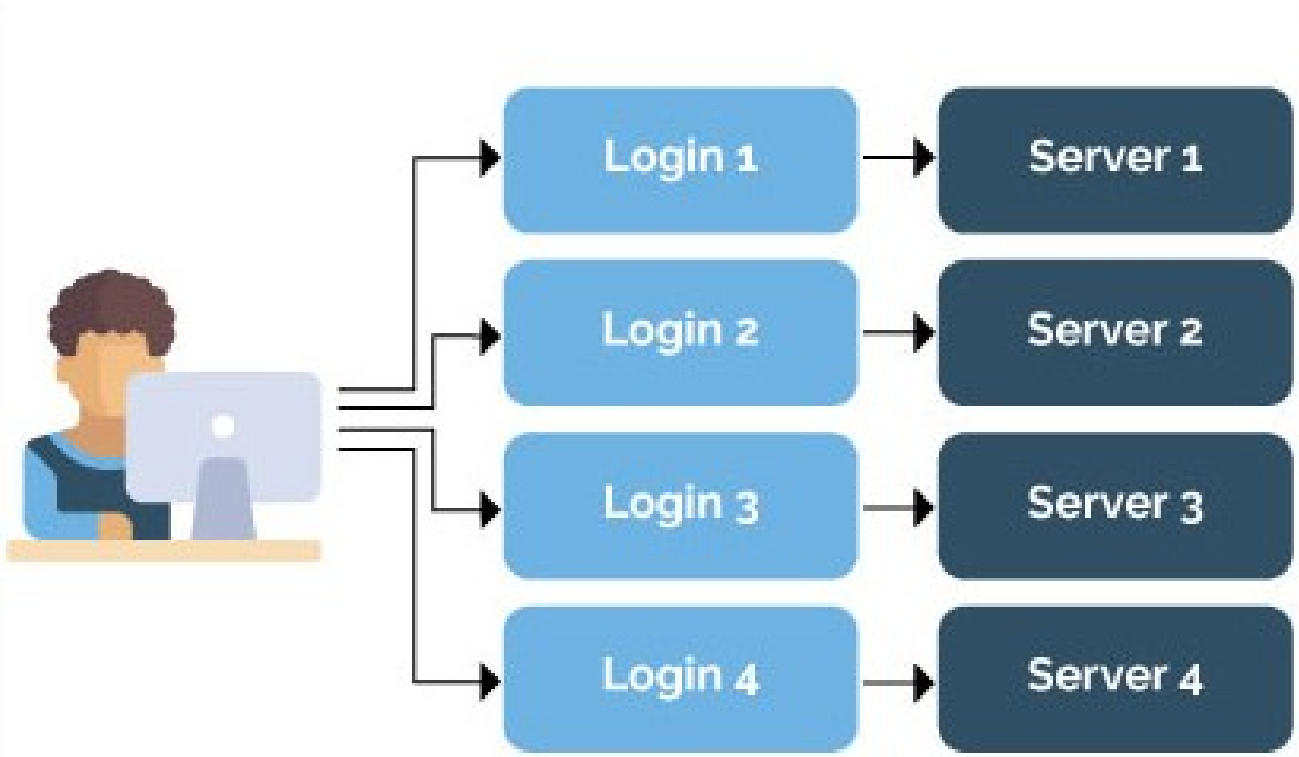

Login 1
Server 1
Login 2
Server 2
Login 3
Server 3
Login 4
Server 4

Quelle: [1]

Single Sign On bei Webanwendungen

Daniel Ebert 65926

Hochschule Aalen

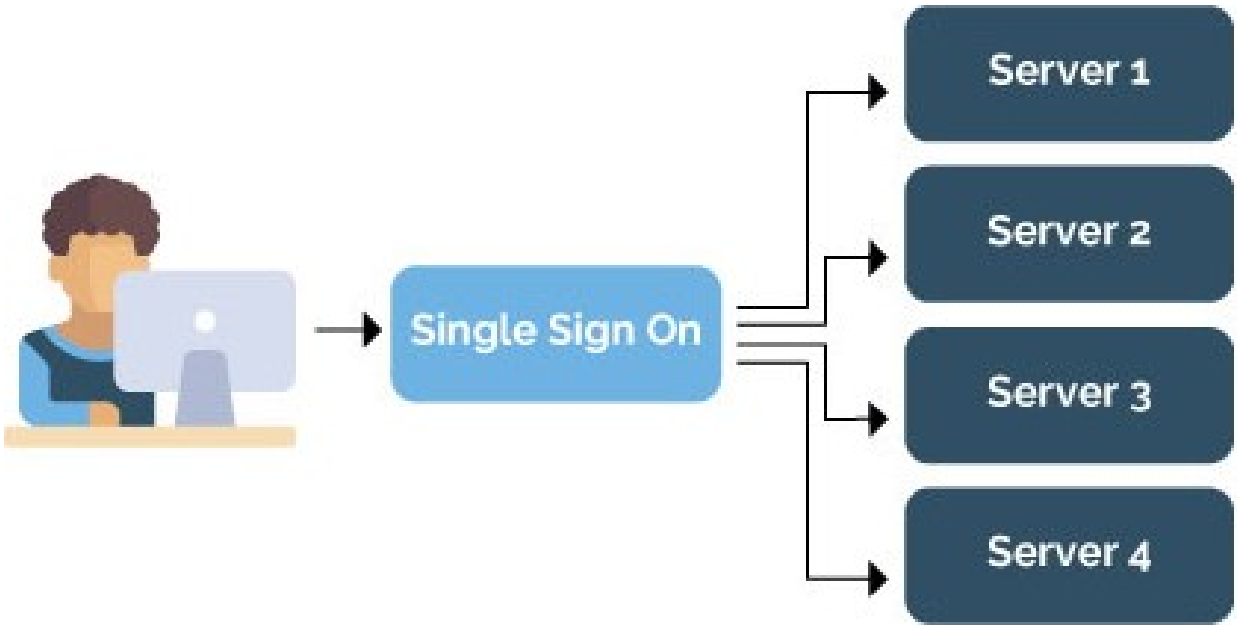
daniel.ebert01@studmail.htw-aalen.de

4. Januar 2021

Overview

Single Sign On (SSO)

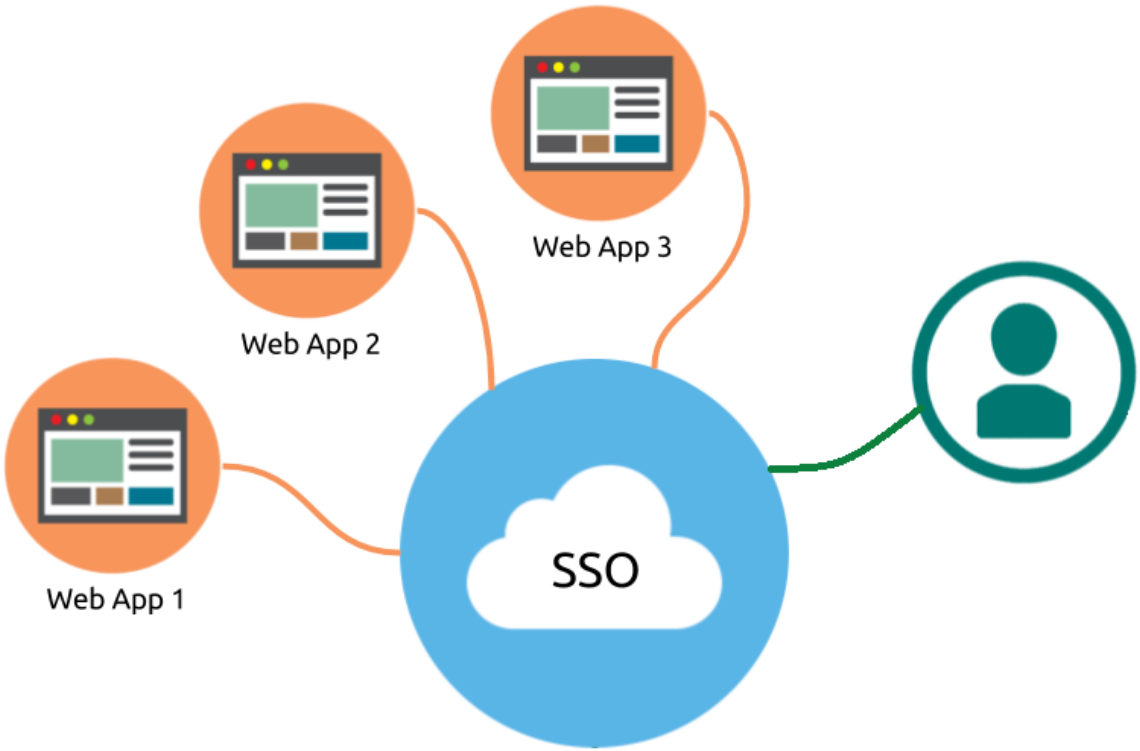
- Benutzer durchläuft ein Authentifizierungsverfahren
- Danach übernimmt SSO-Mechanismus das Authentifizieren



Quelle: [1], [Mar20]

Beispiel SSO Use Case

- Eine Gruppe von Webseiten nimmt am SSO teil
- Benutzer loggt sich auf einer dieser Webseiten mit seinen Anmeldedaten ein
- Dann wird der Benutzer auf anderen Webseiten automatisch eingeloggt



Quelle: [2]

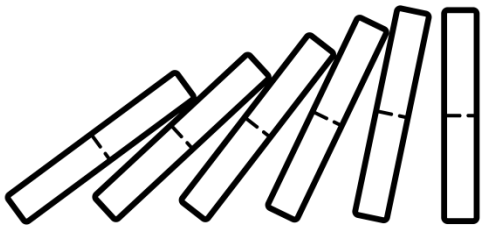
Vorteile

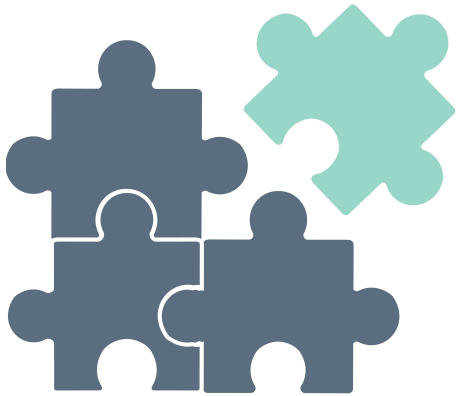
- Psychological Acceptability
  - Keine mehrfachen Anmeldungen
  - Ein Passwort für mehrere Webanwendungen
- Einfachere Entwicklung
  - Ein System zur Authentifizierung
- Einfachere Administration
  - Alle Benutzerdaten an einem Ort

Quelle: [BK16]

Nachteile

- Single Point of Failure
  - Redundanz durch mehrere Instanzen
- Bereits existierende Systeme an SSO anbinden kann schwierig sein
  - Keycloak Adapter vereinfachen Anbindung





Quelle: [14], [15], [Mat20], [Red20a]

Alternativen

- Verschiedene Arten von SSO
  - SSO für Webanwendungen
  - Alternative: SSO für Intranet/Enterprise
- Verschiedene Protokolle für SSO
  - OpenID Connect (OIDC)
  - Alternativen: SAML 2.0, Kerberos
- Verschiedene Implementierungen von SSO-Systemen
  - Keine zwei SSO Implementierungen sind gleich
  - Keycloak
  - Alternativen: Okta, Auth0, Apereo CAS

Quelle: [RR12], [Red20b]

OIDC Teilnehmer/Rollen

- Benutzer
- Clients
- OpenID Provider

Benutzer

- Menschliche Teilnehmer
- Kann sich mittels Authentifizierungsverfahrens authentifizieren
- Jeder Benutzer hat Claims

Quelle: [3], [Red20b], [N S14]

Claims

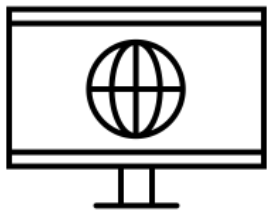
- Key-Value Paare
- Information über Benutzer, z.B. Name, Adresse, E-Mail
- Information über Authentifizierung, z.B. wann und bei wem hat Authentifizierung des Benutzers stattgefunden
- Beispiel Claims:
  - 'sub' (Subject) Claim: Benutzer ID, erstellt von Issuer
  - 'iss' (Issuer) Claim: Bei welcher Instanz authentifiziert

Quelle: [7], [16], [N S14]

Clients

Drei Arten von Clients:

1. Ruft andere Clients/Services im Namen des authentifizierten Benutzers auf
   - z.B. Frontend Webanwendungen
   - Rufen geschützte Ressourcen von Backend Services ab
2. Backend Services, die Ressourcen bereitstellen
   - z.B. Ressourcenserver
   - Ressourcen beschränkt für authentifizierte und autorisierte Benutzer
3. Native Anwendungen
   - Laufen auf dem Gerät des Benutzers
   - Für Web SSO nicht interessant



Quelle: [4], [5], [Har12], [RS17]

OpenID Provider

- Authentifiziert Benutzer
- Speichert z.B. Claims der Benutzer und Konfiguration der Clients
- Stellt Endpunkte bereit für:
  - Authentifizierung eines Benutzers
  - UserInfo Endpunkt (Informationen über Benutzer)
  - Token Endpunkt (Tokens mit späterer Verfallszeit und/oder weniger Rechten)

Quelle: [6], [Red20b], [N S14]

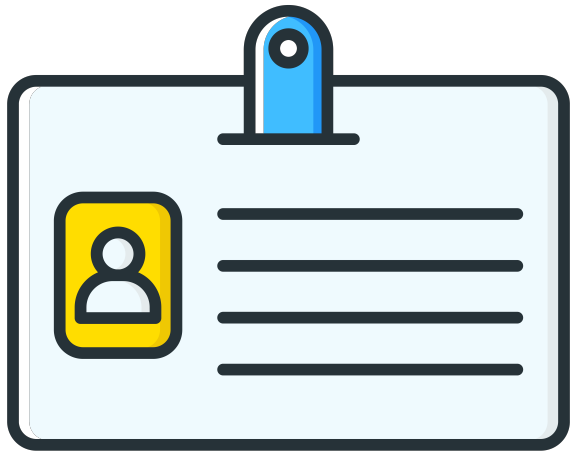
Tokens

- Tokens im JWT-Format
- Erstellt und signiert vom OpenID Provider
- Verschiedene Arten von Tokens:
  - ID Token
  - Access Token
  - Refresh Token
- 3 teilige Struktur:

Quelle: [8], [JBS15]

ID Token

- Enthält Claims über die Authentifizierung des Benutzers
  - z.B. 'auth_time' Claim: Wann hat die Authentifizierung stattgefunden?
- Optional weitere Claims mit Informationen über den Benutzer
- Ist nur für den an der Authentifizierung des Benutzers beteiligten Client gedacht
- Client validiert mit ID Token die Authentifizierung des Benutzers



Quelle: [9], [N S14]

Access Token

- Schlüssel für Ressourcen, welche für die authentifizierten und autorisierten Benutzer beschränkt ist
- Kann z.B. als Bearer Token im HTTP Authorization Header an Backend Services enthalten sein
- Sollte außer 'sub' Claim keine Informationen über Benutzer enthalten
- Informationen über Benutzer kann mit Access Token bei UserInfo Endpunkt angefordert werden

Quelle: [10], [Har12], [Red20b], [Ide20]

Access Token

- Mit Access Token Zugriff hat man Zugriff auf:
  - Ressourcen bei Backend Services
  - Benutzer Claims bei UserInfo Endpunkt
- Enthält 'scope' Claim
  - Spezifiziert Ressourcen und Benutzer Claims

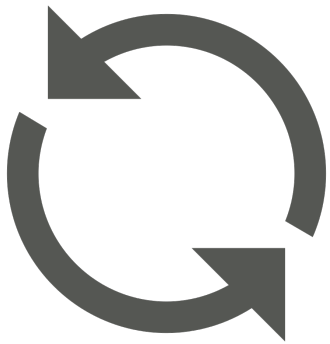
Access Token

Benutzer Claims
und Ressourcen
im scope Claim

Quelle: [10], [11], [Har12], [Red20b]

Refresh Token

- OIDC Tokens haben Verfallszeit ('exp' Claim)
- Abgelaufene Tokens sind nicht mehr gültig
- Mit Refresh Tokens können über den Token Endpunkt neue Tokens angefordert werden
  - Optional neue Tokens mit weniger Rechten

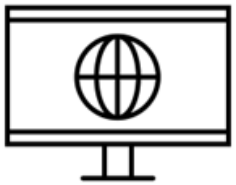


Quelle: [12], [JBS15], [N S14], [Har12]

OIDC Protokoll Flows

- Authorization Code Flow
- Zugriff auf geschützte Ressourcen
- Validieren des Access Tokens

Authorization Code Flow Schritt 1



Client

1. Authorization Request

Benutzer

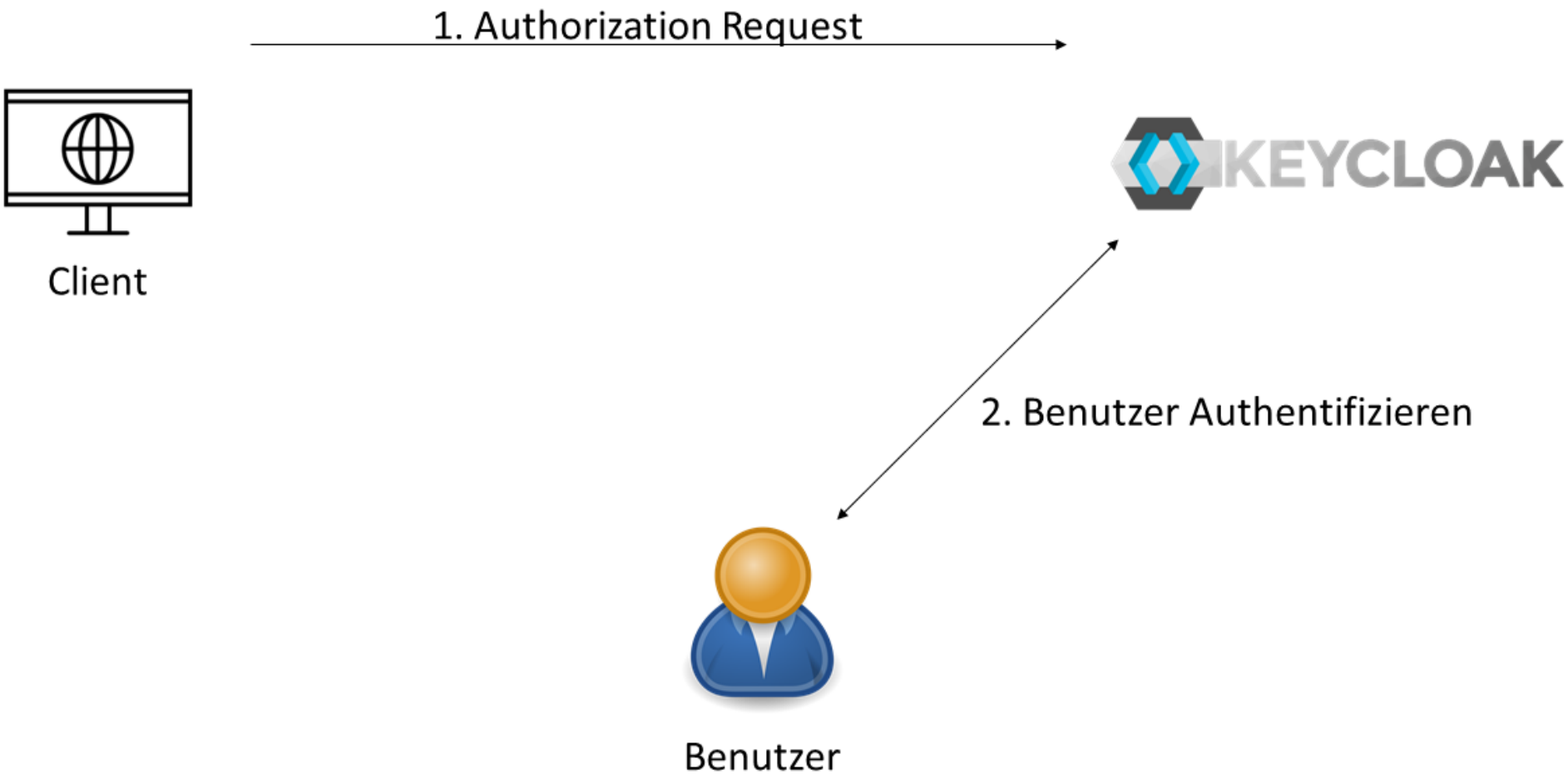
Quelle: [3], [4], [5], [N S14]

Authorization Code Flow Schritt 1 Beispiel

```
GET https://keycloak/auth/realms/.../auth?
        client_id=frontend1
        &redirect_uri=https%3A%2F%2Ffrontend.com%2F
        &state=0909ff6a-53b2-4253-8690-aff72d2cfff1
        &response_mode=fragment
        &response_type=code
        &scope=openid%20profile
        &nonce=67bad316-d8c1-45d1-9559-2a1c4726ce91
```

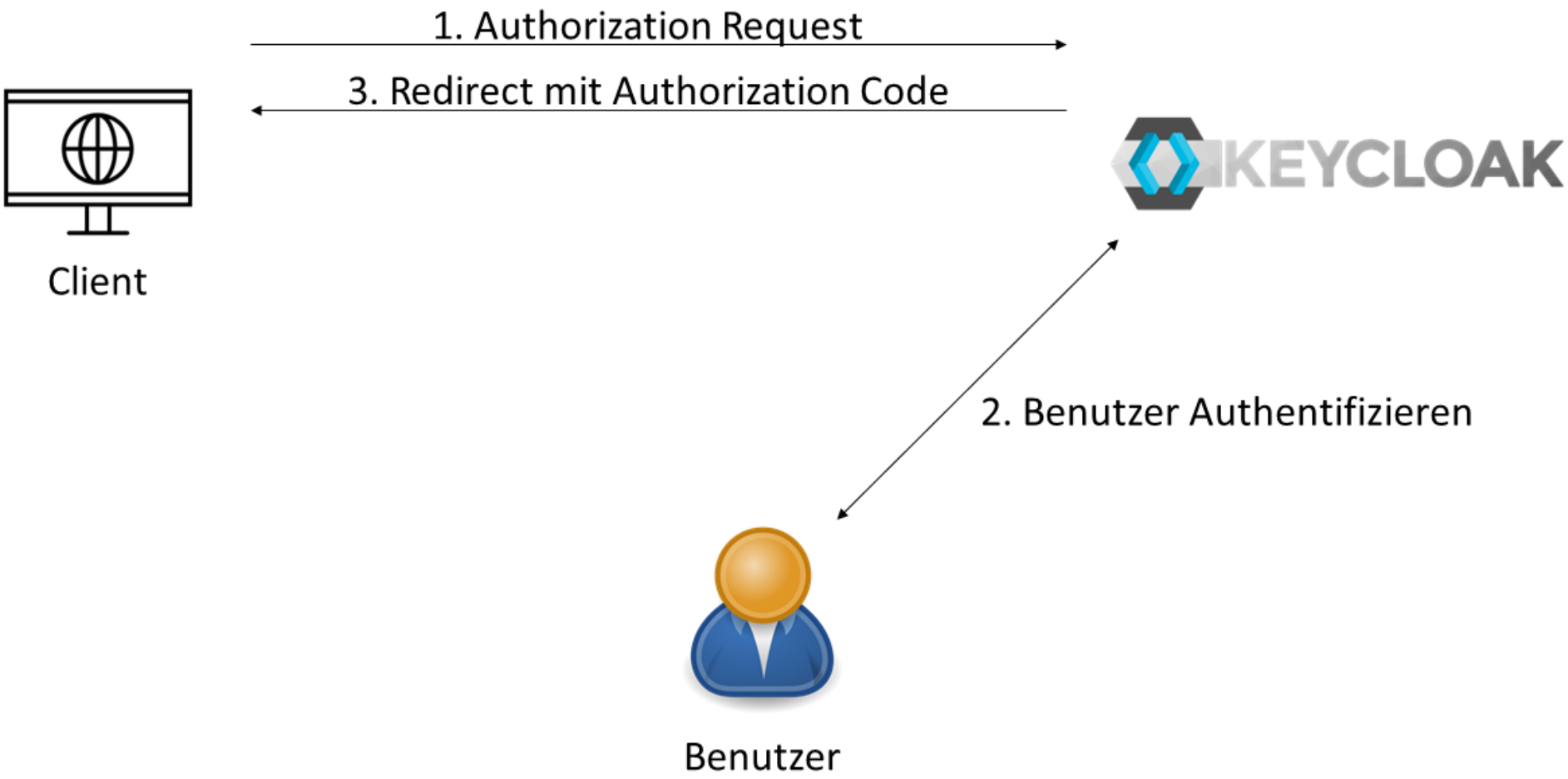
Listing 1: Beispiel Authorization Request

Authorization Code Flow Schritt 2



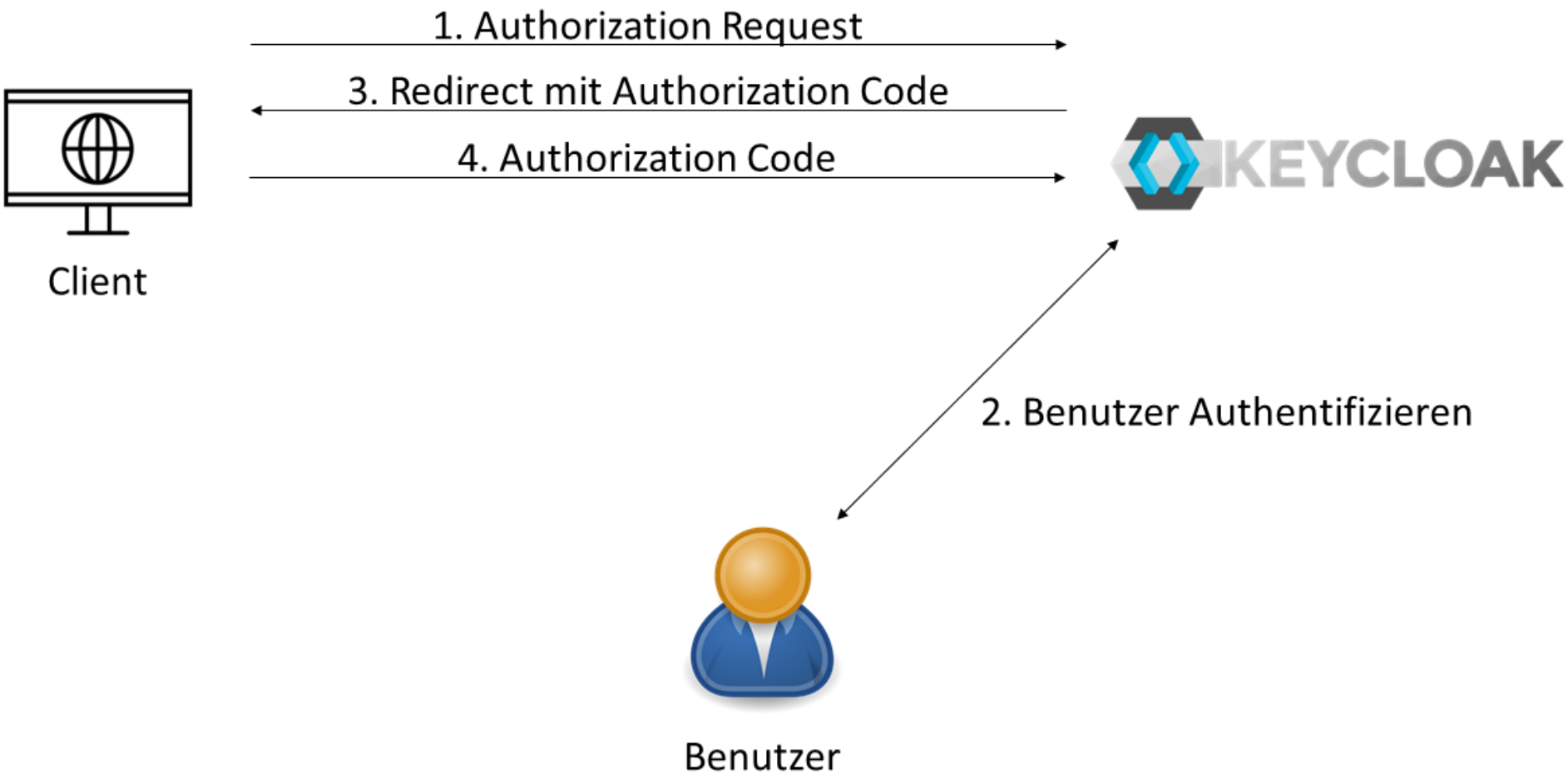
Quelle: [3], [4], [5], [N S14]

Authorization Code Flow Schritt 3



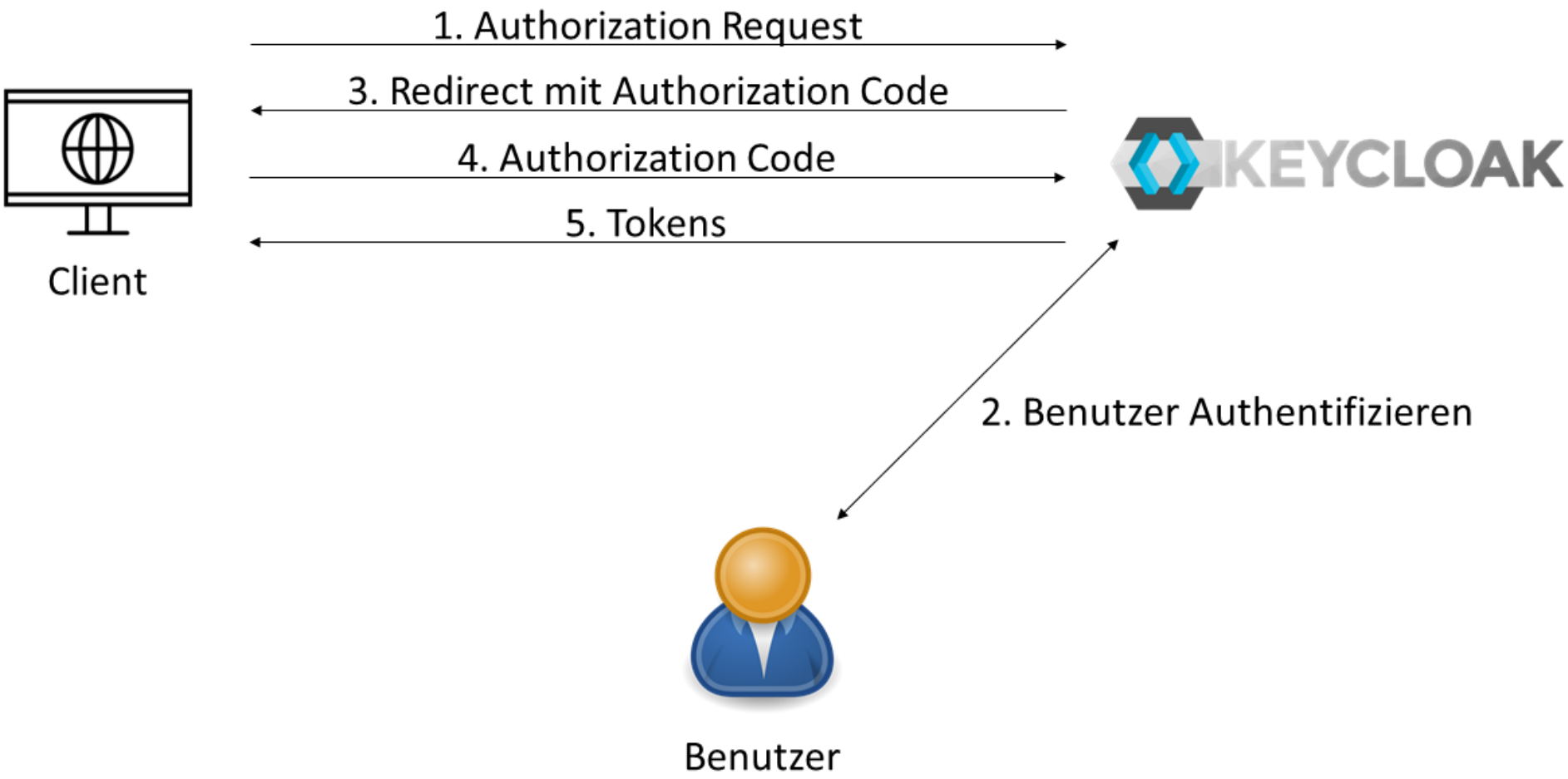
Quelle: [3], [4], [5], [N S14]

Authorization Code Flow Schritt 4



Quelle: [3], [4], [5], [N S14]

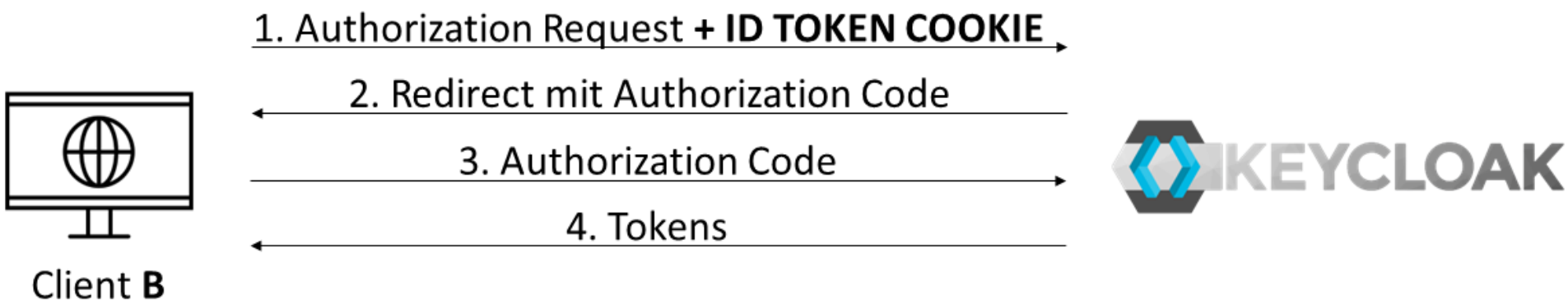
Authorization Code Flow Schritt 5



Quelle: [3], [4], [5], [N S14]

Zweite Authentifizierung

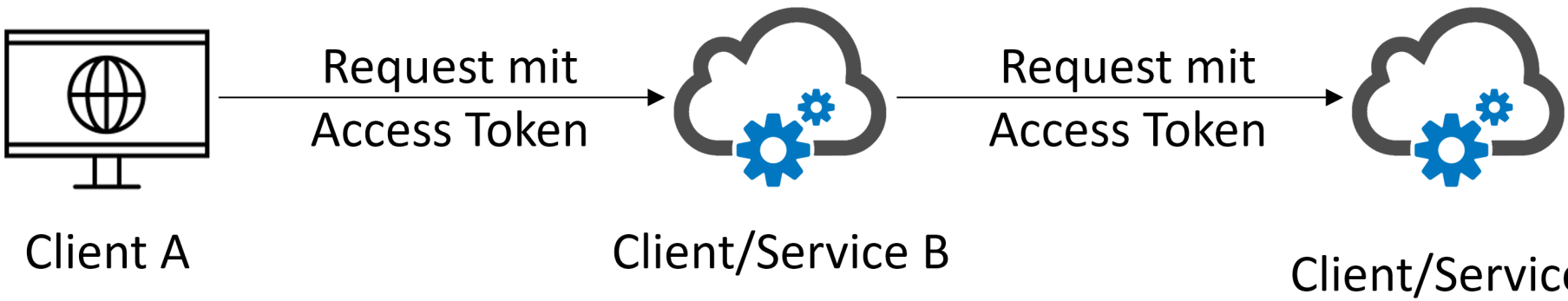
- Nach erster erfolgreicher Authentifizierung wird Cookie mit ID Token für Keycloaks Authorization Endpunkt gesetzt



Quelle: [3], [4], [5], [N S14], [Jan16]

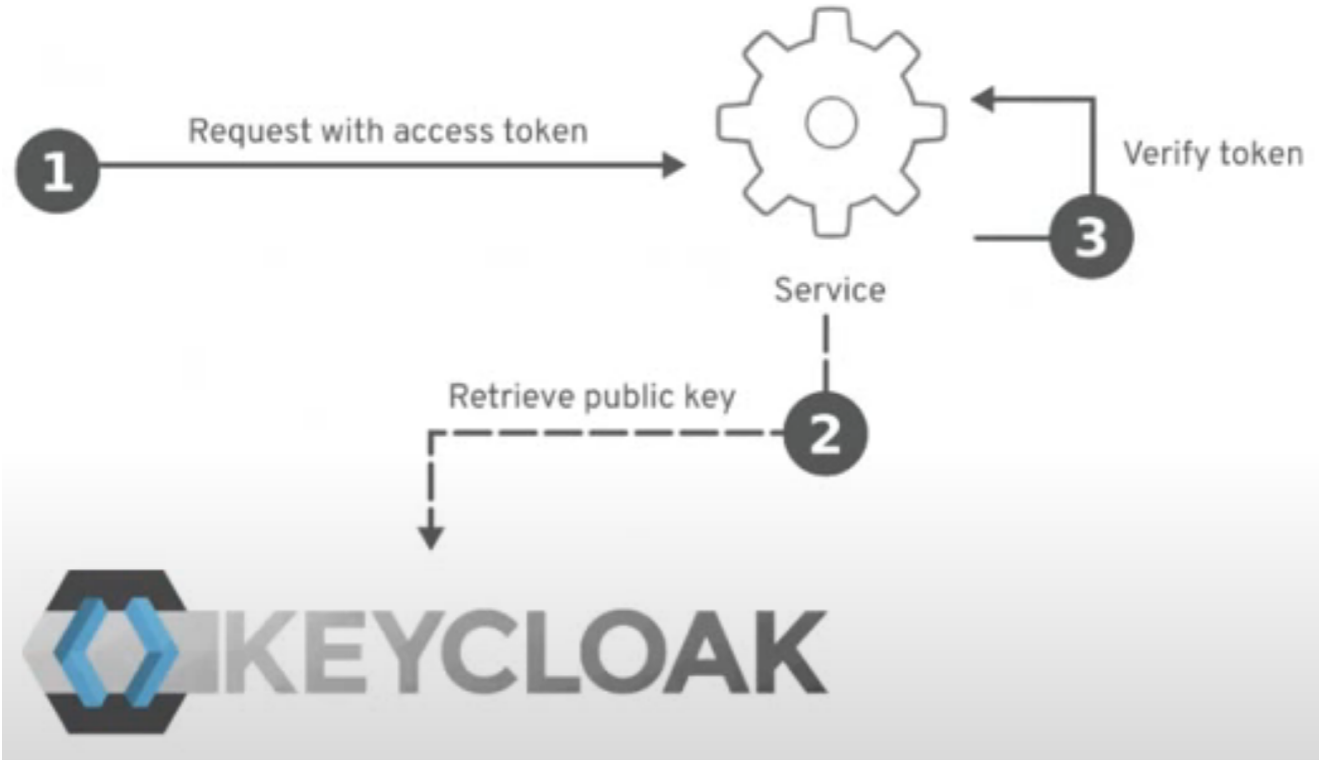
Zugriff auf geschützte Ressourcen

- Access Token im Authorization Header der HTTP Anfrage
- Jeder kann Access Token einsetzen
- Einschränken der abrufbaren Ressourcen und Benutzer Claims durch 'scope' und Audience Claims des Access Tokens
- Empfänger muss 'scope' und Audience Claims überprüfen



Quelle: [4], [5], [Har12], [JH12]

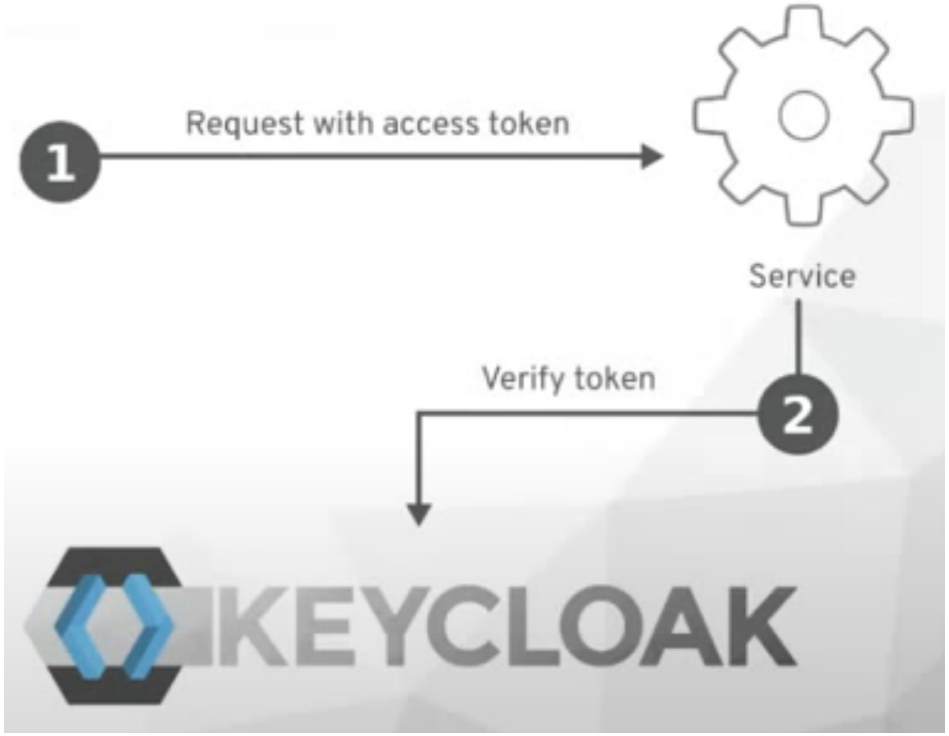
Validieren des Access Tokens - Offline Variante



- Public Key gecached
- Verringerte Latenz
- Verringerte Auslastung des Keycloak Servers
- Access Token kann nicht widerrufen und invalidiert werden

Quelle: [13], [JBS15]

Validieren des Access Tokens - Online Variante



- Höhere Latenz
- Höhere Auslastung des Keycloak Servers
- Access Token kann widerrufen und invalidiert werden

Quelle: [13], [JBS15]

Threat Model

- Client Schwachstellen
- Authorization Server Schwachstellen
- Token Schwachstellen

Client Schwachstellen

- Cross-Site-Request-Forgery (CSRF)
  - z.B. Angreifer sendet gefälschte Authentifizierungsantwort an Clients redirect URI
  - State Token (Anti-CSRF Token)
- Registrierung der redirect_uri
  - Angreifer gibt sich als Client aus
  - Möglich, wenn redirect_uri zu unspezifisch
- Diebstahl von Tokens
  - z.B. durch XSS
- OIDC frei von Kryptographie
  - Verwendung von TLS voraussetzen, z.B. mit HSTS

Quelle: [RS17], [Red20b], [Har12]

Authorization Server Schwachstellen

- Redirection URI manipulieren
  - Authorization Code zu URI des Angreifers umleiten
- Session Hijacking
  - z.B. Authorization Code aus Browser-Verlauf auslesen
  - Deshalb darf Authorization Code nur einmal eingelöst werden

Quelle: [Har12], [Red20b]

Token Schwachstellen

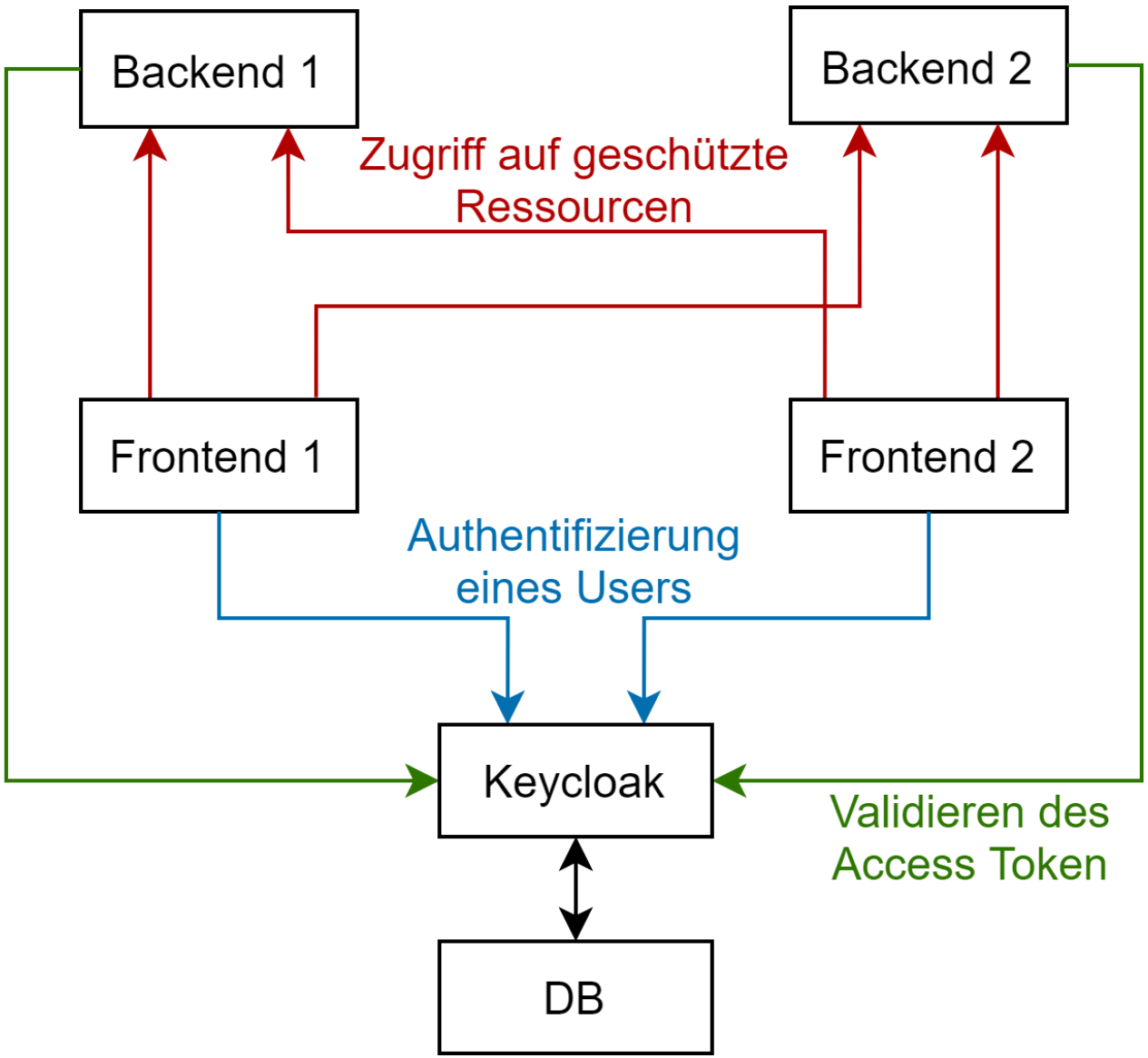
- Token Modification
  - Empfänger von Access Token muss digitale Signatur überprüfen
- Token Replay
  - Gute Verschlüsselung, geringe Token Verfallszeit
- Token Redirect
  - Empfänger ID (Audience Claim) im Token
- Token Disclosure
  - Token selbst könnte sensible Informationen enthalten

Quelle: [JH12], [Har12], [Red20b]

Notiz an Herrn Professor Karg

- Notiz an Herrn Professor Karg: Abschließend folgt noch eine Demo der implementierten Beispielwebanwendung in der VM. Vom Inhalt wird diese Demo ungefähr so sein wie das Kapitel 1.4 "Umsetzung von SSO innerhalb einer Beispielwebanwendung" unserer Ausarbeitung.

Überblick der Beispielwebanwendung



Zusammenfassung

- Benutzer durchläuft ein Authentifizierungsverfahren
- Danach automatische Authentifizierung durch SSO-Mechanismus
- Drei Rollen bei OpenID Connect:
  - Benutzer: Menschlicher Teilnehmer
  - Client: z.B. Webanwendungen oder Ressourcenserver, die am SSO teilnehmen
  - OpenID Provider: Authentifiziert Benutzer, stellt Endpunkte bereit
- Drei Arten von Tokens bei OpenID Connect:
  - ID Token: Informationen über Authentifizierung des Benutzers
  - Access Token: Schlüssel für Ressourcen und Benutzerinformationen
  - Refresh Token: Anfordern neuer Tokens

Quellen Teil 1

Margaret Rouse (2020)

single sign-on (SSO)

url: https://searchsecurity.techtarget.com/definition/single-sign-on (besucht am 01.12.2020)

Tayibia Bazaz et al. (2016)

A Review on Single Sign on Enabling Technologies and Protocols

International Journal of Computer Applications 151, p. 18-25

Mattia Panzeri et al. (2020)

@react-keycloak/web

url: https://www.npmjs.com/package/@react-keycloak/web (besucht am 29.11.2020)

Redhat. Inc (2020)

Server Administration Guide - Red Hat Single Sign-On 7.0.

url: https://access.redhat.com/documentation/en-us/red_hat_single_sign-on/7.0/html/server_administration_guide/index (besucht am 25.11.2020)

Quellen Teil 2

Vedala Radha und D. Reddy (2012)

A Survey on Single Sign-On Techniques

Procedia Technology 4, S. 134-139

Redhat. Inc (2020)

Server Administration Guide

url: https://www.keycloak.org/docs/latest/server_admin/index.html (besucht am 28. 11. 2020)

N. Sakimura et al. (2014)

OpenID Connect Core 1.0.

url: https://openid.net/specs/openid-connect-core-1_0.html

D. Hardt (2012)

The OAuth 2.0 Authorization Framework

url: https://tools.ietf.org/html/rfc6749

Quellen Teil 3

Justin Richer und Antonio Sanso (2017)

OAuth 2 in Action

1. Aufl. Shelter Island, NY 11964: Manning Publications

M. Jones, J. Bradley und N. Sakimura (2015)

JSON Web Token (JWT)

RFC 7519. RFC Editor, S. 1-30 *url:* https://www.rfc-editor.org/rfc/rfc7519.txt

IdentityServer (2020)

IdentityServer3 - Terminology

url: https://identityserver.github.io/Documentation/docsv2/overview/terminology.html (besucht am 29.11.2020)

Jannik Hüls (2016)

Single Sign-On mit Keycloak als OpenID Connect Provider

url: https://blog.codecentric.de/2016/08/single-sign-mit-keycloak-als-openid-connect-provider/ (besucht am 29.11.2020)

Quellen Teil 4

M. Jones und D. Hardt (2012)

The OAuth 2.0 Authorization Framework: Bearer Token Usage

RFC 6750. RFC Editor, S. 1–18 *url:* https://tools.ietf.org/html/rfc6750

Amin Saqi (2019)

A Survey on SSO Authentication Protocols: Security and Performance

url: https://medium.com/@aminsaqi/a-survey-on-sso-authentication-protocols-security-and-performance-287dcb6 (besucht am 29.11.2020)

Quellen (Abbildungen) Teil 5

1. https://www.renovodata.com/blog/2019/01/17/single-sign-on (besucht am 23.12.2020)
2. https://insready.com/en/blog/single-sign-using-oauth2-and-jwt-distributed-architecture von Jingsheng Wang (besucht am 17.12.2020)
3. https://commons.wikimedia.org/wiki/File:User_icon_2.svg (besucht am 23.12.2020)
4. https://thenounproject.com/term/web-application/3085910/ by Justin Blake from Noun Project (besucht am 23.12.2020)
5. https://icon-library.com/icon/web-service-icon-9.html (besucht am 1.01.2021)
6. https://design.jboss.org/keycloak/logo/images/keycloak_logo_600px.png von Keycloak (besucht am 1.01.2021)
7. https://upload.wikimedia.org/wikipedia/commons/5/59/OneDrive_Folder_Icon.svg (besucht am 2.01.2021)

Quellen (Abbildungen) Teil 6

8 https://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs von Tino Tkalec (besucht am 2.01.2021)

9 https://iconarchive.com/show/job-seeker-icons-by-inipagi/id-card-icon.html von Inipagi Studio (besucht am 2.01.2021)

10 https://de.cleanpng.com/png-jry6v5/ (besucht am 2.01.2021)

11 https://thenounproject.com/search/?q=resource&i=2979093 resource by Hrbon from the Noun Project (besucht am 2.01.2021)

12 https://commons.wikimedia.org/wiki/File:Refresh_icon.svg (besucht am 2.01.2021)

13 https://www.youtube.com/watch?v=mdZauKsMDiI von Stian Thorgersen (besucht am 2.01.2021)

Quellen (Abbildungen) Teil 7

14 https://thenounproject.com/term/domino-effect/2020716/ by Oleksandr Panasovskyi from the Noun Project (besucht am 2.01.2021)

15 https://www.freepik.com/free-vector/illustration-jigsaw-icon_2606569.htm Designed by rawpixel.com / Freepik (besucht am 3.01.2021)

16 https://thenounproject.com/pentool.knight/uploads/?i=1393568 by Mohamad Arif Prasetyo from the Noun Project (besucht am 3.01.2021)

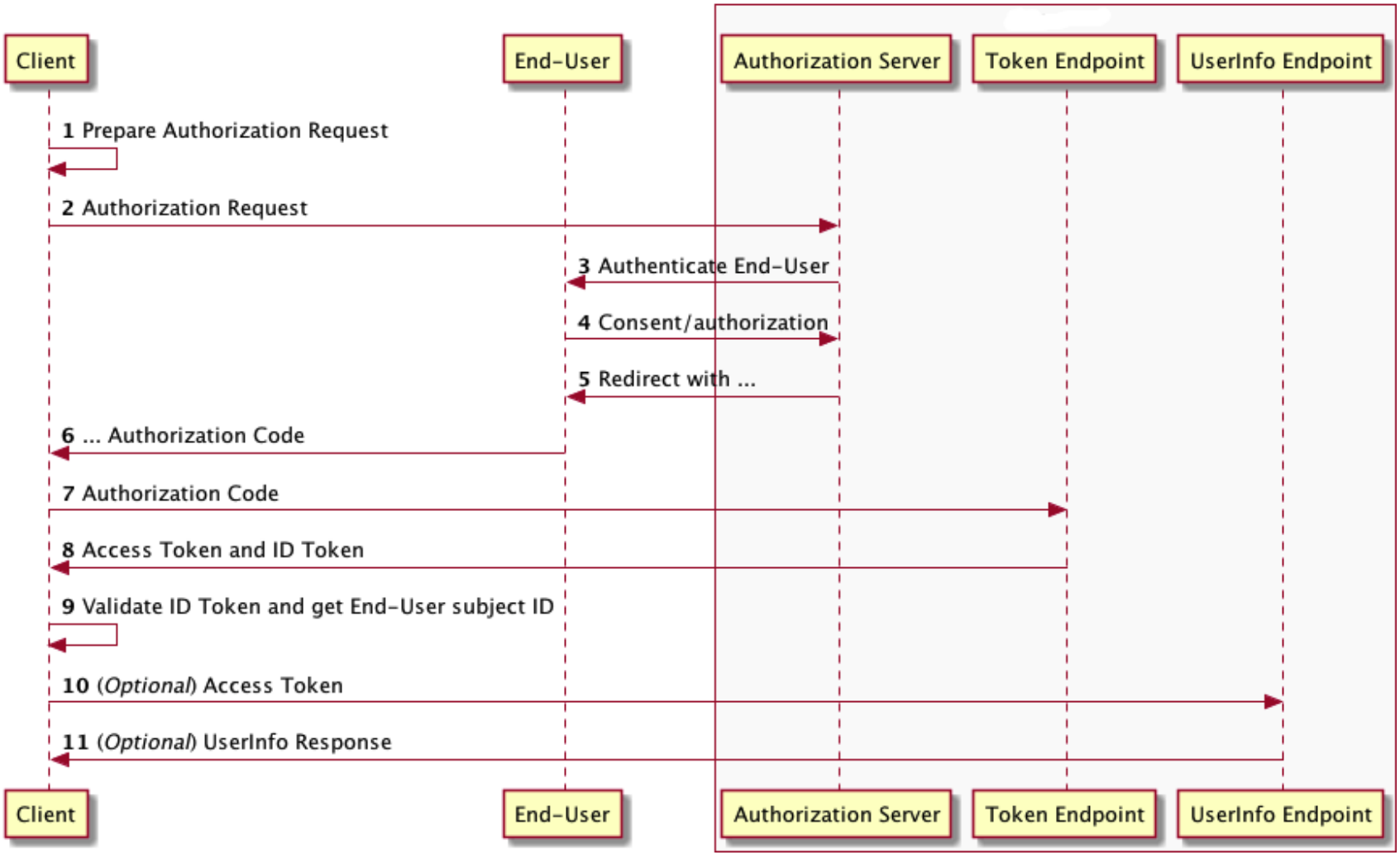
17 https://developers.redhat.com/blog/wp-content/uploads/2019/11/keycloak3-1024x576.png von Abhishek Koserwal (besucht am 3.01.2021)

Danke & Ihre Fragen

Notiz an Herrn Professor Karg

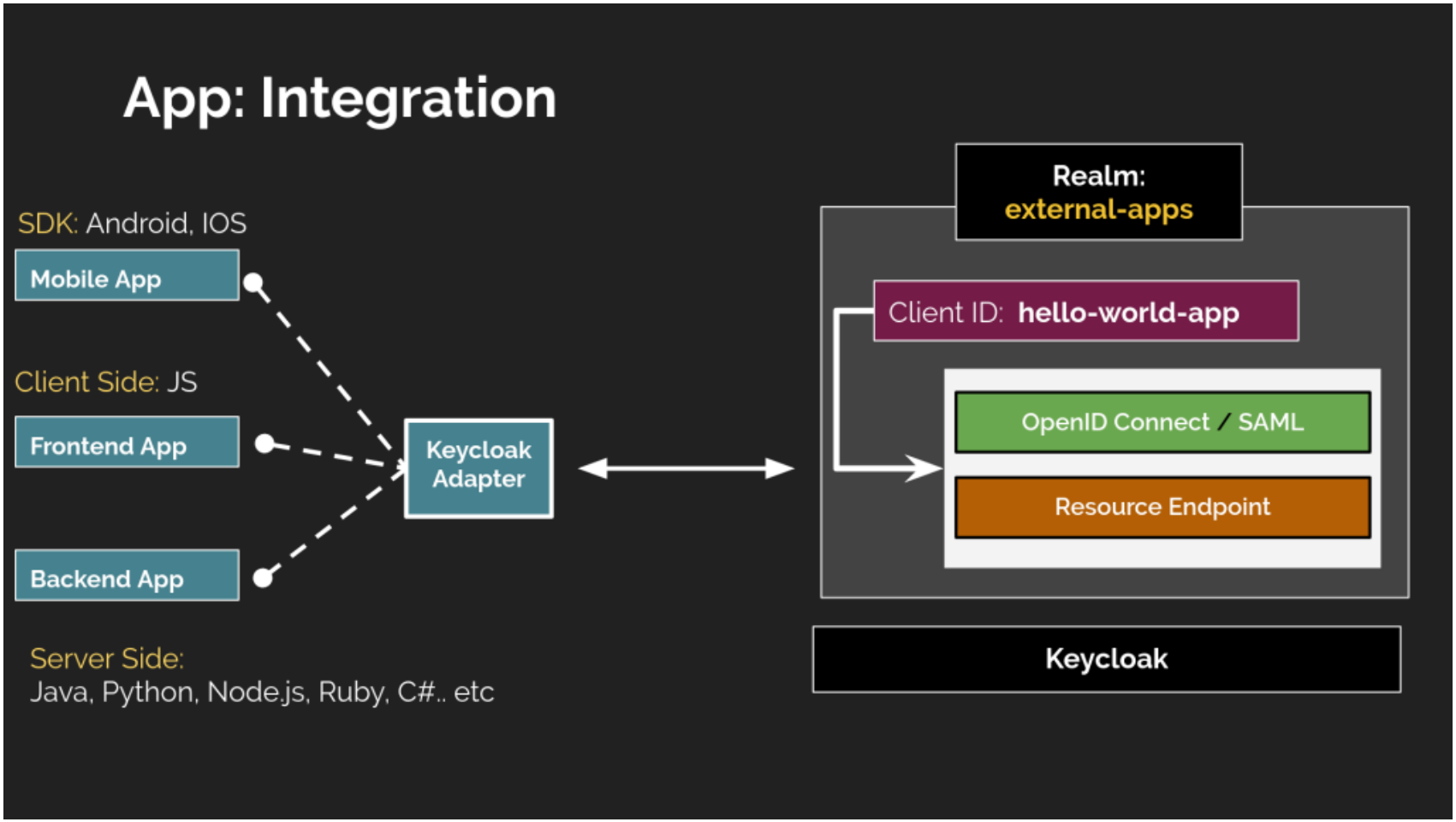
- Notiz an Herrn Professor Karg: Die folgenden Seiten sind nicht Teil der normalen Präsentation, sondern werden nur für die Beantwortung von möglichen und passenden Fragen von den Zuhörern verwendet.

Extra - Authorization Code Flow im Detail



Quelle: [14]

Extra - Client Adapter



Quelle: [17]