FACULDADE FACINT

PÓS-GRADUAÇÃO LATO SENSU

Modalidade EaD

Arquitetura de Software em Java

Carga Horária Total: 380h

Email: secretaria@facint.edu.br
Telefone: +55 (44) 3024-4242
Website: www.facint.edu.br

Apresentação do Curso

Domine as competências críticas para projetar sistemas complexos, escaláveis e resilientes utilizando o ecossistema Java de última geração. Desenvolva uma visão estratégica sobre decisões arquiteturais que impactam diretamente a performance e a manutenção de grandes ativos digitais. Eleve seu patamar profissional com metodologias avançadas exigidas pelas principais empresas de tecnologia do mercado global.

Pós-Graduação (Lato Sensu) TIPO DE CURSO	**EaD** MODALIDADE
380h CARGA HORÁRIA	**4, 6 ou 12 meses** DURAÇÃO

Perfil do Egresso

O especialista formado estará apto a liderar a concepção de soluções robustas, aplicando padrões de projeto e escolhas tecnológicas fundamentadas em requisitos de alta disponibilidade. Este profissional dominará a transição entre a codificação avançada e o planejamento macro de sistemas distribuídos, atuando como um facilitador técnico e estratégico. Sua expertise incluirá a governança de microsserviços e a otimização de infraestruturas modernas em nuvem.

Campo de Atuação

Atue como Arquiteto de Software, Arquiteto de Soluções ou Team Lead em consultorias de TI, fintechs e grandes corporações que utilizam Java no core business. O profissional terá portas abertas para posições de liderança técnica em projetos internacionais que demandam alta escalabilidade e integração de sistemas complexos. Além disso, poderá atuar na governança técnica e na definição de padrões de engenharia de software em centros de inovação.

Diferenciais do Curso

- ✓ **Currículo focado nas demandas reais de sistemas críticos de alta performance.**
- ✓ **Corpo docente composto por arquitetos de referência no mercado corporativo de TI.**
- ✓ **Metodologia EaD flexível que permite a conclusão acelerada conforme seu ritmo.**
- ✓ **Acesso a bibliotecas digitais atualizadas e conteúdos focados em tecnologias Java Moderno.**

Matriz Curricular

Nº	Disciplina
1	Fundamentos de Arquitetura de Software
2	Design Patterns e Clean Code em Java
3	Arquitetura de Microsserviços com Spring Cloud
4	Persistência de Dados e Modelagem Avançada
5	Arquiteturas Orientadas a Eventos (EDA)
6	Cloud Native Java e Containers
7	Qualidade, Testes e Integração Contínua
8	Segurança e Proteção de Dados (LGPD)
9	Performance Tuning e Multithreading
10	DevOps, Observabilidade e Monitoramento

Ementas das Disciplinas

1. Fundamentos de Arquitetura de Software

- Análise profunda dos atributos de qualidade (ilities) e sua influência direta no design de sistemas escaláveis.
- Estudo comparativo entre estilos arquiteturais como Monólitos Modulares, Client-Server e Service-Oriented Architecture (SOA).
- Técnicas de documentação arquitetural utilizando modelos visuais como C4 Model e diagramas UML 2.0 avançados.
- Gestão de tradeoff e tomada de decisão fundamentada em métricas de acoplamento e coesão sob a ótica arquitetural.
- Implementação do Processo de Desenvolvimento de Software com foco na arquitetura e ciclo de vida de ativos.

2. Design Patterns e Clean Code em Java

- Aplicação prática de padrões de projeto GoF (Creational, Structural e Behavioral) em cenários de alta complexidade.
- Refatoração de código legado com foco em Clean Code e princípios SOLID para garantir testabilidade e longevidade.
- Implementação de padrões arquiteturais emergentes como Fluent Interfaces e Immutability no ecossistema Java Moderno.
- Estratégias de Defensive Programming e tratamento de erros avançado para aumentar a resiliência do software.
- Utilização de Aspect-Oriented Programming (AOP) para separação transversal de preocupações como logging e segurança.

3. Arquitetura de Microsserviços com Spring Cloud

- Desenvolvimento de ecossistemas distribuídos utilizando Spring Boot e governança de serviços com Spring Cloud.
- Implementação de padrões de resiliência como Circuit Breaker, Bulkhead e Retry com Resilience4j.
- Configuração de Service Discovery e API Gateways escaláveis para gerenciamento de tráfego e segurança perimetral.
- Estratégias de Distributed Configuration Management para manter ambientes de desenvolvimento, homologação e produção.
- Monitoramento e rastreamento distribuído de requisições utilizando tecnologias como Spring Cloud Sleuth e Zipkin.

4. Persistência de Dados e Modelagem Avançada

- Otimização de performance com Hibernate e JPA através de estratégias de caching, fetch tuning e batch processing.
- Modelagem de dados para bancos NoSQL (Document, Key-Value, Graph) integrados a aplicações Java via Spring Data.
- Gerenciamento de transações distribuídas e consistência eventual em ambientes de alta disponibilidade.
- Implementação de CQRS (Command Query Responsibility Segregation) para separação de fluxos de leitura e escrita.
- Migração de esquemas e evolução de bancos de dados em ambientes CI/CD utilizando ferramentas como Flyway ou Liquibase.

5. Arquiteturas Orientadas a Eventos (EDA)

- Concepção de sistemas baseados em eventos utilizando Apache Kafka e RabbitMQ para desacoplamento de serviços.
- Implementação do padrão Event Sourcing para manutenção do estado de entidades através de históricos de eventos.
- Processamento de streams em tempo real com Kafka Streams e integração com ecossistema Spring Cloud Stream.
- Padrões de confiabilidade em mensageria como Outbox Pattern e Idempotency para garantir a entrega exata de mensagens.
- Orquestração vs. Coreografia em fluxos de negócios complexos dentro de arquiteturas distribuídas.

6. Cloud Native Java e Containers

- Containerização avançada de aplicações Spring Boot com Docker e otimização de imagens para baixo footprint.
- Orquestração de microsserviços com Kubernetes, abordando deployments, serviços, ingress e auto-scaling.
- Desenvolvimento de aplicações 'Cloud Native' seguindo rigorosamente a metodologia 12-Factor App.
- Migração de aplicações On-Premise para arquiteturas Serverless utilizando AWS Lambda com Java e GraalVM.
- Implementação de Service Mesh com Istio para controle de tráfego, observabilidade e políticas de segurança.

7. Qualidade, Testes e Integração Contínua

- Estratégias de Testing Pyramid: do JUnit 5 e Mockito até testes de integração complexos com Testcontainers.
- Implementação de Test-Driven Development (TDD) e Behavior-Driven Development (BDD) em fluxos de engenharia.
- Automação de pipelines de CI/CD utilizando Jenkins ou GitHub Actions personalizados para o ecossistema Java.
- Análise estática de código e governança técnica com SonarQube para medição de débito técnico e segurança.
- Testes de performance e carga em APIs RESTful utilizando ferramentas como JMeter ou Gatling.

8. Segurança e Proteção de Dados (LGPD)

- Implementação de autenticação e autorização robustas utilizando Spring Security, OAuth2 e OpenID Connect (OIDC).
- Gestão de identidades e acesso centralizado com Keycloak integrado a ecossistemas Java distribuídos.
- Criptografia de dados sensíveis e conformidade técnica com as normas da LGPD no desenvolvimento de software.
- Prevenção ativa contra vulnerabilidades do OWASP Top 10 em aplicações web e APIs Java.
- Auditoria e logging de segurança com integração em ferramentas de SIEM para monitoramento em tempo real.

9. Performance Tuning e Multithreading

- Deep dive no funcionamento da JVM: Garbage Collection tuning, JIT Compiler e gerenciamento de Heap Memory.
- Desenvolvimento concorrente avançado utilizando Java Concurrency API, Virtual Threads (Project Loom) e Locks.
- Profiling de aplicações com VisualVM e JProfiler para identificação de memory leaks e gargalos de processamento.
- Otimização de tempo de resposta através de programação reativa com Project Reactor e Spring WebFlux.
- Uso estratégico de mecanismos de cache distribuído como Redis e Hazelcast para redução de latência.

10. DevOps, Observabilidade e Monitoramento

- Implementação de Infrastructure as Code (IaC) com Terraform para provisionamento de ambientes Java na nuvem.
- Configuração de dashboards de monitoramento com Prometheus e Grafana para visualização de métricas de negócio e sistema.
- Centralização e análise de logs estruturados com a stack ELK (Elasticsearch, Logstash e Kibana).
- Práticas de Site Reliability Engineering (SRE) aplicadas ao ciclo de vida de desenvolvimento de software Java.
- Gestão de incidentes e análise de causa raiz (Post-mortem) apoiada por telemetria avançada e APM.