ti&m

Authentisierung und Autorisierung in einer Microservice-Applikation

CH-Open Workshop-Tage 2021

Matthias Graf, Software Architekt

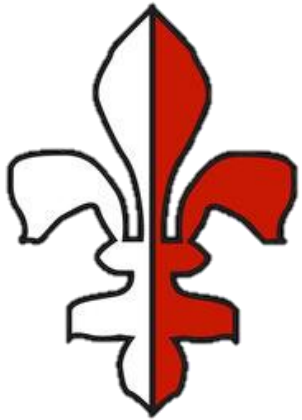
Zürich, September 2021

ti&m

Einführung

ti&m

big ideas. creative technology.



BERN

ETH zürich

SLRG SSS

09:10	Einführung
10:30	Pause
10:45	Keycloak und OIDC
11:15	Grundstuktur Applikation
12:45	Mittagspause
14:00	Applikation Erweitern
15:30	Pause
16:45	Abschluss

Informationen

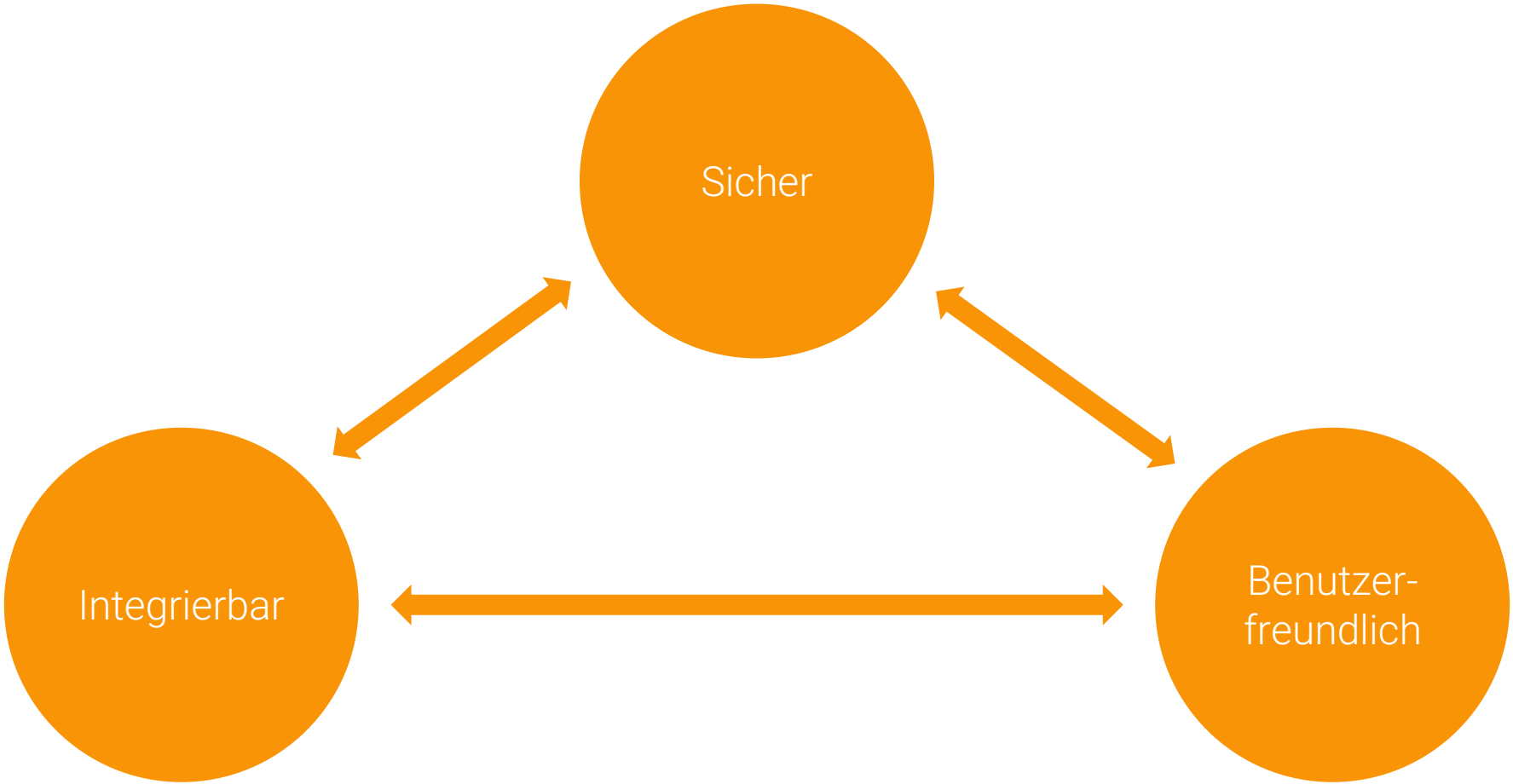
Unterlagen und Beispiele zum Workshop findet ihr auf Github

https://github.com/lizzyTheLizard/ch-open

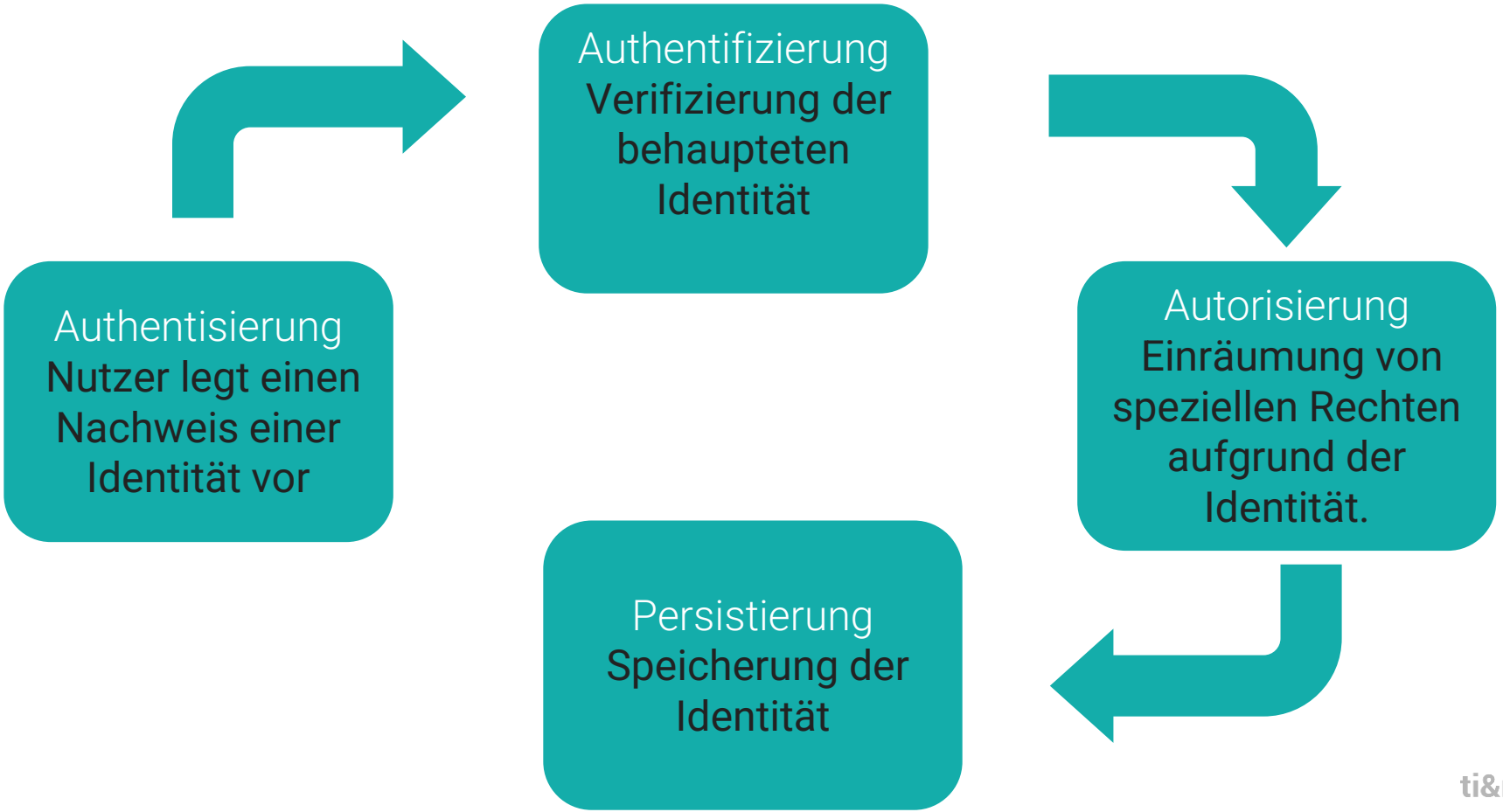
Die Unterlagen und Beispiele werden auch nach dem Workshop zur Verfügung stehen, bei Fragen könnt ihr euch an mich wenden

matthias.graf@ti8m.ch

Theorie Authentisierung


Sicher
Integrierbar
Benutzer-
freundlich

Wer bist du und was darfst du?



Wir kann ich meine Identität beweisen?

Wir kann ich meine Identität beweisen?

Ich weiss etwas

- Passwort
- Pin
- Sicherheitsfrage

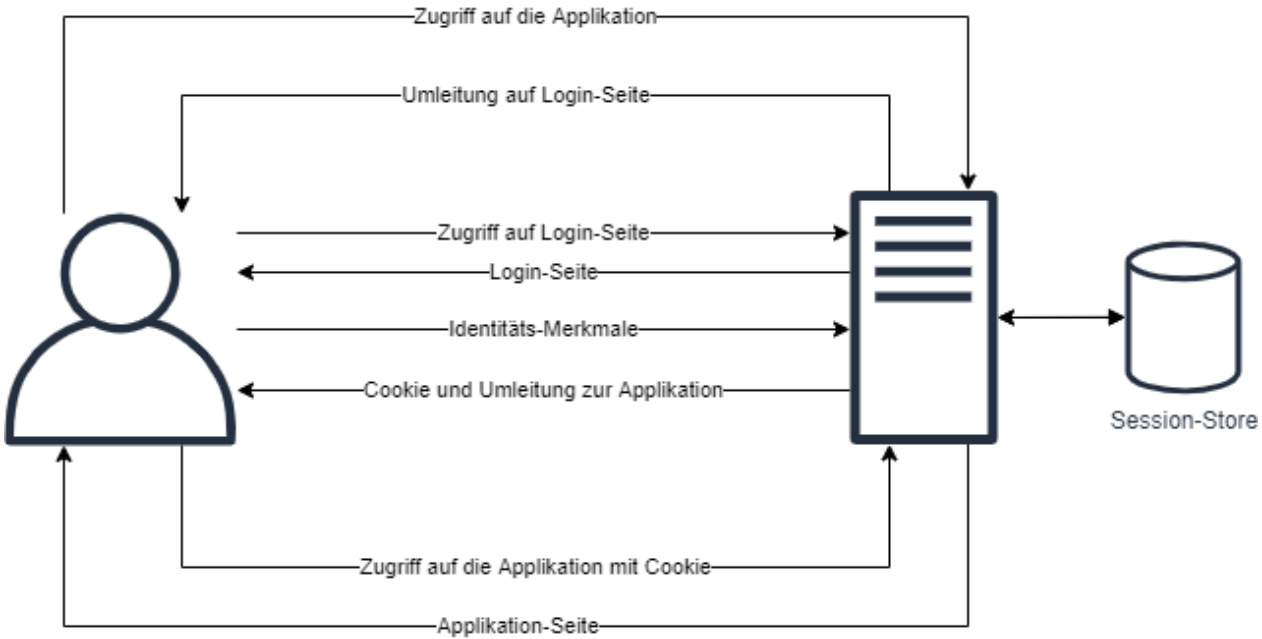
Ich habe etwas

- SmartCard
- SMS-Tan
- App
- Token
- Schlüssel

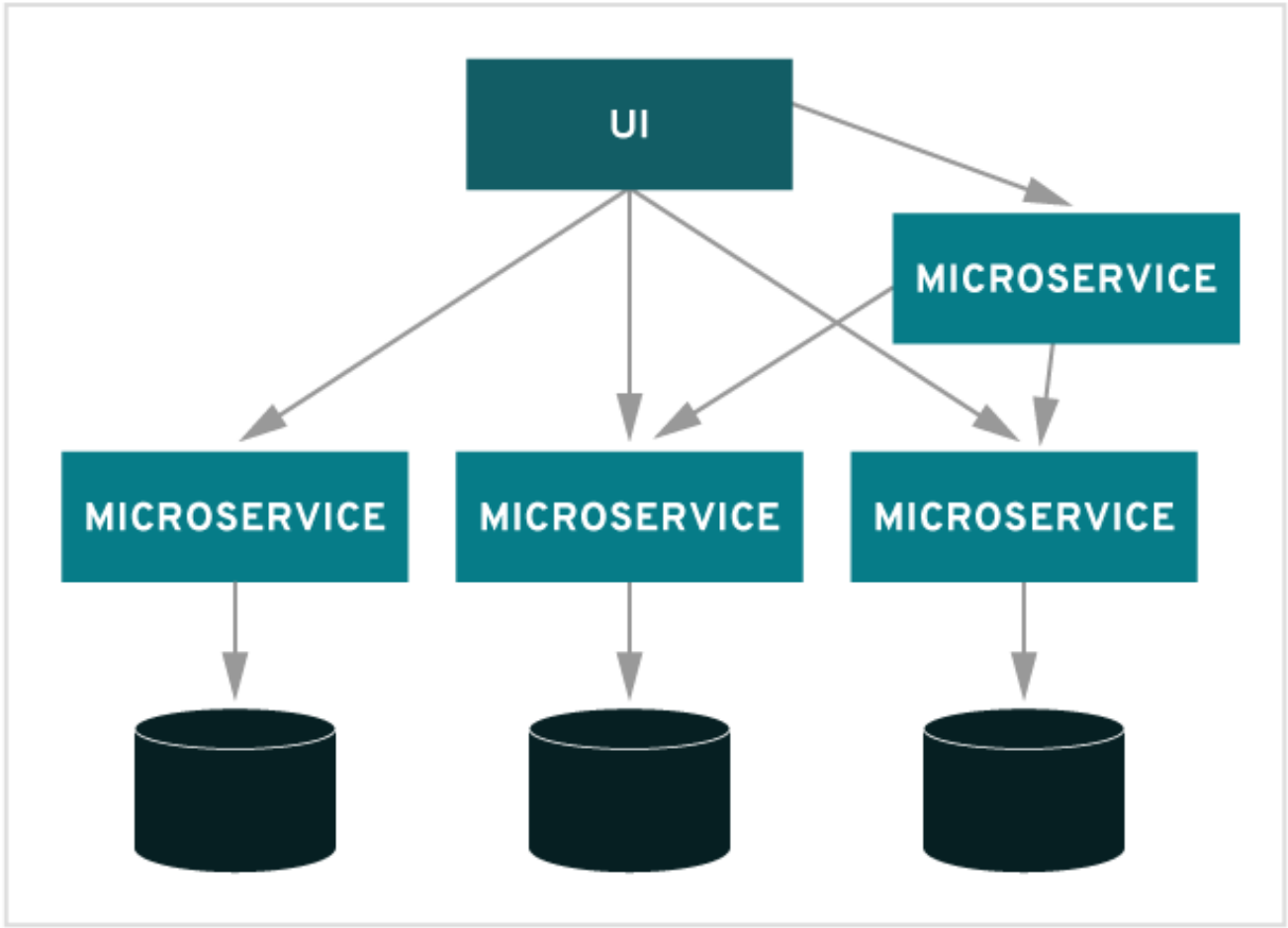
Ich bin etwas

- Fingerabdruck
- Gesichts
- Sprache
- Iris-Scan

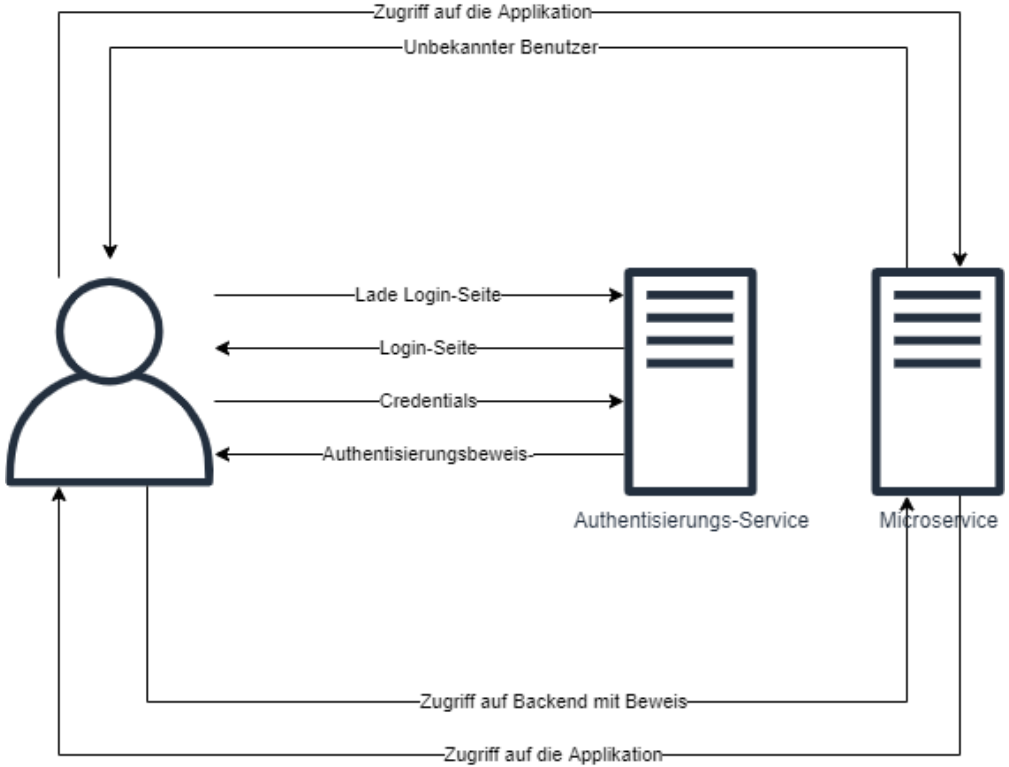
Authentisierung in einem Monolithen



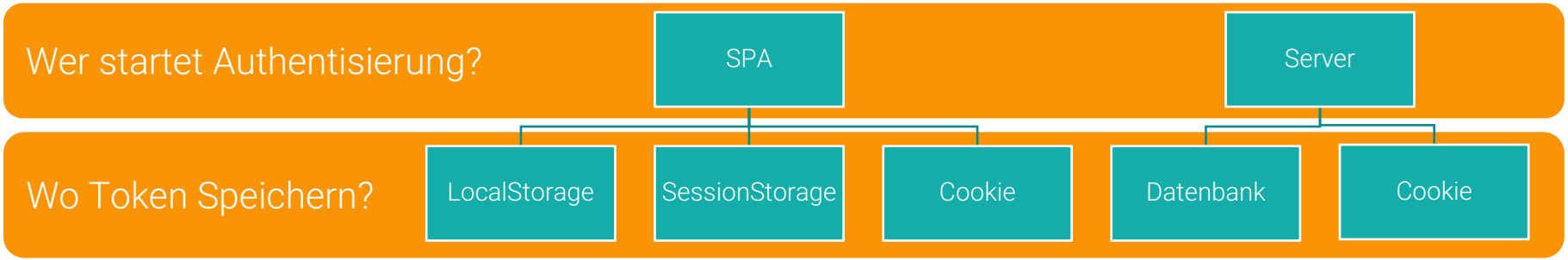
Microservices



Authentisierung mit Microservices

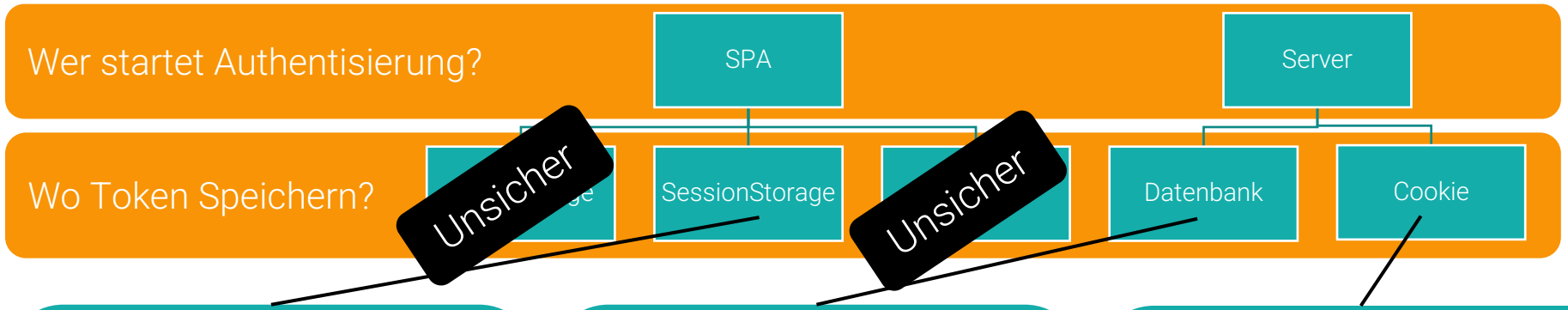


Verschiedene Möglichkeiten zum Persistieren (nicht abschliessend)



ti&m

Verschiedene Möglichkeiten zum Persistieren (nicht abschliessend)



+ Einfach Umzusetzen
- Statefull
- Kein SPA mit Public / Private möglich
- Jeder Microservice braucht eigenen Login

+ Einfach Umzusetzen
+ Stateless
- Kein SPA mit Public / Private
- Token muss in jedem Request übertragen werden
- Gemeinsame Domain notwendig

+ Stateless
+ Flexibel
- Token muss in jedem Request übertragen werden
- Jedes Tab braucht eigenen Login

Keycloak Installieren

Keycloak

– Open-Source Authorization Server
– Unterstützt OAuth2, OIDC und SAML
– Unterstützt unterschiedliche Authentisierungsarten
– Kann einfach (SAML und OIDC) Authorization Server anbinden
– Kann erweitert werden (Java)
– Wird kommerziell von RedHat als RH-SSO vertrieben

Keycloak Installieren

Ziel: Starten von Keycloak auf den eigenen Computer

Die notwendigen Informationen findet ihr unter

https://github.com/lizzyTheLizard/ch-open/blob/main/docs/keycloak.md

OIDC

OAuth 2.0 und OpenID Connect

OpenID: Als Benutzer will ich mich einmal bei einem OIDC-Provider anmelden und mein Login für mehrere Websites (Relying Party) verwenden (Single-Sign-On)

OAuth: Ein Benutzer (Resource Owner) besitze eine Ressource auf einem Server (Resource Server) und will einer Applikation (Client) mit einem Access-Token Zugriff auf diese geben.

OAuth seit 2006, OpenID seit 2005. Seit 2014 OpenID-Connect basierend auf OAuth 2.0

Definiert nicht wie die Authentisierung durchgeführt wird, noch wie der Access-Token aussieht (in der Regel aber JWT)

OAuth 2.0 Rollen

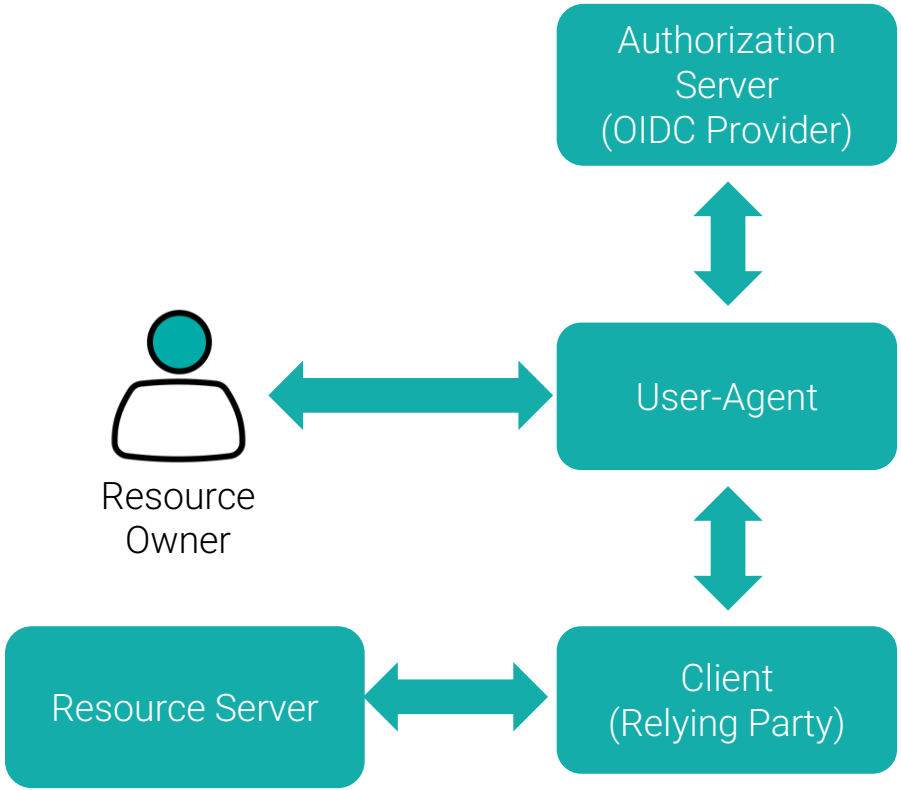
Resource Owner: Benutzer

User-Agent: Browser, kann aber auch Teil einer Applikation sein (z.B. FatClient, Mobile-App)

Client: Eine Applikation, kann eine SPA (läuft Lokal im Browser oder eine Web-Applikation sein)

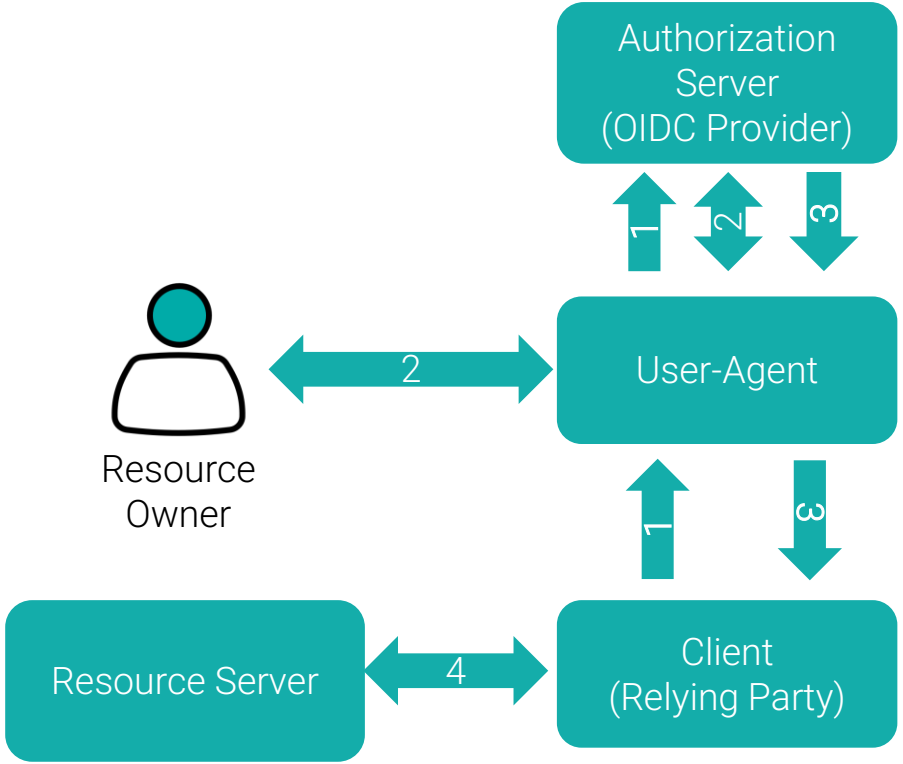
Resource Server: Service auf den der Client zugreifen will (z.B. ein Microservice)

Authorization Server: Service der Benutzer authentisieren kann



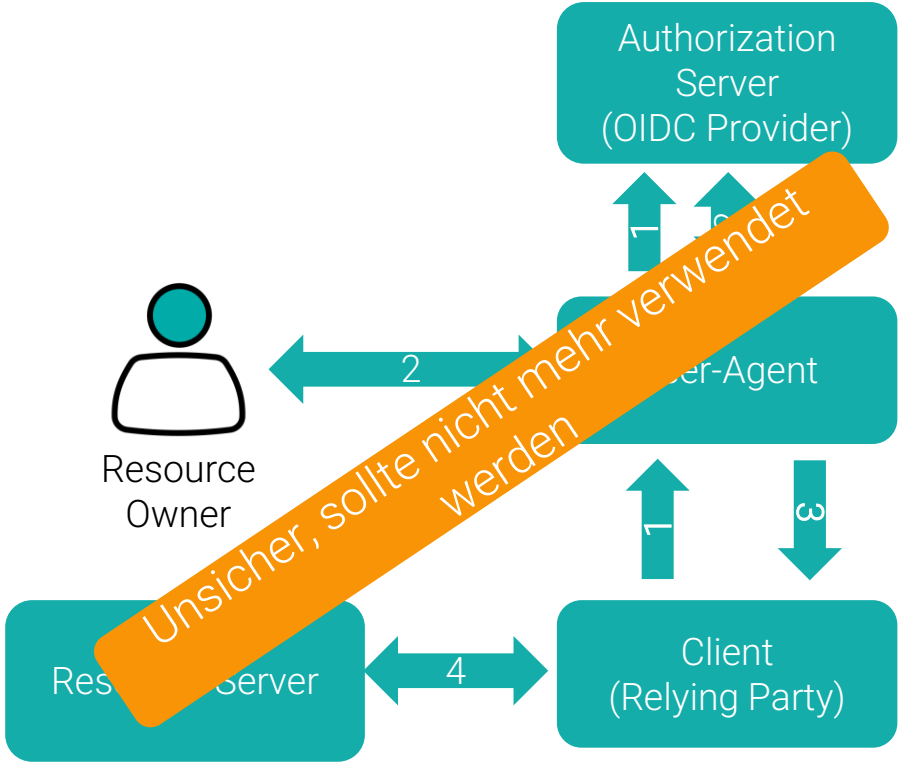
OIDC / OAuth2 Implicit Flow

1. Client will Zugriff auf ein e Ressource und schickt einen **AuthenticationRequest** über User-Agent an Authorization Server
2. Benutzer authentisiert sich beim Authorization Server (falls notwendig)
3. Authorization Server schickt einen **Token-Response** an den Client
4. Client kann mit Token auf Ressource zugreifen



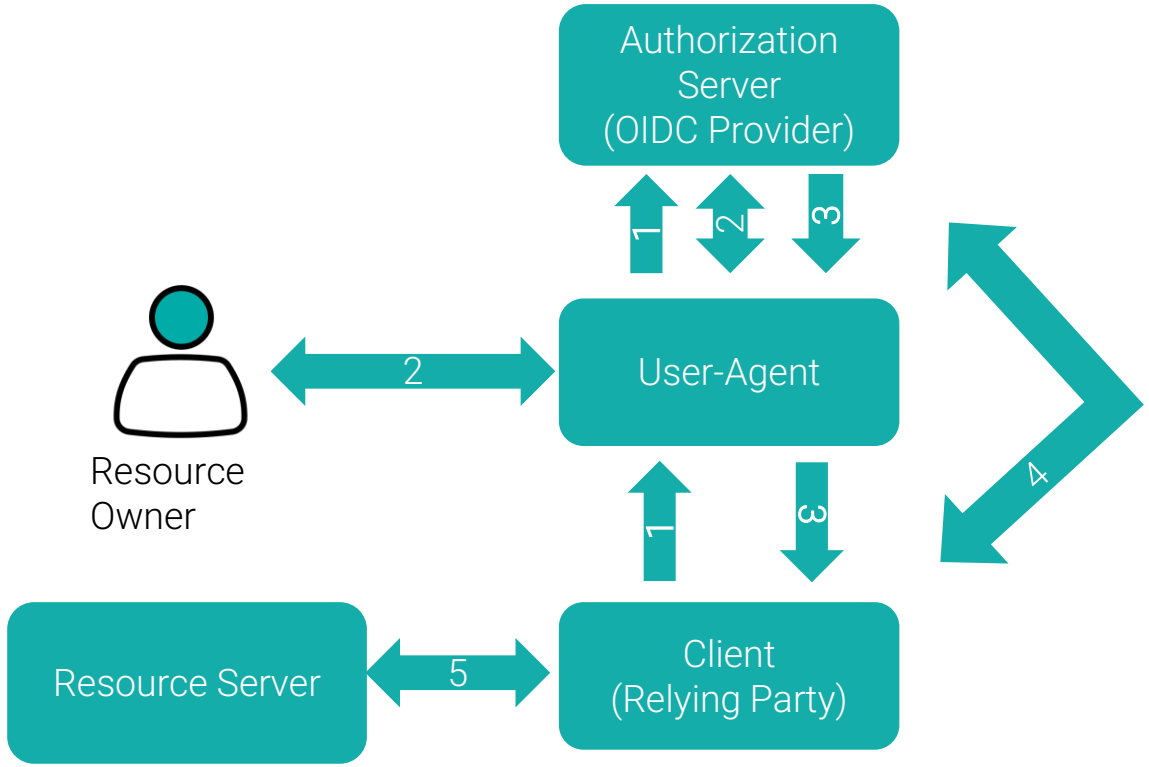
OIDC / OAuth2 Implicit Flow

1. Client will Zugriff auf eine Ressource und schickt einen **AuthenticationRequest** über User-Agent an Authorization Server
2. Benutzer authentisiert sich beim Authorization Server (falls notwendig)
3. Authorization Server schickt einen **Token-Response** an den Client
4. Client kann mit Token auf Ressource zugreifen



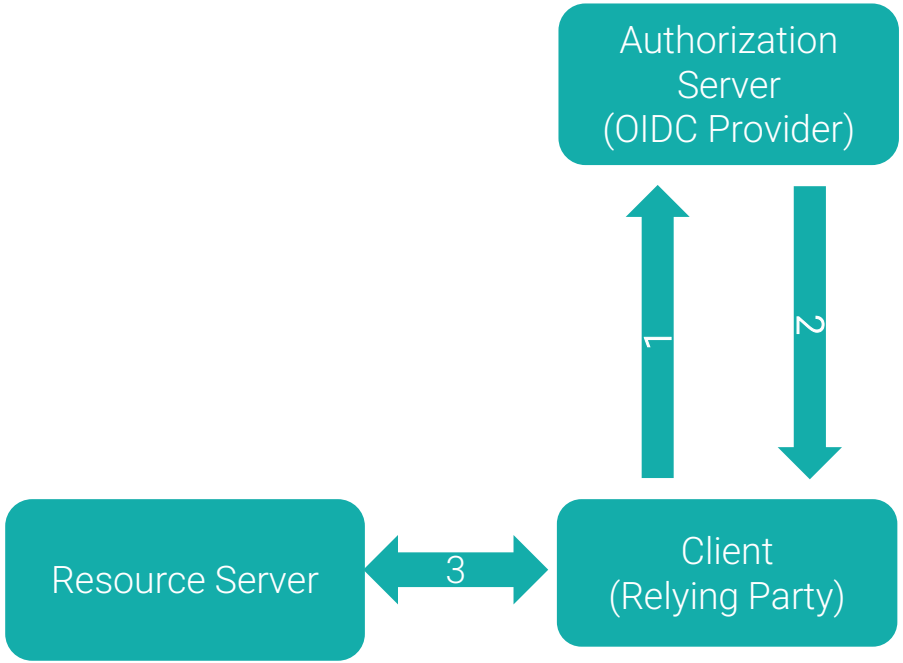
OIDC / OAuth2 Code Flow

1. Client will Zugriff auf eine Ressource und schickt einen **AuthenticationRequest** über User-Agent an Authorization Server
2. Benutzer authentisiert sich beim Authorization Server (falls notwendig)
3. Authorization Server schickt eine **Code-Response** an den Client
4. Client kann sich mit Code einen Token holen
5. Client kann mit Token auf Ressource zugreifen

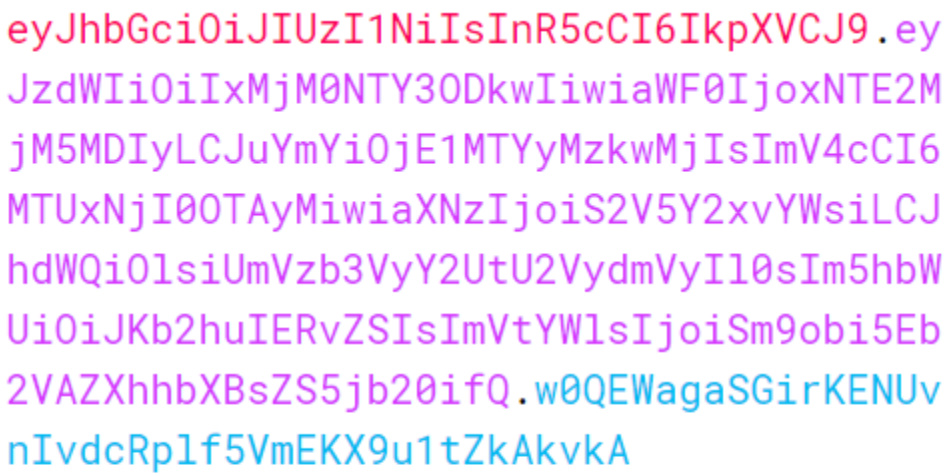


OIDC / OAuth2 Client Credentials Flow

1. Client will zugriff auf Ressource, schickt AuthenticationRequest inkl. seinem Credential an Authorization Server
2. Authorization Server schickt einen Access-Token an den Client
3. Client kann auf Ressource zugreifen



JWT



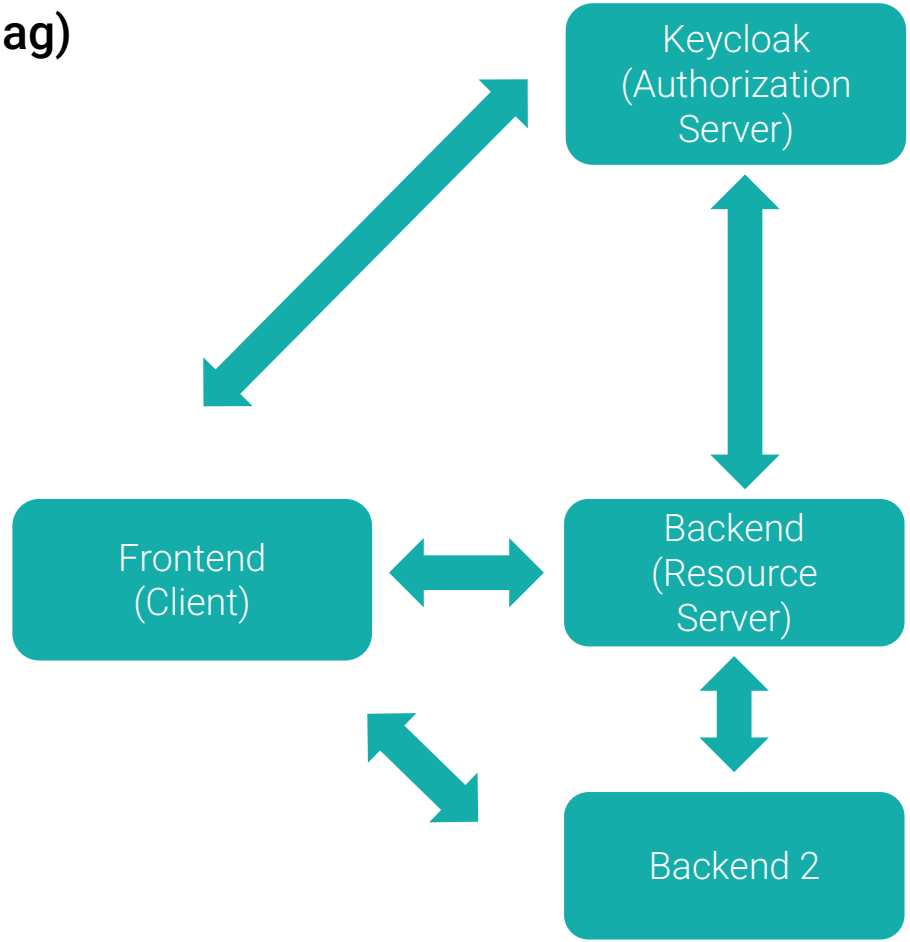

```
{
  "alg": "HS256",
  "typ": "JWT"
}

{
  "sub": "1234567890",
  "iat": 1516239022,
  "nbf": 1516239022,
  "exp": 1516249022,
  "iss": "Keycloak",
  "aud": ["Resource-Server"],

  "name": "John Doe",
  "email": "John.Doe@example.com"
}
```

Grundstruktur

Grundlegende Architektur (Vorschlag)

Wir bauen eine Microservice-Applikation mit einem Frontend (WebSite, SPA, MobileApp, RichClient) das mit einem oder mehreren Backend-Services sicher kommuniziert.

Back- und Frontend sollen zwei Applikationen sein, Umsetzung individuell z.B. Angular SPA und Java Backend



Notwendige Schritte

Konfigurieren Keycloak	https://github.com/lizzyTheLizard/ch-open/blob/main/docs/keycloak.md
Erstellen SPA mit Basis-Anbindung	https://github.com/lizzyTheLizard/ch-open/blob/main/docs/frontend.md
Erstellen einer SpringBoot-Applikation	https://github.com/lizzyTheLizard/ch-open/blob/main/docs/backend.md

Erweiterungen

Silent-Refresh

Ziel

Automatischer Neubezug des Tokens

Vorgehen

- Hinzufügen silent-refresh.html
- Hinzufügen neue Redirect-URI in Keycloak
- Konfigurieren Silent-Refresh im Frontend
- Listener für Authentisierungs-Events im Frontend

Vorteile

– Session läuft nicht mehr ab

Auto-Login

Ziel

Erkennen bei Login ob der Benutzer bereits eingeloggt ist

Vorgehen

- Wenn Benutzer nicht eingeloggt ist, manuelles durchführen eines Silent-Refresh bei Start

Vorteile

– Benutzer ist in mehreren Tabs eingeloggt

Rollenüberprüfung

Ziel

Ich kann eine Benutzer Rollen zuweisen und nur die Benutzer mit diesen Rollen könne gewisse Funktionen ausführen.

Vorgehen

- In Keycloak Rollen vergeben
- GrantedAuthoritiesExtractor konfigurieren
- Rollen-Check implementieren

Vorteile

- Einfacher Rechte-Check

Identitäts-Federation

Ziel

Ich will mein Google (Microsoft, LinkedIn…)- Account benutzen um mich einloggen zu können. Das Back- und Frontend sollte von dieser Änderung nicht betroffen sein.

Vorgehen

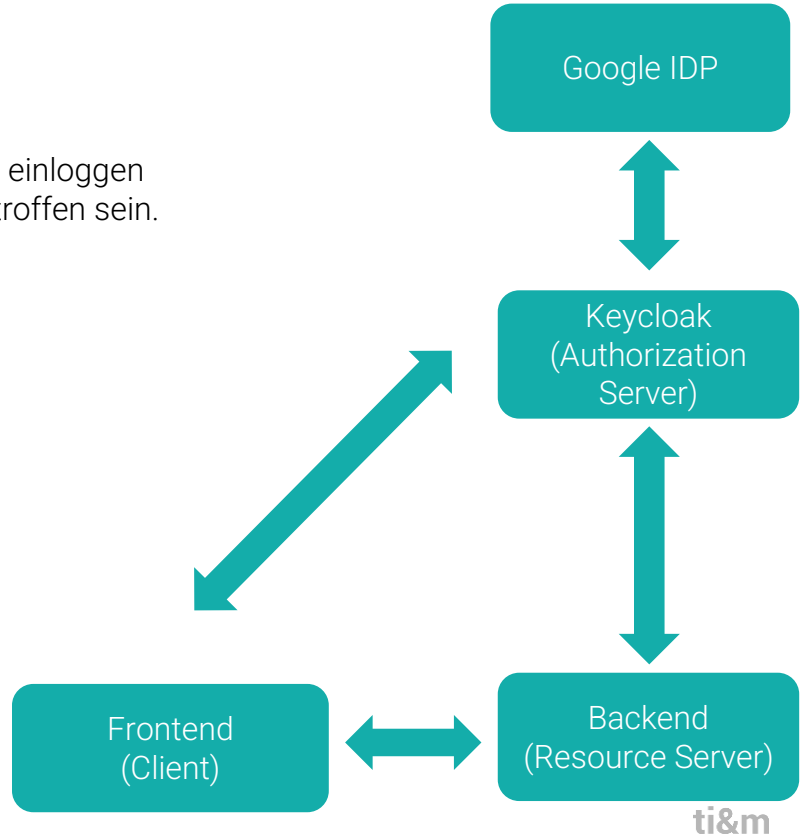
- Konfigurieren IDP in Keycloak

Vorteile

– Einfach für Benutzer, können ihre bestehenden Accounts benutzen

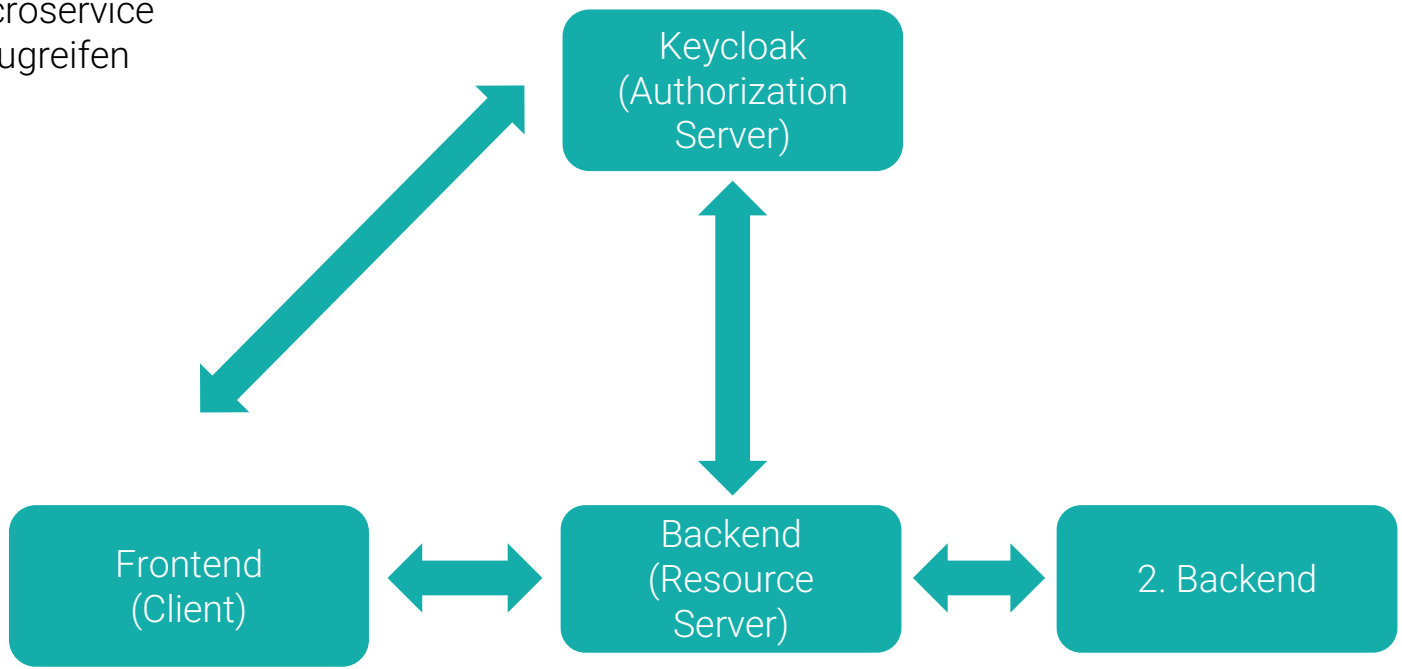
– Einfach im Support (keine Password-Vergessen o.ä)

– Sicherer Login ohne viel Aufwand

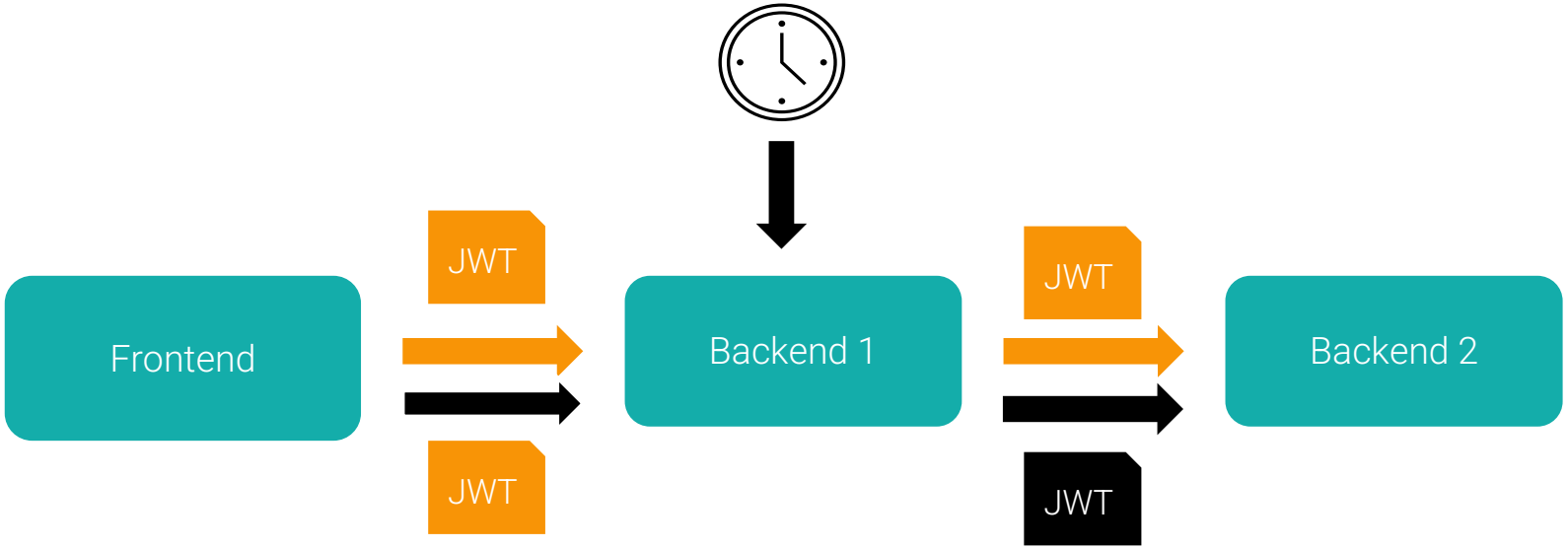


System-2-System Kommunikation

Ziel: Ich will aus einem Microservice sicher auf einen zweiten Zugreifen können



System-2-System Kommunikation: 2 Möglichkeiten



User-Kontext: Backend 1 leitet Benutzertoken weiter an Backend 2

System-Kontext: Backend 1 authentisiert sich mittels Client-Credentials selbst und nutzt seinen JWT für die Kommunikation zu Backend 2

Keycloak-Plugins

Ziel

Keycloak mit Plugins ergänzen

Vorgehen

- Keycloak SPI-Klassen erweitern und in Docker-Image hochladen
- States etc. konfigurieren

Vorteile

– Eigene Authentisierungsschritte können umgesetzt werden

We digitalize your company.

ti8m.com

ti&m AG
Zurich
Buckhauserstrasse 24
CH-8048 Zurich
+41 44 497 75 00

ti&m AG
Bern
Monbijoustrasse 68
CH-3007 Bern
+41 44 497 75 00

ti&m GmbH
Frankfurt am Main
Schaumainkai 91
D-60596 Frankfurt am Main
+49 69 247 5584 20

ti&m Pte. Ltd.
18 Robinson Road #15-16
Singapore 048547
Singapore
+65 6955 7755

ti&m