🚀 Ultimate Backend Development Course – Spring Boot Edition

Learn Spring Boot from Scratch and Level Up to a Senior Software Engineer Role

---

🔥 Top Highlights

This course is your complete guide to backend mastery. Designed to match real-world engineering standards, it not only teaches backend programming—it trains you to think like a systems designer, a scalable architect, and a performance-first engineer.

🧠 Most Advanced Concepts You'll Master:

- **CQRS (Command Query Responsibility Segregation)**: Scale high-read or high-write workloads independently. A must-have in fintech and analytics-heavy domains.
- **API Composition & Materialized View Pattern**: Aggregate data across microservices in milliseconds—vital for dashboards, home feeds, and listings.
- **SAGA Pattern**:
  - **Orchestration**: Use centralized services to coordinate workflows like booking + payment.
  - **Choreography**: Design event-driven microservices where each component reacts to events.
- **Event Sourcing with Kafka**: Reconstruct entire system state from immutable logs. Used for audit compliance, data replay, and analytics.
- **Transactional Outbox Pattern + CDC (Debezium)**: Guarantee safe event publishing from the database—used in payment, messaging, and supply-chain systems.

- **Consistency Models (Eventual, Causal, Immediate)**: Choose the right trade-offs for high-availability systems. Understand when and why banks use strict consistency and when social apps use eventual.
- **DB Internals**: Learn WAL (Write-Ahead Logs), MVCC, buffer pool, and LSM Trees. Used by every modern RDBMS and NoSQL engine.
- **GeoHashing & Location Indexes**: Build Uber-like location lookups using Redis GEO, spatial indexes, and quad tree comparisons.
- **Distributed Locks (Redis, DB)**: Learn when to use pessimistic vs optimistic locking and how Redis helps coordinate across pods and servers.
- **API Gateways & Service Meshes**: Secure, throttle, route, and manage API access using Kong or Spring Gateway.
- **Service Discovery**: Learn Eureka, Consul, and dynamic resolution in scalable service meshes.
- **Database Replication & Sharding**: Implement master-slave, multi-master, and leaderless replication. Learn partition strategies for scale.

💻 Flagship Projects Included

🔷 Major Projects:

1. **Uber Backend**
    - Geo-search for drivers using Redis GEO and Haversine
    - Real-time updates with WebSocket and Kafka
    - Surge pricing, driver-location tracking, trip lifecycle
2. **Payment Wallet System**
    - CQRS + Event Sourcing + Outbox Pattern + Kafka
    - Transactional integrity with rollback and retry

- Used for P2P transfers like Paytm or Uber Wallet

3. **Airbnb Booking System**
   - Calendar sync, host/guest flows, messaging
   - JWT-secured microservices, role-based auth
   - Booking race conditions handled with distributed locks

🔷 Minor Projects:

4. **Quora Clone**
   - Feed ranking, upvote/downvote scoring, reply trees
   - MongoDB aggregations, nested population
   - Full-text search with ElasticSearch integration
5. **Hotel Management System**
   - Room allocation, billing, service management
   - Shift scheduling and availability sync
   - Monolith → microservice refactoring journey

🧱 Build Systems & Java Ecosystem

- Master Gradle for real-world CI/CD and packaging.
- Write reusable modules with shared interfaces, contracts, DTOs.
- Create fat JARs for Dockerization and production-ready apps.
- JVM GC types, tuning memory, JIT compiler internals.
- Build AOP annotations for tracing, logging, auth injection.

🧠 Low-Level Design (LLD)

- Implement common patterns: Singleton, Builder, Strategy, Observer.
- DTO → Mapper → Domain flow using MapStruct.
- Create reusable validators, request interceptors, and service contracts.
- Understand how good LLD unlocks scalable HLD.

🔌 REST API Development

- REST principles done the right way: idempotency, status codes, HATEOAS.
- Pagination, filtering, rate-limiting, and versioned APIs.
- Exception handling with @ControllerAdvice and global error handling middleware.
- Integrate Swagger/OpenAPI for real-time documentation.

🧩 Microservices & Project Architecture

- Transition from Monolith to Modular Monolith to Microservices.
- DDD-driven boundaries for independent deployments.
- Use Spring Cloud Config, Eureka for centralized configuration & discovery.
- Retry, circuit breakers (Resilience4J), fallback and bulkheads.
- Feign client for service-to-service internal calls.

🧵 Messaging & Event-Driven Systems

- Kafka setup, brokers, zookeepers, replication factors, partitions.
- Kafka with Spring Boot using KafkaTemplate and listeners.
- Kafka Streams for ETL, sliding windows, and aggregations.
- Outbox pattern implementation with PostgreSQL + Debezium.

✍️ CQRS + Outbox + Debezium CDC

- Split read/write DBs for scaling independently.
- Use CDC to detect inserts in outbox tables.
- Publish those changes as Kafka events reliably.
- Ideal for financial apps, audit systems, logistics.

🔁 Redis & Caching

- Spring Cache abstraction over Redis.
- Read-through, write-around, and cache invalidation strategies.
- Distributed locks using Redisson.
- Use Redis streams and pub/sub for real-time communication.

🔗 gRPC & Protobuf

- Define Protobuf messages and RPC services.
- Generate Java code and create gRPC servers & clients.

- Use streaming RPCs for real-time messaging.
- Benchmark gRPC vs REST vs Thrift in payload & latency.

🗃️ Spring Data JPA & ORM

- Entity inheritance: Table-per-class, Joined, Single Table.
- Relationships with Cascade and Fetch types.
- Lazy loading vs Eager loading and N+1 problem.
- Flyway integration for versioned schema changes.

💾 Advanced Databases

- MVCC, WAL, isolation levels, redo logs explained.
- Indexes: B-tree, Hash, GIN/GiST in PostgreSQL.
- Triggers: before/after insert/update for automation.
- ACID & CAP theorem applications in real-world systems.

⚙️ Auth & Security

- Spring Security, JWT, refresh tokens, method-level security.
- Role-based access control (RBAC), fine-grained permissions.
- OAuth2 integration for social login.

📡 Real-Time Systems

- WebSocket + SockJS + STOMP for messaging.
- Kafka as backbone for message queue in large systems.
- Use rooms/namespaces for scalable WebSocket architecture.

✅ Testing & TDD

- Unit testing with JUnit5 and Mockito.
- Integration testing with Testcontainers (Kafka, MySQL, Redis).
- REST Assured for API contract tests.
- Mutation testing and test coverage metrics.

📦 CI/CD, Docker & Observability

- Dockerfile best practices + Docker Compose stacks.
- GitHub Actions workflows for test/build/deploy.
- Prometheus + Grafana + Micrometer metrics.
- ELK Stack: Filebeat, Logstash, Elasticsearch, Kibana.
- Distributed tracing with OpenTelemetry and Jaeger.

🧠 System Design & Patterns (Throughout the Course)

- Consistent Hashing, Cache Invalidation, Circuit Breakers.
- Database sharding, replication, quorums, CAP tradeoffs.
- Design Twitter timelines, Uber geo-routing, Airbnb booking flow.
- Apply trade-offs of availability vs consistency in real scenarios.

🎯 **Final Outcome:** Walk away not just with knowledge—but real systems engineering wisdom. You'll be able to:

- Design and build cloud-native backend systems end-to-end
- Understand how large-scale systems are architected
- Implement real-world backend features like payments, bookings, chat, and more
- Ace backend interviews at top product companies

If you're serious about backend engineering—this course is your launchpad.