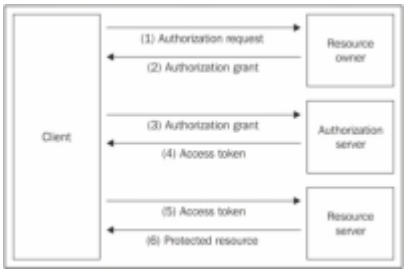
# Keycloak

Bei Keycloak handelt es sich um ein open-source Identitäts- und Zugriffsverwaltungssystem, dass es vereinfachen soll Applikationen und Dienste zu sichern. Dabei implementiert es wichtige Standards wie OpenID Connect und SAML. Im Rahmen dieser Arbeit ist lediglich OpenID Connect relevant. Auch ermöglicht Keycloak es Logins von anderen Plattformen wie GitHub oder Twitter zu implementieren. Außerdem kann Keycloak sich auch mit beispielsweise einem Active Directory verbinden, welches oftmals in Firmen eingesetzt wird, um die Nutzer-Daten mit diesem zu teilen. Ein weiteres wichtiges Feature ist die Bedienoberfläche. (Keycloak.org, o.J.)

## OAuth und OpenID Connect

Die Unterstützung von OAuth 2.0 in Keycloak stellt eines der wichtigsten Features da. Spasovski beschreibt OAuth 2.0 wie eine schützende Schicht für einen Dienst, sodass die Nutzer-Applikation eine Methode hat, um an geschütze Daten zu gelangen. OAuth 2.0 (voller Titel: “The OAuth 2.0 Authorization Framework”) ist eine Spezifikation eines Protokolls zur Sicherung von Diensten, wobei die Spezifikation Freiraum für verschiedene Implementierung offen hält. Der häufigste Anwendungsfall für OAuth 2.0 sind der Schutz von RESTful APIs und webbasierten Applikationen. (Spasovski, 2013)

### Grundfunktionalität

Die Grundfunktionalität lässt sich kurz zusammenfassen: Möchte eine Applikation auf geschützte Daten zugreifen, macht diese sogenannte HTTP Anfragen an einen Server. Dabei wird ein Zugriffstoken mitgeliefert. Dieser Token enthält Informationen, welcher Nutzer der Applikation gestattet auf die Daten zuzugreifen. 

In Abbildung SPASOVSKI_OAUTH wird ein Beispiel einer Authentifizierung gezeigt. Zunächst fragt die Nutzerapplikation den Zugriff auf die geschütze Ressource an. Wird dies gestattet erhält der Nutzer ein “Authorization grant”. Dies enthält dann Informationen über die Authentifizierung. Folgend gibt die Nutzerapplikation den “Authorization grant” weiter an den Authentifizierungsserver. Dieser überprüft den “Authorization grant” und gibt bei Bestätigung einen “Access Token” aus. Dieser spezielle Token ist mit dem Nutzer verbunden und kann dafür genutzt werden auf eine geschützte Ressource zuzugreifen. Der Token besteht mindestens aus dem Token an sich und einer Zeitangabe wie lange der Token gültig ist. Hat die Nutzerapplikation den Token erhalten, kann sie die gewünschte geschützte Ressource bei dem Server anfragen. Bei der Anfrage wird der Token mitgeschickt. Der Server mit der angefragten Ressource überprüft den Token dann und sendet bei Bestätigung die angefragten Ressource zurück. (Spasovski, 2013) OpenID Connect (OIDC) ist eine weitere Ebene, welche direkt auf OAuth 2.0 aufbaut. So erweitert OIDC OAuth 2.0 hauptsächlich um zwei wichtige Features. Zum einen ermöglicht es Applikationen die Identität von Endnutzern zu bestätigen, und zum anderen besteht auch die Möglichkeit einfache Profil-Informationen des Endnutzers zuerhalten. Zur Datenübertragung werden

JSON Web Token verwendet, diese enthalten alle Informationen. (Sakimura et al., 2014)

#### Weitere Features

OAuth 2.0 und damit auch OIDC enthalten weitere wichtige Features. So ist es für einen Nutzer möglich einem weiteren Dienst einen “Access Token” zu geben. Spasovski beschreibt dies wie folgt. Der Nutzer könnte sich auf einer Fotowebseite anmelden, auf welcher er Bilder mit anderen Nutzern teilen kann. Gleichzeitig ist er auch auf einer Webseite angemeldet, auf welcher Bilder drucken lassen kann. Unterstützen beide Webseiten nun OAuth 2.0 und haben eine entsprechende Schnittstelle um Daten auszutauschen, so kann der Nutzer dem Druckdienst einen “Access Token” geben, um auf seine Bilder bei dem anderen Foto-Teil-Dienst zuzugreifen. Dieser Ablauf wird “Authorization Delegation” genannt. Der Zugriff kann natürlich auch Widerrufen werden. Ein weiteres wichtiges Feature ist das Verbinden von mehreren unterschiedlichen Dienste, genannt “Federated Identity” Der Nutzer registriert sich bei Webseite A. Wenn er sich nun auf einer anderen Webseite B registrieren will, kann er sich mit seinem Konto von Webseite A dort direkt anmelden, ohne dass Webseite B einen Nutzernamen oder Passwort von ihm erhalten hätte. Webseite A würden in diesem Fall als “Identity Provider” agieren. Bekannte Beispiele für solche Provider sind Google und Facebook. (Spasovski, 2013)

## Implementierung in Keycloak

Keycloak übernimmt im OAuth 2.0-Authentifizierungsablauf gleich zwei wichtige Rollen. So fungiert es gleichzeitig als Resource Owner und Authorization Server (vgl. Abbildung SPASOVSKI_OAUTH). In manchen Fällen kann Keycloak sogar der Resource Server sein, da es auch Funktionen bietet, wie die Admin Konsole, über welche Keycloak konfiguriert werden kann. (Hartenstein, 2018, S.4) In Keycloak können mehrere Resource Server angegeben und konfiguriert werden. Sie werden Clients genannt. (Keycloak.org, 2021) Bei solch einem Client kann es sich um jedwede Art von Dienst handeln. Beispielsweise eine Web-, Desktop- oder Mobilapplikation (Spasovski, 2013). Durch die Admin Konsole und die Account Management Konsole kann Keycloak sehr schnell und einfach eingerichtet und genutzt werden (Keycloak.org, o.J.). Die Admin Konsole erlaubt es dabei Clienten und Rollen anzulegen, aber integriert auch gleichzeitig eine komplette Nutzerverwaltung. Auch die Verbindungen zu anderen OAuth 2.0 Diensten lassen sich konfigurieren (Siehe KAPITEL WEITERE FEATURES). Mit der Account Management Konsole bietet Keycloak auch die Möglichkeit für den Endnutzer seine Nutzerdaten zuändern oder weitere Sicherheitsmaßnahmen hinzuzufügen.

From:
https://wiki.eolab.de/ - **HSRW EOLab Wiki**

Permanent link:
**https://wiki.eolab.de/doku.php?id=user:jan001:ba:keycloak&rev=1612976951**

Last update: **2021/08/24 17:34**