Architektur-Überblick

Diese Übersicht beschreibt die Architektur von D-DOK. Code-Stand: Version **0.0.8** (Branch `master`). Quelle: **git.agrarforschung.at/ddok/ddok** (https://git.agrarforschung.at/ddok/ddok).

Architektur-Diagramm

```
flowchart LR
  subgraph Client["Browser-Client"]
    NG["Angular 21 SPA
(client/)"]
    PDFX["PDF-Volltext-Extraktion
im Browser (pdfparse)"]
    NG --- PDFX
  end

  subgraph Server["D-DOK Server (Node.js 22)"]
    API["Express 5 + Awilix (DI)
TypeScript, server/"]
  end

  subgraph SQL["MySQL: eine gemeinsame Datenbank (D-DOK + D-ESS)"]
    MY[("ddok_*-Tabellen
Projekte, Publikationen
Volltext-Index (FULLTEXT)")]
    DESSDB[("D-ESS tbl_*-Tabellen
Kostenträger, Benutzer:innen")]
  end

  MO[("MongoDB 7.0, GridFS
PDF-Dateien (Bucket je Projekt)")]

  subgraph Auth["Authentifizierung"]
    AD["Active Directory (LDAPS)"]
    KC["Keycloak OIDC
sso.agrarforschung.at"]
  end

  subgraph Web["Öffentliche BAB-Website (optionaler Publish)"]
    WEBMY[("MySQL WEB_DATABASE")]
    WEBMO[("MongoDB WEB_MONGO")]
    SITE["Public-Frontend"]
  end

  NG -- "HTTPS / REST" --> API
  PDFX -- "pdfText im Upload" --> API
  API -- "Metadaten + Volltext" --> MY
  API -- "Lookup (gleiche Verbindung)" --> DESSDB
  API -- "PDF-Blob" --> MO
  API -. OIDC .-> KC
  API -. LDAPS .-> AD
  API -- Publish --> WEBMY
  API -- Publish --> WEBMO
  WEBMY --> SITE
  WEBMO --> SITE
```

Komponenten im Überblick

KOMPONENTE	STACK	QUELLE
Frontend (SPA)	Angular 21 · Angular Material · Wijmo (@mescius) · Quill (ngx-quill). Upgrade auf Angular 21 in 0.0.8 (zuvor Angular 20).	`client/`
Backend-API	Node.js 22 · Express 5 · **Awilix** / awilix-express (Dependency Injection) · TypeScript · TypeORM	`server/`
PDF-Volltext-Extraktion	Läuft **im Browser** (Bibliothek `pdfparse`, Git-Submodul `modules/pdfparse`); der extrahierte Text wird beim Upload an den Server mitgesendet – kein serverseitiger Parser-Worker.	`client/` (file-upload), `modules/pdfparse/`

Relationale DB	MySQL (Port 3306) · `ddok_*`-Tabellen: Projekte, Publikationen, Stammdaten + **Volltext-Index (MySQL-FULLTEXT auf `file_text`)** · TypeORM mit `synchronize:true`	Gemeinsame MySQL-DB mit D-ESS
Dokumenten-DB	MongoDB 7.0 (Port 27017) · PDF-Dateien als **GridFS**, ein Bucket pro Projekt (`PROJECT_<id>`). Kein Volltext-Index – der liegt in MySQL.	Lokale MongoDB
D-ESS-Anbindung	Lese-Zugriff auf die D-ESS-Tabellen (`tbl_*`) über **dieselbe MySQL-Verbindung**: Kostenträger/KS/KT + Benutzer:innen-/Berechtigungsdaten	`DESS_LINK_ENABLED` (Default `true`)
Authentifizierung	Lokal · Active Directory (LDAPS, `passport-ldapauth`) · Keycloak OIDC (`openid-client`, Realm `ddok` auf sso.agrarforschung.at)	`AD_ENABLED` / `OIDC_ENABLED` in `server/config/config.js`
Website-Publish (optional)	Manueller Publish-Vorgang: ausgewählte Projekte/Publikationen samt Dateien werden in separate Website-Datenbanken kopiert (eigener Publish-Worker)	`WEB_DATABASE_*` + `WEB_MONGO_DATABASE_*`

Deployment

D-DOK läuft in Produktion **nicht als Container**. Das im Repository vorhandene `Dockerfile` ist ausschließlich eine reproduzierbare **Build-Umgebung** für die CI – die Laufzeit ist ein klassischer **Plesk-/Passenger-Node.js-Prozess**.

ASPEKT	DETAILS
Laufzeit-Host	Plesk-Server mit Phusion Passenger (LXC-Container). Geteilt mit D-ESS, Datenpool und Adressdatenbank (Co-Tenancy). Node 22 via `nodenv`.
App-Pfade	Live: `/var/www/vhosts/agrarforschung.at/ddok.agrarforschung.at` Staging: `/var/www/vhosts/agrarforschung.at/staging-ddok.agrarforschung.at`
SSO	Keycloak-Realm `ddok` auf `sso.agrarforschung.at`

Build – `npm run build`

Führt drei Workspace-Builds nacheinander aus (`package.json`):

- `build:pdfparse` – baut das PDF-Volltext-Modul (`modules/pdfparse/`).
- `build:client` – Angular-Produktions-Build (`client/`).
- `build:server` – bündelt den TypeScript-Server via **esbuild** (`node esbuild.mjs`) und erzeugt das Laufzeit-Lockfile (`npm --prefix ./dist install --package-lock-only`).

Resultat ist der Ordner `dist/` – das eigentliche Deployment-Artefakt.

Rolle des `Dockerfile`

Multi-Stage-Build: Die *builder*-Stage führt `npm ci` + `npm run build` aus und erzeugt `dist/`; die finale Stage übernimmt nur `dist/`, exponiert Port 3000 und definiert `CMD ["npm","run","start"]`. In der CI wird dieses Image jedoch **nicht als Runtime ausgerollt**, sondern dient als hermetische Build-Box: Nach dem Build wird ein Wegwerf-Container erzeugt und dessen `/usr/src/app` (= `dist/`) per `docker cp` als CI-Artefakt herauskopiert.

CI/CD – `.gitlab-ci.yml`

Drei Stages, ausschließlich auf Branch `master`:

1. **build** – Docker-Image bauen (Docker-in-Docker, BuildKit-Registry-Cache), nach **Harbor** pushen und `dist/` als Artefakt extrahieren.
2. **deploy:staging** (automatisch) – bestehendes Verzeichnis nach `backup/` sichern (außer `backup`/`log`/`config`/`config.js`), `dist/*` per `scp` auf den Staging-Host kopieren, `.npmrc` schreiben, `npm ci` (nodenv Node 22), `touch tmp/restart.txt` → Passenger-Reload.
3. **deploy:live** (manueller Trigger) – identischer Ablauf auf den Live-Host.

Das Deployment ist somit ein **scp-basiertes Datei-Rollout des `dist/`-Ordners** mit anschließendem Passenger-Restart – kein `docker run` und kein Image-Pull auf dem Zielhost.

Datenflüsse

1. **Projekt-/Publikationsanlage:** Benutzer:innen erstellen Projekte und Publikationen im Angular-Frontend; die Metadaten werden über die Express-API in MySQL persistiert.
2. **PDF-Upload:** Der Browser extrahiert beim Hochladen den PDF-Volltext (Bibliothek `pdfparse`). Die Datei wird als GridFS-Blob in MongoDB abgelegt, der extrahierte Text in der MySQL-Spalte `file_text` (FULLTEXT-indiziert für die Suche).
3. **D-ESS-Lookup:** Kostenstellen, Kostenträger und Benutzer:innen-Stammdaten werden zur Laufzeit aus den D-ESS-Tabellen (`tbl_*`) in **derselben** MySQL-Datenbank gelesen (`DESS_LINK_ENABLED`).
4. **Website-Publish (optional):** Über einen manuellen Publish-Vorgang werden freigegebene Projekte/Publikationen samt Dateien in die separaten Website-Datenbanken (`WEB_*`) kopiert und auf der öffentlichen BAB-Website angezeigt.

Letzte Aktualisierung: 2026-06-10 · Pflege: Roland Neissl · Quelle: git.agrarforschung.at/ddok/ddok (https://git.agrarforschung.at/ddok/ddok)

Version #5
★Erstellt: 2026-05-23 15:39:11 CEST von roland.neissl
✎Zuletzt aktualisiert: 2026-06-10 14:59:11 CEST von roland.neissl