IU Internationale Hochschule
Studiengang Informatik (M.Sc.)

PJ SkillTrack

Architekturdokument
im Fach
Projekt: Software Engineering (DLMCSPSE01_D)

Luca Witt (IU14101147)
am 19. September 2025

Tutor: Davin Kuhlen

Inhaltsverzeichnis

1 Technologieübersicht

1.1 Backend

1.1.1 Java & Gradle

PJ SkillTrack soll als Software as a Service (SaaS) im Internet betrieben werden. Demnach muss für die Entwicklung des zugrundeliegenden Webservers eine etablierte und zugängliche Technologie verwendet werden. Die Programmiersprache Java gehört seit Jahren zu den populärsten Programmiersprachen (“Technology | 2025 Stack Overflow Developer Survey”, n. d.) und zeichnet sich durch dessen Plattformunabhängigkeit aus, indem Java innerhalb der Java Virtual Machine (JVM) auf verschiedensten Betriebssystemen ausgeführt werden kann. Um Java-Code auslieferbar und ausführbar zu machen, werden sogenannte Build Tools eingesetzt. Diese nehmen den Entwickler:innen Aufgaben wie das Dependency Management, die Ausführung von Tests und den Bau der Java-JAR-Dateien selbst ab (mansisha8y75, 2025). Populäre Tools hierfür sind unter anderem Apache Maven und Gradle. Beide Tools stellen eine sinnvolle Wahl dar; für PJ SkillTrack wird aufgrund des bestehenden Know-Hows auf Gradle zurückgegriffen.

1.1.2 Spring-Framework

Das Spring Framework ist ein weit verbreitetes Java-Framework zur Entwicklung von Unternehmensanwendungen. Es bietet eine modulare Architektur und unterstützt vor allem die Entwicklung von Anwendungen nach dem Prinzip der Inversion of Control (IoC) und Dependency Injection (DI). Dadurch wird eine lose Kopplung zwischen Komponenten erreicht und die Wartbarkeit erhöht (“Why Spring”, n. d.; Wicksell et al., 2018).

1.1.3 Spring Boot

Spring Boot, eine Erweiterung des Spring Frameworks, vereinfacht die Entwicklung von Anwendungen erheblich, indem vorkonfigurierte Anwendungstemplates, eingebettete Webserver wie Apache Tomcat oder Auto-Konfigurationen den Overhead für die Einrichtung, Implementierung und Konfigurationen minimieren (“Spring Boot”, n. d.).

1.1.4 Spring Security

Ein vielgenutztes Modul ist Spring Security. Es stellt Mechanismen zur Authentifizierung und Autorisierung an Java- bzw. Spring-Anwendungen bereit, beispielsweise über Basic-Authentication, OAuth2 oder LDAP. Auch häufige Sicherheitslücken wie Cross Site Request Forgery (CSRF) können durch Spring Security abgedeckt werden (“Spring Security”, n. d.).

1.2 Persistenzschicht

1.2.1 PostgreSQL

Für die Datenpersistierung bietet sich eine relationale Datenbank an, welche die Abhängigkeiten zwischen den Entitäten wie *Skill* oder *Progress* optimal abbilden kann. Obwohl es für diese Technologie verschiedene Anbieter gibt, zeichnet sich besonders *PostgreSQL* aus. *PostgreSQL* ist ein open-source relationales Datenbank-System mit einer großen Community, welches sich durch Zuverlässigkeit und Performanz auszeichnet ("PostgreSQL: About", n. d.).

1.2.2 Spring Data & Flyway

Die Datenbankentitäten werden in der Businesslogik, dem Backend, durch das Spring-Modul *Spring Data* abgebildet. Spring Data ist ein Spring-Modul, das den Zugriff auf Datenbanken abstrahiert und vereinfacht. Es unterstützt relationale Datenbanken (z.B. über JPA/Hibernate) und NoSQL-Datenbanken und bietet einheitliche Repositories mit automatischer SQL-Query-Generierung ("Spring Data", n. d.). Für eventuelle Änderungen an dem Datenbank-Schema o.Ä. hat sich das Open-Source-Tool *Flyway* bewährt, welches Datenbankmigrationen mittels einfacher SQL-Skripte oder aber für komplexe Änderungen über Java-Code ermöglicht (baeldung, 2024).

1.3 Frontend

1.3.1 Typescript & Vite

Für die Gestaltung von Webseiten mit interaktiven und dynamischen Inhalten ist JavaScript die gängige Programmiersprache. TypeScript, eine von Microsoft entwickelte Programmiersprache, bildet eine Obermenge von JavaScript und lässt sich durch den TypeScript compiler (tsc) in gültigen JavaScript-Source-Code transkompilieren. Besonderheit dieser Programmiersprache ist die statische Typisierung, welche bei zunehmender Komplexität von Software unerwartetes Verhalten dieser verhindert. Aufgrund dessen nahm die Popularität der Sprache die letzten Jahre kontinuierlich zu ("Technology | 2025 Stack Overflow Developer Survey", n. d.) und wird auch in PJ SkillTrack eingesetzt.

Ein ebenfalls beliebtes, häufig eingesetztes Tool ist das Frontend-Build-Tool Vite, welches den TypeScript- bzw. JavaScript-Code bündelt und dem Browser bereitstellt. Es zeichnet sich durch dessen Schnelligkeit und die Anwenderfreundlichkeit gegenüber den Entwickler:innen aus ("Vite", n. d.).

1.3.2 React & Mantine

React ist eines der gängigsten Frameworks zur Erstellung von Benutzeroberflächen. Einzelne Oberflächen-Elemente werden dort in Form sogenannter Komponenten abstrahiert, hierarchisch aufgebaut und können außerdem wiederverwendet werden. Insbesondere bei verhältnismäßig wenig komplexen Webanwendungen wird die Nutzung von React der von zum Beispiel Vue oder Angular bevorzugt. Nicht zuletzt ist die Community hinter React enorm und es existiert ein breites Ökosystem an Erweiterungen wie z.B. für Frontend-Routing ("React", n. d.). Die Komponenten-Bibliothek Mantine ist eine mächtige React-Erweiterung, die Entwickler:innen eine Vielzahl moderner, robuster und anpassbarer

Oberflächen-Komponenten und individuelles Theming bereitstellt (“Mantine”, n. d.). Insbesondere in kleinen Teams und für Prototyping bzw. eine kurze Time-to-Market ist das Zurückgreifen auf vorgefertigte Komponenten eine geeignete Möglichkeit.

1.4 Sonstige Technologie

Der Projektablauf sowie der Softwarelebenszyklus können durch weitere Technologie sinnvoll ergänzt werden. Erwähnenswert ist unter anderem die Containerorchestrierungs-Software Docker. Container stellen eine isolierte Laufzeitumgebung für einzelne Prozesse/Programme dar und kapseln diese somit von der ausführenden Umgebung (Susnjara & Smalley, 2024). Einerseits ermöglicht dies die Portabilität einer Software und andererseits wird bei korrekter Konfiguration auch die Sicherheit erhöht. Die Kommunikation mit Containern findet über die Definition von klaren Schnittstellen wie ein Port-Mapping statt. Neben dem Deployment von PJ SkillTrack als Container-Anwendung selbst kann z.B. auch der Betrieb der dazugehörigen PostgreSQL-Datenbank auf einem containerbasierten Datenbank-Server erfolgen.

Eine weitere wichtige Komponente ist der Prozess der Continuous Integration (CI) beziehungsweise des Continuous Deployment (CD). Erstes beschreibt die Fähigkeit, jederzeit eine lauffähige und auslieferbare Software auf dem neusten Stand zur Verfügung zu haben, inklusive dem eigentlichen Bauen der Software, der Ausführung von Tests, dem Scan auf Sicherheitslücken, der Überprüfung von Coding-Guidelines uvm. CD ist ergänzend hierzu die fortlaufende Veröffentlichung von Softwareupdates auf Produktions- oder Staging-Systeme, oft in Kombination mit einer Cloud-Infrastruktur und SaaS. Für diese Prozesse gibt es eine Vielzahl an bewährten Anbietern. Eine einfache Möglichkeit ist die Nutzung von GitHub Actions, welche die regelmäßige Ausführung von sogenannten Workflows, also Schrittfolgen von Aufgaben im Bereich von CI/CD, ermöglichen (“GitHub Actions”, 2025). Dies erspart insbesondere im Fall von PJ SkillTrack übermäßige Konfiguration, da das Repository selbst auf GitHub verwaltet wird.

2 Archiktekturübersicht

2.1 Architektur & Schichtentrennung

Die entwickelte Anwendung basiert auf einer klassischen Client-Server-Architektur mit einer klaren Trennung zwischen Frontend, Backend und Datenbank. Dieses Modell ermöglicht eine saubere Verantwortlichkeitsverteilung und erleichtert somit die Wartung. Zudem stellt diese Architektur, neben Peer-to-Peer-Netzwerken, den klassischen Aufbau von Applikationen im Internet dar, indem eine zentrale Instanz (der Server, das Backend) viele Clients (das Frontend) mit Daten versorgt.

Das Frontend wurde mit React(1.3.2) in Kombination mit Vite(1.3.1) umgesetzt. React bietet eine komponentenbasierte Struktur, die eine modulare Entwicklung fördert und die Wiederverwendung von UI-Elementen ermöglicht. Vite dient als moderner Build- und Entwicklungsserver, der durch u.a. optimierte Bundling-Prozesse die Entwicklungsproduktivität erheblich steigert. Über eine REST-API kommuniziert das Frontend mit dem Backend und bleibt so unabhängig von der konkreten Implementierung der Geschäftslogik.

Das Backend basiert auf Spring Boot(1.1.3), das die Entwicklung von Java-Anwendungen im Unternehmens-Umfeld erheblich vereinfacht. Durch den Einsatz des Spring-Ökosystems(1.1.2) kann eine Schichtenarchitektur realisiert werden:

- Präsentationsschicht: Bereitstellung der REST-Controller, die HTTP-Anfragen entgegennehmen und Antworten an das Frontend zurückliefern.
- Geschäftsschicht: Implementierung der Geschäftslogik und Orchestrierung der Abläufe.
- Persistenzschicht: Verwaltung der Datenzugriffe mittels JPA/Hibernate und Anbindung an die Datenbank.

Diese Aufteilung unterstützt eine klare Trennung von Zuständigkeiten, wodurch Änderungen in einer Schicht nur minimale Auswirkungen auf andere Schichten haben. Eine korrekt implementierte Schichtenarchitektur im Backend fördert insgesamt die Wartbarkeit und Erweiterbarkeit. Im Zusammenspiel mit der Client-Server-Architektur ergibt sich eine robuste, skalierbare und verständliche Architektur, die sich sowohl für den Anwendungsfall von PJ SkillTrack als auch für zukünftige Erweiterungen eignet.

2.2 Innere Struktur

2.2.1 Persistenzschicht

Das gewählte Design der Klassenstrukturen im Quellcode unterstützt die vorangegangene Differenzierung der Schichtenarchitektur im Backend. Die Entitäten bilden die Datenstruktur des Schemas der zugrundeliegenden Datenbank ab und werden im Folgenden Klassendiagramm dargestellt [1]:

[1] Für die Erstellung der Diagramme wurde die Webseite PlantUML(https://www.plantuml.com/) verwendet

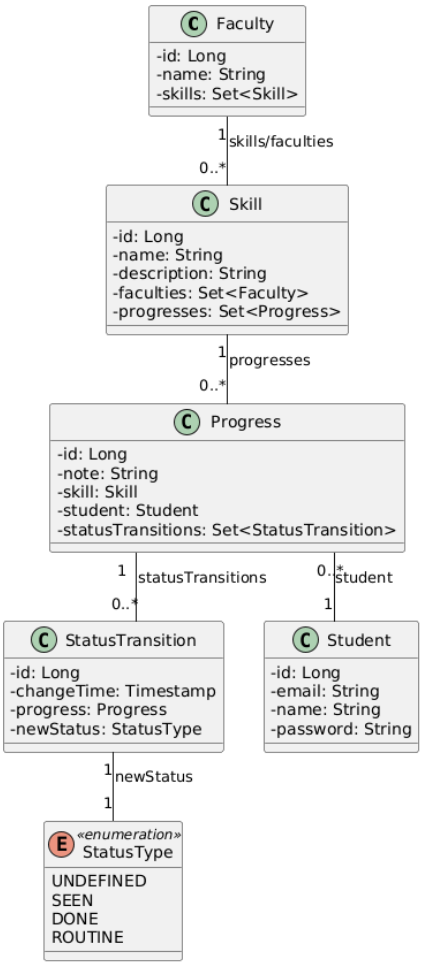

Abbildung 2.1: UML-Klassendiagramm der zentralen Entitäten im Backend.

Neben den Entitäten spielen die sogenannten JPA-Repositories eine zentrale Rolle. Diese stellen eine Abstraktion der Datenbank dar und erlauben direkte Zugriffe auf die in der jeweiligen Tabelle enthaltenen Elemente. So werden für PJ SkillTrack folgende Repositories definiert:

- SkillRepository
- FacultyRepository
- StudentRepository
- ProgressRepository
- StatusTransitionRepository
- StatisticsRepository: Hier werden Statistiken über mehrere Tabellen hinweg aus der Datenbank ausgelesen.

2.2.2 Geschäftsschicht

Die Geschäftslogik wird im Spring-Kontext über sogenannte Service-Klassen abgebildet. Der Application-Context, welcher durch Spring bereitgestellt wird, erlaubt die einmalige Instantiierung von entsprechend annotierten Klassen (z.B. Controller, Services), welche dann projektweit in weiteren im Context enthaltenen Klassen verwendet werden können (ähnlich dem Singleton-Pattern).

PJ Skill-Track stellt folgende Services bereit, welche in Form eines Interfaces abstrahiert werden. Durch die Nutzung von Interfaces wird der nutzenden Klasse eine Schnittstelle geboten, welche auf verschiedene Art implementiert werden kann.

- FacultyService, bzw. FacultyServiceImpl
- SkillService, bzw. SkillServiceImpl
- ProgressService, bzw ProgressServiceImpl
- StudentService, bzw. StudentServiceImpl
- StatisticsService, bzw. StatisticsServiceImpl

Ebenfalls erwähnenswert ist der *SkillConverter*, eine Hilfskomponente, die einen Skill je nach authentifizierten Student mit den Informationen des dazugehören Progresses bzw. aktuellen Status verbindet.

2.2.3 Präsentationsschicht

Schließlich stellen sogenannte Controller eine einheitliche Rest-Schnittstelle nach außen hin bereit, welche durch den Client angesprochen werden kann. Diese Controller verwenden jeweils die passenden Service-Klassen, um die Client-Anfragen zu bearbeiten.

Controller	Schnittstelle	Beschreibung
FacultyController	GET /api/faculty	Liefert eine Liste aller Fakultäten.
SkillController	GET /api/skill?facultyName	Liefert die Skills einer Fakultät
SkillController	PUT /api/skill/skillId	Aktualisiert die Notiz und/oder den Status eines Progresses (bzw. Skills pro Student)
StatisticsController	GET /api/statusPerFaculty	Liefert Statistiken zum Status von Skills pro Fakultät
StudentController	GET /api/auth/me	Liefert Informationen zum authentifizierten Nutzer/Studenten

Tabelle 2.1: Verwendete Controller und implementierte Rest-Schnittstellen.

2.2.4 Beispielhafte Abläufe

Ein zentraler, viele Bereiche der Anwendung durchlaufender Prozess ist die Abfrage nach den Skills einer bestimmten Fakultät. Hierfür wird aus dem Client die Anfrage an den Controller gestellt, der wiederum mithilfe des SkillServices und dem SkillRepository die Informationen aus der Datenbank erhält. Anschließend werden diese Abbildungen der Tabelleneinträge der Skill-Tabelle über den Skill-Converter - wiederum mithilfe des Student- sowie Progress-Service - zu handlichen Transferobjekten konvertiert, mit denen im Frontend alle nötigen Informationen in strukturierter Form dargestellt werden können:

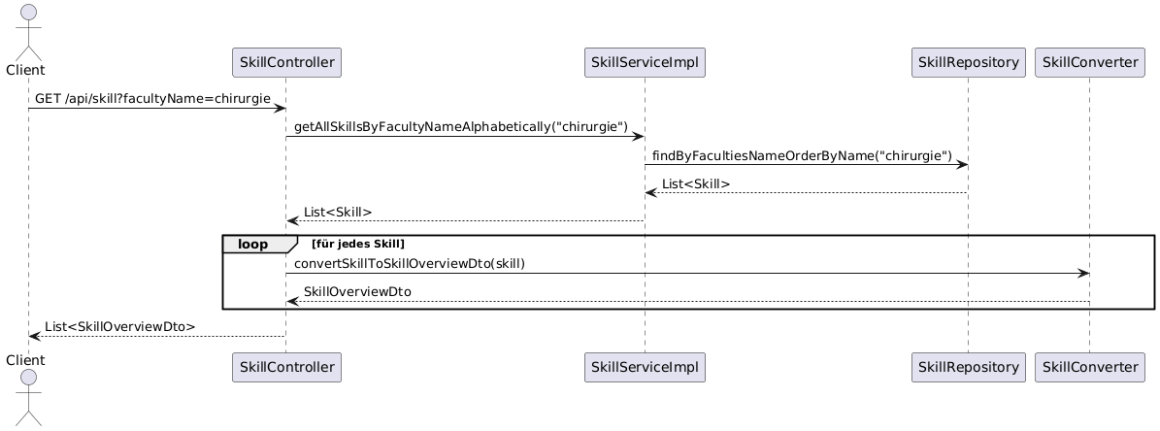

Abbildung 2.2: UML-Sequenzdiagramm für den Prozess zum Erhalten der Skill-Informationen.

Ein weitaus komplexerer Ablauf ist das Aktualisieren eines SKills bzw. der dazugehörigen Entitäten *Progress* oder *StatusTransition*. Hier sind deutlich mehr Klassen involviert:

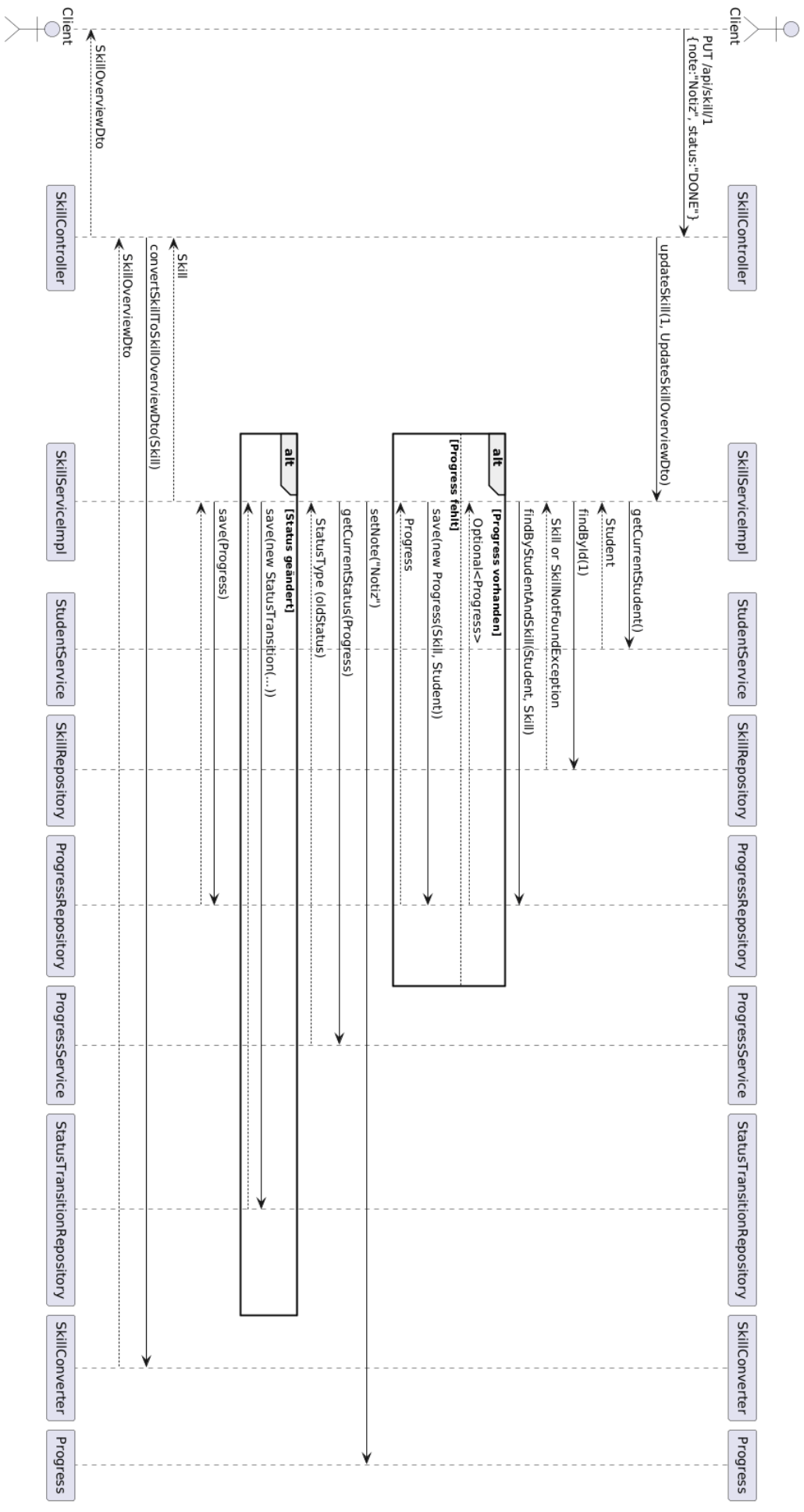

Abbildung 2.3: UML-Sequenzdiagramm für den Prozess zum Aktualisieren eines Skills.

A Abbildungsverzeichnis

B Tabellenverzeichnis

C Abkürzungsverzeichnis

D Literatur

baeldung. (2024, Januar). Database Migrations with Flyway | Baeldung. Verfügbar 4. September 2025 unter https://www.baeldung.com/database-migrations-with-flyway.

GitHub Actions. (2025). Verfügbar 4. September 2025 unter https://github.com/features/actions.

mansisha8y75. (2025, August). Introduction to Build Tools [Section: Java]. Verfügbar 4. September 2025 unter https://www.geeksforgeeks.org/java/what-is-a-build-tool/.

Mantine. (n. d.). Verfügbar 4. September 2025 unter https://mantine.dev/.

PostgreSQL: About. (n. d.). Verfügbar 4. September 2025 unter https://www.postgresql.org/about/.

React. (n. d.). Verfügbar 4. September 2025 unter https://react.dev/.

Spring Boot. (n. d.). Verfügbar 4. September 2025 unter https://spring.io/projects/spring-boot.

Spring Data. (n. d.). Verfügbar 4. September 2025 unter https://spring.io/projects/spring-data.

Spring Security. (n. d.). Verfügbar 4. September 2025 unter https://spring.io/projects/spring-security.

Susnjara, S., & Smalley, I. (2024, Juni). Was ist Docker und wie wird es benutzt? | IBM. Verfügbar 4. September 2025 unter https://www.ibm.com/de-de/think/topics/docker.

Technology | 2025 Stack Overflow Developer Survey. (n. d.). Verfügbar 4. September 2025 unter https://survey.stackoverflow.co/2025/technology#most-popular-technologies-language-prof.

Vite. (n. d.). Verfügbar 4. September 2025 unter https://vite.dev.

Why Spring. (n. d.). Verfügbar 4. September 2025 unter https://spring.io/why-spring.

Wicksell, T., Cellucci, T., Yuan, H., Bross, A., Yap, N., & Liu, D. (2018, Dezember). Netflix OSS and Spring Boot — Coming Full Circle. Verfügbar 4. September 2025 unter https://netflixtechblog.com/netflix-oss-and-spring-boot-coming-full-circle-4855947713a0.