|  |
| --- |
| OPDRACHT 3: GET TO KNOW ADVANCED IAM PRACTICES  3ITCSC1 |
| **Opleiding: Cloud & Cybersecurity**  **Academiejaar: 2023-2024**  **Joey de Waal** |

AP.BE

Inhoud

[Titel 1](#_Toc455153993)

[1 Kop 1 1](#_Toc455153994)

[1.1 Kop 2 1](#_Toc455153995)

[1.1.1 Kop 3 1](#_Toc455153996)

# Voorwoord

Tijdens

# API

Mijn applicatie is geen bestaand project waar ik verder aan heb gewerkt. Het is een SPA waar je bij kunt inloggen via Auth0 (OIDC). De backend zorgt voor een verbinding met een eenvoudige in memory database. De functionaliteit van de webapplicatie is heel beperkt, het is een basic todo app. Het is mogelijk om een activiteit toe te voegen en te lezen. Indien de gebruiker een email heeft die niet van school is kan deze zijn activiteiten verwijderen.

# Open ID Connect uitwerking

## Frontend

Langs de voor kant heb ik gebruik gemaakt van React om de SPA te laten werken.

Binnen de webpagina werk ik met de oidc-client-ts library om de integratie met OIDC eenvoudig te maken. Met redelijk weinig code kon ik OAuth2.0 samen met Open ID Connect toepassen. Deze code gebruik ik om een nieuwe OidcClient aant te maken met bepaalde parameters zoals client\_id en authority.



De startpagina bestaat uit een eenvoudige link om in te loggen. De code die gebruikt wordt om in te loggen is redelijk eenvoudig, door gebruik te maken van de OidcClient wordt er een URL aangemaakt en wordt de browser hiernaar verwezen.



Als de gebruiker is ingelogd wordt deze terugverwezen naar de website, dan worden gegevens zoals email, naam, ID Token, acces token in de sessie storage gestoken. Wanneer de gebruiker de website refresht moet er niet nog eens worden ingelogd. Hierna wordt deze gerouteerd naar de “/home” route zodat de webapplicatie gebruikt kan worden.



De homepagina ziet er zeker niet fancy uit maar ze werkt.

A screenshot of a computer

Description automatically generated

De code hiervoor lijk me net iets te veel om in deze pdf te zetten, indien u geïnteresseerd bent is **hier!!!!!** een link naar GitHub.

## Backend

Als backend maak ik gebruik van het framework [Axum](https://github.com/tokio-rs/axum) in [Rust](https://www.rust-lang.org/). De API heeft 1 route die 3 verschillende methodes accepteert, GET, POST, DELETE.



De database is een hele eenvoudige in memory HashMap. De key van de HashMap is de naam van de gebruiker, de waarde is een lijst van todos.



Telkens wanneer de SPA een verzoek doet richting de API wordt er een Authorization header bij in gedaan. Deze wordt telkens er uitgehaald en naar de Auth0 gestuurd om gevalideerd te worden. Hier heb ik wat middleware voor moeten schrijven die er zo uit ziet.



## Opmerkingen

Tijdens het uitwerken van het OIDC inlog systeem waren er enkele dingen die misliepen of niet werken zoals ik zou willen. De acces token die Auth0 terugstuurt is niet in JWT-formaat waardoor ik deze naar een endpoint moet sturen om deze zo te valideren. Dit zorgt voor meer complexiteit en voor vertraging aan de kant van de API om verzoeken te valideren.

# Open Policy Agent integratie

Na veel documentatie te lezen heb ik een hele minimale integratie met OPA. De policy kijkt of de email die doorgestuurd wordt, begint met de letter s. Zo weet ik of de email van de gebruiker een school email is.

Deze code wordt gebruikt om via de API een policy aan te maken.



De API gebruikt de volgende code om de OPA rest API aan te sturen om de email van te valideren.



In de API kan ik redelijk eenvoudig verifiëren of de gebruiker er mag zijn.



# Docker integratie

# Threat model