15 Fazit und Plattformvergleich

Thema

Backend-Konzepte auf verschiedene Programmiersprachen und Frameworks übertragen.

Ausgangslage

Du hast in diesem Modul eine Backend-API geplant, umgesetzt, getestet und abgesichert. Die grundlegenden Konzepte bleiben dabei unabhängig von der verwendeten Plattform gleich. Die Namen, Dateistrukturen und technischen Werkzeuge können sich jedoch unterscheiden.

Auftrag

Erstelle zum Abschluss eine Vergleichsübersicht für die folgenden Plattformen:

- ASP.NET Core mit C#
- Spring Boot mit Java
- Node.js mit Express und TypeScript
- NestJS mit TypeScript

Vergleiche mindestens diese Konzepte:

- Einstiegspunkt und Anwendungskonfiguration
- Routing und Controller
- Middleware und Pipeline
- Dependency Injection
- Services und Geschäftslogik
- Repository und ORM
- JSON-Serialisierung und Validierung
- Unit Tests und Mocking
- End-to-End-Tests
- Session-basierte Authentifizierung
- JWT-Authentifizierung
- Rollen- und funktionsbasierte Autorisierung
- Fehlerbehandlung

Beschreibe zusätzlich mindestens fünf grosse Unterschiede. Unterscheide dabei zwischen:

- einem Unterschied des Programmiermodells;
- einem Unterschied des Frameworks;
- einem Unterschied einer Bibliothek oder eines Werkzeugs;
- einem Unterschied, der nur durch die gewählte Projektstruktur entsteht.

Vergleichstabelle

Plattformunabhängiges Konzept	ASP.NET Core / C#	Spring Boot / Java	Express / TypeScript	NestJS / TypeScript
Controller / Route	Controller oder Minimal API	`@RestController`, `@GetMapping`	`app.get()`, `router.get()`	`@Controller`, `@Get()`

Plattformunabhängiges Konzept	ASP.NET Core / C#	Spring Boot / Java	Express / TypeScript	NestJS / TypeScript
Middleware / Pipeline	ASP.NET Core Middleware	Servlet Filter, Interceptor, Security Filter Chain	Middleware mit `next()`	Middleware, Guards, Pipes, Interceptors
Service	Service-Klasse per Dependency Injection	`@Service`	Modul oder Service-Klasse	`@Injectable()`
Repository / ORM	Entity Framework Core, `DbContext`	JPA, Hibernate, Spring Data	Prisma oder TypeORM	Prisma oder TypeORM
Validierung	Model Binding, Data Annotations, Validatoren	Bean Validation, `@Valid`	Bibliothek wie Zod oder Joi	DTOs und Validation Pipes
Authentifizierung	Cookie- oder JWT-Handler	Spring Security, Session oder JWT	Middleware, Passport oder eigene Strategie	Guards, Passport und Auth-Module
Autorisierung	Rollen und Policies	Rollen, Authorities und Method Security	Middleware oder Berechtigungs-funktionen	Guards und Berechti-gungslogik
Unit Tests / Mocking	xUnit, NUnit oder Moq	JUnit und Mockito	Jest oder Node Test Runner	Jest und Nest Testing Utilities
End-to-End-Tests	ASP.NET TestServer oder externe API-Tools	MockMvc, Testcontainers oder externe API-Tools	Supertest, `curl` oder Postman	Supertest, `curl` oder Postman

Vorgehen

- Lies die ausgefüllte Vergleichstabelle und die Erläuterungen.
- Markiere die Begriffe, die du bereits aus dem Modul kennst.
- Notiere dir drei Unterschiede, die du für deine weitere Ausbildung besonders wichtig findest.

Vergleiche zusätzlich einen selbst erstellten Backend-Skill mit der Umsetzung in zwei Plattformen. Beurteile, welche Teile fachlich gleich bleiben und welche Teile nur durch Framework, Sprache oder Werkzeug anders aussehen.

Fazit

Du musst nicht jede Backend-Plattform vollständig kennen, bevor du mit ihr arbeiten kannst. Entscheidend ist, dass du die Grundprinzipien verstehst: HTTP, REST, JSON, Statuscodes, Schichten, Datenzugriff, Authentifizierung, Autorisierung und Tests.

Die Plattformen verwenden dafür unterschiedliche Sprachen, Begriffe und Werkzeuge. Wenn du die gemeinsamen Prinzipien verstanden hast, kannst du die Dokumentation einer neuen Technologie lesen, bekannte Konzepte wiedererkennen und dich vergleichsweise schnell in die konkrete Plattform einarbeiten.

Zeitbudget

Zeit: ca. 3–4 Stunden