AUTHENTICATION RESEARCH

Kevin Oomen

FONTYS EINDHOVEN

Inhoud

What (What is the research question)

I want to know what I could use for authentication my application to make my application more secure and more friendly towards users. If I integrate authentication in my application, it will be a lot safer to use the application because all of the user's data will only be able to be accessed by people who have the right to view the data. In this research I want to know what authentication exactly entails and how I can use it in my application.

How (how will the research be performed)

The DOT-framework makes use of 5 research strategies:

Library

You use information that has already been made. You do this by checking if the information fits in the scope of your project and if it has any value to your project. If this is the case, you can use the information to further help you in your research.

Field

You use information about your users. You will have to make sure that your user's requirements get fulfilled. The way this is done is by doing interviews with your users and investigate what they would want to see.

Workshop

You mainly do this by prototyping. With prototyping I mean making a proof of concept for example. The reason for doing this is to see if you will get the desired results.

Lab

You use test methods to see if your solution has delivered the desired results, you can do this by using: security, system, unit and static test methods.

Showroom

Show what makes your solution different from others by communicating with experts or by comparing similar solutions to yours.

The information I have gathered to create this part can be found here.

Why (the reason for the research)

What does the research solve?

I want to be able to manage my users by using an identity provider. This way I have decentralised way of authenticating users across multiple services.

What is authentication?

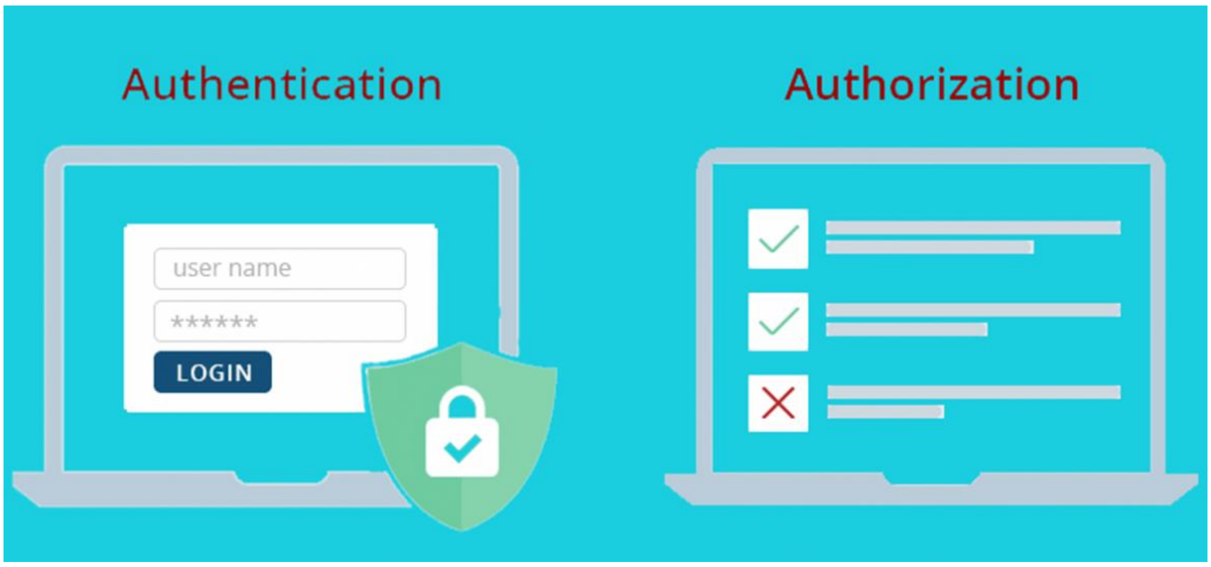
Definition

Authentication (from Greek: αὐθεντικός authentikos, "real, genuine", from αὐθέντης authentes, "author") is the act of proving an assertion, such as the identity of a computer system user. In contrast with identification, the act of indicating a person or thing's identity, authentication is the process of verifying that identity. It might involve validating personal identity documents, verifying the authenticity of a website with a digital certificate, determining the age of an artifact by carbon dating, or ensuring that a product or document is not counterfeit. (Source)

My interpretation

In my interpretation authentication means the process of validating a user's identity. If for example someone wants to buy alcohol in the store, they will most likely need to show their ID to show that they are who they are and to show that their age is above the age of 18. This is an example of authentication. Someone requests something the person gets asked to show a method of identification and therefore gets authenticate to see if he/she is the person who he/she claims to be. In Software engineering this process is seen in the login screens of applications most of the time.

What is authentication?



Definition

Authorization is the function of specifying access rights/privileges to resources, which is related to general information security and computer security, and to access control in particular. More formally, "to authorize" is to define an access policy. For example, human resources staff are normally authorized to access employee records and this policy is often formalized as access control rules in a computer system. During operation, the system uses the access control rules to decide whether access requests from (authenticated) consumers shall be approved (granted) or disapproved (rejected). Resources include individual files or an item's data, computer programs, computer devices and functionality provided by computer applications. Examples of consumers are computer users, computer software and other hardware on the computer. (Source)

My interpretation

Authorization is about having or not having access to something for example: If you would like to change user data of other users then yourself you will have to be a admin. If you are not an admin you won't have the privilege required to perform this action. But an admin will be able to do so because he/she has the privilege to do so.

Authorization V.S. authentication

The difference between authorization and authentication is that with authentication you check if someone is who they claim to be. And with authorization you check if someone is allowed to access a certain resource.

What is Openid?

Definition

OpenID is an open standard and decentralized authentication protocol. Promoted by the non-profit OpenID Foundation, it allows users to be authenticated by cooperating sites (known as relying parties, or RP) using a third-party service, eliminating the need for webmasters to provide their own ad hoc login systems, and allowing users to log into multiple unrelated websites without having to have a separate identity and password for each. Users create accounts by selecting an OpenID identity provider and then use those accounts to sign onto any website that accepts OpenID authentication. Several large organizations either issue or accept OpenIDs on their websites, according to the OpenID Foundation. (Source)

My interpretation

OpenID is an authentication mechanism that makes it possible for a user to login to an application without username or password. Because of this reason it's very popular for application that make use of multiple services. If the user would have to login for each service every task would take a very long time. It also makes it more secure by not making the services public to the service.

What is Oauth?

Definition

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites. Generally, OAuth provides clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without providing credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server. (Source)

My interpretation

Oauth makes use of tokens to authenticate users to third party services. Instead of having to give the username and password to third party services Oauth makes it possible to generate a token that can be used to login instead. By using this method, you can keep your username and password secure to 1 service.

Keycloak

What is Keycloak?

Definition

Keycloak is an open-source Identity and Access Management solution aimed at modern applications and services. It makes it easy to secure applications and services with little to no code. (Source)

My interpretation

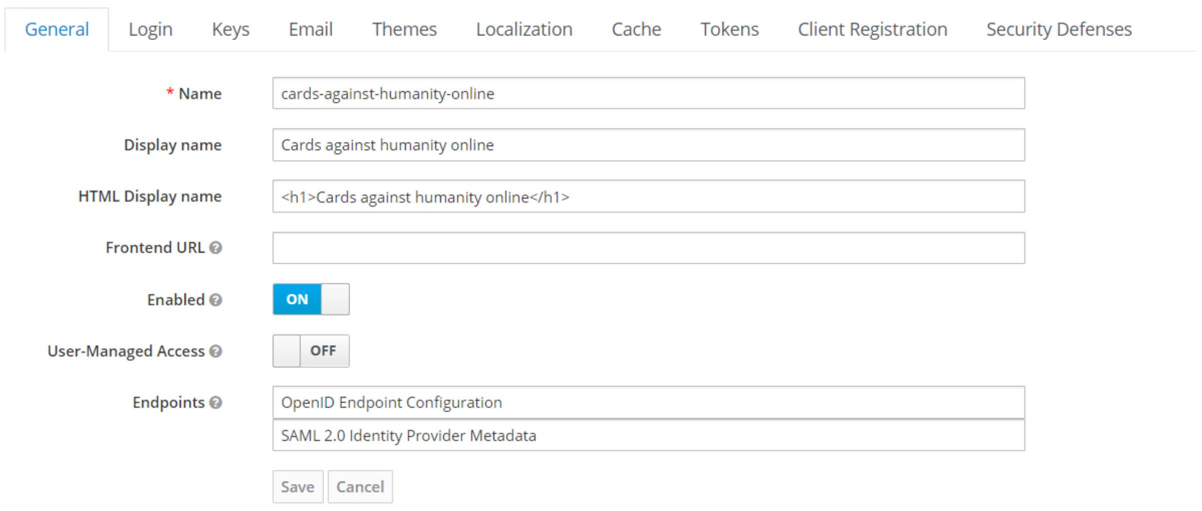
Keycloak is an identity provider meaning that it will handle all user related actions. It's very good for distributed software because you can keep user session alive through multiple services which will make it easier to mange users across multiple services. Keycloak implements the Oauth and OpenID protocols. The Oauth protocol is a protocol that handles authentication and the OpenID protocol builds on top of Oauth to implement authorization.

How to integrate Keycloak into a React frontend

I started with implementing Keycloak in my docker compose file like so:

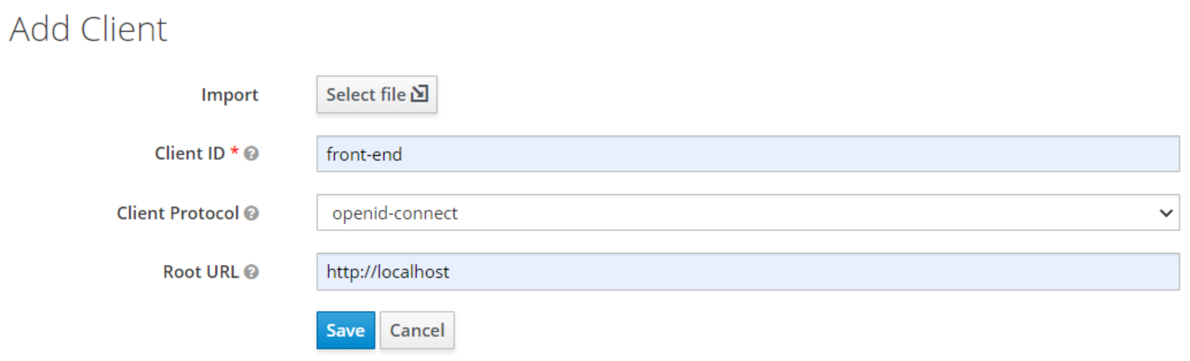
```
keycloak:
        container_name: keycloak
        image: jboss/keycloak:13.0.0
        restart: always
        volumes:
            - "./realm-export.json:/tmp/realm-export.json"
        ports:
            - "8080:8080"
        environment:
            KEYCLOAK_USER: ${KEYCLOAK_USER:-admin}
            KEYCLOAK_PASSWORD: ${KEYCLOAK_PASSWORD:-admin}
            DB_VENDOR: H2
            KEYCLOAK_IMPORT: /tmp/realm-export.json
            KEYCLOAK_FRONTEND_URL: http://localhost:8080/auth
```

This code snippet creates a Keycloak service which I can use for my application. After that I will need to create and configure a realm in Keycloak like so:



* Name	cards-against-humanity-online
Display name	Cards against humanity online
HTML Display name	<h1>Cards against humanity online</h1>
Frontend URL	
Enabled	ON
User-Managed Access	OFF
Endpoints	OpenID Endpoint Configuration SAML 2.0 Identity Provider Metadata

When the configuration is done it’s time to create and configure a client for our React frontend. We will add a client like so:

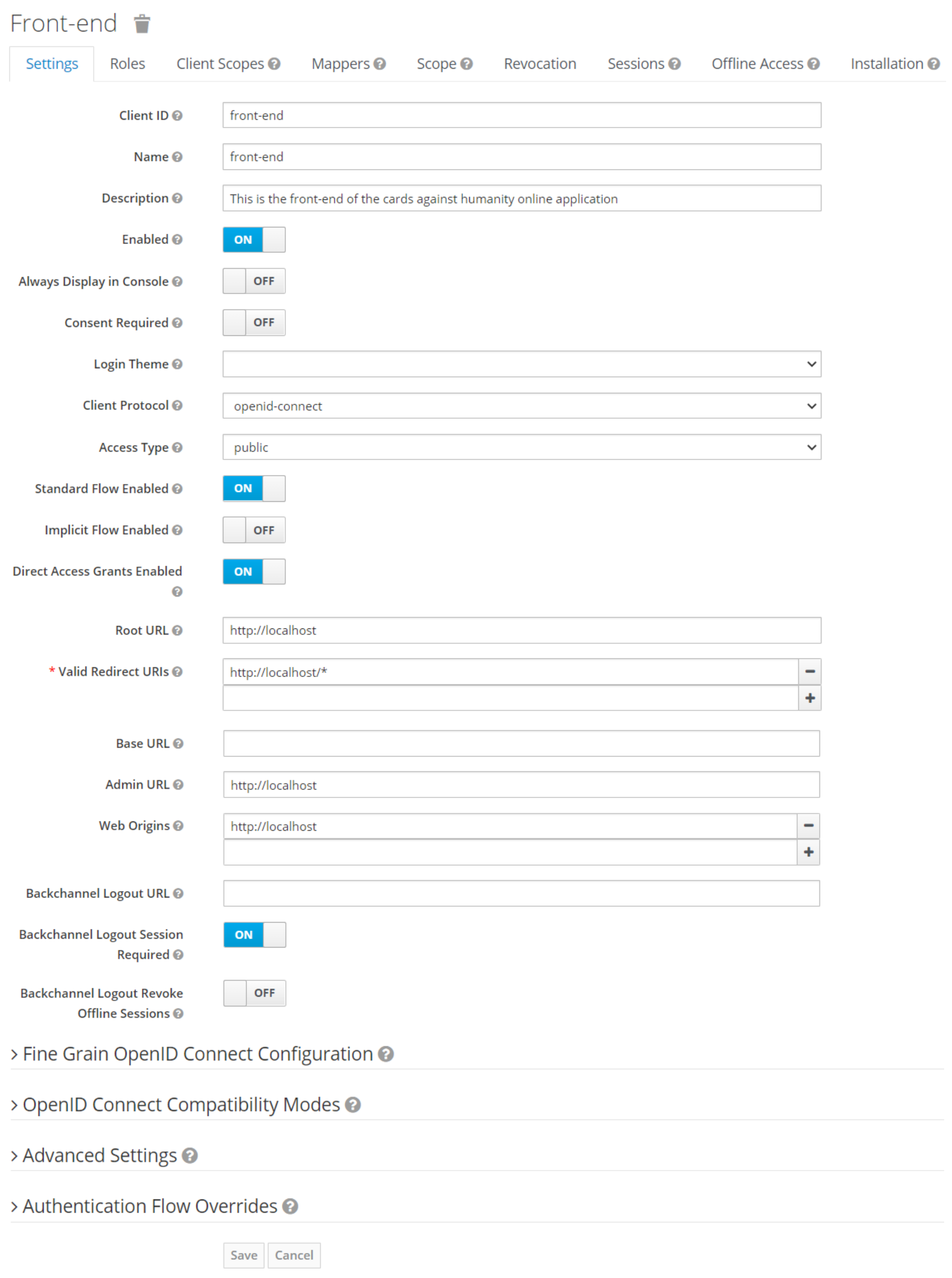


Add Client

Import	Select file
Client ID *	front-end
Client Protocol	openid-connect
Root URL	http://localhost

Save Cancel

And after this client is added we can configure it here:



Front-end

Settings | Roles | Client Scopes | Mappers | Scope | Revocation | Sessions | Offline Access | Installation

Client ID	front-end
Name	front-end
Description	This is the front-end of the cards against humanity online application
Enabled	ON
Always Display in Console	OFF
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	public
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	ON
Root URL	http://localhost
* Valid Redirect URIs	http://localhost/*
Base URL	
Admin URL	http://localhost
Web Origins	http://localhost
Backchannel Logout URL	
Backchannel Logout Session Required	ON
Backchannel Logout Revoke Offline Sessions	OFF

> Fine Grain OpenID Connect Configuration

> OpenID Connect Compatibility Modes

> Advanced Settings

> Authentication Flow Overrides

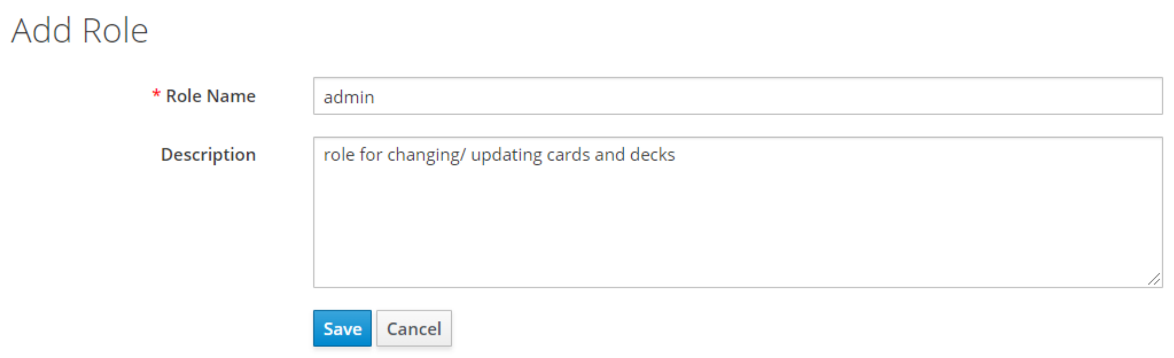
Save Cancel

Because we have configured a client for our React front-end now, we can actually start and use it. Now we can use Keycloak to send request to our services by sending the keycloak user token with the request who need it. Like so:

```
const config = {
     headers: {
          Authorization: `Bearer ${this.token}`
     }
}
await axios.post(`${backendUrl}/message/create`, {
     description: this.state.typedMessage
}, config);
```

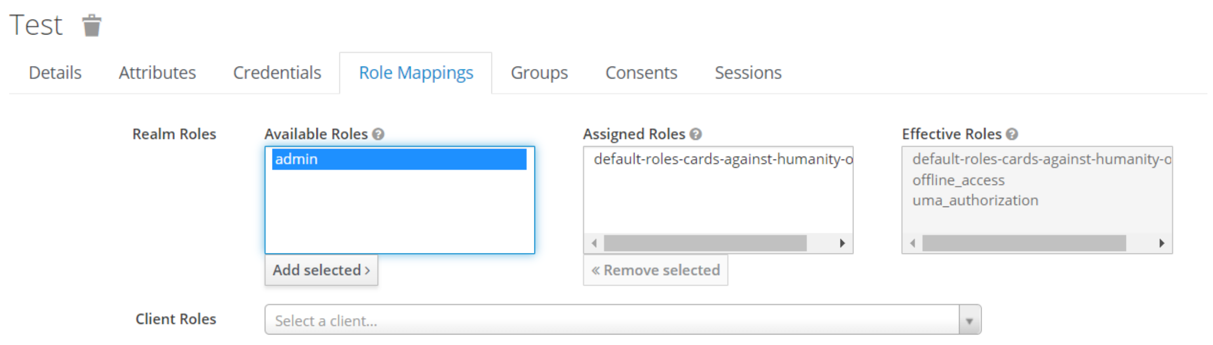
This request sends the Keycloak token with it through the header. This makes it so that services who are also integrated with keylcloak can authenticate that it is the right user.

We can also add roles to Keycloak which will give us the ability to check which roles someone has. To achieve this, we first need to go back to the Keycloak configuration. In the Keycloak configuration we're going to go to the roles section, and we are going to add a role:


Add Role

* Role Name	admin
Description	role for changing/ updating cards and decks

Now that we have added a role, we can give this role to a user by going to a user and clicking on "Role Mappings":


Test

	Available Roles	Assigned Roles	Effective Roles
Realm Roles	admin	default-roles-cards-against-humanity-o	default-roles-cards-against-humanity-o offline_access uma_authorization

You can add the role to the user by selecting the role you want give and by clicking the "add selected" button.

Now that we have added a role to a user, we can implement private routing in the front-end like so:

```
import {Redirect, Route} from 'react-router'

export default function PrivateRoute({isAuthorized, redirectPath,
...routerProps}) {
  if(isAuthorized) {
    return <Route {...routerProps} />
  }

  return <Redirect to={{pathname: redirectPath}} />
}
```

First, we create a new file called PrivateRoute.js and we add the code above. After that we can implement this class in the routing like so:

```
import PrivateRoute from "../PrivateRoute/PrivateRoute";

function App() {
    const {initialized, keycloak} = useKeycloak();
    return (
        <div className="App">
            {initialized ? (
                <Switch>
                    <PrivateRoute isAuthorized={keycloak.hasRealmRole('admin')
} redirectPath='/accessdenied' exact path='/admin'>
                        <Layout title='admin'>
                            <b>You are a admin!</b>
                        </Layout>
                    </PrivateRoute>
                    <Route exact path='/home'>
                        <Layout title="chat">
                            <Chat/>
                        </Layout>
                    </Route>
                    <Route exact path='/accessdenied'>
                        <Layout title='access denied'>
                            Access denied!
                        </Layout>
                    </Route>
                </Switch>
            ):(
                <h1>Loading...</h1>
            )}
        </div>
    );
}
```

This code makes sure that only people with the “admin” role can access the “/admin” path. It also makes sure that a loading page will be shown whenever Keycloak hasn’t been loaded in yet. Also, if someone without a admin role tries to go to “/admin” he/she will be redirected to “/accessdenied”.

How to Integrate Keycloak into a Spring backend

Assuming you have first followed the steps above I will show you how I have implemented Keycloak in a spring backend service. First of let’s configure a backend service. Follow the steps above from how to integrate Keycloak into a React frontend about adding and configuring a Keycloak client. When you have added and configured the backend client we can continue. Now in our Spring backend we will need to add a dependency in our Pom.xml file like so:

```
<dependency>
   <groupId>org.keycloak</groupId>
   <artifactId>keycloak-spring-boot-starter</artifactId>
</dependency>
```

When you have added the dependency, you will need to create a new file called: “SecurityConfig.js” which will contain the following code:

```
@KeycloakConfiguration
@Import(KeycloakSpringBootConfigResolver.class)
@EnableGlobalMethodSecurity(jsr250Enabled = true, prePostEnabled = true,
securedEnabled = true)
public class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter
{
    /**
     * Registers the KeycloakAuthenticationProvider with the authentication
manager.
     */
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
        KeycloakAuthenticationProvider
keycloakAuthenticationProvider=keycloakAuthenticationProvider();
        keycloakAuthenticationProvider.setGrantedAuthoritiesMapper(new
SimpleAuthorityMapper());
        auth.authenticationProvider(keycloakAuthenticationProvider);
    }

    /**
     * Defines the session authentication strategy.
     */
    @Bean
    @Override
    protected SessionAuthenticationStrategy sessionAuthenticationStrategy()
{
        return new RegisterSessionAuthenticationStrategy(new
SessionRegistryImpl());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        super.configure(http);
        http.authorizeRequests().anyRequest().permitAll();
        http.csrf().disable();
        http.cors().disable();
    }

}
```

You might have to do some cors configuration in here but for the sake of this research I have disabled this.

Now that we have added a security config file for Keycloak we can start implementing it in our service. You will now for example be able to get Keycloak user ids by using the following sample code:

```
@GetMapping("/logs")
public List<Message> getLogs(HttpServletRequest request) {
    KeycloakAuthenticationToken token = (KeycloakAuthenticationToken)
request.getUserPrincipal();
    KeycloakPrincipal principal=(KeycloakPrincipal)token.getPrincipal();
    KeycloakSecurityContext session =
principal.getKeycloakSecurityContext();
    String id = session.getToken().getSubject();
    return messageService.getLogs(id);
}
```

This code will extract the user id from the Keycloak user. You will also be able to set role limitations to routes by using:

```
@RolesAllowed("admin")
```

This sample code can be set on top of a route to make sure that only users with the role “admin” can access this route. This makes the backend more secure so that users can’t access data restricted to other roles. In the making of this proof of concept I mainly used this video

Implemented strategies

In this research I have mainly searched for existing information of the subject and have used this information to create an implementation for my individual project. In this research I have used the following strategies:

Library

Methods

- Research on available articles
- Research in my interpretation

Implemented where?

- What is authentication?
- What is authorization?
- What is OpenID?
- What is Oauth?
- What is Keycloak?

Workshop

Methods

- Prototyping

Implemented where?

- How to integrate Keycloak into a React frontend

Conclusion

Because of this research I now know how to implement authentication in my own project by using Keycloak as an identity provider I can manage users across all my services. And I can make sure that my users won’t have to login multiple times in a row. This is a very good development for the user experience and security in my application.

Sources

DOT-framework (EN) - Lesmateriaal - Wikiwijs. (z.d.). wikiwijs. Geraadpleegd op 19 juni 2021, van https://maken.wikiwijs.nl/129804/DOT_framework__EN_#!page-4594586

Wikipedia contributors. (2021b, juni 20). *Authentication*. Wikipedia. https://en.wikipedia.org/wiki/Authentication

Wikipedia contributors. (2021a, april 6). *Authorization*. Wikipedia. https://en.wikipedia.org/wiki/Authorization

Wikipedia contributors. (2021, 11 juni). *OpenID*. Wikipedia. https://en.wikipedia.org/wiki/OpenID

Wikipedia contributors. (2021a, juni 7). *OAuth*. Wikipedia. https://en.wikipedia.org/wiki/OAuth

Keycloak - About. (z.d.). Https://Www.Keycloak.Org. Geraadpleegd op 19 juni 2021, van https://www.keycloak.org/about

Secure Spring Boot Microservices with Keycloak. (2021, 9 maart). YouTube. https://www.youtube.com/watch?v=rcvAmBoDlLk

Wikipedia contributors. (2021a, mei 26). *Integration testing*. Wikipedia. https://en.wikipedia.org/wiki/Integration_testing