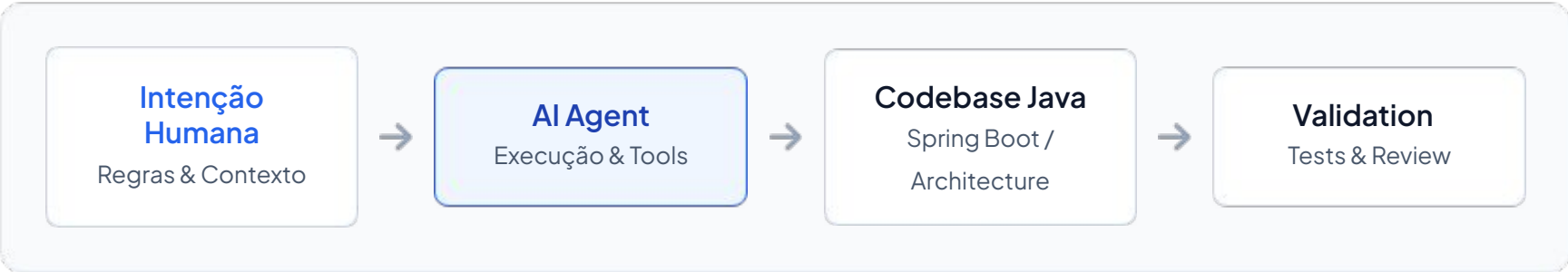
Luiz Real

PALESTRA TÉCNICA JAVA

O futuro do desenvolvimento Java:

AI-first SDLC

Java, agentes e o novo papel do desenvolvedor em um mundo orientado por intenção



Agenda

- **A mudança:** de syntax para intent
- **Comparativo:** Vibe coding vs agentic engineering
- **Infraestrutura de IA:** Context engineering e harness
- **Evolução de carreira:** O novo papel do desenvolvedor Java
- **Demo ao vivo:** Construção de CRUD Spring Boot do zero com Devin

A Tese Central

> ”A maior transformação atual não é uma nova linguagem ou framework. É a mudança de escrever código para especificar intenção.”

Engenharia Permanece

AI não elimina a necessidade de software engineering; o rigor técnico continua indispensável.

Mudança de Foco

O esforço do desenvolvedor migra da digitação de sintaxe para a modelagem do problema.

Pilares Críticos

Architecture, tests, verification e judgment humano tornam-se os ativos mais valiosos.

Por Que Isso Importa para Java

Estrutura Previsível

Java trabalha nativamente com contratos, camadas claras, convenções e foco em maintainability.

Essas características facilitam para que AI agents naveguem e ajam com segurança em codebases reais.

Ecosistema Auditável

O ecossistema Java favorece software fortemente especificado, testável, auditável e pronto para evolução.

A disciplina histórica do ecossistema Java se torna um super poder na era dos AI agents.

Key Takeaway: Java não está sendo deixado para trás. A disciplina técnica que sempre foi marcante na comunidade Java ganha valor multiplicado com agents.

De Syntax para Intent

Menos tempo em código boilerplate

Mais tempo em goals, arquitetura e critérios

Novo foco: desenhar o sistema de execução

Do Coding Assistant ao Agent

Coding Assistant

- Sugere linhas e pequenos blocos de código.
- Atua de forma reativa, focado no editor IDE.
- Exige orientação, cópia e ajuste contínuo.

AI Agent

- Analisa e entende o repositório por completo.
- Planeja mudanças estruturadas em múltiplos arquivos.
- Usa ferramentas, roda testes e prepara Pull Requests completos.

O salto real: O impacto não é apenas gerar sintaxe melhor, é aumentar drasticamente o **escopo da tarefa que pode ser delegada**.

Vibe Coding vs Agentic Engineering

Vibe Coding

Exploração rápida e prototipação guiada por intuição.

- Baixa verificação formal inicial
- Ideal para POCs e experimentação rápida
- Risco alto em produção

Agentic Engineering

Desenvolvimento rigoroso ancorado em regras e automação.

- Ambiente e restrições altamente estruturados
- Verification, guardrails e testes contínuos
- Projetado para ambientes corporativos e missão crítica

Conclusão: Não é uma divisão binária. É um **spectrum** definido pela quantidade de estrutura, verificação e julgamento humano colocados ao redor do agent.

A Diferença Real

”A diferença não é usar AI.
A diferença é como o output é validado.”

PERGUNTA FRÁGIL

”Parece funcionar?”

Foco visual ou amostragem superficial sem garantias de regras de negócio ou estabilidade.

PERGUNTA ENGENHARADA

”Respeitou constraints, passou nos testes e está pronto para review?”

Validação rigorosa automatizada alinhada à arquitetura do sistema.

Quanto maior o risco e a complexidade do domínio, menos espaço existe para o improviso.

Context Engineering

Prompt engineering é apenas uma pequena fração. **Context engineering** é preparar todo o ambiente para que o agent execute com precisão.

Regras & Style

Convenções de código, naming patterns e padrões da equipe.

Arquitetura

Divisão de camadas (Controller, Service, Repository) e DTOs.

Exemplos

Trechos de referência do repositório para manter a consistência.

Memória

Decisões anteriores (ADRs) e estado atual da codebase.

Tools

Acesso ao compilador (Maven/Gradle), CLI e suíte de testes.

Critérios

Definition of Done clara e validações de qualidade.

Static vs Dynamic Context

Static Context

Contexto fixo e sempre presente no ambiente do agent:

- Instruções globais do repositório
- Padrões arquiteturais definidos
- Contratos de API e políticas institucionais
- Architecture Decision Records (ADRs)

Dynamic Context

Informações carregadas estritamente sob demanda:

- Requisito ou issue atual em execução
- Arquivos relevantes e dependências do trecho
- Logs de execução e falhas de testes
- Diffs e alterações recentes

A divisão inteligente entre contexto permanente e sob demanda é uma **decisão de arquitetura do processo**.

O Que É Harness Engineering

Agent = Model + Harness

Tools & Sandbox

Ambiente isolado para compilação Java, execução de comandos terminal e ferramentas do sistema.

Tests & Middleware

Execução contínua de JUnit/Mockito e conectores que guiam o ciclo de refatoração.

Guardrails & Observability

Limites de segurança, telemetria de ações do agent e regras que impedem degradação.

Mensagem-chave: O modelo é apenas a CPU. A confiabilidade do resultado depende do **harness (sistema)** ao redor dele.

O Novo Papel do Desenvolvedor Java

MODO 1

Conductor Mode

Atuação tática no nível do código:

- Interação próxima dentro da IDE
- Controle fino de cada classe/método
- Uso de chat e sugestões inline

MODO 2

Orchestrator Mode

Atuação estratégica no nível do sistema:

- Define goals, contexto e boundaries
- Delega tarefas complexas para agents
- Avalia diffs, arquitetura e decisões

O futuro não elimina o desenvolvedor. Ele **eleva o nível de abstração** em que o desenvolvedor atua.

O Que Muda nos Perfis Profissionais

Para Estudantes

- **Aceleração:** IA acelera a exploração e o feedback de aprendizado.
- **Fundamentos:** Algoritmos, orientação a objetos e redes continuam essenciais.
- **Novo foco:** É preciso aprender a *verificar* o código, não apenas gerá-lo.

Para Experientes

- **Produtividade:** Drástica redução de tempo gasto em tarefas repetitivas.
- **Foco Estratégico:** Mais tempo dedicado à arquitetura, especificação e review.
- **Alavancagem:** Capacidade de multiplicar o impacto individual através de agents.

O Que Muda para Times Java

- **Além das Ferramentas:** Não basta adotar um assistente; é necessário reestruturar workflow, governança e quality gates.
- **Convenções Explícitas:** Padrões antes implícitos precisam ser documentados formalmente no contexto do projeto.
- **Automação Integrada:** Testes automatizados e pipelines de CI tornam-se parte ativa do fluxo do agent.
- **Gargalo na Verificação:** A alta velocidade de geração de código só se converte em vantagem se a **capacidade de verificação acompanhar**.

Onde Começar na Prática

1. Endpoints Delimitados

Criar novos endpoints simples ou pequenas expansões de API.

2. Geração de Testes

Aumentar a cobertura de testes unitários e de integração existentes.

3. Refatorações Locais

Modernização pontual de código antigo ou otimizações puras.

4. Documentação Técnica

Gerar Javadoc, esquemas OpenAPI e resumos de arquitetura.

5. Pequenas Migrações

Atualizações pontuais de bibliotecas e utilitários.

Estratégia: Começar pequeno permite aprender a configurar contexto, medir retrabalho e expandir a autonomia gradualmente.

Demo Ao Vivo: Cenário Real

Do Requisito ao CRUD Completo com Devin

Projeto & Agent

- **Projeto Base:** API Spring Boot real existente em produção.
- **AI Agent:** Devin (autônomo via ambiente completo).

Objetivo da Demo

- Criar um novo CRUD do zero absoluto.
- **Escopo:** Nova Entidade, Repository, Service, Endpoints Controller e Suíte de Testes.

Ponto de atenção na demo: A IA acelera a execução; o desenvolvedor estabelece contexto, restrições e aceitação final.

Fluxo da Demonstração

1

Arquitetura

Apresentação do projeto existente e convenções.

2

Requisito

Definição clara do novo CRUD a ser criado.

3

Contexto

Fornecimento de padrões, pacotes e aceite.

4

Execução

Devin analisa e implementa as camadas.

5

Review

Análise de diff, testes e decisão final.

https://github.com/lgsreal/api-rest-pw/pull/1

CONCLUSÃO

”O futuro do Java não é menos engineering. É mais engineering em torno de agentes.”

1. Interface

Intent becomes the new primary interface.

2. Skill Core

Context engineering becomes a core developer skill.

3. Responsabilidade

Verification and judgment remain human responsibilities.

Java continua altamente relevante, mas o desenvolvedor Java precisa aprender a atuar em um SDLC AI-first.

Referências

1. **The New SDLC with Vibe Coding**
   Google Whitepaper

2. **AI-Driven Development**
   Context Engineering

3. **Devin University**
   Learn Devin by Doing

4. **The AI Engineering Skills Map**
   Andrew Ng

5. **The Community Dev**
   Personal tech blog