Theorie Backend 01

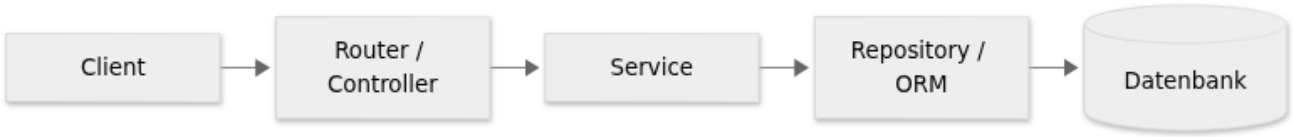
Ziel

Du lernst, wie Backend-Anwendungen HTTP-Anfragen verarbeiten, Daten liefern, Geschäftslogik ausführen und Zugriffe absichern. Die Konzepte werden zuerst mit `curl` und Node.js/Express sichtbar gemacht und anschliessend mit NestJS, ASP.NET Core und Spring Boot verglichen.

Zentrale Konzepte

- HTTP Semantics
- HTTP request methods
- HTTP response status codes
- REST
- JSON
- Express Middleware
- NestJS Request Lifecycle
- ASP.NET Core Middleware
- Spring Boot REST Services
- Jest Mock Functions
- Postman Tests

Schichtenmodell



Die Aufgabengebiete bauen aufeinander auf. Lies die passenden Abschnitte und bearbeite danach die Aufgaben im jeweiligen Verzeichnis.

Teststrategie

Unit Tests prüfen eine kleine Einheit isoliert, meistens eine Funktion oder einen Service. Abhängigkeiten wie Repositories oder externe APIs können gemockt werden, damit der Test schnell, deterministisch und gezielt fehlerhafte Rückgaben prüfen kann.

End-to-End-Tests prüfen die Anwendung vom HTTP-Eingang bis zur Antwort. Sie verwenden die laufende Anwendung und möglichst realistische Infrastruktur. `curl` macht die HTTP-Kommunikation sichtbar; Postman eignet sich später für Collections, Variablen und automatisierte Assertions.

Eine In-Memory-Datenbank ist schneller und einfacher als eine echte Datenbank, kann aber sich bei SQL, Datentypen, Constraints, Migrationen und Transaktionen anders verhalten. Deshalb ersetzen In-Memory-Tests keine Tests gegen die produktive Datenbanktechnologie.

KI-gestützte Backend-Entwicklung

KI wird im Abschlussprojekt zuerst für die Spezifikation und erst danach für die Implementierung eingesetzt. Ein Backend-Skill macht fachliche Anforderungen prüfbar und verbindet API, Geschäftslogik, Datenbank, Sicherheit und Tests.

```
Fachliche Funktion → Backend-Skill → API-Vertrag und Akzeptanzkriterien
                   → Implementierung → Tests und Review
```

Ein KI-Vorschlag ist kein Beweis für eine korrekte Lösung. Besonders Authentifizierung, Autorisierung, Validierung, SQL, personenbezogene Daten und Fehlerbehandlung müssen anhand der eigenen Anforderungen geprüft werden.

Werkzeuge

- Node.js und Express für die erste Umsetzung
- `curl` für die ersten HTTP-Tests
- PostgreSQL und ein ORM für Datenzugriffe
- Postman als spätere Ergänzung für organisierte API-Tests