Curso ATVC200 - Vibe coding com o Claude Code | Do protótipo à produção

Duração: 16h

Introdução

Fazer um protótipo com IA é rápido; pô-lo em produção e mantê-lo é outro trabalho. Este curso é para programadores que já usam o Claude Code e querem levar o código gerado por IA até à produção com testes, revisão, segurança e integração contínua, sempre na versão do Claude Code em vigor à data de cada turma.

Em quatro sessões, cada participante leva uma aplicação, a sua ou a de exemplo do curso, do protótipo a uma entrega: especifica e planeia, fixa as regras do projeto no CLAUDE.md, cobre o código com testes, automatiza verificações com hooks, revê e audita o que o agente escreveu, limita o que ele pode fazer e liga o Claude Code ao pipeline de CI/CD.

É a continuação do curso Claude Code | Construir Aplicações Reais com IA. Os exercícios usam JavaScript e TypeScript, mas os métodos aplicam-se a qualquer linguagem.

Público-alvo

Este curso destina-se a programadores e a equipas de desenvolvimento que já usam o Claude Code, ou outro agente de programação, e querem passar dos protótipos a código que vai para produção. É indicado também para responsáveis técnicos e de qualidade que têm de definir como a equipa trabalha com agentes de IA.

No final do curso

Após a conclusão deste curso, os formandos deverão ser capazes de:
• Transformar uma ideia numa especificação com critérios de aceitação e num plano de alterações pequenas.
• Organizar o CLAUDE.md e as regras do projeto para obter resultados consistentes.
• Gerir o contexto, o modelo, o nível de esforço e o consumo.
• Usar testes, linters e hooks como verificações que o agente corre sozinho.
• Depurar com checkpoints, Git e worktrees.
• Rever código gerado por IA, com o /code-review e com subagentes de revisão.
• Identificar e corrigir as falhas de segurança mais frequentes no código gerado, incluindo segredos e dependências.
• Limitar o agente com modos de permissão, regras, sandboxing e definições geridas pela organização.
• Integrar o Claude Code no GitHub Actions e em scripts sem interação.
• Partilhar skills, subagentes e plugins com a equipa.

Pré-requisitos

• Experiência de programação e de Git: ramos, commits e pull requests.
• Utilização prévia do Claude Code, por exemplo no curso Claude Code | Construir Aplicações Reais com IA.
• Uma subscrição do Claude com acesso ao Claude Code durante toda a formação: o plano Pro ou superior, ou uma licença Team ou Enterprise da organização. O plano gratuito não inclui o Claude Code.
• Computador com Windows, macOS ou Linux, com o Claude Code, o Git e o Node.js instalados, e uma conta no GitHub.
• A subscrição do Claude não está incluída no preço do curso.

Exames de certificação

(não existem exames)

Conteúdo programático em detalhe

Do protótipo a um sistema em produção

• Vibe coding e engenharia de software com agentes: o que muda quando há utilizadores, dados e manutenção
• Os riscos do código que ninguém leu: erros silenciosos, dependências desnecessárias e dívida técnica
• O ciclo de trabalho do curso: especificar, planear, implementar em passos pequenos, verificar e rever
• A aplicação de trabalho: o projeto de cada participante ou a aplicação de exemplo do curso

Especificação e plano

• Da ideia à especificação: requisitos, critérios de aceitação e o que fica de fora
• O modo Plan: explorar o código e discutir a solução antes de qualquer alteração
• Partir o trabalho em alterações pequenas e fáceis de rever
• Exercício: especificar e planear uma funcionalidade nova

Contexto que dura: CLAUDE.md e regras

• Os níveis do CLAUDE.md: da organização, do utilizador, do projeto e local
• Importar outros ficheiros e definir regras por pasta em .claude/rules
• O que escrever e o que deixar de fora; corrigir o ficheiro quando o mesmo erro se repete
• Exercício: o CLAUDE.md e as regras do projeto

Gerir o contexto e o consumo

• A janela de contexto: /context, /compact e quando começar uma sessão nova
• Escolher o modelo e o nível de esforço para cada tarefa: /model e /effort
• Acompanhar o consumo com o /usage
• Subagentes para pesquisar sem encher o contexto principal

Testes como rede de segurança

• Dar ao Claude uma verificação que ele próprio pode correr: testes, linters e compilação
• Escrever os testes antes da implementação e impedir que o agente os altere para passarem
• Testes unitários, de integração e ponta a ponta: o que pedir em cada nível
• Exercício: cobrir com testes uma funcionalidade gerada sem eles

Hooks: regras sem exceções

• O que é um hook e os eventos mais usados: PreToolUse, PostToolUse e Stop
• Formatar, correr os testes e bloquear comandos de forma automática
• Hooks do projeto e hooks do utilizador no settings.json
• Exercício: um hook que corre os testes depois de cada alteração

Depurar com método

• Reproduzir o erro primeiro; registos e mensagens como prova
• Checkpoints e /rewind: voltar atrás no código, na conversa ou nos dois, e o que os checkpoints não cobrem
• O Git como registo: commits pequenos, ramos e worktrees para sessões em paralelo
• Exercício: encontrar e corrigir um erro introduzido de propósito

Rever código gerado por IA

• Ler diffs grandes: por onde começar e o que procurar
• O /code-review: rever as alterações antes do commit e aplicar as correções
• Subagentes de revisão com instruções próprias
• O que a revisão humana não delega: arquitetura, dados e decisões de negócio

• Exercício: rever um pull request gerado por IA

Segurança do código gerado

• As falhas mais frequentes: validação de entradas, autenticação e autorização, injeção e exposição de dados
• Segredos e chaves: fora do código, fora do repositório e fora da conversa
• Dependências: confirmar que existem, que são mantidas e que licença têm
• O /security-review: a análise das alterações de um ramo
• Exercício: auditar e corrigir a aplicação de trabalho

Limitar o que o agente pode fazer

• Modos de permissão e regras allow, ask e deny no settings.json
• Sandboxing dos comandos: isolamento de ficheiros e de rede, e as plataformas suportadas
• Injeção de instruções (prompt injection) em páginas, ficheiros e respostas de ferramentas
• Servidores MCP: o que ligam, que permissões pedem e como os avaliar
• Definições geridas pela organização (managed settings)

O Claude Code no CI/CD

• Claude Code GitHub Actions: instalar, responder a menções e correr com um pedido definido
• Revisão automática de pull requests e os seus limites
• Correr o Claude Code sem interação (claude -p) em scripts e pipelines
• Segredos, permissões e custos num pipeline
• Exercício: um fluxo que revê cada pull request do projeto

Skills, subagentes e plugins para a equipa

• Skills: os procedimentos da equipa num SKILL.md, partilhados no repositório
• Subagentes especializados em .claude/agents
• Plugins e marketplaces: distribuir skills, agentes, hooks e servidores MCP
• Exercício: uma skill para um procedimento da equipa (por exemplo, a preparação de uma entrega)

Entrega e operação

• A lista de verificação antes de publicar: testes, revisão, segurança, dados e reversão
• Publicar por etapas e voltar atrás
• Registos e monitorização: dar ao agente os dados para diagnosticar
• Documentação gerada e mantida com o projeto
• Exercício final: levar uma funcionalidade da especificação à produção