Anlage 1: Technische Bestandsbeschreibung und aktuelle Systemarchitektur

Im Folgenden wird der technische Kontext des StandortTOOLs grob umrissen. Nachdem der Projektzuschlag erteilt ist, wird selbstverständlich der Zugang zu allen Dokumentationsseiten gewährt.

Frontend-Technologien

Das Frontend nutzt folgende Technologien und Frameworks:

- **TypeScript** - Typsichere Entwicklung der fachlichen Logik, Datenmodelle und API-Anbindungen im Web-Frontend.
- **Vue 3** - Framework für die komponentenbasierte Umsetzung der Benutzeroberfläche und der interaktiven Webanwendung.
- **Vite 5** - Build- und Entwicklungswerkzeug für lokale Entwicklung, optimierte Produktions-Builds und unterschiedliche Zielumgebungen.
- **MapLibre GL 4** - Darstellung und Interaktion mit der zentralen Webkarte, einschließlich Kartenebenen, Markern und geografischen Daten.
- **ECharts 5** - Visualisierung von Kennzahlen und Daten in Diagrammen und Dashboards.
- **Cypress 13** - Automatisierte End-to-End-Tests der wesentlichen Nutzerprozesse im Frontend.

Für die Anpassung des Keycloak-Themes wird die standardmäßige Keycloak-Theming-Engine (Freemaker) zusammen mit Tailwind verwendet, wobei SCSS zum Einsatz kommt.

Es wird kein externes Component-Framework eingesetzt. Stattdessen wird ein eigenes Design-System auf Basis von Tailwind CSS 3 mit definierten Design-Tokens verwendet.

Die UX/UI-Vorgaben sind in **Figma** dokumentiert und basieren auf den internen Corporate Design Vorgaben der NOW GmbH. Neue Features müssen sich konsistent in das bestehende Design integrieren.

Als nutzerfreundliches und flexibles Content Management System (CMS) wird **Strapi** genutzt.

Backend-Technologie

Das Frontend nutzt folgende Technologien und Frameworks:

- **Java 21** - Zentrale Programmiersprache für die Backend-Services und die Umsetzung der fachlichen Geschäftslogik.
- **Spring Boot 3.x** - Basisframework für die Entwicklung und den Betrieb der unabhängig deploybaren Backend-Services.
- **Spring Cloud** - Unterstützung verteilter Systemkomponenten sowie der Anbindung und Konfiguration von Cloud-Services.
- **Spring Data** - Abstraktion des Datenzugriffs und Anbindung der PostgreSQL-Datenbanken.
- **Spring Security** - Absicherung der Backend-Endpunkte sowie Integration der Authentifizierungs- und Autorisierungsmechanismen.
- **Keycloak / OAuth 2.0 / OpenID Connect** - Zentrale Nutzeranmeldung, Rollenverwaltung und Zugriffskontrolle.
- **OpenAPI 3 / Swagger** - Beschreibung, Dokumentation und testbare Bereitstellung der REST-Schnittstellen.
- **PostgreSQL** - Persistente Speicherung der Ladeinfrastruktur-, Nutzer-, Berechtigungs- und Onboarding-Daten.
- **PostgreSQL JDBC Driver 42.x** - Technische Datenbankanbindung der Java-Services an PostgreSQL.
- **Liquibase** - Versionierung und automatisierte Ausführung von Datenbankschemaänderungen.
- **Maven** - Verwaltung von Abhängigkeiten sowie automatisierter Build und Test der Backend-Anwendungen.
- **Docker** - Standardisierte Paketierung und reproduzierbarer Betrieb der einzelnen Anwendungen und Services.
- **AWS** - Hosting- und Betriebsplattform für Services, Datenbanken, Batch-Verarbeitung, Synchronisation und weitere Infrastrukturkomponenten.

Das System folgt einer **Microservices-Architektur** mit zwei unabhängig deploybaren Services.

Die Kommunikation zwischen Frontend und Backend erfolgt über eine **REST API** mit versionierten HTTP-Endpunkten. Die API ist vollständig mit OpenAPI/Swagger dokumentiert.

Das Backend implementiert eine **Kombination aus technischen Schichten und fachlichen Domänen**, dabei gibt es drei Schichten pro Service: *domain* → *infrastructure* → *application*

Innerhalb jeder Schicht findet eine Gruppierung nach Fachdomäne statt: lis/ (Ladeinfrastruktur) und user/ (Nutzerverwaltung, Onboarding, Netzbetreiber)

Datenbackend

Das Sytem basiert auf zwei zentralen Datenbanken: 1. LIS-Service Database und 2. User-Service Database.

Dabei stellt die **LIS-Service Datenbank** („LIS“ steht hier für Ladeinfrastruktur) alle inhaltlichen Daten zur Verfügung, die auf der Karte oder in den Dashboards des Tools angezeigt werden. Diese werden regelmäßig, einmal pro Woche, vom Datawarehouse der NOW GmbH in die Backend-Datenbank des StandortTOOL (AWS RDS PostgreSQL) synchronisiert.

Die **User-Service Datenbank** speichert alle nutzerbezogenen Daten des Tools — darunter Nutzerprofile, Registrierungen, Berechtigungen, Onboarding-Status, Netzbetreiber-Zugehörigkeiten sowie die Annahme von Nutzungsbedingungen. Sie ist eng mit dem Identity-Provider Keycloak verknüpft, der die Authentifizierung und Rollenverwaltung übernimmt.

Hosting/Infrastruktur

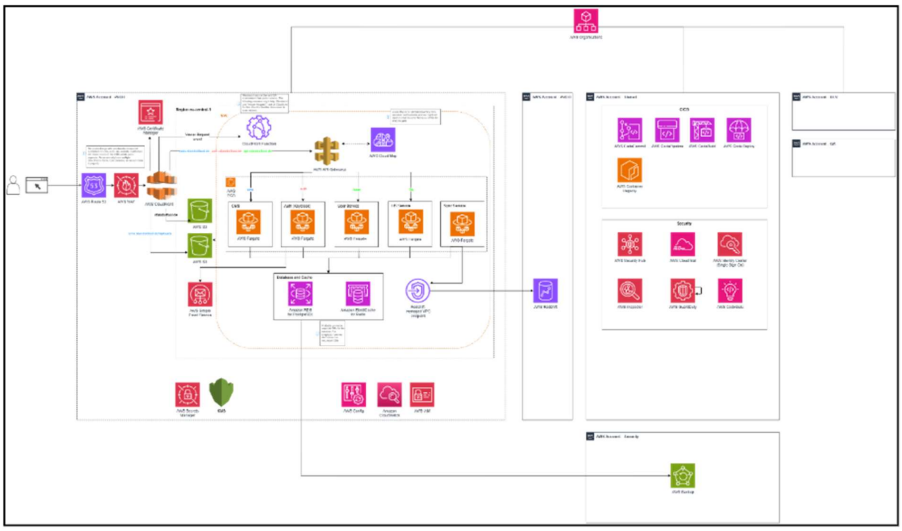
Das StandortTOOL wird hauptsächlich in der AWS-Cloud gehostet. Für die Cloud-Infrastruktur werden ausschließlich C5-zertifizierte Services eingesetzt, um die Einhaltung von Branchenstandards für Sicherheit und Performance zu gewährleisten. Dabei kann es zu einzelnen Ausnahmen dieser Regel kommen. Das Hosting muss in Europa erfolgen, bevorzugt in der Region Frankfurt am Main. Dabei sind Ausnahmen wie Metadaten-Services oder spezielle Services dieser Anforderung ausgenommen.

Soweit technisch machbar, werden Serverless-Services bevorzugt eingesetzt, um Skalierbarkeit zu verbessern, den operativen Overhead zu reduzieren und Kosten zu optimieren.

Die Cloud-Infrastruktur ist als Infrastructure-as-Code in **Terraform** definiert und dokumentiert.

Die folgende Abbildung zeigt das **Cloud-Architekturdiagramm** des Webtools.

Die fünf zentralen Anwendungen des Systems sind dabei als AWS Container Jobs mit Fargate angelegt (CMS, Auth (Keycloak)), User Service, LIS Service und Sync Service. Das Datenbackend wird durch eine Amazon RDS Postgre SQL Datenbank abgebildet, ergänzt durch ElastiCache for Redis.



Externe Schnittstellen

Webkarte

Für die interaktive Webkarte wird MapLibre GL JS (https://maplibre.org) Library genutzt.

Data Warehouse der NOW

Der größte Teil der im StandortTOOL angezeigten Informationen wird durch das Data Warehouse der NOW bereitgestellt. Das Data Warehouse besteht aus einer AWS Redshift Datenbank. Die Daten werden wöchentlich durch einen „Sync-Service“ in das Datenbackend des StandortTOOLs übertragen. Der Sync-Service wird durch einen AWS Schedule ausgelöst und basiert auf einer Lambda Funktion, die die relevanten Inputs an einen AWS Batch Job übergibt.

FlächenTOOL der NOW

Ausgewählte Informationen werden per API-Abfrage aus dem FlächenTOOL (https://flaechentool.de/) abgerufen.

MasterplanTOOL der NOW

Die Daten aus dem StandortTOOL Datenbackend speisen auch das MasterplanTOOL der NOW (https://masterplantool.de). Dies muss bei allen Änderungen an der Datenbank berücksichtigt werden.

Vorhandene technische Dokumentation

Die vorhandene technische Dokumentation liegt in englischer Sprache vor und umfasst folgende Themenbereiche:

Infrastruktur Konventionen, Prozesse und Workflows, Systemarchitektur und Infrastruktur, IAM, Backend und Data Layer, Frontend und CMS, Testing & QA, sowie einzelne domain-spezifische Prozesse.

Teststrategie

Wir setzen auf eine kontinuierliche, risikobasierte Qualitätssicherung mit **automatisierten** und **manuellen** Tests.

Automatisierte Unit-, Integrations- und bei Bedarf End-to-End-Tests sind in die CI/CD-Pipeline integriert und werden bei Builds, Pull Requests und Releases ausgeführt. Unit-Tests prüfen einzelne Komponenten isoliert, Integrationstests das Zusammenspiel von APIs, Datenbanken und externen Systemen.

Jira dient als Ticket- und Testmanagementsystem. Dort werden feste Release-Testfälle sowie Testergebnisse, Fehler, Reproduktionsschritte, Prioritäten und Retests nachvollziehbar dokumentiert.

Die Teststrategie setzt auf vollständig automatisierte Tests in der CI/CD-Pipeline, mit 1.079 Testfällen vorwiegend als Integrationstests (JUnit 5, Testcontainers, Spring Boot Test), die HTTP-Endpunkte end-to-end gegen eine vollständige Laufzeitumgebung testen.

Entwicklungs- und Deployment-Prozess

Alle Entwicklungen folgen einem standardisierten Git-Workflow mit Pull-Request-Review. Nach Genehmigung werden Änderungen über die AWS-CodePipeline mit automatisierten Build-, QA- und Production-Deployment-Schritten bereitgestellt. Manuelle Genehmigungen vor QA- und Production-Deployment sind erforderlich.

Für die Versionenverwaltung wird Git genutzt. Die Codebase ist aktuell in AWS CodeCommit mit entsprechenden AWS CodePipelines gehostet. Da CodeCommit als deprecated gilt, ist eine Migration auf das Unternehmens-GitHub inkl. der

Deployment pipelines erforderlich. Diese Migration steht aus. Alle weiteren Entwicklungs- und Wartungsarbeiten erfolgen anschließend auf GitHub.

Kommentiert [RL1]: Sollen wir das explizit als konkrete Aufgabe nennen? Oder einfach hier löschen und dann als eines der ersten Aufgaben übergeben?