Ementa Java Elite

1. Fundamentos do Java

- **História da linguagem e evolução**
- **Por que aprender Java?**
- **Estruturas de dados:**
  - List: ArrayList e LinkedList
  - Map: HashMap, LinkedHashMap, TreeMap, Hashtable, ConcurrentHashMap, WeakHashMap, IdentityHashMap, EnumMap
  - Set: HashSet, LinkedHashSet, TreeSet, EnumSet, CopyOnWriteArraySet, **ConcurrentSkipListSet**
- **Orientação a objetos: Herança, Polimorfismo, Abstração e Encapsulamento**
- **Principais APIs do Java:**
  - Concurrency Framework
  - File I/O (NIO)
  - Date and Time API
  - Optional API
  - Reflection API
  - Annotation Processor
  - Annotations API
  - Logging API
  - Networking API
  - Security API
  - Serialization API
  - JDBC APIInternationalization (I18N) API
  - Regular Expressions API
  - Stream API
  - Lambda Expression e Functional interface

- **Manipulação de arquivos e entrada/saída com NIO 2**
- **Tratamento de erros e exceções**
- **Configuração do ambiente de desenvolvimento no IntelliJ IDEA e Eclipse**
- **Maven para gestão de dependências**
- **Introdução ao Gradle**

2. Desenvolvimento de Aplicações Back-End + IA

- **Spring Framework e Spring Boot:**
  - Introdução e configuração de projetos
  - Desenvolvimento de APIs REST
  - Autenticação e autorização com OAuth2 e JWT
  - Consumo e exposição de APIs usando OpenAPI/Swagger
  - Integração com bancos de dados usando Spring Data JPA
  - Gerenciamento de transações
  - Trabalhando com WebFlux para programação reativa

- **Quarkus:**
  - Introdução e configuração de projetos
  - Desenvolvimento de APIs REST com Quarkus
  - Autenticação e segurança com Quarkus Security e JWT
  - Consumo e exposição de APIs usando Swagger/OpenAPI no Quarkus
  - Integração com bancos de dados usando Panache
  - Otimização de desempenho através de compilação nativa (usando Graal VM)
  - Configuração e uso de extensões para expandir a funcionalidade
  - Implementação de microsserviços com Quarkus
  - Integração com Kafka e outros sistemas de mensagens
  - Implementação de serviços resilientes através da Fault Tolerance API
  - Desenvolvimento de serviços Kubernetes Native
  - Observabilidade com Open Telemetry

- **IA Corporativa com Java e Langchain4j:**
  - **Fundamentos e Configuração do Ambiente**
    - Entender o papel do desenvolvedor Java na nova fronteira da IA corporativa
    - Diferença entre IA preditiva e IA generativa
    - Princípio central: foco em integração e não em ciência de dados
    - Visão arquitetural de uma aplicação de IA e como Java se encaixa nesse ecossistema

  Retrieval Augmented Generation (RAG)
    - RAG como alternativa prática ao fine tuning de modelos
    - Fundamentos de ingestão, recuperação e aumento de contexto
    - O papel dos embeddings como base da busca semântica
    - Do RAG em memória a bancos vetoriais de produção
    - Princípios de qualidade da resposta: filtragem, transformação de consultas e reordenação de resultados

  Construindo Agentes Inteligentes (Agentic AI)
    - De chatbots passivos para agentes autônomos ativos

- O ciclo de raciocínio como motor de tomada de decisão
- “Tools” como extensões de capacidade de negócio

- Importância da memória e do gerenciamento de estado em interações de múltiplos passos
- Princípio de orquestração: agentes como coordenadores de lógica e execução

Integração Avançada com Model Context Protocol (MCP)

- Problema do acoplamento entre agentes e ferramentas
- MCP como padrão para interoperabilidade e desacoplamento
- Princípio de serviços distribuídos aplicado a ferramentas de IA
- Arquitetura distribuída: agentes que descobrem e consomem capacidades remotas de forma transparente

Segurança e Governança em Aplicações de IA

- Novos vetores de ataque introduzidos por LLMs (injeção de prompt, vazamento de dados)
- Guardrails como camada de defesa
- Princípios de validação e moderação de entradas e saídas
- Menor privilégio como prática essencial em ferramentas de agentes

Produção, Cache, Resiliência e Observabilidade

- Cache como princípio de eficiência em performance e custos
- Resiliência com timeouts, retries e tratamento robusto de falhas
- Princípio de tolerância a erros em execução de ferramentas
- Observabilidade em sistemas não determinísticos
- Tracing e métricas

3. Fundamentos de Front-End com React

- **Introdução ao desenvolvimento front-end para programadores back-end**
- **Conceitos fundamentais do React:**
  - Componentes funcionais e estado
  - Propriedades (props) e eventos
  - Ciclo de vida de componentes e hooks básicos (useState, useEffect)
- **Construção de interfaces simples:**
  - Criação de formulários com validação
  - Consumo de APIs REST utilizando Axios e Fetch
- **Gerenciamento de estado:**
  - Introdução ao Context API
  - Noções básicas de Redux para projetos maiores

- **Configuração de projetos:**
  - Ambiente de desenvolvimento com Create React App e Vite
  - Integração com ferramentas como ESLint e Prettier
- **Boas práticas de organização de código em front-end**
- **Deploy de aplicações front-end simples:**
  - Uso do Netlify e Vercel para deploy rápido
  - Integração com APIs back-end desenvolvidas em Java

4. Arquitetura de Sistemas

- **Software Architecture vs Software Design**
- **Arquitetura em camadas e Clean Architecture**
- **Arquitetura baseada em eventos:**
  - Utilização de Apache Kafka e RabbitMQ
  - Implementação de CQRS e Event Sourcing
- **Coupling/Decoupling**
- **Refactoring**

5. Software Design & System Design

- **Software Design**
  - **Padrões de projeto (Design Patterns):**
    - Singleton
    - Factory
    - Strategy
    - Observer
    - Builder
    - Prototype
    - Adapter
    - Decorator
    - Proxy
  - **Introdução ao DDD (Domain-Driven Design)**
    - Introdução ao DDD estratégico
    - Bounded Context
    - Context Mapping
    - Linguagem Ubíqua
    - Patterns of integrations
  - **DDD tático**
    - Patterns
    - Overview
    - Entity

- Aggregator
- Repository

- **System Design**
  - **Princípios Básicos de Design de Sistemas:**
    - Componentes de um sistema: front-end, back-end, bases de dados, caches
    - Escalabilidade: vertical vs horizontal
    - Disponibilidade e consistência: CAP theorem e trade-offs
    - Latência e throughput: como otimizar ambos
  - **Banco de Dados**
    - SQL
    - NoSQL
    - NewSQL
    - CAP/PACELC
  - **Projetando Sistemas Escaláveis:**
    - Particionamento de dados (sharding)
    - Balanceadores de carga (Load Balancers)
    - Uso de caches (Redis, Memcached) para otimização
    - Bancos de dados distribuídos e replicação
  - **Comunicação Entre Componentes:**
    - APIs síncronas (REST, gRPC)
    - Comunicação assíncrona com filas e brokers de mensagens (Kafka, RabbitMQ)
    - Design de interfaces robustas para integração
  - **Alta Disponibilidade e Tolerância a Falhas:**
    - Projetos resilientes com Circuit Breakers e retries
    - Estratégias de replicação e failover
    - Ferramentas para monitoramento e recuperação automática (auto-healing)
  - **Design de Casos Práticos:**
    - Sistema de reserva (como Uber ou Booking)
    - Feed de redes sociais (como Instagram ou Twitter)
    - E-commerce de alta escala (como Amazon)

6. Concorrência e Multithreading

- **Threading em Java**
- **Executors, Fork/Join Framework**
- **CompletableFuture e programação reativa**
- **Gerenciamento de concorrência com locks, semáforos e synchronizers**
- **Virtual Thread**

7. Infraestrutura e Cloud Computing

- **Docker e Kubernetes:**
  - Construção e otimização de imagens Docker
  - Construção e otimização de imagens Podman
  - Kubernetes para desenvolvedores (POD, ReplicaSet, Deployment, Service, Secrets, etc)
- **Introdução ao AWS para Java:**
  - Integração com serviços como S3, Lambda, e DynamoDB

- **CI/CD:**
  - Configuração de pipelines com Jenkins e GitHub Actions

8. Bancos de Dados

- **Modelagem de banco de dados relacional**
- **Frameworks de Persistência de Dados**
  - JPA
  - Jakarta Data
  - Hibernate

Consultas nativas vs JPQL

- **Integração com bancos NoSQL**
  - Introdução ao Jakarta NoSQL
  - Redis
  - Cassandra
  - MongoDB
  - Neo4J
- **Ferramentas de migração: Flyway e Liquibase**

9. Testes

- **Estratégias de TDD (Test-Driven Development)**
- **Unit tests com JUnit Jupiter, AssertJ eMockito**
- **Tests estáticos (Spotbug/PMD)**
- **Testes de integração com Spring e Quarkus**
- **Usando Testcontainers no Teste de Integração**
- **Metricas de Qualidade e Sonar**
- **Test Data Driven Approach**
- **ArchUnit (Testando o código Design)**
- **Testes de performance e benchmarking**

10. Como Atrair as Melhores Vagas do Mercado

- **Otimização de perfis no LinkedIn para atrair recrutadores**
- **Estratégias para aumentar o Social Selling Index (SSI)**
- **Networking estratégico: como se conectar com recrutadores e líderes técnicos**
- **Uso de palavras-chave e técnicas de SEO para ser encontrado por empresas**
- **Construção de uma marca pessoal forte e posicionamento no mercado**
- **Criação de um portfólio técnico atrativo no GitHub**
- **O que se espera de você no cargo?**
  - Junior Developer
  - Pleno Developer
  - Senior Developer
  - Tech lead
  - Staff Engineer
  - Principal Engineer
  - Software Manager

11. Como Passar em Qualquer Entrevista de Emprego

- **Entrevistas de RH:**
  - Técnicas para estruturar respostas baseadas na metodologia STAR (Situação, Tarefa, Ação, Resultado)
  - Como transmitir confiança e alinhamento cultural
  - Estratégias para lidar com perguntas sobre pontos fracos e expectativas salariais
- **Entrevistas Técnicas:**
  - Preparação para live coding e resolução de problemas em tempo real
  - Simulações de entrevistas técnicas para desafios em Java
  - Explicação de escolhas técnicas e decisões arquiteturais
- **Negociação Salarial e Vendas:**
  - Como comunicar seu valor e negociar salários competitivos
  - Estratégias para pedir aumentos ou renegociar contratos Técnicas de diferenciação para se destacar como o candidato ideal.