

Analytics as a Service

Table of Contents

1. Aufwand.....	2
2. Dependencies	2
3. Installationsanleitung	4
4. Aufbau des Projektes.....	5
5. Übersicht der Komponenten	6
5.1. Navbar	6
5.2. Logs	7
5.3. Metrics.....	9
5.4. Detectors	12
5.5. Clients	15
5.6. Toasts.....	17
6. Authentifizierung	20
7. Loader-Interceptor.....	22

1. Aufwand

Aufwand liegt bei ca. 30h.

2. Dependencies

Wichtige Dependencies sind:

- `tailwindcss`: Functional CSS Library
- `chartjs`: Charting Library
- `angular-oauth2-oidc`: Für Authentication

Listing 1. package.json

```
{
  "name": "analytics-as-a-service",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~13.1.0",
    "@angular/common": "~13.1.0",
    "@angular/compiler": "~13.1.0",
    "@angular/core": "~13.1.0",
    "@angular/forms": "~13.1.0",
    "@angular/platform-browser": "~13.1.0",
    "@angular/platform-browser-dynamic": "~13.1.0",
    "@angular/router": "~13.1.0",
    "@ngneat/tailwind": "^7.0.3",
    "angular-oauth2-oidc": "^13.0.1",
    "chart.js": "^3.7.0",
    "chartjs-adaptor-moment": "^1.0.0",
    "d3": "^7.2.1",
    "ng2-charts": "^3.0.6",
    "rxjs": "~7.4.0",
    "tailwindcss": "^3.0.10",
```

```
    "tslib": "^2.3.0",
    "tslint": "^6.1.3",
    "zone.js": "~0.11.4"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~13.1.2",
    "@angular/cli": "~13.1.2",
    "@angular/compiler-cli": "~13.1.0",
    "@types/d3": "^7.1.0",
    "@types/jasmine": "~3.10.0",
    "@types/node": "^12.11.1",
    "jasmine-core": "~3.10.0",
    "karma": "~6.3.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.1.0",
    "karma-jasmine": "~4.0.0",
    "karma-jasmine-html-reporter": "~1.7.0",
    "typescript": "~4.5.2"
  }
}
```

3. Installationsanleitung

Schritt für Schritt:

1. [AaaS.API](#) starten
2. `docker run -it -e KEYCLOAK_USER=aaas -e KEYCLOAK_PASSWORD=aaas --name keycloak -p 8080:8080 -d jboss/keycloak`: Keycloak starten
3. `npm ci`: installiert die Dependencies ohne sie zu aktualisieren
4. `npm start`: Webserver wird auf `4200` gestartet
5. <http://localhost:4200>

Zum Konfigurieren des Ganzen wurde `environment.ts` benutzt. Hier können auch alle Endpunkte für API oder Keycloak geändert werden.

Listing 2. `environment.ts`

```
// This file can be replaced during build by using the
`fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.

export const environment = {
  production: false,
  url: "http://localhost:5000",
  oauthUrl: 'http://localhost:8080',
  realmName: 'aaas'
};

/*
 * For easier debugging in development mode, you can import the
 * following file
 * to ignore zone related error stack frames such as `zone.run`,
 * `zoneDelegate.invokeTask`.
 *
 * This import should be commented out in production mode because it
 * will have a negative impact
 * on performance if an error is thrown.
 */
// import 'zone.js/plugins/zone-error'; // Included with Angular CLI.
```

4. Aufbau des Projektes

Das Angular Projekt wurde wie folgt strukturiert:

```
AaaS.Web/src/app
├─ components: Beinhaltet alle Angular/UI Komponenten
├─ services: Service für REST, LocalStorage und für Loading-Animation
├─ model: Interfaces und Enums
├─ pipes: Pipes zum Formattieren von AppKeys und Timespans
├─ validators: Validatoren für bestimmte Reactive-Forms
├─ guards: AuthGuard
├─ interceptors: Interceptor für HttpRequests zum Aktivieren der Loading-Animation
└─ utils: Convenience-Functions
```

Figure 1. Structure

Nachdem Tailwind benutzt wurde, sind einige Standard-Komponenten wie bei angular-material nicht verfügbar. Da ich allerdings schon einige Erfahrung mit Angular hatte, wollte ich auch was zusätzlich neues ausprobieren. Daher gibt es eigene Implementierungen für Modals und Toasts.

5. Übersicht der Komponenten

5.1. Navbar

Navigation und Login/Logout

ANALYTICS AS A SERVICE **LOGS** **GRAPHS** **DETECTORS** **CLIENTS**

Logout

Figure 2. Navbar logged-in

ANALYTICS AS A SERVICE

Login

Figure 3. Navbar logged-out

5.2. Logs

Ein einfacher Log-Overview, mit Filter-Komponente, diese filtert die Logs nach gewissen Kriterien.

Logs

Name:

Instance:

Type:

None

From:

To:

Filter

Name	Type	Instance	Created At	Message
FizzBuzz	Trace	pop-os	2022-01-23 05:02:52+0100	Buzz
FizzBuzz	Trace	pop-os	2022-01-23 05:02:50+0100	Fizz
agent.ping	Warning	pop-os	2022-01-23 05:02:46+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:44+0100	Fizz
agent.ping	Warning	pop-os	2022-01-23 05:02:44+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:42+0100	Buzz
agent.ping	Warning	pop-os	2022-01-23 05:02:42+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:40+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:38+0100	Fizz
agent.ping	Warning	pop-os	2022-01-23 05:02:38+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:35+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:33+0100	Could not ping 9.9.9.9
FizzBuzz	Error	pop-os	2022-01-23 05:02:31+0100	FizzBuzz
agent.ping	Warning	pop-os	2022-01-23 05:02:31+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:29+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:27+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:25+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:25+0100	Fizz
agent.ping	Warning	pop-os	2022-01-23 05:02:22+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:20+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:20+0100	Buzz
FizzBuzz	Trace	pop-os	2022-01-23 05:02:18+0100	Fizz
agent.ping	Warning	pop-os	2022-01-23 05:02:18+0100	Could not ping 9.9.9.9
agent.ping	Warning	pop-os	2022-01-23 05:02:14+0100	Could not ping 9.9.9.9
FizzBuzz	Trace	pop-os	2022-01-23 05:02:12+0100	Fizz

1 >>

25 >

Figure 4. Logs

5.3. Metrics

Metrics ist die Kernkomponente von AaaS.Web, es zeigt verschiedene Graphen an. Es können wieder Graphen hinzugefügt, bearbeitet und gelöscht werden.

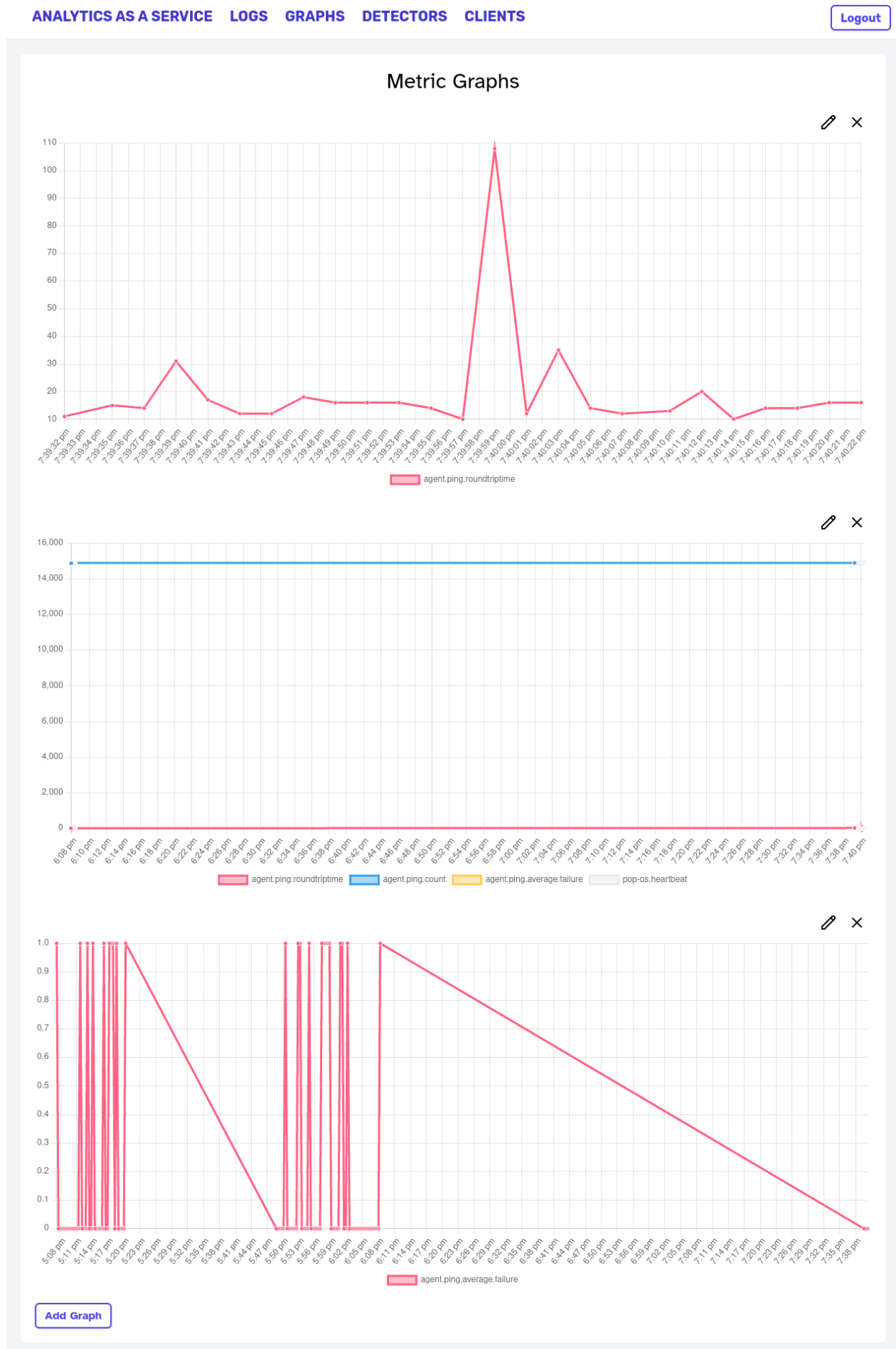


Figure 5. Metrics

5.3. Metrics

Zum Erstellen von neuen Graphen gibt es ein Modal, welches auf einem Reactive Form basiert und einen Preview von dem neuen Graphen anzeigt. Der Modal zum Editieren von bestehenden Graphen sieht ident aus. Das Reactive Form nutzt Autocomplete, um Vorschläge zu geben. Es können natürlich mehr Metriken angegeben werden, um diese in einem Graphen anzeigen zu können, oder es wird der Metrik-Filter komplett weggelassen um wirklich alle Metriken sehen zu können.



Figure 6. New Graph

Die Graphen werden alle im Local-Storage gespeichert, und werden nicht am Backend persistiert.

5.4. Detectors

Hier werden alle Detectors, inklusive Heartbeat Detektoren angezeigt. Heartbeat-Detektoren sind dafür zuständig zu überprüfen, ob eine Instanz eines Clients noch aktiv ist. Daher werden diese extra gekennzeichnet und können nicht editiert werden. Diese können wieder bearbeitet, erstellt oder gelöscht werden, mit bestimmten Einschränkungen. Aktivierte Detektoren werden mit einem roten Herz gekennzeichnet. Zusätzlich gibt es einen einfachen Toggle um Detektoren deaktivieren und aktivieren zu können.

Bestehende Detektoren können bearbeitet werden. Das Form ist wieder ein Reactive Form, welches mit eigenen Validatoren validiert wird.

Update Detector

Metric Name: agent.ping.roundtriptime

Execution Interval: 0 : 30 : 0

Offset: 0 : 30 : 0

Activated ☒

Detector Type: Interval

Interval Detector

Aggregate Operation: Max

Compare Type: Greater

Threshold: 40

Action

Endpoint: S1910307103@fhooe.at

Endpoint Type: Email

Submit

Figure 7. Update Detector

Für neue Detektoren gibt es wieder ein eigenes Form, welches auch validiert wird.

Add Detector

Metric Name:

Field is required.

Execution Interval:

hh

:

mm

:

ss

At least hours, minutes or seconds is required and have to be numbers

Offset:

hh

:

mm

:

ss

At least hours, minutes or seconds is required and have to be numbers

Activated

☐

Detector Type:

MinMax

MinMax Detector

Lower Threshold:

Upper Threshold:

Max Hits:

Action

Endpoint:

Field is required.

Endpoint Type:

Field is required.

"" is not in defined range [Email, Webhook]


Lower threshold is required and has to be a number.

Figure 8. new Detector With Errors

Detectors





POP-OS.HEARTBEAT DETECTOR

Metric Name:	pop-os.heartbeat
Execution Interval:	00h 30m 00s
Offset:	00h 30m 00s
Last Executed:	2022-01-23 09:03
Aggregate Operation:	CurrentValue
Compare Type:	Smaller
Threshold:	1
Endpoint:	andreas.wenzelhuemer@gmail.com







AGENT.PING.ROUNDTRIP TIME DETECTOR

Metric Name:	agent.ping.roundtrip time
Execution Interval:	00h 30m 00s
Offset:	00h 30m 00s
Last Executed:	2022-01-23 08:23
Aggregate Operation:	Max
Compare Type:	Greater
Threshold:	40
Endpoint:	S1910307103@fhoee.at







AGENT.PING.ROUNDTRIP TIME DETECTOR

Metric Name:	agent.ping.roundtrip time
Execution Interval:	00h 30m 00s
Offset:	00h 30m 00s
Last Executed:	2022-01-23 08:52
Lower Threshold:	10
Upper Threshold:	40
Max Hits:	10
Endpoint:	google.com







AGENT.PING.AVERAGE.FAILURE DETECTOR

Metric Name:	agent.ping.average.failure
Execution Interval:	00h 30m 00s
Offset:	00h 30m 00s
Last Executed:	2022-01-23 08:23
Aggregate Operation:	Max
Compare Type:	Greater
Threshold:	0
Endpoint:	S1910307106@fhoee.at



AGENT.PING.AVERAGE.FAILURE DETECTOR

Metric Name:	agent.ping.average.failure
Execution Interval:	00h 30m 00s
Offset:	00h 30m 00s
Last Executed:	0001-01-01 12:00
Aggregate Operation:	Min
Compare Type:	Smaller
Threshold:	1
Endpoint:	http://google.com



Create Detector

Figure 9. Detectors

5.5. Clients

Zuletzt noch eine Übersicht von Clients und deren AppKeys. AppKeys werden standardmässig versteckt angezeigt und die Klartext Ansicht muss getogglt werden. AppKeys können auch einfach kopiert werden ohne sie explizit anzeigen zu müssen.

ANALYTICS AS A SERVICE









































LOGS

GRAPHS

DETECTORS

CLIENTS

Logout

Id	App Key	Actions
Client-1	*****	 
Client-2	*****	 
Client-3	*****	 
Client-4	*****	 
Client-5	*****	 
Client-6	*****	 
Client-7	*****	 
Client-8	*****	 
Client-9	*****	 
Client-10	*****	 
Client-11	*****	 
Client-12	*****	 
Client-13	*****	 
Client-14	*****	 
Client-15	*****	 
Client-16	*****	 
Client-17	*****	 
Client-18	*****	 
Client-19	*****	 
Client-20	*****	 

Create Client

Figure 10. Clients

Einen neuen Client zu erstellen ist auch sehr einfach:

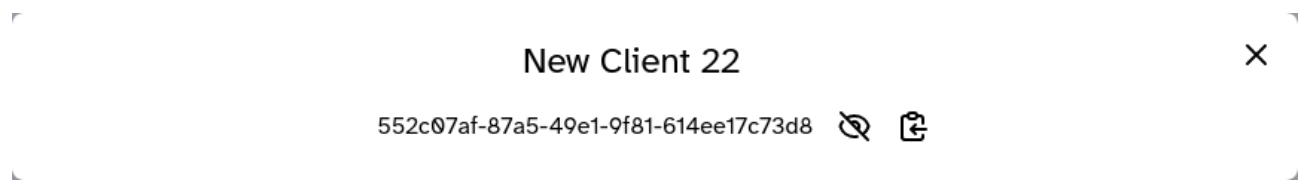


Figure 11. New Client

5.6. Toasts

Zum Darstellen von Error-Messages oder auch Success-Messages wurde eine Toast-Komponente und ein dazugehöriger Service implementiert.

Listing 3. toast-service

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';
import { Toast, ToastType } from '../model/toast';

@Injectable({
  providedIn: 'root'
})
export class ToastService {

  toasts = new Subject<Toast>()

  constructor() { }

  sendToast(toast: Toast) {
    this.toasts.next(toast)
  }

  sendFine(message: string) {
    this.toasts.next({
      message: message,
      type: ToastType.Fine
    })
  }

  sendWarning(message: string) {
    this.toasts.next({
      message: message,
      type: ToastType.Warning
    })
  }

  sendError(message: string) {
    this.toasts.next({
      message: message,
      type: ToastType.Error
    })
  }
}
```

Listing 4. toast-component

```
import {Component, ElementRef, OnInit} from '@angular/core';
import {ToastService} from '../../services/toast.service';
import {Toast} from '../../model/toast';

@Component({
  selector: 'app-toast',
  templateUrl: './toast.component.html',
  styleUrls: ['./toast.component.css']
})
export class ToastComponent implements OnInit {

  toasts = this.toastService.toasts
  toast: Toast
  timeout: number | null

  constructor(private toastService: ToastService,
               private el: ElementRef) {

  }

  ngOnInit(): void {
    this.toasts.subscribe(toast => {
      this.toast = toast
      this.el.nativeElement.classList.remove('fade-out')
      this.el.nativeElement.classList.add('fade-in')
      if(this.timeout) {
        clearTimeout(this.timeout)
      }
      this.timeout = setTimeout(() => {
        this.el.nativeElement.classList.remove('fade-in')
        this.el.nativeElement.classList.add('fade-out')
      }, 4000)
    })
  }
}
```

6. Authentifizierung

Es wird ein eigener Keycloak-Server verwendet, da dieser sehr einfach zu starten ist. Die Konfiguration wurde zum Teil mit dem `environment.ts` gelöst.

Listing 5. auth.config.ts

```
import {AuthConfig} from 'angular-oauth2-oidc';
import {environment} from "../environments/environment";

// INTROSPECTION
// with Keycloak 11.0.2
// siehe
https://www.keycloak.org/docs/latest/securing\_apps/index.html#endpoints-2

export const authConfig: AuthConfig = {
  issuer: `${environment.oauthUrl}/auth/realms/${environment.realmName}`,
  loginUrl: `${environment.oauthUrl}/auth/realms/${environment.realmName}/protocol/openid-connect/auth`,
  logoutUrl: `${environment.oauthUrl}/auth/realms/${environment.realmName}/protocol/openid-connect/logout`,
  tokenEndpoint: `${environment.oauthUrl}/auth/realms/${environment.realmName}/protocol/openid-connect/token`,
  sessionCheckIframeUrl: `${environment.oauthUrl}/auth/realms/${environment.realmName}/protocol/openid-connect/login-status-iframe.html`,
  userinfoEndpoint: `${environment.oauthUrl}/auth/realms/${environment.realmName}/protocol/openid-connect/userinfo`,
  clientId: environment.realmName,
  redirectUri: window.location.origin + '/index.html',
  silentRefreshRedirectUri: window.location.origin + '/silent-refresh.html',
  scope: 'profile email',
  silentRefreshTimeout: 5000, // For faster testing
  timeoutFactor: 0.25, // For faster testing
  sessionChecksEnabled: true,
  showDebugInformation: false, // Also requires enabling "Verbose" level in devtools
  clearHashAfterLogin: false,
  requireHttps: false
};
```

Für Zugriffe etc wurde ein eigener auth-service erstellt.

Listing 6. auth-service

```
import {Injectable} from '@angular/core';
import {JwksValidationHandler, OAuthService} from 'angular-oauth2-oidc';
import {authConfig} from '../auth.config';
import {ToastService} from './toast.service';

@Injectable({
  providedIn: 'root'
})
export class AuthService {

  constructor(private oauthService: OAuthService,
               private toastService: ToastService) {

  }

  configure() {
    this.oauthService.configure(authConfig)
    this.oauthService.tokenValidationHandler = new
JwksValidationHandler()
    this.oauthService.loadDiscoveryDocumentAndTryLogin()
      .catch(() => {
        this.toastService.sendError('Could not connect to auth-server')
        this.logout()
      })
  }

  login(): boolean {
    this.oauthService.initImplicitFlow()
    return true
  }

  isLoggedIn(): boolean {
    return this.oauthService.hasValidAccessToken() && this.
oauthService.hasValidIdToken();
  }

  logout() {
    this.oauthService.logout()
  }
}
```

7. Loader-Interceptor

Der Interceptor zeigt eine Lade-Animation an während die Daten geladen werden.

Listing 7. loader-service

```
import {Injectable} from '@angular/core';
import {Subject} from "rxjs";

@Injectable({
  providedIn: 'root'
})
export class LoaderService {

  isLoading = new Subject<boolean>()

  constructor() {
  }

  show() {
    this.isLoading.next(true)
  }

  hide() {
    this.isLoading.next(false)
  }
}
```

Listing 8. loader-interceptor

```
import {Injectable} from '@angular/core';
import {HttpEvent, HttpHandler, HttpInterceptor, HttpRequest} from
 '@angular/common/http';
import {finalize, Observable} from 'rxjs';
import {LoaderService} from "../services/loader.service";

@Injectable()
export class LoaderInterceptor implements HttpInterceptor {
  constructor(public loaderService: LoaderService) {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable
  <HttpEvent<any>> {
    this.loaderService.show();
    return next.handle(req).pipe(
      finalize(() => this.loaderService.hide())
    );
  }
}
```