

Service-Portal Architektur

schooltech.ch

February 2, 2026

1 Zielsetzung

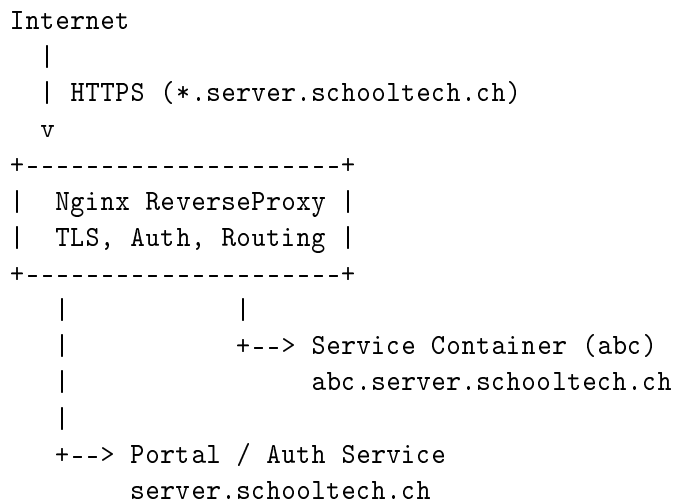
Ziel ist der Aufbau eines zentralen Login- und Navigations-Portals für mehrere unabhängige Web-Services, die in Docker-Containern betrieben werden. Benutzer authentifizieren sich einmal zentral und können anschließend zwischen Services wechseln, die jeweils unter eigenen Subdomains bereitgestellt werden.

2 Architekturübersicht (final)

2.1 Domänenstruktur

- `server.schooltech.ch` – Login- und Navigations-Portal (SPA)
- `<service>.server.schooltech.ch` – einzelne Services (Docker Container)

2.2 High-Level Architektur



2.3 Zentrale Prinzipien

- Ein Einstiegspunkt (Nginx)
- Zentrale Authentifikation (`auth_request`)
- Services kennen kein Login
- Keine iFrames, kein Subpath-Rewrite
- Services laufen auf Root-Pfad (`/`)

3 Authentifikation (Variante A: auth_request)

3.1 Prinzip

Nginx prüft jeden Request zu einem Service mittels `auth_request` gegen den Auth-Service. Der Auth-Service entscheidet ausschließlich anhand der Session (Cookie).

3.2 Ablauf

1. Benutzer loggt sich am Portal ein
2. Portal setzt Session-Cookie:

`Domain=.server.schooltech.ch`
`Secure; HttpOnly; SameSite=None`
3. Benutzer ruft Service-Subdomain auf
4. Nginx fragt `/internal/auth` beim Auth-Service an
5. Bei Erfolg: Request wird an Service weitergeleitet

3.3 Header-Weitergabe

Optional injiziert Nginx folgende Header:

- X-Remote-User
- X-User-Roles

4 Nginx Konfiguration

4.1 Wildcard Server Block

```
server {
    listen 443 ssl;
    server_name *.server.schooltech.ch;

    ssl_certificate      /etc/letsencrypt/live/server.schooltech.ch/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/server.schooltech.ch/privkey.pem;

    location / {
        auth_request /internal/auth;
        proxy_pass http://$upstream;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location = /internal/auth {
        proxy_pass http://auth:3000/internal/auth;
        proxy_pass_request_body off;
        proxy_set_header Content-Length "";
        proxy_set_header X-Original-URI $request_uri;
    }
}
```

4.2 Upstream Mapping

```
map $host $upstream {
    default          portal:3000;
    abc.server.schooltech.ch abc:8080;
    xyz.server.schooltech.ch xyz:8080;
}
```

5 Docker Compose (Übersicht)

```
version: "3.9"
```

```
services:
```

```
  nginx:
    image: nginx:alpine
    ports:
      - "443:443"
    volumes:
      - ./nginx:/etc/nginx/conf.d
      - ./certs:/etc/letsencrypt
    depends_on:
      - auth
      - portal
    networks:
      - internal
```

```
  auth:
    image: node:18
    command: node auth.js
    networks:
      - internal
```

```
  portal:
    image: node:18
    command: node portal.js
    networks:
      - internal
```

```
  abc:
    image: service-abc
    networks:
      - internal
```

```
networks:
  internal:
    driver: bridge
```

6 UI-Konzept (Navigations-Portal)

6.1 Funktion

- Login

- Anzeige verfügbarer Services
- Wechsel per Full Navigation
- Kollabierbare Kopfzeile
- Floating Launcher (Bookmark)

6.2 UI-Layout

+	-----	+
	LOGO Service A Service B User	
+	-----	+
	Service Overview / Last Service / Status	
+	-----	+

6.3 HTML-Prototyp (Portal)

```

<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8" />
  <title>Service Portal</title>
  <style>
    header {
      position: fixed;
      top: 0; left: 0; right: 0;
      height: 60px;
      backdrop-filter: blur(6px);
      background: rgba(0,0,0,0.3);
      color: white;
      display: flex;
      align-items: center;
      padding: 0 20px;
    }
    main {
      padding-top: 80px;
    }
    .service {
      display: inline-block;
      margin: 10px;
      padding: 20px;
      border: 1px solid #ccc;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <header>
    <strong>Portal</strong>
  </header>

```

```

<main>
  <div class="service" onclick="go('abc')">Service ABC</div>
  <div class="service" onclick="go('xyz')">Service XYZ</div>
</main>

<script>
  function go(name) {
    localStorage.setItem("lastService", name);
    window.location.href = "https://" + name + ".server.schooltech.ch";
  }
</script>
</body>
</html>

```

6.4 Portal als Docker Container

Das Portal wird als eigener Container betrieben und statisch oder via Node.js ausgeliefert. Es besitzt keine direkte Kopplung zu den Services außer über URLs.

7 Zusammenfassung

Diese Architektur ermöglicht:

- saubere Trennung von Portal und Services
- zentrale Sicherheit
- einfache Erweiterbarkeit
- wartbare Reverse-Proxy-Konfiguration