen de belangrijkste functionaliteiten zien, om zo het diagram overzichtelijk te houden.

4.3 Views van Kruchten 4 + 1

Er zijn binnen Kruchten 4 + 1 een viertal views gerealiseerd. Hiermee wordt het ontwerp voor het vaccinatieregister beschreven. De data die in het vaccinatieregister wordt opgeslagen is weergegeven in de logical view. Voor deze logical view is gebruik gemaakt van een klassendiagram. Dit diagram is te zien in Bijlage B – Klassendiagram.

Omdat het uitgangspunt een microservice-architectuur is, is dit ook weergegeven in de development view. Deze development view in de vorm van een componentendiagram laat alle componenten zien voor een vaccinatieregister en geeft daarbij ook de verbindingen tussen de verschillende componenten aan. Deze view is te zien in Bijlage C – Componentendiagram.

De derde view is de process view, waarin verschillende kernprocessen van het vaccinatieregister te zien zijn. De kernprocessen die zijn weergegeven zijn het zetten en registreren van een vaccinatie door een vaccinatiecentrummedewerker, het zetten en registeren van een vaccinatie door medisch personeel en het bijwerken van een geregistreerde vaccinatie door een GGD-medewerker. Elk van deze processen is weergegeven middels een sequentiediagram. Deze drie diagrammen zijn te zien in Bijlage D – Sequentiediagrammen.

De vierde en laatste view is de security view, welke de cross-cutting concerns op security gebied beschrijft. Deze view is verder beschreven in hoofdstuk 4.4 Security view.

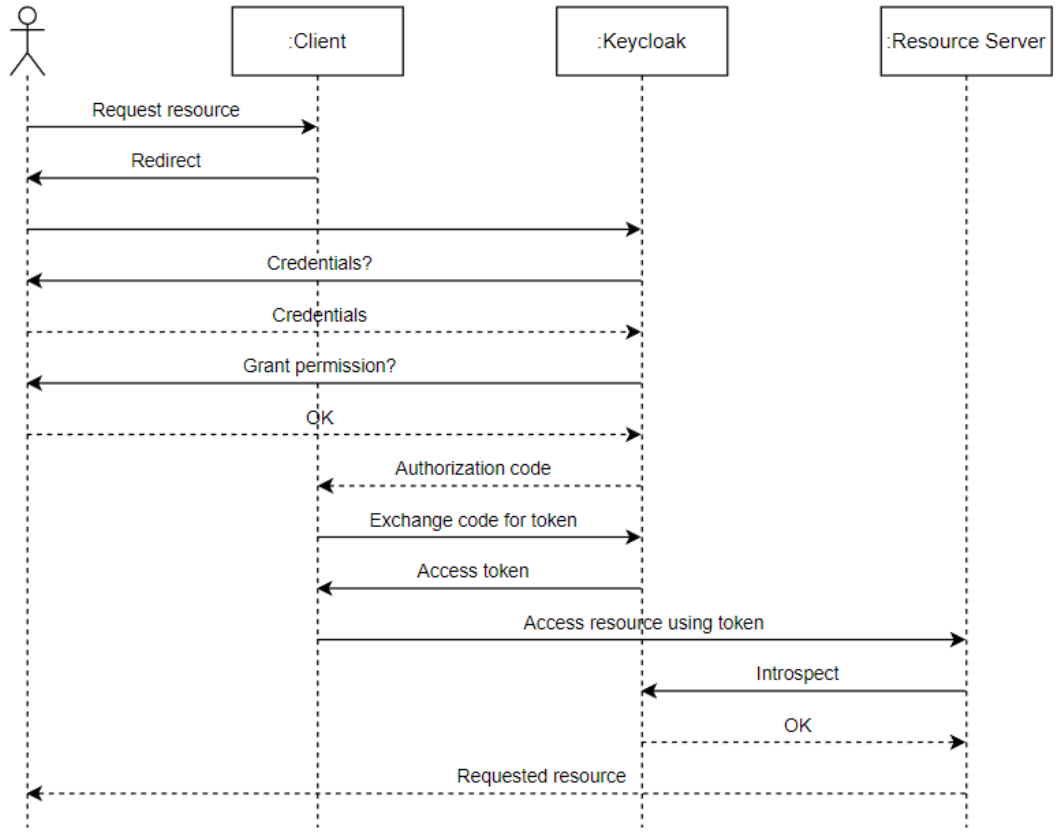
4.4 Security view

4.4.1 Identity and access management

Zoals eerder benoemd wordt voor IAM gebruik gemaakt van OAuth 2.0 en OpenID Connect. Binnen deze protocollen zijn verschillende flows mogelijk, met ieder zijn eigen use-cases. Voor het vaccinatieregister, wordt gebruik gemaakt van twee verschillende flows.

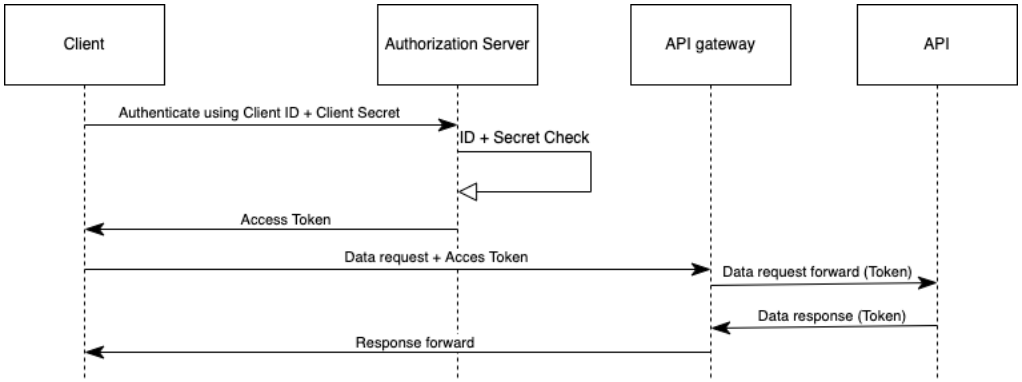
Voor de implementatie van OAuth en OpenID Connect wordt gebruik gemaakt van de open source IAM oplossing Keycloak (Keycloak, sd). Keycloak biedt de verschillende endpoints aan die nodig zijn voor de implementatie van OAuth en OpenID Connect. Hierbinnen worden ook de rollen en gebruikers gedefinieerd. Er is voor Keycloak gekozen, omdat deze open source is (wat wordt vereist vanuit de eisen) en een lage complexiteit heeft.

Voor communicatie met de client bij de eindgebruiker wordt gebruik gemaakt van de authorization code flow (Auth0, sd), te zien in Figuur 3. Binnen deze flow ontvangt de client eerst een authorization code. De client onvangt deze code en wisselt deze samen met zijn client secret (een code om de client te identificeren) in voor een access token. Deze access token biedt toegang tot de gevraagde functionaliteit. Doordat de access token nooit bij de gebruiker zelf terecht komt (enkel bij de client), wordt een extra laag van beveiliging gecreëerd.



Figuur 3 Authorization code flow

De tweede flow die wordt gebruikt is de client credentials flow (Auth0, sd). Deze flow wordt gebruikt voor communicatie tussen microservices onderling. Omdat hierbij geen gebruiker is, is het niet nodig om gebruik te maken van autorization codes. In plaats daarvan wisselt een microservice zijn client credentials (bestaande uit een client ID en client secret) direct om voor een access token, waarmee hij toegang verkrijgt. Deze flow is te zien in Figuur 4.



Figuur 4 Client credentials flow

Deze access tokens zijn beschikbaar als Opaque Token of als JWT-token. Een Opaque Token bevat enkel een code, waarmee de backend terug kan naar de autorisatieserver om de rechten op te halen. Bij JWT-tokens staan de rechten in de inhoud van de token zelf. Dit betekent dat een Opaque Token in de basis veiliger is, omdat het manipuleren van een token niet mogelijk is. Voor het vaccinatieregister wordt, vanwege de extra beveiliging, gekozen om gebruik te maken van Opaque Tokens.

Er wordt in het vaccinatieregister gebruik gemaakt van het zero-trust model. Dit wil zeggen dat voor iedere actie die een actor uit wil voeren opnieuw wordt gecontroleerd of de actor hiervoor gemachtigd is. Deze controle wordt meerdere malen uitgevoerd, wanneer een actie over meerdere microservices gaat. In dat geval vindt deze controle bij iedere microservice opnieuw plaats.

4.4.2 Encryptie

Om de vertrouwelijkheid van de data te waarborgen, wordt gebruik gemaakt van encryptie. Dit wordt gedaan voor zowel data in transport als data at rest.

Voor data at rest wordt gebruik gemaakt van het AES-256 encryptiealgoritme. AES is een zeer sterk encryptiealgoritme wat ook door de Amerikaanse overheid wordt gebruikt (Simplilearn, 2022). Door gebruik te maken van een sleutellengte van 256 bits is deze vorm van encryptie ook kwantumbestendig (Tobias, 2021). Hiermee is de encryptie ook toekomstbestendig.

Bij data in transit wordt gebruik gemaakt van TLS versie 1.3. TLS is momenteel de standaard voor het versleutelen van data in transport. Versie 1.3 is de meest recente en veilige versie van het protocol. (Cloudflare, sd)

4.5 Prototype

Het gerealiseerde prototype is een zeer versimpelde versie van het vaccinatieregister, gerealiseerd in Java met Quarkus (Quarkus, sd). Het prototype bestaat uit een API die verschillende CRUD-operaties aanbiedt voor de verschillende data-objecten. Voor ieder van deze API-endpoints is gedefinieerd welke rollen toegang hiertoe hebben. Deze toegang wordt ook afgedwongen. De verschillende rollen (en gebruikers) worden in Keycloak (Keycloak, sd) gedefinieerd. Het prototype is niet opgebouwd met microservices.

Vanwege beperkingen in de tijdsscope was het niet mogelijk om zowel een volwaardige front- als backend te realiseren. Doordat deze twee in één applicatie gecombineerd zijn, is het niet mogelijk de autorization code flow te implementeren. Daarom is gebruik gemaakt van de password code flow. Dit betekent dat de gebruiker middels een HTTP-request met gebruikersnaam en wachtwoord direct een access token kan ophalen. Daarnaast is er ook voor gekozen om in het prototype gebruik

te maken van JWT-tokens (in plaats van de veiligere Opaque Tokens). Buiten dat dit een lagere complexiteit heeft, maakt een JWT-token het ook beter mogelijk om te zien wat er gebeurd tijdens de flow. Hierdoor kan de flow en implementatie hiervan beter worden begrepen. In een productieapplicatie heeft een Opaque Token echter nog steeds sterk de voorkeur.

Om een API-endpoint aan te roepen is dus een JWT-access token nodig. Deze kan via een los API-endpoint worden opgehaald. Hierbij wordt de gebruiker omgeleid naar een inlogpagina. Vervolgens krijgt de gebruiker de access token, welke hij kan gebruiken om de verschillende API-endpoints (waar hij op basis van zijn rol voor geautoriseerd is) aan te roepen.

5 Discussie

Ondanks dat er veel kennis en ervaring is opgedaan op gebied van IAM met een zero-trust model, is er enige ruimte voor discussie. Een eerst discussiepunt is dat het microservices-architectuur aspect enigszins naar de achtergrond is gedreven. Het gerealiseerde prototype is niet opgebouwd uit microservices. De reden hiervoor is dat het begrijpen en implementeren van OAuth en OpenID Connect (welke nodig waren voor het realiseren van IAM) een groot deel van de tijd in beslag namen. Hierdoor was er binnen de tijdsscope geen ruimte om het prototype op te bouwen volgens een microservice-architectuur. In het gemaakte ontwerp is echter wel onderzocht hoe OAuth en OpenID Connect binnen microservices onderling werken. De client credentials flow die hiervoor kan worden gebruikt is helaas niet in het prototype gerealiseerd.

Een tweede punt van discussie is dat het prototype als redelijk beperkt kan worden gezien. Belangrijk is om hierbij op te merken dat het prototype vooral moet worden gezien als applicatie, waarmee geëxperimenteerd kan worden om de OAuth en OpenID Connect flows beter te begrijpen. Daarnaast biedt het prototype een hands-on ervaring. Helaas was het vanwege de tijdsscope niet mogelijk om een uitgebreider prototype te realiseren.

Een derde en laatste punt van discussie is dat er is afgeweken van de standaard views van Kruchten 4 + 1. In plaats van een physical view is er voor een security view gekozen. Er kan beargumenteerd worden dat er vanuit een security oogpunt ook goed moet worden gekeken naar de infrastructuur waarop een systeem draait. Voor IAM is deze view echter minder belangrijk. Daarnaast is de wens om verder onderzoek te doen naar IAM middels OAuth en OpenID Connect in de cloud met Kubernetes. Dit is een aspect wat buiten de scope van dit project valt.

6 Conclusie

De doelstelling van dit project was het vergaren van kennis en ervaring over IAM met een zero-trust model en de implementatie hiervan binnen een microservice-architectuur.

Binnen dit project is kennis en ervaring opgedaan over IAM met OAuth 2.0 en OpenID Connect. Hierbij is uitgegaan van een zero-trust model. De opgedane kennis is voornamelijk theoretisch en is uitgewerkt in een aantal views binnen Kruchten 4 + 1. Vooral binnen de security view is veel kennis verzameld, met name over de verschillende flows binnen OAuth die gebruikt kunnen worden. Hierbij staan de authorization code flow en client credentials code flow centraal voor een microservice-architectuur.

Een deel van deze theoretische kennis is ook terug te zien in het gerealiseerde prototype. Dit prototype, waarbij Keycloak wordt gebruikt als identity en access server, laat vooral zien hoe rechten afgedwongen kunnen worden, zodra een actor een access token heeft ontvangen. De wijze waarop een actor een access token ontvangt verschilt per flow.

Verder onderzoek is nodig om meer kennis en ervaring op te doen over de implementatie van OAuth 2.0 en OpenID Connect. Omdat de vergaarde kennis binnen dit project voornamelijk theoretisch is, zou een vervolgproject zich voornamelijk op meer praktijkervaring moeten richten. Het gerealiseerde prototype kan hiervoor als basis worden genomen.

Er kan op een aantal punten meer praktijkervaring worden opgedaan. Een eerste punt is het onderzoeken hoe het scheiden van de front- en backend in de praktijk werkt. Daarnaast kan ook worden onderzocht hoe de autorization code flow geïmplementeerd kan worden. Verder is een aspect, waarvoor meer onderzoek in de praktijk noodzakelijk is, het implementeren van IAM met OAuth en OpenID Connect binnen microservices. Het laatste aspect, waarvoor een vervolgonderzoek kan worden uitgevoerd, is het onderzoeken van IAM binnen de cloud met Kubernetes.

7 Verwijzingen

Auth0. (sd). *Authorization Code Flow*. Opgehaald van Auth0 Docs: https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow

Auth0. (sd). *Client Credentials Flow*. Opgehaald van Auth0 Docs: https://auth0.com/docs/get-started/authentication-and-authorization-flow/client-credentials-flow

Auth0. (sd). *Resource Owner Password Flow*. Opgehaald van Auth0 Docs: https://auth0.com/docs/get-started/authentication-and-authorization-flow/resource-owner-password-flow

Bennett, S., McRobb, S., & Farmer, R. (2010). *Object-Oriented Systems Analysis and Design Using UML 4th Edition.* Maidenhead: McGraw-Hill Higher Education.

Brady, S. (2019, Maart 28). *Fact Sheet: The Dangers of Using the Password Grant Type with Mobile Applications*. Opgehaald van IdentityServer: https://www.identityserver.com/articles/fact-sheet-the-dangers-of-using-the-password-grant-type-with-mobile-applications

CGI. (2022). *Offices*. Opgehaald van CGI: https://www.cgi.com/en/offices

Cloudflare. (sd). *What is the difference between TLS 1.3 and TLS 1.2?* Opgehaald van Cloudflare: https://www.cloudflare.com/learning/ssl/why-use-tls-1.3/

Havinga, E. (2021, april 30). *Security by Design concreet gemaakt*. Opgehaald van IB&P: https://ib-p.nl/2021/04/security-by-design-concreet-gemaakt/

Keycloak. (sd). *Open Source Identity and Access Management*. Opgehaald van Keycloak: https://www.keycloak.org/

Kubernetes. (sd). *Kubernetes*. Opgehaald van Kubernetes: https://kubernetes.io/

Melony, S. (2021, juli 2). *Zero Trust vs. Perimeter-Based Security*. Opgehaald van ITBriefcase: https://www.itbriefcase.net/zero-trust-vs-perimeter-based-security

OAuth. (sd). *OAuth 2.0*. Opgehaald van OAuth: https://oauth.net/2/

OpenID. (sd). *Welcome to OpenID Connect*. Opgehaald van OpenID: https://openid.net/connect/

OWASP. (2021, oktober 1). *OWASP Top Ten*. Opgehaald van OWASP: https://owasp.org/www-project-top-ten/

Quarkus. (sd). *What is Quarkus*. Opgehaald van Quarkus: https://quarkus.io/about/

Raible, M. (2018, December 19). *Spring Boot 2.1: Outstanding OIDC, OAuth 2.0, and Reactive API Support*. Opgehaald van Okta: https://developer.okta.com/blog/2018/11/26/spring-boot-2-dot-1-oidc-oauth2-reactive-apis

Richards, M., & Ford, N. (2020). *Fundementals of Software Architecture.* Sebastopol: O'Reilly Media.

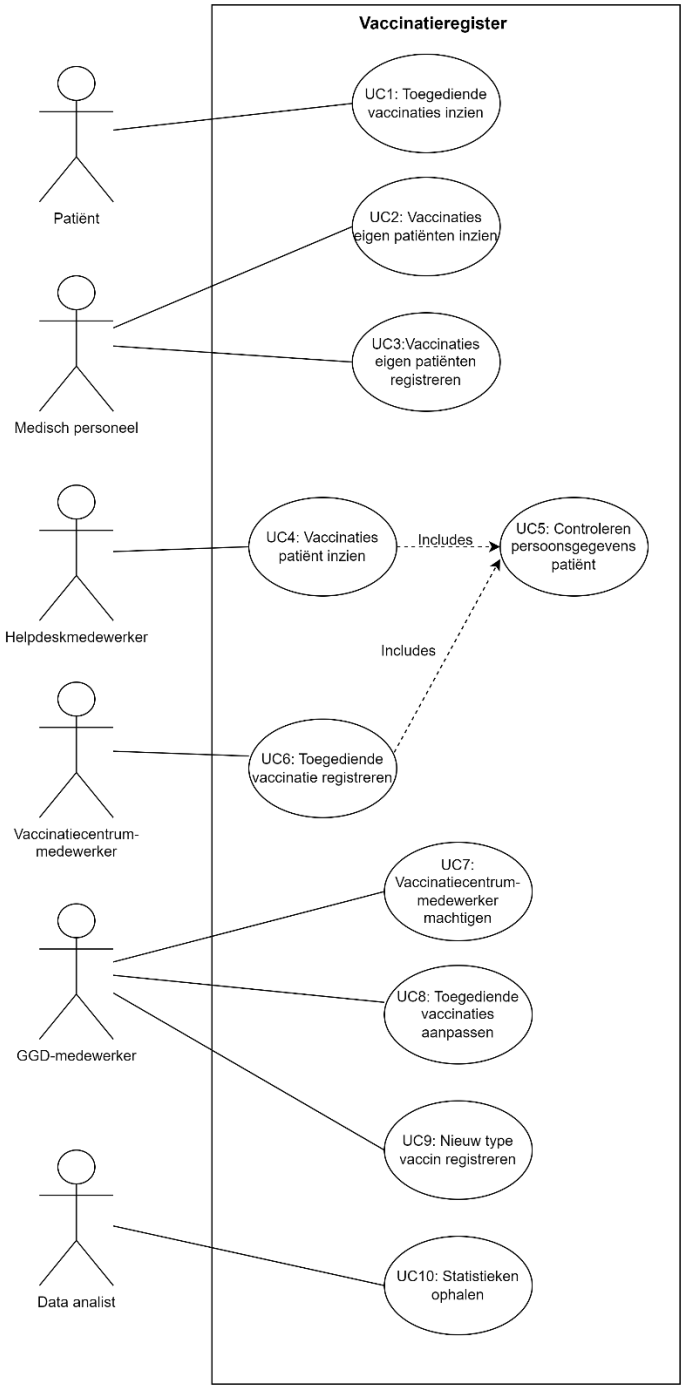
Simplilearn. (2022, september 14). *What Is Data Encryption: Types, Algorithms, Techniques and Methods*. Opgehaald van Simplilearn: https://www.simplilearn.com/data-encryption-methods-article

Tobias, E. (2021, februari 15). *128 or 256 bit Encryption: Which Should I Use?* Opgehaald van Ubiq: https://www.ubiqsecurity.com/128bit-or-256bit-encryption-which-to-use/

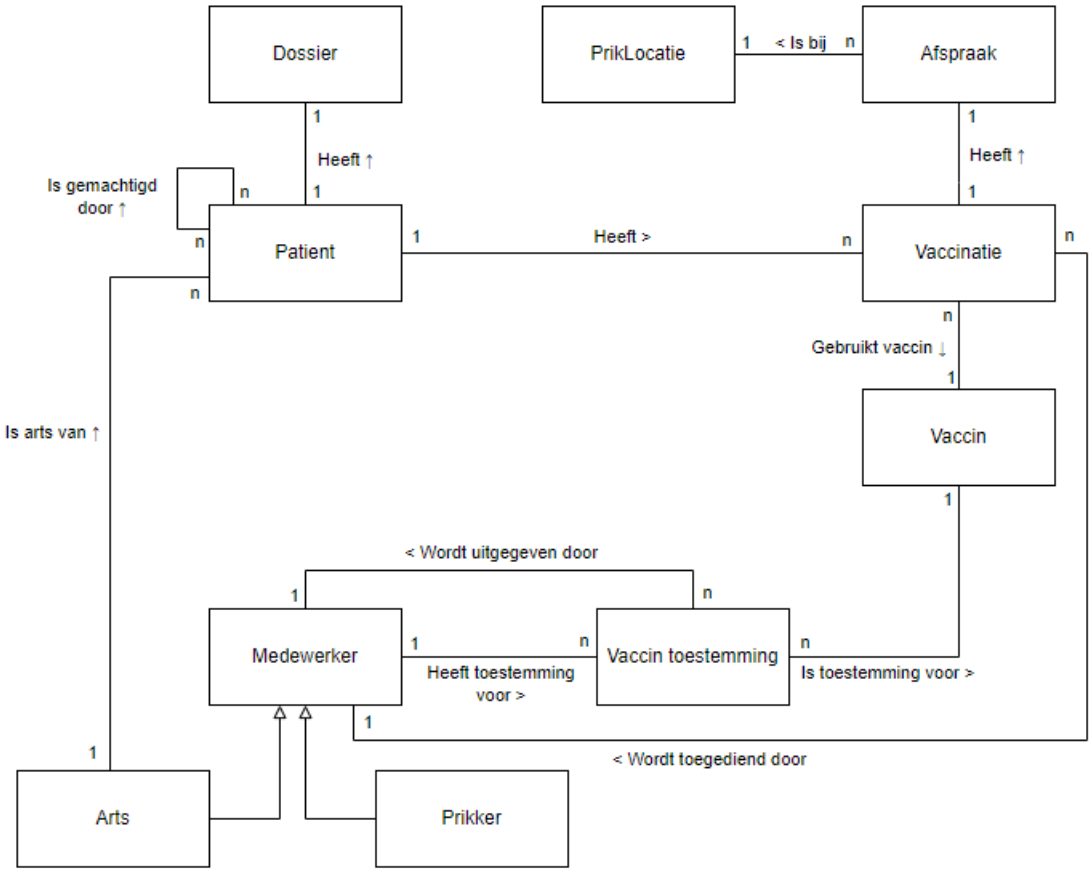
Wikipedia-community. (2021, december 13). *4+1 architectural view model*. Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model

Wikipedia-community. (2021, april 30). *MoSCoW-methode*. Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/MoSCoW-methode

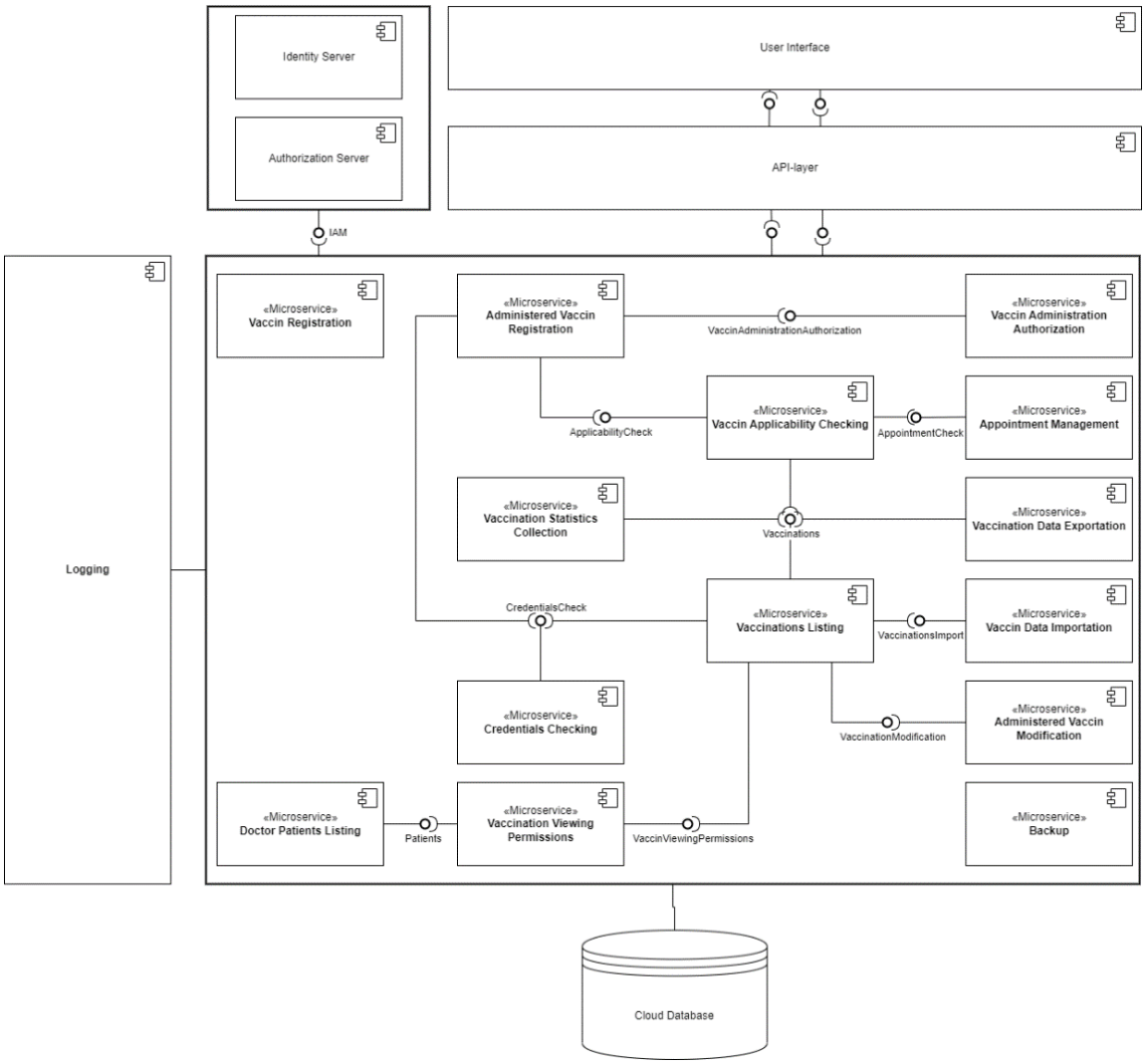
Bijlage A – Use-casediagram



Bijlage B – Klassendiagram

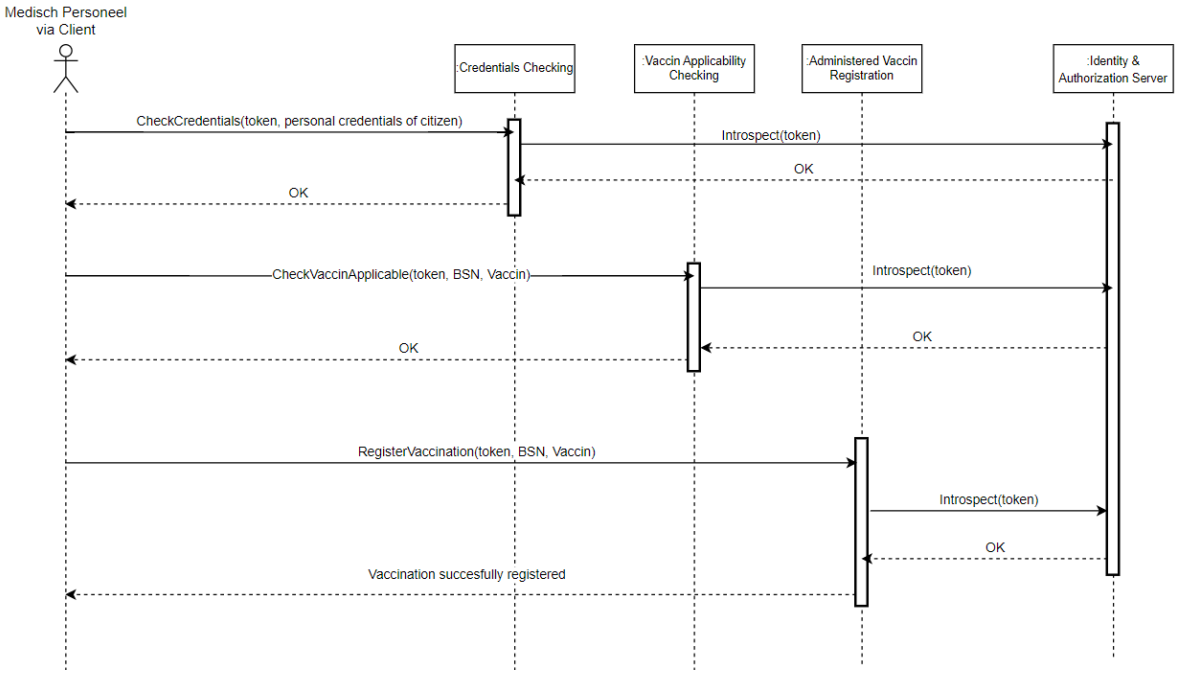


Bijlage C – Componentendiagram

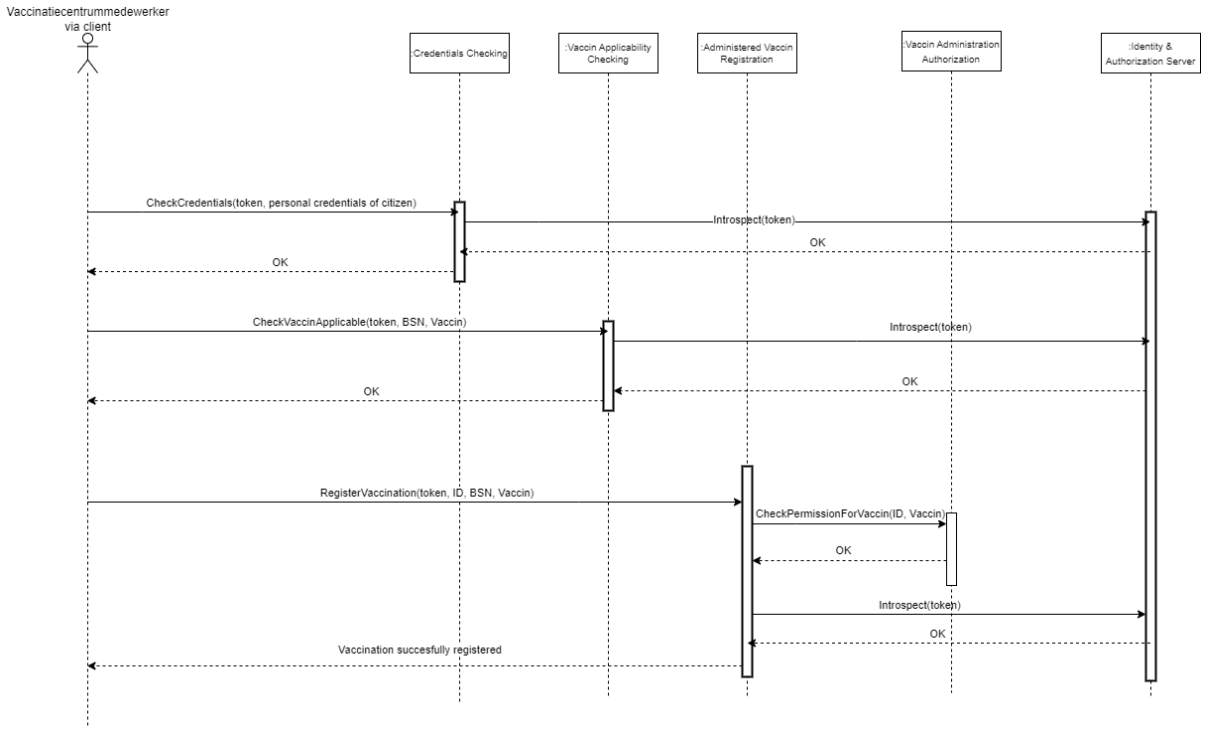


Bijlage D – Sequentiediagrammen

Registeren van vaccinatie door medisch personeel



Registreren van vaccinatie door vaccinatiecentrummedewerker



Bijwerken van vaccinatie door GGD-medewerker