Robert-Bosch-Schule Ulm

Lernfeld 12, E3FI2

Projektdokumentation

Single Sign On mit verschiedenen Plattformen realisieren

Jannik Pongratz, Lukas Butscher, Daniel Högerle
E3FI2

1. Einleitung

1.1 Projektumfeld

Der Kunde des Projekts ist die RBS-IT GmbH, welche sich für seine Buchhaltungsmitarbeiter und Geschäftsführung einen einheitlichen Login wünscht. Die RBS-IT GmbH hat 20 Mitarbeiter, von denen drei in der IT die Webserver monitoren und zwei Mitarbeiter Werbebilder in einer gemeinsamen Bilder Cloud anschauen und zum Weiterverarbeiten hochladen sollen. Alle Mitarbeiter laden ihre Dokumente auf den gemeinsamen Next Cloud Share hoch. Außerdem muss mühsam bei einem angeforderten Kundenzugriff ein neues Benutzerkonto durch die IT-Abteilung in den jeweiligen Diensten angelegt werden.

1.2 Projektziel

Das Ziel des Projektes ist das Aufsetzen von Keycloak auf einem Raspberry Pi und die Einbindung der Dienste Grafana, Next Cloud und Immich in diesen, sowie eine Einbindung in die Login-Dienste von GitHub und Discord, um die Geschäftsprozesse der RBS-IT GmbH zu vereinfachen und zu unterstützen.

1.3 Projektschnittstellen

Für das Projekt wurden folgende Projektschnittstellen benötigt:

Name	Firma	Abteilung	Beschreibung
Jannik Pongratz	RBS-IT GmbH	IT Connectivity	Implementation
Daniel Högerle	RBS-IT GmbH	IT Connectivity	Implementation
Lukas Butscher	RBS-IT GmbH	IT Connectivity	Implementation
Hr. Batke	RBS	Lehrer	Beratung, Projekt Ansprechpartner

1.4 Projektabgrenzung

Der Raspberry Pi ist schon für die Mitarbeiter der RBS-IT GmbH in der Vergangenheit eingerichtet worden und hostet bereits die Dienste Immich, Grafana und Next Cloud, welche jeweils schon länger mit eigenen Logins verwaltet und betrieben wurden.
Außerdem wurde ein Vergleich von Single Sign-On Anbietern getätigt und Keycloak als kostenlose Open Source Variante gewählt.

1.5 Abweichung vom Projektantrag

Es gab keine grundlegenden Änderungen des Antrags. Hauptsächlich wurden lediglich Dinge spezifiziert und sich auf die jeweiligen Systeme für den Login und die Implementation geeinigt.

2. Projektplanung

2.1 IST-Analyse

Aktuell verwendet die RBS-IT GmbH die Dienste mit ihren eigenen eingebauten Login Funktionen, wodurch sich die Nutzer aufgrund von verschiedenen Passwortrichtlinien und anderen Loginfeld restriktionen für die Dienste Grafana, Immich und NextCloud ein eigenes Passwort und Benutzername/E-Mail Adresse merken müssen. Zusätzlich muss mühsam, wenn ein neuer Mitarbeiter eingestellt wird oder ein Kunde Zugriff auf die Dienste fordert, ein neues Benutzerkonto durch das IT-Team angelegt werden, obwohl die meisten Kunden oder Mitarbeiter bereits für ihre Arbeit einen Github oder Discord-Account besitzen.

2.2 SOLL-Konzept

In Zukunft sollen die Nutzer sich einheitlich über den Keycloak Single Sign-On Dienst einloggen können. Damit sollen die oben genannten Passwort/Benutzernamen Probleme aus dem Weg geräumt und durch den einheitlichen Keycloak Login ersetzt werden. Der Login soll dann im Normalfall mit entweder GitHub oder Discord erfolgen.

2.3 Anforderungen an das neue System

Keycloak soll ein zentraler Login-Dienst innerhalb der RBS-IT GmbH werden und auch auf zukünftige Systeme erweitert werden können, die SSO Services unterstützen.
Außerdem sollen sich Mitarbeiter und Kunden unkompliziert mit ihren schon für die Arbeit bestehenden Konten in Github oder Discord in den jeweiligen Services einloggen können.
Außerdem soll der Keycloak Identity-Server aus dem Internet erreichbar sein und weltweit sicher und zuverlässig eine Login-Möglichkeit zur Verfügung stellen.

2.4 Ressourcen und Kostenplanung

2.4.1 Materialien

Da Keycloak nicht so viele Nutzer haben wird, wird ein Raspberry Pi aufgesetzt, um Kosten für einen gemieteten oder gehosteten Server zu sparen. Wir haben uns hier für einen Raspberry Pi 5 mit 8GB RAM entschieden, um ein aktuelles Modell mit genug Rechenkapazität zu haben, um weitere Lizenzkosten zu sparen, fiel die Wahl auf das

kostenlose Keycloak Tool. Alle externen Identity Provider verlangen ebenfalls keine Lizenzkosten. Somit ergibt sich für die Materialien folgende Rechnung:

Art	Anzahl	Kosten
Raspberry Pi 5	1	97,90€
SD Karte 64GB	1	16,99€
USB-C Kabel für Pi	1	12,40€
Raspberry Pi Gehäuse	1	9,90€
Seagate Portable 2Tb External Harddrive	1	80,99€
Cloudflare Domain Name	1	10,99€

Die Domaine bei Cloudflare wird jährlich abgerechnet. Wir haben mit einer Laufzeit von **5 Jahren** gerechnet, womit dies **54,95€** kosten würde.

Somit ergeben sich Gesamtkosten für die Materialien von 273,13€.

2.4.2 Personalkosten

Person	fiktiver Stundenlohn (Brutto)	Gesamtkosten
Jannik Pongratz	10€	270€
Lukas Butscher	10€	270€
Daniel Högerle	10€	270€

Somit ergeben sich Gesamtkosten für das Personal von 810€.

2.4.3 Gesamtkosten

Die Kosten für Material 273,13€ + die Personalkosten von 810€ ergeben einen Gesamtpreis für die Keycloak Implementation von **1083,13€**.

2.5 Ablaufplanung

Im ersten Schritt muss selbstverständlich Keycloak auf dem Raspberry Pi eingerichtet und konfiguriert werden. Dies erfolgt zur einfacheren Verwaltung und Sicherheit in einem

Docker-Container. Im nächsten Schritt werden die externen Identity Provider eingerichtet, um ein schnelles und praxisnahes Testen der angebundenen Clients zu ermöglichen. Folgende Zeiten wurden für die Durchführung eingeplant und eingehalten:

Projektphase	Geplante Zeit
Analyse und Planung	1 Stunde
Auswahl und Evaluierung des Single Sign-On Anbieters	2 Stunden
Skalierung und Hardware Wahl	1 Stunde
Auswahl und Evaluierung der anzubindenden Clients	2 Stunden
Auswahl und Evaluierung der anzubindenden	2 Stunden
Aufsetzen des Raspberry Pis	1 Stunde
Implementierung/Einrichtung von Keycloak	1 Stunde
Implementierung/Einrichtung von eingebunden Clients	8 Stunden
Implementierung/Einrichtung von Externen Identity Providern	1 Stunden
Dokumentation	8 Stunden

Somit ergibt sich eine Gesamtzeit von 27 Stunden für das Projekt.

3. Implementierung der Dienste

3.1 Aufsetzung Raspberry Pi

3.1.1 Raspberry Pi

Als Basis dient ein aktuelles Raspberry Pi OS. Nach der initialen Installation und dem ersten Boot-Vorgang wird das System zunächst vollständig aktualisiert, um sicherzustellen, dass alle Sicherheitspatches eingespielt sind.

Da Datenbankanwendungen wie Nextcloud und Immich intensive Schreib- und Lesezugriffe verursachen, ist der Einsatz einer SD-Karte als primärer Datenspeicher nicht empfehlenswert, da diese schnell verschleißen würde.

Stattdessen wird eine externe Hard Disk Drive (HDD) über USB 3.0 angebunden. Diese HDD wird mit dem Dateisystem **ext4** formatiert und fest in das System eingebunden. Hierfür wird ein Eintrag in der Datei **/etc/sda** erstellt, sodass das Laufwerk bei jedem Neustart automatisch unter dem Pfad **/mnt/sda** zur Verfügung steht.

3.1.2 Docker

Die gesamte Anwendungslandschaft wird mittels Docker realisiert. Dies hat den Vorteil, dass Anwendungen und ihre Abhängigkeiten isoliert vom Host-System laufen. Zunächst wird das offizielle Docker-Repository den Paketquellen hinzugefügt, um die aktuellste Version der Docker Engine und des Docker Compose Plugins zu installieren.

Nach der Installation wird der Standard-Benutzer des Raspberry Pis der Benutzergruppe **docker** hinzugefügt. Dies ist ein wichtiger Komfort-Schritt, da er es ermöglicht, Docker-Befehle ohne die ständige Eingabe von **sudo** auszuführen.

Um die Daten der Container persistent und strukturiert zu speichern, wird auf der externen HDD eine klare Ordnerstruktur angelegt. Für jeden Dienst (Keycloak, Immich, Nextcloud, Grafana) wird ein eigenes Unterverzeichnis erstellt, in dem Konfigurationsdateien, Datenbanken und hochgeladene Medien getrennt voneinander abgelegt werden.

Das Aufsetzen aller Dienste erfolgt durch eine jeweilige **docker-compose.yml** Datei, welche sowohl das Aufsetzen als auch etwaige Fehlerbehebung erleichtern soll.

3.1.3 Cloudflare

Um mit dem Cloudflare-Setup anzufangen, muss eine Cloudflare Domaine vorhanden sein. Hier ist bereits die Domain **"pngrtz.com"** vorhanden, weshalb keine neue angelegt und gekauft werden muss.

Mit dem Command **"cloudflared tunnel login"** öffnet sich der Cloudflare Login im Browser. Nach dem erfolgreichen Anmelden wird ein Zertifikat lokal gespeichert. Danach müssen für die verschiedenen Anwendungen Tunnel erstellt werden. Diese sind für die spätere Umleitung von der öffentlichen URL auf den localhost mitverantwortlich.

Diese Tunnel können mit dem **"cloudflared tunnel create name_tunnel"** Command erstellt werden. Im Anschluss muss noch die DNS Auflösung konfiguriert werden. Hierzu wird der command **"cloudflared tunnel route dns tunnel_name public_url"** verwendet.
In **"/etc/cloudflared/config.yml"** wird die Tunnel-Config gespeichert.

Jetzt müssen diese nur noch gestartet werden. Mit Hilfe der folgenden Commands kann dies erledigt und geprüft werden. Nach dem Ausführen sollte der Status **"Active"** anzeigen.

"sudo cloudflared service install"
"sudo systemctl enable cloudflared"
"sudo systemctl start cloudflared"
"systemctl status cloudflared"

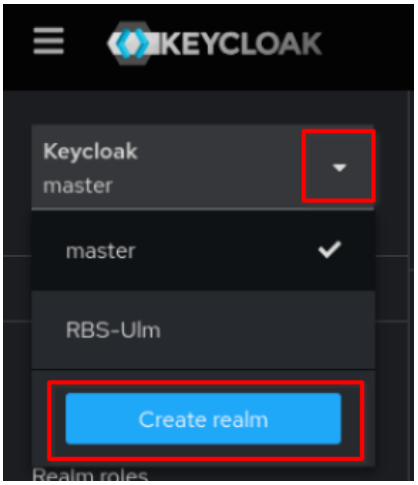
3.2 Keycloak

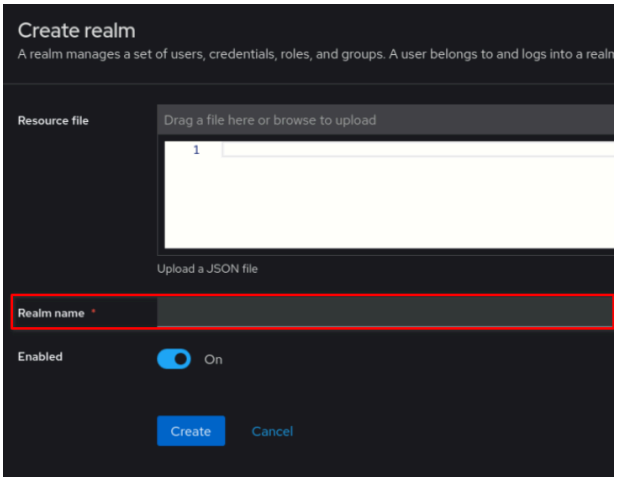
Keycloak fungiert als zentraler Authentifizierungs-Server, auch "Identity Provider" genannt. Für den Betrieb auf dem Raspberry Pi wird Keycloak in einem optimierten Modus betrieben. Als Datenbank-Backend wird PostgreSQL verwendet.

Ein kritischer Punkt in der Konfiguration ist die Behandlung des Reverse Proxies. Über die Umgebungsvariable **KC_PROXY=edge** wird Keycloak mitgeteilt, dass die SSL-Terminierung bereits bei Cloudflare stattfindet. Ohne diese Einstellung würde Keycloak versuchen, Benutzer auf unsichere HTTP-Verbindungen umzuleiten, was zu Endlosschleifen im Browser führen würde.

Keycloak stellt anschließend die Schnittstelle, ein sogenanntes Realm, bereit, über die sich Nextcloud und Grafana via OpenID Connect (OIDC) authentifizieren können. Dies ermöglicht ein "Single Sign-On" (SSO): Der Benutzer meldet sich einmal an und hat danach Zugriff auf alle Dienste.

Nach dem ersten Einloggen in Keycloak mit dem Standard Admin-User wird ein neues Realm mit dem Namen **RBS-Ulm** erstellt, welches für alle anzubindenden Dienste zuständig sein wird. Dies geschieht durch das Anklicken des aktuellen Realms, **master**, und das anschließende Klicken auf den Button **"Create Realm"**. Dann kann, wie unten abgebildet, der Realm-Name vergeben werden, hier **"RBS-Ulm"**.





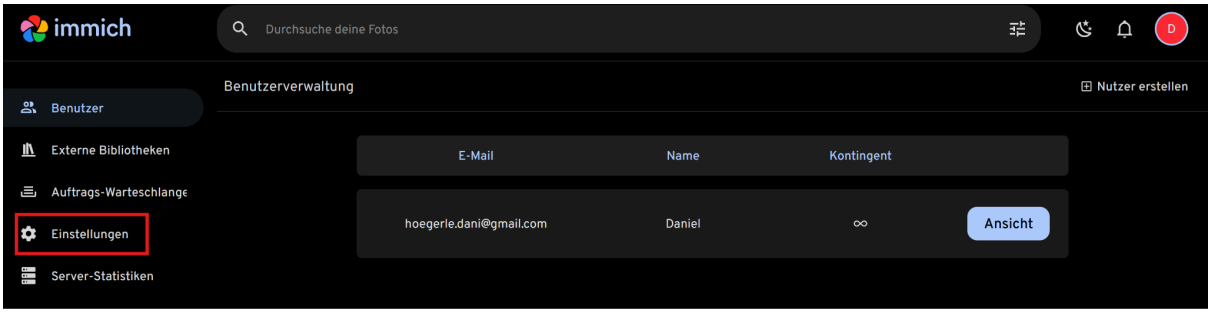
3.3 Angebundene Clients

3.3.1 Immich

3.3.1.1 Konfigurieren in Immich

Um Immich hierfür korrekt konfigurieren zu können, muss nach der Installation ein lokales Admin Konto angelegt werden. Dieses wird verwendet, um den Client zu konfigurieren. Nach dem Einloggen in den Admin-Account, müssen die Immich System-Einstellungen aufgerufen werden. Dies erfolgt durch das Klicken auf das Profilbild (Standardmäßig der Anfangsbuchstabe des Kontoinhabers) in der rechten oberen Ecke, gefolgt auf das Klicken auf “Verwaltung”.

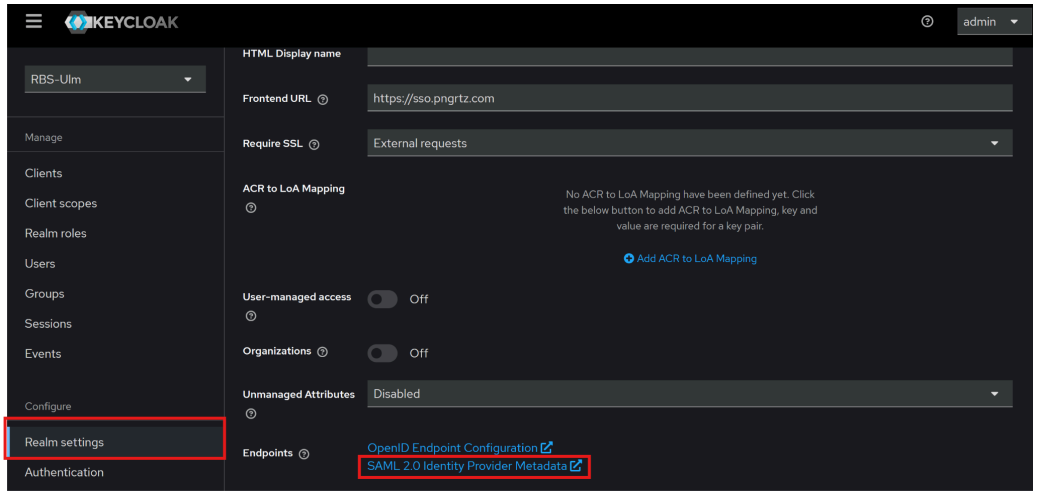
Dann öffnen sich die Immich-Server Optionen. Danach erfolgt ein Klick auf **“Einstellungen”** auf der linken Seite



Im darauf folgenden Menü öffnet man den ersten Punkt “Authentifizierungseinstellungen”, dort ist dann der Punkt OAuth zu wählen. Danach muss man den Schieberegler bei “Anmeldung mit OAuth” nach rechts schieben.

Hier werden dann die Daten aus Keycloak eingetragen. Diese sind aus den SAML 2.0 Settings entnommen.

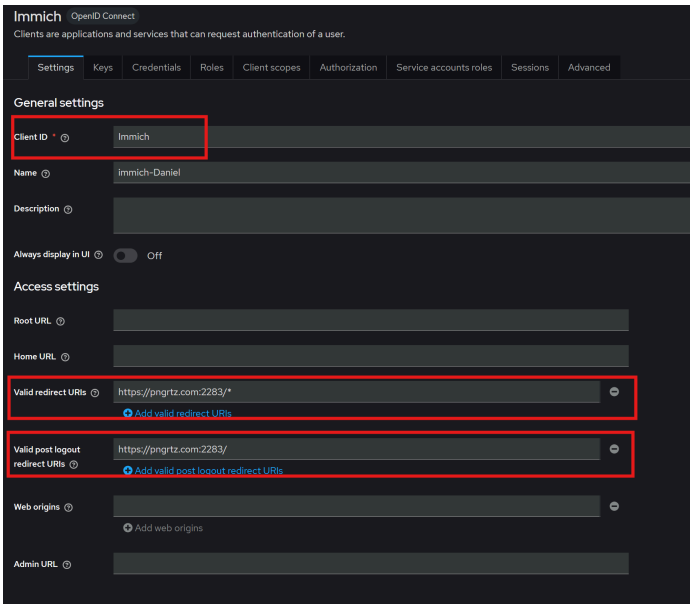
Anschließend muss man in die Keycloak Admin-Ansicht, dort das korrekte Realm auswählen und danach den Punkt “Realm settings”. Unter Endpoints wählt man **“SAML 2.0 Identity Provider Metadata”**, dort ist die Issuer_URL korrekt abgebildet.



Die restlichen Einstellungen mit Ausnahme der Client_ID können beim Standardwert belassen werden. Die Client_ID ist der Name, der später in Keycloak für Immich angelegt wird. Im letzten Schritt muss nur noch unten auf den **“Speichern”**-Knopf gedrückt werden.

3.3.1.2 Konfiguration in Keycloak

Zuletzt müssen noch in Keycloak die richtigen Einstellungen getroffen werden. Hierfür muss man das korrekte Realm auswählen und dann einen neuen Client anlegen. Dafür klickt man auf den Punkt “**Clients**” und wählt “**Create Client**”. Hier legt man anschließend den Namen fest, der ebenfalls in Immich unter **Client_ID** angelegt werden muss. Im Drop-Down Menü “Client Type” wählt man nun den Punkt “SAML” aus und gibt im nächsten Punkt, den Client Settings, die korrekten URIs (in Valid redirect URLs und Valid post logout redirect URLs) von Immich an. Bei Valid redirect URLs muss am Ende der Adresse ein Asterisk folgen.

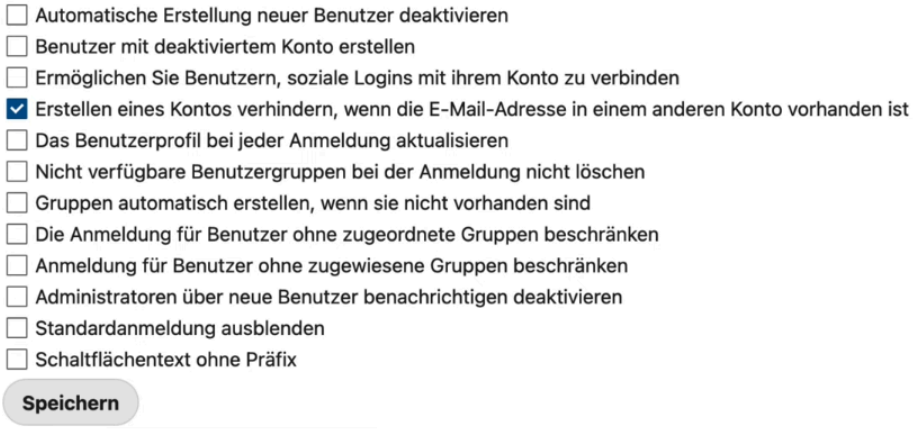


3.3.2 Next-Cloud

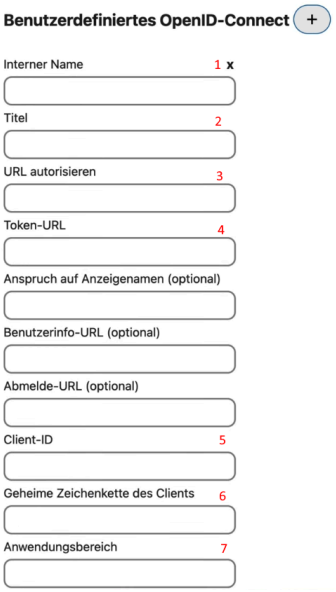
Die Next Cloud Instanz ist schon vorhanden und es muss nur noch die Verbindung zu Keycloak konfiguriert werden.

3.3.2.1 Next-Cloud Konfiguration

Die Next-Cloud-Konfiguration erfolgt wie bei den anderen Clients über die Systemeinstellungen. Jedoch muss zuerst aus dem App Store die App **“Social Login”** heruntergeladen werden, bevor man sich über das Menü in die Administration Einstellungen begibt. Nachdem die App installiert wurde, ist dort der Menüpunkt **“Social Login”** verfügbar. Hierbei sollten diese Einstellungen getroffen werden:

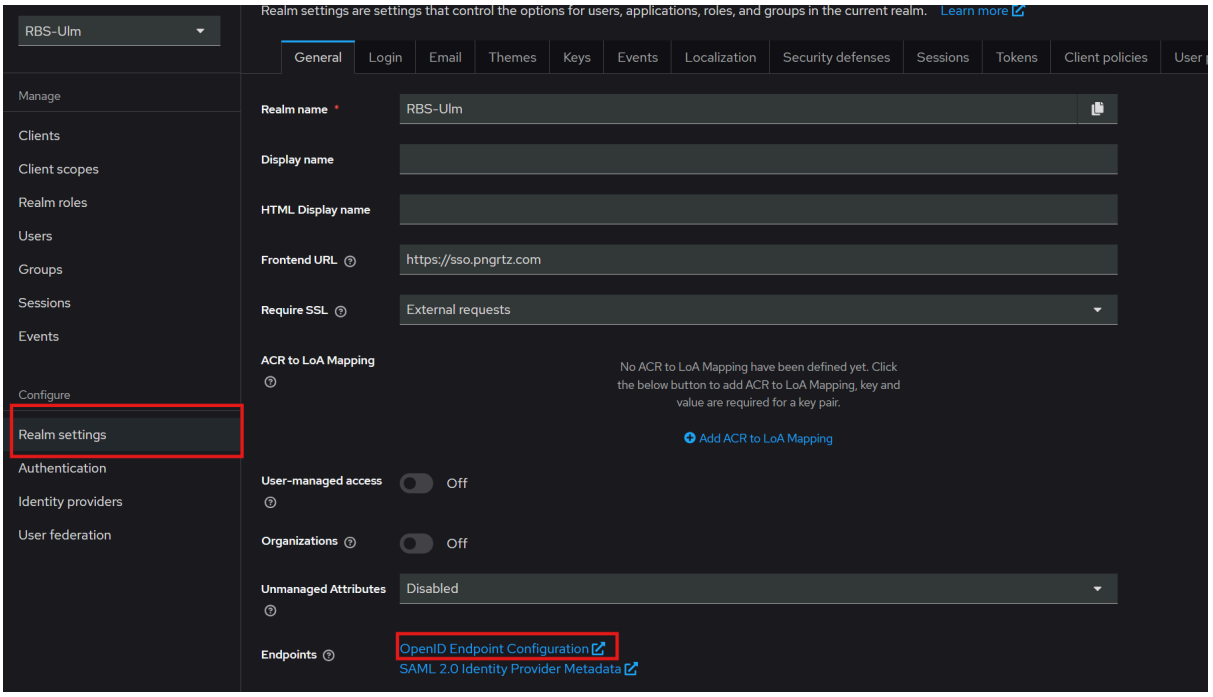


Danach klickt man auf den **“Benutzerdefiniertes OpenID-Connect”**-Button darunter und es öffnet sich das rechts stehende Menü. Hier die Erklärung er einzelnen Felder:



1: Hier wird die für Next-Cloud verwendete Bezeichnung der Keycloak Connection eingefügt
2: Hier der in Next-Cloud angezeigte Titel. Es empfiehlt sich, Verwechslungen zu vermeiden und hier den identischen Wert wie bei Feld 1 einzutragen.
3: Hier wird die **“authorization_endpoint”** URL benötigt, welche aus der Keycloak OID-Config herausgelesen werden kann (siehe folgend)
4: Die Token URL wird, identisch zum 3., aus der Keycloak OID-Config ausgelesen. Diese heißt hier **“token_endpoint”**.
5: Hier muss der Name des Clients eingegeben werden, wie er in Keycloak hinterlegt wurde.
6: Dieser Wert ist das **“Client Secret”** aus Keycloak. Dieses ist in den Client-Einstellungen zu finden.
7: Hier kann einfach der Wert **“openid”** eingetragen werden.

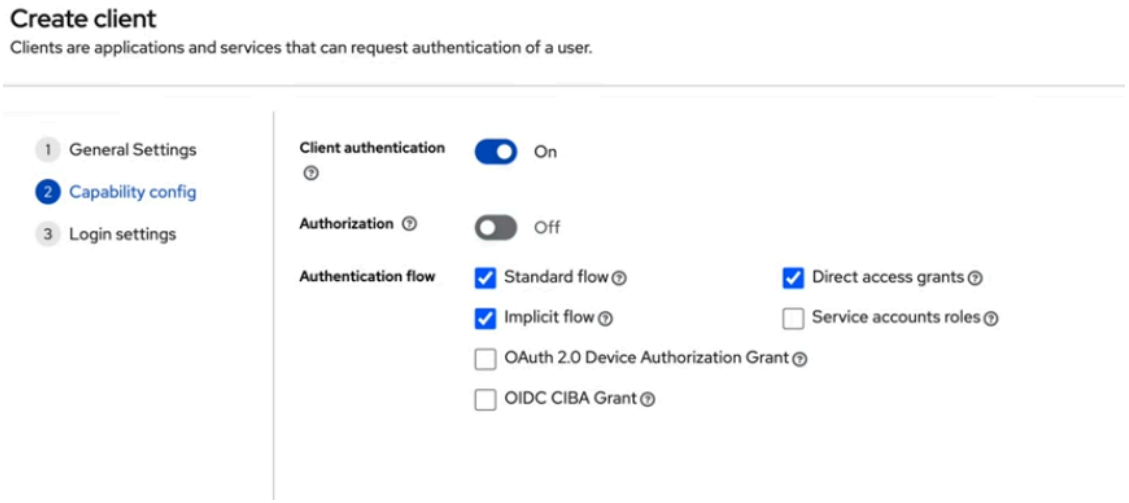
Um die Werte für 3. und 4. zu erhalten, muss in Keycloak das korrekte Realm auswählen und dort auf Realm settings, sowie beim Unterpunkt Endpoints OpenID Endpoints auswählen.



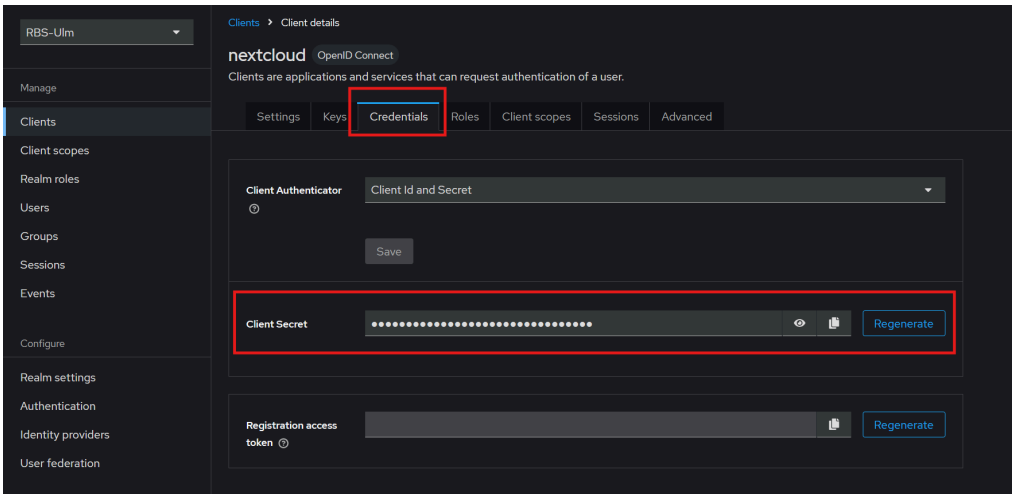
Daraufhin öffnet sich ein Json-File im Browser, in dem die verschiedenen Werte unter den oben beschriebenen Namen auffindbar sind.

3.3.2.2 Konfiguration in Keycloak

Im ersten Schritt muss im jeweiligen Realm in Keycloak der Punkt **“Clients”** ausgewählt werden und dort dann ein neuer Client mit der Schaltfläche **“Create Client”** hinzugefügt werden. Dort stellt man zuerst sicher, dass im Dropdown Menü oben der Punkt **“OpenID Connect”** ausgewählt ist. Danach setzt man bei **“Client ID”** jetzt denselben Wert wie auch in Next-Cloud ein (5.).
Nach dem Klick auf **“Next”** müssen die Werte so gesetzt werden:



Bestätigt werden die Einstellungen mit einem Klick auf **“Next”**. Bei den **“Login settings”** ist es wichtig, bei der **“Root URL”** die URL von Next-Cloud zu wählen. Das Feld **“Home URL”** kann leer gelassen werden. Wichtig ist lediglich, dass bei **“Valid redirect URIs”** die Adresse von der **“Root URL”** erneut einzufügen ist, nur dieses Mal mit einem Asterisk dahinter. Bei **“Web origins"** will Keycloak erneut die Adresse des Next-Cloud-Servers. Am Schluss bestätigt man die Einstellungen mit einem Klick auf **“Save”**.
Zuletzt muss noch der Wert des **“Client Secrets”** für die Next-Cloud Konfiguration ausgelesen werden. Hierfür geht man erneut in Keycloak auf den Unterpunkt **“Clients”** und klickt auf die Next-Cloud Verbindung. Es öffnet sich nun das Konfigurationsfenster. Hier klickt man nun auf den Tab **“Credentials”** und erhält hier das von Next-Cloud verlangte **“Client Secret”**.



3.3.3 Grafana

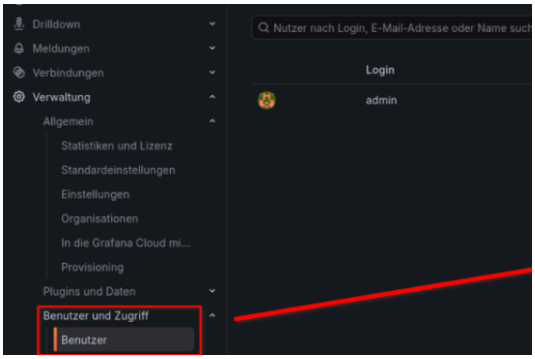
3.3.3.1 Grafana Konfiguration

Für die Grafana-Konfiguration muss man sich nach der Installation im lokalen Admin Konto anmelden. Dieses wird verwendet, um neue Benutzer anzulegen, da bei Grafana, anders als bei Immich, die OAuth-Konfiguration bereits in der **docker-compose.yml** stattfindet.

```
environment:
  - GF_SERVER_ROOT_URL=https://monitor.pngrtz.com
  - GF_SERVER_HTTP_PORT=3000
  - GF_AUTH_GENERIC_OAUTH_ENABLED=true
  - GF_AUTH_GENERIC_OAUTH_NAME=Keycloak
  - GF_AUTH_GENERIC_OAUTH_ALLOW_SIGN_UP=false
  - GF_AUTH_GENERIC_OAUTH_CLIENT_ID=grafana
  - GF_AUTH_GENERIC_OAUTH_ROLE_ATTRIBUTE_PATH=contains(resource_access.grafana.roles[*], 'admin') && 'Admin' || 'Viewer'
  - GF_AUTH_GENERIC_OAUTH_CLIENT_SECRET=SECRET
  - GF_AUTH_GENERIC_OAUTH_AUTH_URL=https://sso.pngrtz.com/realms/RBS-Ulm/protocol/openid-connect/auth
  - GF_AUTH_GENERIC_OAUTH_TOKEN_URL=http://keycloak:8080/realms/RBS-Ulm/protocol/openid-connect/token
  - GF_AUTH_GENERIC_OAUTH_TLS_SKIP_VERIFY_INSECURE=true
  - GF_AUTH_GENERIC_OAUTH_API_URL=http://keycloak:8080/realms/RBS-Ulm/protocol/openid-connect/userinfo
  - GF_AUTH_GENERIC_OAUTH_SCOPES=openid profile email
```

Nach dem Einloggen in den Admin-Account kann durch das Klicken auf **"Verwaltung -> Benutzer -> Neuer Benutzer"** ein neuer Benutzer hinzugefügt werden. Sollte dieser bereits

in Keycloak vorhanden sein, müssen Benutzername und E-Mail übereinstimmen.



3.3.3.2 Anbindung von Prometheus

Damit später die Daten des Raspberry Pis angezeigt werden können, muss eine Anbindung zu Prometheus eingerichtet werden. Prometheus ist eine Zeitreihen-Datenbank und läuft lokal auf dem Raspberry Pi; es wird zusammen mit Grafana im Docker-Container gestartet. Die späteren Daten werden durch den **Node-Exporter** auf dem Pi gesammelt und in Prometheus Importiert.

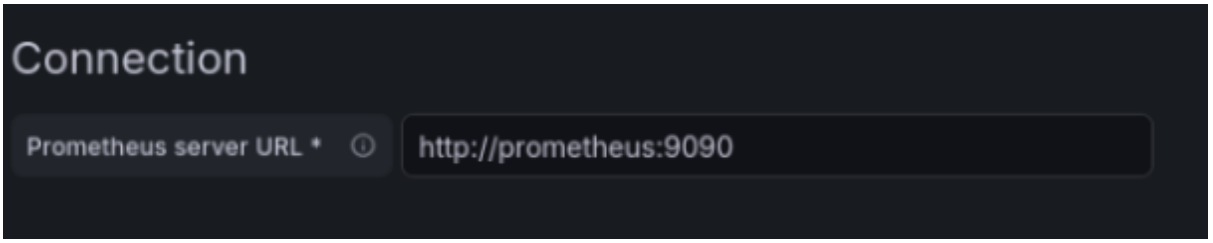
```
global:
  scrape_interval: 15s # How often to fetch data
  evaluation_interval: 15s

scrape_configs:
  - job_name: 'raspberry_pi_stats'
    static_configs:
      - targets: ['node-exporter:9100']

  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

prometheus.yml

```
node-exporter:
  image: prom/node-exporter:latest
  container_name: node-exporter
  ports:
    - "9100:9100"
  restart: always
```

docker-compose.yml

In Grafana muss anschließend eine Verbindung zu Prometheus konfiguriert werden. Hierfür muss auf **“Verbindungen -> Datenquellen -> Prometheus”** und anschließend auf **“Neue Datenquelle”** geklickt werden. Danach muss nur noch die richtige URL eingegeben werden, hier "http://prometheus:9090".

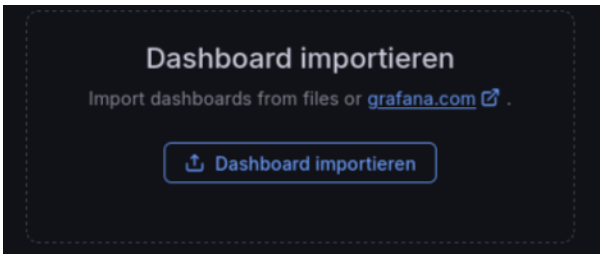


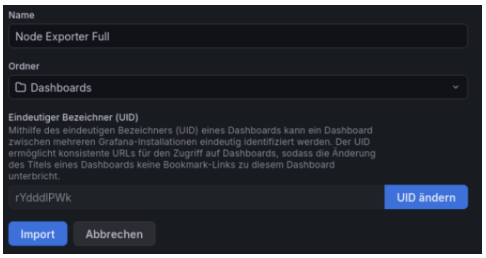
Zum Schluss muss nur noch ganz nach unten gescrollt und auf "Testen und speichern" geklickt werden.

3.3.3.3 Dashboard erstellen

Es gibt bereits Dashboards, welche nur noch importiert werden müssen; Eines davon hat die ID "1860". Um dieses zu importieren, muss auf **"Dashboards -> Dashboard erstellen ->**

Dashboard importieren" geklickt werden. Danach kann die ID eingegeben und die Konfiguration des Dashboards geladen werden. Jetzt muss noch auf **import** geklickt werden.





3.3.3.4 Konfiguration in Keycloak

Die Konfiguration in Keycloak hat den gleichen Ablauf als für Immich. beim Anlegen des Clients muss nur auf die Richtigen URIs geachtet werden.

3.4 Externe Identity Provider

3.4.1 Github

3.4.1.1 Github OAuth Application

Zunächst muss GitHub mitgeteilt werden, dass Keycloak berechtigt ist, Nutzerdaten abzufragen. Hierfür navigiert man in seinem GitHub-Profil zu den **Settings** und wählt im linken Menü ganz unten die **Developer Settings** aus. Unter dem Punkt **OAuth Apps** klickt man auf **New OAuth App**, um eine neue Registrierung zu starten.

In der Eingabemaske vergibt man einen aussagekräftigen Namen für die Applikation. Die **Homepage URL** entspricht der Basis-URL der Keycloak-Instanz. Der wichtigste Punkt ist die **Authorization callback URL**.
Nachdem die App registriert ist, generiert GitHub eine **Client ID**. Zusätzlich muss auf die Schaltfläche **Generate a new client secret** geklickt werden. Diese sollten sicher aufbewahrt werden, da man sie nicht erneut einblicken kann.

3.2.1.2 Konfiguration in Keycloak

In Keycloak gibt es unter **"Identity Provider -> OAuth v2"** bereits eine Blaupause für Github. Wählt man diese, müssen nur noch der **Name** geändert und **Client ID** sowie **Client Secret** eingefügt werden.

3.4.2 Discord

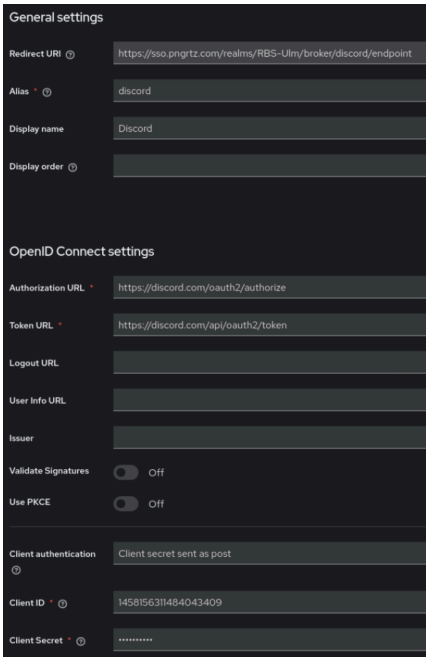
3.4.2.1 Discord App erstellen

Um Discord erfolgreich anzubinden, muss zuerst eine neue Discord App im **Discord Developer Portal** erstellt werden. Wichtig ist hier, dass im Folgenden nicht die normale

Discord Anwendung gemeint ist. Sollte sich das ändern, wird dies explizit erwähnt.
Diese ist über folgende URL erreichbar: https://discord.com/developers/applications.

Eine neue Anwendung kann einfach erstellt werden. Es muss nur ein Name vergeben und auf "erstellen" geklickt werden. Im Anschluss muss unter **OAuth** die **RedirectURI** hinzugefügt werden. Die **RedirectURI** kann später aus Keycloak kopiert werden.

3.4.2.2 Keycloack Konfiguration



In Keycloak kann unter **"Identity Provider -> OAuth v2"** Discord hinzugefügt werden.

Nun müssen die Felder nur noch wie links im Bild ausgefüllt werden.

Durch das Ausfüllen des **Namens** wird die **RedirectURI** vervollständigt. Diese kann nun unter dem Tab **OAuth** in der Discord App hinzugefügt werden; hier können auch die **Client ID** und das **Client Secret** gefunden werden. Diese können einfach kopiert und in Keycloak eingefügt werden.

4. Fazit

Die Implementierung von Keycloak war erfolgreich. Alle Anforderungen wurden erfüllt. Die Integration verlief größtenteils reibungslos und ist auch ohne größere Probleme funktionsfähig.
Durch das Einbauen von externen Identity Providern, hat sich die allgemeine User-Experience deutlich verbessert. Die Auslagerung der Passwörter auf die externen Dienste, sorgen zusätzlich für Sicherheit vor Hackerangriffen oder Phishing-Attacken auf die RBS-IT GmbH.
Durch die Verwendung von Keycloaks mit den Sicherheitsstandards OIDC und SAML 2.0 besitzt die RBS-IT GmbH eine zukunftssichere Login-Technik, die von vielen Diensten unterstützt wird.

5. Anlagen

5.1 Textquellen

https://de.wikipedia.org/wiki/Single_Sign-on
https://www.oracle.com/de/security/identity-management/single-sign-on-sso/
https://www.keycloak.org/
https://immich.app/
https://nextcloud.com/
https://grafana.com/
https://www.keycloak.org/server/configuration
https://docs.immich.app/administration/oauth/
https://www.berrybase.de/
https://medium.com/@ravipatel.it/building-a-monitoring-stack-with-prometheus-grafana-and-alerting-a-docker-compose-ef78127e4a19
https://discord.com/developers/docs/intro
https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps
https://janikvonrotz.ch/2020/10/20/openid-connect-with-nextcloud-and-keycloak/
https://www.muehlencord.de/wordpress/2019/12/14/nextcloud-sso-using-keycloak/
https://www.youtube.com/watch?v=Pb2OkR0Pt8c

5.2 Bildquellen

Alle Screenshots der verwendeten Software sind selbstgemacht.

5.3 Soll/Ist Vergleich

Alle Anforderungen konnten schlussendlich erfüllt werden und Keycloak funktioniert, wie erwartet. Der von uns im Vorhinein festgelegte Zeitplan konnte gut eingehalten werden und es wurde eher weniger, als zu viel Zeit benötigt.

5.4 Qualitätssicherung

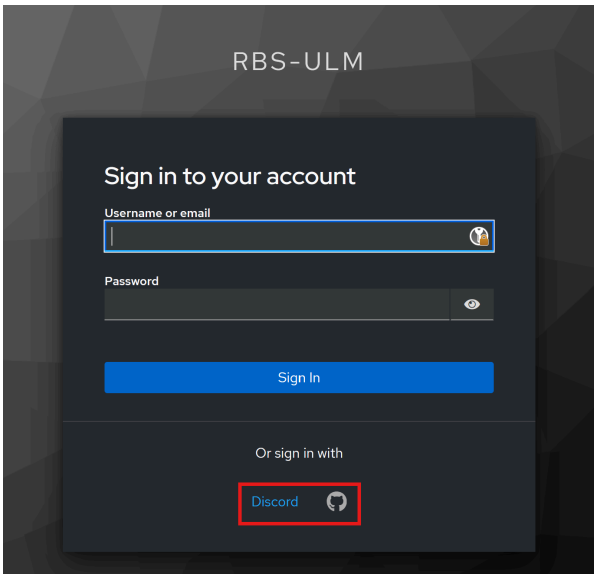
Um eine gewisse Qualität und Sicherheit des Dienstes in Zukunft sicherzustellen, sind regelmäßige Updates von Keycloak sowie den angebundenen Clients essenziell. Wichtig ist auch das sorgfältige Pflegen der Einträge, sowie die Benamung innerhalb von Keycloak, wie auch bei den verbundenen Diensten. Um Sicherheitslücken zu vermeiden, sollten Sicherheits-Workshops und Anti-Phishing Kurse geplant werden, damit die Mitarbeiter nicht ihr eigenes Passwort weggeben und ein potentieller Angreifer dann auf alle Systeme Zugriff erhält.

5.5 Kundendokumentation Mitarbeiter der RBS-IT GmbH

Diese Kundendokumentation ist für Mitarbeiter und Kunden der RBS-IT GmbH, die Immich, Nextcloud oder Grafana für ihre Arbeit verwenden. Und GitHub bzw. Discord als Login Möglichkeit verwenden.

5.5.1 Genereller Login über externe Identity Provider

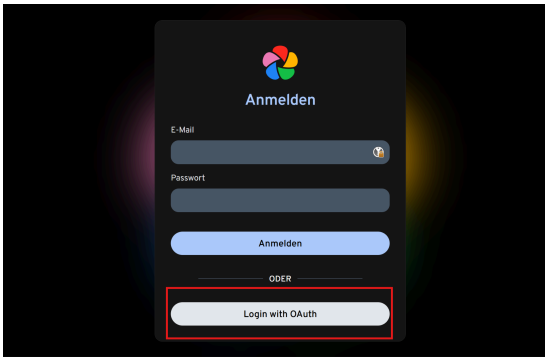
Nach dem Login-Klick auf der jeweiligen Applikation (folgend unten) erscheint das Keycloak Login-Fenster. Dort wählt man nun die Art des Log-ins. Entweder über Discord, GitHub, oder ein ganz normaler Login mit Username und Passwort, falls ein Account in Keycloak hinterlegt ist.



Sollte man sich über Github oder Discord einloggen, wird man nach dem Klick auf das jeweilige Logo auf die vertraute Login-Seite des Dienstes weitergeleitet. Sobald das Passwort und der Nutzername erfolgreich eingegeben ist, erfolgt die Weiterleitung auf die Seite des jeweiligen Dienstes und der Login Prozess ist abgeschlossen.

5.5.2 Immich Login

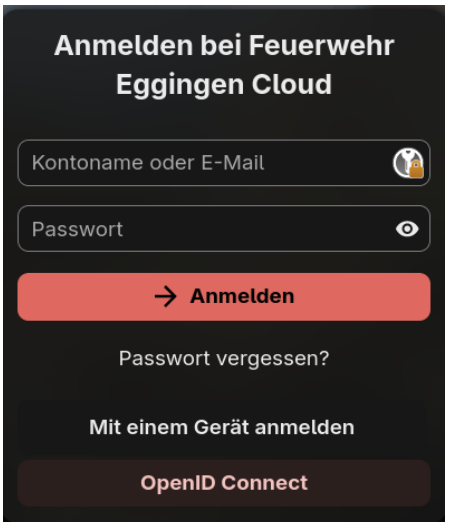
Um sich bei Immich einzuloggen, muss man den Knopf “Login with OAuth” betätigen.



Dieser führt anschließend auf die Keycloak Login-Seite. Nach der Auswahl der Login Option, ist man anschließend in Immich eingeloggt.

5.5.3 Next Cloud Login

Bei Next Cloud erfolgt dies über die Option “OpenID Connect” im Login-Fenster. Nach dem Klick auf den Knopf, wird man erneut zu Keycloak umgeleitet und der Ablauf ist identisch, wie bei Immich.



5.5.4 Grafana Login

Das Einloggen in Grafana funktioniert ähnlich wie bei den anderen Anwendungen. Durch das Klicken auf “Anmelden mit Keycloak" wird man auf Keycloak umgeleitet. Dort ist der Ablauf identisch mit Immich und Next Cloud.